VAPT: Week-2 Report

1. Introduction

I conducted a comprehensive Vulnerability Assessment and Penetration Testing (VAPT) exercise in a controlled laboratory environment as part of my Week 2 internship assignment at CYART. I performed this assessment to identify security weaknesses in network services and web applications and to validate their exploitability using industry-standard tools and methodologies.

I carried out testing against intentionally vulnerable systems, including Metasploitable2 and Damn Vulnerable Web Application (DVWA), hosted locally in an isolated environment. I ensured that I tested only authorized targets within the defined scope.

During this engagement, I performed reconnaissance, vulnerability scanning, exploitation, post-exploitation validation, and reporting. I aligned my methodology with the Penetration Testing Execution Standard (PTES) to ensure a structured and professional approach.

This report documents my methodology, findings, exploitation results, risk analysis, and remediation recommendations.

2. Objective and Scope

2.1 Objective

I conducted this VAPT engagement to identify, analyze, and exploit security vulnerabilities in a controlled lab environment. I aimed to simulate real-world attack scenarios and validate whether identified vulnerabilities could lead to unauthorized access, data exposure, or system compromise.

During this assessment, I focused on:

- Identifying open ports and exposed services
- Detecting known vulnerabilities using automated tools
- Validating findings manually to eliminate false positives
- Exploiting confirmed vulnerabilities
- Extracting sensitive information where possible
- Recommending remediation strategies

I performed the assessment using both network-level and application-level testing techniques.

2.2 Scope

I strictly limited my testing to the following authorized targets:

Target Systems

- **Metasploitable2 (Local Virtual Machine)**
  - Purpose: Network and service exploitation
  - IP Address: 192.168.20.129

- **DVWA (Damn Vulnerable Web Application)**
  - URL: http://localhost/dvwa
  - Purpose: Web application exploitation
- **OWASP Juice Shop (Local Web Application)**
  - URL: http://localhost:3000
  - Purpose: Modern web application reconnaissance and security assessment
  - Testing Focus: Service detection, technology stack identification, directory enumeration, API endpoint discovery, and configuration analysis

I performed all testing inside an isolated lab environment using Kali Linux as the attacking machine. I did not test any production systems, external domains, or unauthorized assets. I conducted all activities in a controlled and legally authorized virtual lab setup.

2.3 Out of Scope

To maintain ethical standards and controlled testing, I excluded:

- External internet-facing systems
- Third-party services
- Any systems without explicit authorization
- Denial-of-service attacks

---

3. Tools and Environment

3.1 Testing Environment

I conducted the assessment in a controlled and isolated lab environment using virtual machines. I used Kali Linux as the attacking machine and hosted vulnerable targets within the same internal network to simulate a real-world penetration testing scenario.

Lab Setup

- Attacker Machine: Kali Linux
- Target 1: Metasploitable2
- Target 2: DVWA (hosted on Apache)
- Network Type: Internal virtual network
- Testing Date: 11 February 2026

I verified connectivity between machines before initiating testing to ensure proper communication and realistic attack simulation.

3.2 Tools Used

During this engagement, I used the following tools:

Kali Linux
I used Kali Linux as the primary penetration testing platform. Kali Linux provides pre-installed security tools required for reconnaissance, scanning, exploitation, and reporting.

Nmap
I used Nmap to perform network scanning and service enumeration. I identified open ports, running services, and service versions on the target system. I used aggressive scan options to gather detailed information about exposed services.

OpenVAS (Greenbone Vulnerability Manager)
I used OpenVAS to perform automated vulnerability scanning. I configured a full scan against Metasploitable2 and analyzed vulnerabilities based on severity and CVSS scoring.

Nikto
I used Nikto to perform web server vulnerability scanning. Nikto helped me identify outdated software, misconfigurations, and exposed components.

Metasploit Framework
I used the Metasploit Framework to exploit identified vulnerabilities. I configured payloads, established reverse shells, and validated successful exploitation using Meterpreter sessions.

sqlmap
I used sqlmap to automate SQL Injection exploitation against DVWA. I enumerated databases, extracted tables, dumped user credentials, and cracked password hashes.

DVWA (Damn Vulnerable Web Application)
I used DVWA to simulate web application vulnerabilities, specifically SQL Injection.

Metasploitable2
I used Metasploitable2 as a deliberately vulnerable Linux system for network exploitation practice.

4. Methodology (PTES-Based Approach)

I structured this assessment according to the Penetration Testing Execution Standard (PTES) to ensure a systematic and professional testing process. I followed each phase carefully to simulate a real-world security assessment.

4.1 Pre-Engagement Planning

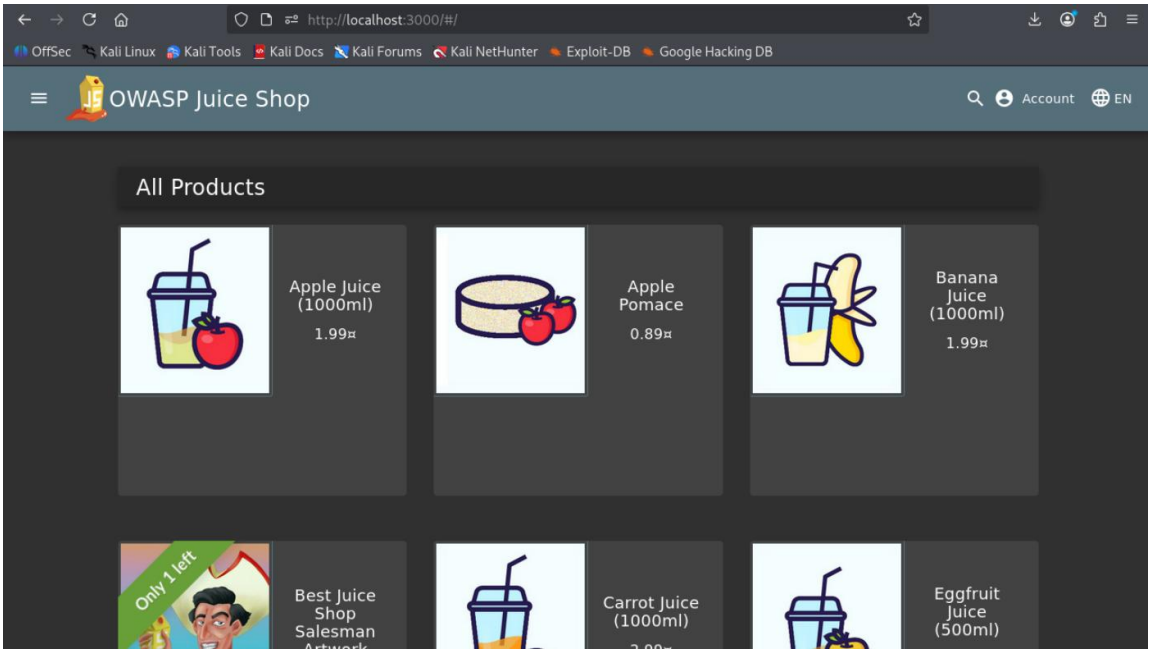
Before initiating testing, I defined the scope, identified authorized targets, and ensured that I tested only within the lab environment. I confirmed IP addresses and verified connectivity between the attacker and target machines.

I avoided testing external systems and maintained ethical standards throughout the engagement.

4.2 Reconnaissance

During this phase, I gathered initial information about the target systems. I identified IP addresses, open ports, and exposed services. I performed basic enumeration to understand the attack surface.

This phase helped me identify which services required deeper analysis.



```
┌──(kali㉿kali)-[~]
└─$ nmap -sV localhost
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-11 04:57 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000013s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE    VERSION
80/tcp   open  http
3000/tcp open  ppp?
5432/tcp open  postgresql PostgreSQL DB 9.6.0 or later
2 services unrecognized despite returning data. If you know the service/version, please submit the follo
wing fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port80-TCP:V=7.95%I=7%D=2/11%Time=698C529C%P=x86_64-pc-linux-gnu%r(HTTP
SF:Options,266,"HTTP/1\.1\x20406\x20Not\x20Acceptable\r\nDate:\x20Wed,\x20
SF:11\x20Feb\x202026\x2009:57:48\x20GMT\r\nConnection:\x20close\r\nContent
SF:-Type:\x20text/html;\x20charset=utf-8\r\nExpires:\x20-1\r\nCache-Contro
SF:l:\x20no-cache,\x20no-store\r\nPragma:\x20no-cache\r\nX-Frame-Options:\
SF:x20SAMEORIGIN\r\nContent-Security-Policy:\x20default-src\x20'none';\x20
SF:object-src\x20'none';\x20base-uri\x20'none';\x20connect-src\x20'self';\
SF:x20script-src\x20'self';\x20script-src-elem\x20'self'\x20'unsafe-inline
SF:';frame-ancestors\x20'none';\x20form-action\x20'self';\x20style-src-ele
SF:m\x20'self'\x20'unsafe-inline';\x20style-src\x20'self'\x20'unsafe-inlin
SF:e';\x20font-src\x20'self';img-src\x20'self'\x20blob:\x20data:;\r\nConte
SF:nt-Length:\x2051\r\n\r\n<html><body>HTTP\x20Method\x20not\x20supported<
SF:/body></html>")%r(RTSPRequest,D8,"HTTP/1\.1\x20400\x20Bad\x20Request\r\
SF:nDate:\x20Wed,\x2011\x20Feb\x202026\x2009:57:48\x20GMT\r\nConnection:\x
SF:20close\r\nContent-Length:\x20111\r\n\r\n<html><head><title>Request\x20
SF:malformed</title></head><body>HTTP\x20request\x20is\x20syntactically\x2
SF:0incorrect\.</body></html>")%r(SIPOptions,D8,"HTTP/1\.1\x20400\x20Bad\x
SF:20Request\r\nDate:\x20Wed,\x2011\x20Feb\x202026\x2009:57:53\x20GMT\r\nC
SF:onnection:\x20close\r\nContent-Length:\x20111\r\n\r\n<html><head><title
SF:>Request\x20malformed</title></head><body>HTTP\x20request\x20is\x20synt
SF:actically\x20incorrect\.</body></html>");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port3000-TCP:V=7.95%I=7%D=2/11%Time=698C52A1%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,101B9,"HTTP/1\.1\x20200\x20OK\r\nAccess-Control-Allow-Origin:\
SF:x20\*\r\nX-Content-Type-Options:\x20nosniff\r\nX-Frame-Options:\x20SAME
SF:ORIGIN\r\nFeature-Policy:\x20payment\x20'self'\r\nX-Recruiting:\x20/#/j
SF:obs\r\nAccept-Ranges:\x20bytes\r\nCache-Control:\x20public,\x20max-age=
SF:0\r\nLast-Modified:\x20Wed,\x2011\x20Feb\x202026\x2009:54:10\x20GMT\r\n
SF:ETag:\x20W/\"1252f-19c4c1f5ffe\"\r\nContent-Type:\x20text/html;\x20char
SF:set=UTF-8\r\nContent-Length:\x2075055\r\nVary:\x20Accept-Encoding\r\nDa
SF:te:\x20Wed,\x2011\x20Feb\x202026\x2009:57:53\x20GMT\r\nConnection:\x20c
SF:lose\r\n\r\n<!--\n\x20\x20~\x20Copyright\x20\(c\)\x202014-2026\x20Bjoer
SF:n\x20Kimminich\x20&\x20the\x20OWASP\x20Juice\x20Shop\x20contributors\.\
SF:n\x20\x20~\x20SPDX-License-Identifier:\x20MIT\n\x20\x20-->\n\n<!doctype
SF:\x20html>\n<html\x20lang=\"en\"\x20data-beasties-container>\n<head>\n\x
SF:20\x20<meta\x20charset=\"utf-8\">\n\x20\x20<title>OWASP\x20Juice\x20Sho
SF:p</title>\n\x20\x20<meta\x20name=\"description\"\x20content=\"Probably\
SF:x20the\x20most\x20modern\x20and\x20sophisticated\x20insecure\x20web\x20
SF:application\">\n\x20\x20<meta\x20name=\"viewport\"\x20content=\"width=d
SF:evice-width,\x20initial-scale=1\">\n\x20\x20<link\x20id=\"favicon\"\x20
SF:rel=\"icon\"\x20")%r(Help,2F,"HTTP/1\.1\x20400\x20Bad\x20Request\r\nCon
SF:nection:\x20close\r\n\r\n")%r(NCP,2F,"HTTP/1\.1\x20400\x20Bad\x20Reques
SF:t\r\nConnection:\x20close\r\n\r\n")%r(HTTPOptions,EA,"HTTP/1\.1\x20204\
SF:x20No\x20Content\r\nAccess-Control-Allow-Origin:\x20\*\r\nAccess-Contro
SF:l-Allow-Methods:\x20GET,HEAD,PUT,PATCH,POST,DELETE\r\nVary:\x20Access-C
SF:ontrol-Request-Headers\r\nContent-Length:\x200\r\nDate:\x20Wed,\x2011\x
SF:20Feb\x202026\x2009:57:53\x20GMT\r\nConnection:\x20close\r\n\r\n")%r(RT
SF:SPRequest,EA,"HTTP/1\.1\x20204\x20No\x20Content\r\nAccess-Control-Allow
SF:-Origin:\x20\*\r\nAccess-Control-Allow-Methods:\x20GET,HEAD,PUT,PATCH,P
SF:OST,DELETE\r\nVary:\x20Access-Control-Request-Headers\r\nContent-Length
SF::\x200\r\nDate:\x20Wed,\x2011\x20Feb\x202026\x2009:57:53\x20GMT\r\nConn
SF:ection:\x20close\r\n\r\n");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.35 seconds
```

```
┌──(kali㉿kali)-[~]
└─$ whatweb http://localhost:3000
http://localhost:3000 [200 OK] HTML5, IP[::1], JQuery[2.2.4], Script[module], Title[OWASP Juice Shop], UncommonHeaders[access
-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Options[SAMEORIGIN]
```

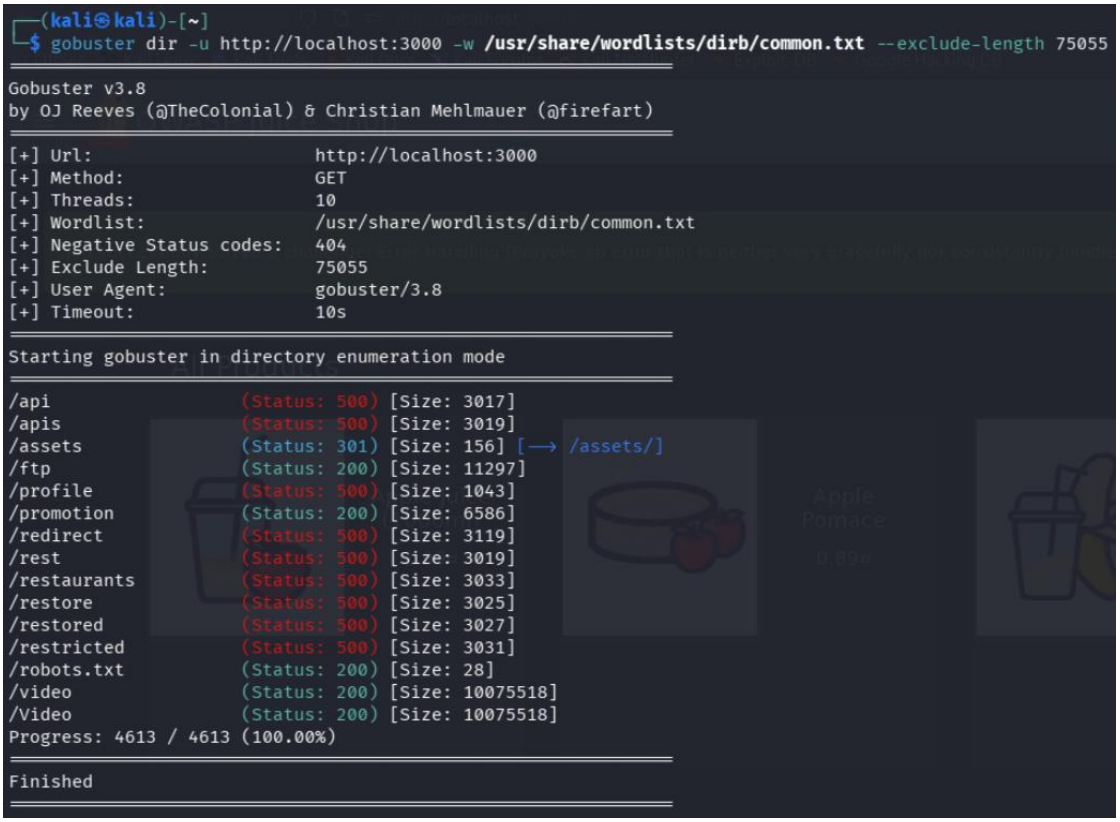


Figure 1-5: OWASP Juice Shop reconnaissance and directory enumeration results

4.3 Vulnerability Scanning

I performed automated vulnerability scanning using Nmap and OpenVAS. I identified outdated services, weak configurations, and high-risk vulnerabilities.

I analyzed scan results and prioritized vulnerabilities based on severity and CVSS scoring. I validated key findings manually to reduce false positives.

4.4 Exploitation

After identifying exploitable vulnerabilities, I attempted controlled exploitation. I used the Metasploit Framework to exploit exposed services on Metasploitable2. I also performed SQL Injection attacks against DVWA using manual testing and sqlmap.

I confirmed successful exploitation by obtaining remote shells and extracting sensitive data.

4.5 Post-Exploitation

After gaining access, I validated the level of compromise. I identified privilege levels, extracted system information, and generated cryptographic hashes to demonstrate evidence handling.

I ensured that I did not cause system instability during exploitation.

4.6 Reporting

Finally, I documented all findings, exploitation results, and remediation recommendations in this report. I categorized risks based on severity and provided actionable security improvements.

---

5. Phase 1 – Planning and Network Verification

5.1 IP Address Identification

Before initiating any scanning activity, I identified and confirmed the IP addresses of both the attacker and target systems.

- Kali Linux (Attacker): 192.168.20.128
- Metasploitable2 (Target): 192.168.20.129

I verified that both machines were operating within the same internal virtual network to ensure proper communication.

5.2 Connectivity Verification

I tested network connectivity between Kali Linux and Metasploitable2 using ICMP ping.
I executed:

```
ping 192.168.20.129
```

The target responded successfully, which confirmed network communication and readiness for further testing.

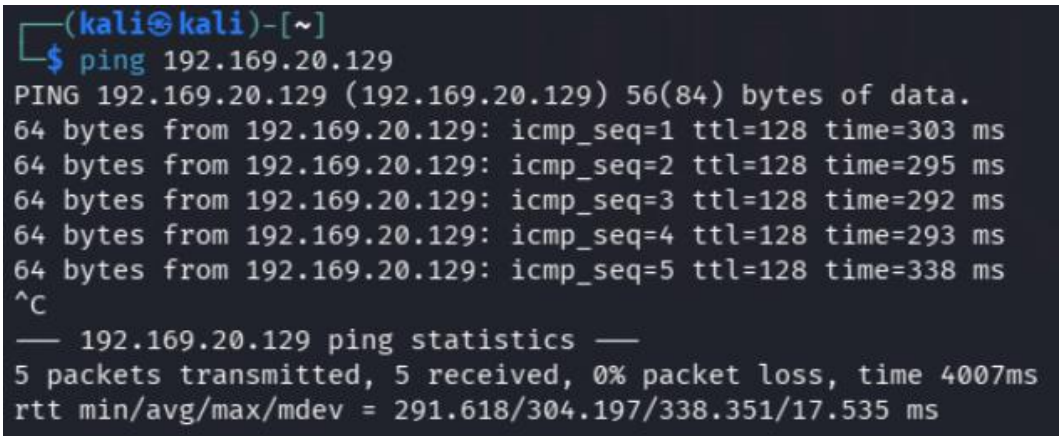


Figure 6: ICMP Ping confirming network connectivity between Kali Linux and Metasploitable2

5.3 Service Availability Confirmation

Before launching vulnerability scans, I confirmed that the target system was running and accessible. I verified that Metasploitable2 was operational and responding to network requests.

This step ensured that subsequent scanning results would be accurate and reliable.

---

6. Phase 2 – Discovery and Vulnerability Scanning

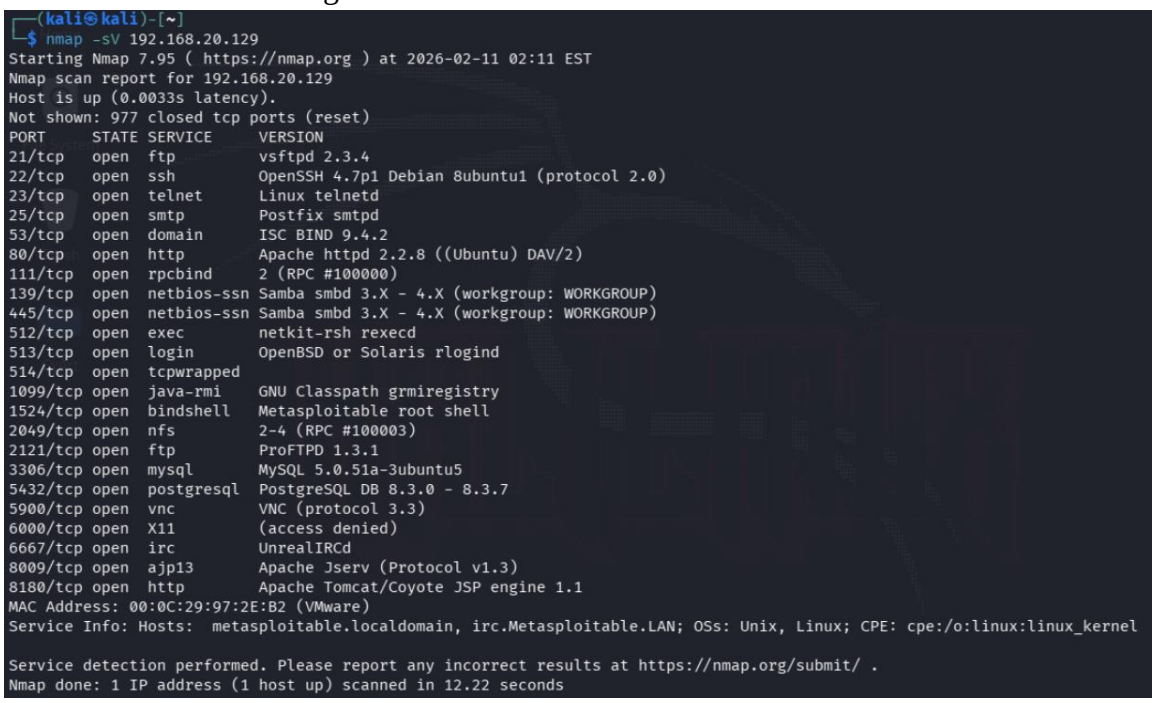
6.1 Network Scanning Using Nmap

I performed an aggressive scan against the Metasploitable2 target using Nmap to identify open ports, running services, and service versions.

I executed the following command:

```
nmap -A 192.168.20.129
```

This scan allowed me to gather detailed service and version information.



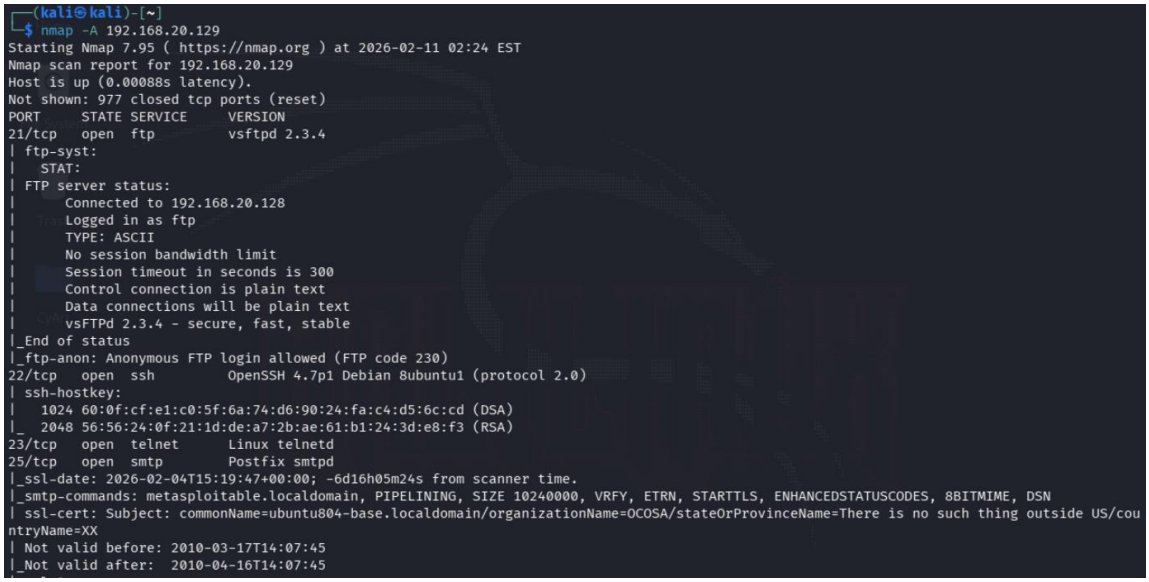


Figure 7-8: Nmap Service Version Scan results identifying open ports and service versions

6.1.1 Key Findings

The scan revealed multiple open ports and outdated services, including:

- Port 21 – FTP (vsftpd 2.3.4)
- Port 22 – SSH
- Port 23 – Telnet
- Port 80 – Apache HTTP Server 2.2.8
- Port 445 – Samba 3.0.20
- Port 8180 – Apache Tomcat 5.5
- Port 3306 – MySQL
- Port 5432 – PostgreSQL

These services significantly expanded the attack surface.

```
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|_    SSL2_RC2_128_CBC_WITH_MD5
53/tcp   open  domain      ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
111/tcp  open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/udp   nfs
|   100005  1,2,3      41119/tcp   mountd
|   100005  1,2,3      60586/udp   mountd
|   100021  1,3,4      36424/tcp   nlockmgr
|   100021  1,3,4      43345/udp   nlockmgr
|   100024  1          37067/tcp   status
|_  100024  1          56926/udp   status
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
```

```
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
| mysql-info:
|   Protocol: 10
|   Version: 5.0.51a-3ubuntu5
|   Thread ID: 2870
|   Capabilities flags: 43564
|   Some Capabilities: ConnectWithDatabase, SwitchToSSLAfterHandshake, SupportsCompression, SupportsTransactions, LongColumnFlag, Support41
Auth, Speaks41ProtocolNew
|   Status: Autocommit
|_  Salt: -JnkM"]2}DwN_r?%8*iS
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
| ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/cou
ntryName=XX
| Not valid before: 2010-03-17T14:07:45
|_Not valid after:  2010-04-16T14:07:45
|_ssl-date: 2026-02-04T15:19:45+00:00; -6d16h05m24s from scanner time.
5900/tcp open  vnc         VNC (protocol 3.3)
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    VNC Authentication (2)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
| irc-info:
|   users: 1
|   servers: 1
|   lusers: 1
|   lservers: 0
|   server: irc.Metasploitable.LAN
```

```
|   version: Unreal3.2.8.1. irc.Metasploitable.LAN
|   uptime: 0 days, 3:57:46
|   source ident: nmap
|   source host: E0738896.7E0A9117.FFFA6D49.IP
|_  error: Closing Link: qmmmpfyyq[192.168.20.128] (Quit: qmmmpfyyq)
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-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
MAC Address: 00:0C:29:97:2E:B2 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
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: 2026-02-04T10:19:35-05:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)
```

```
|_clock-skew: mean: -6d14h50m24s, deviation: 2h30m00s, median: -6d16h05m24s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)

TRACEROUTE
HOP RTT     ADDRESS
1   0.88 ms 192.168.20.129

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 30.04 seconds
```

Figure 9-12: Nmap Aggressive Scan (-A) showing detailed service, OS detection, and script results

6.2 Vulnerability Scanning Using OpenVAS

After identifying exposed services, I performed a full vulnerability scan using OpenVAS (Greenbone Vulnerability Manager).

I configured:

- Target: 192.168.20.129
- Scan Profile: Full and Fast

The scan completed successfully and identified multiple vulnerabilities across different severity levels.

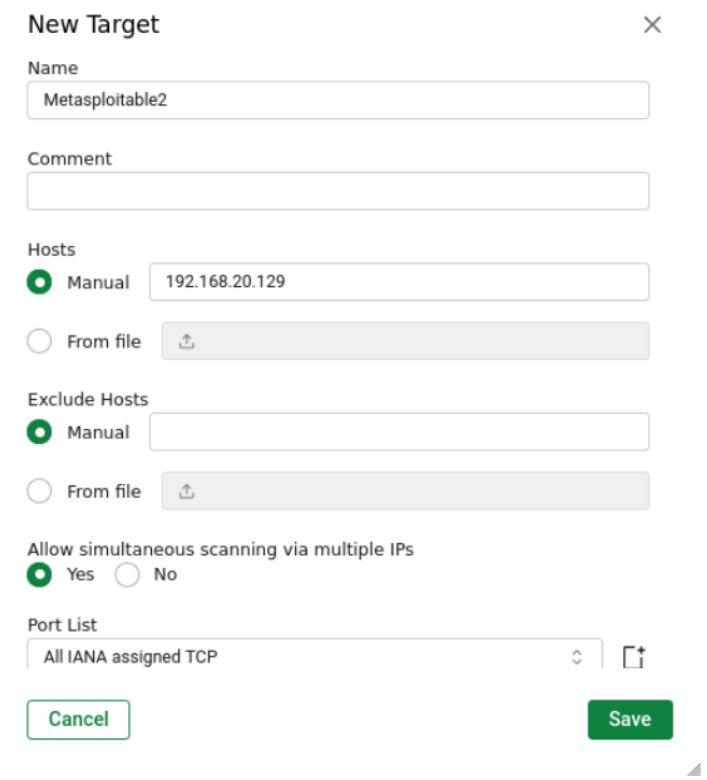


Figure 13: OpenVAS target configuration for Metasploitable2 full scan

6.2.1 Severity Breakdown

OpenVAS categorized vulnerabilities as follows:

- Critical: 14
- High: 10
- Medium: 40
- Low: 6

The highest CVSS score observed was 10.0, indicating critical system compromise potential.

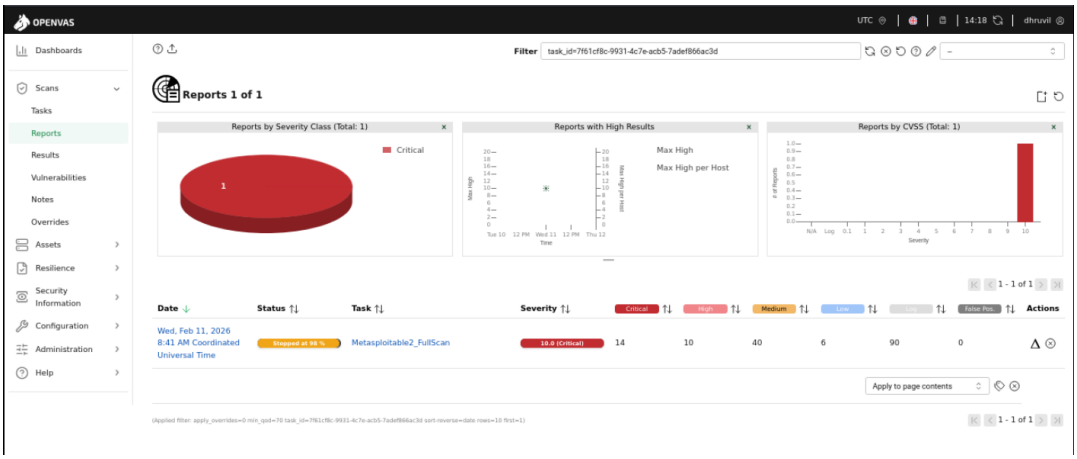


Figure 14: OpenVAS vulnerability severity distribution

6.2.2 Example Critical Vulnerability

OpenVAS identified a critical vulnerability related to the rlogin service:

- Vulnerability: Passwordless login exposure
- Port: 513
- Severity: Critical
- CVSS Score: 10.0

This vulnerability allowed potential root-level access without proper authentication.

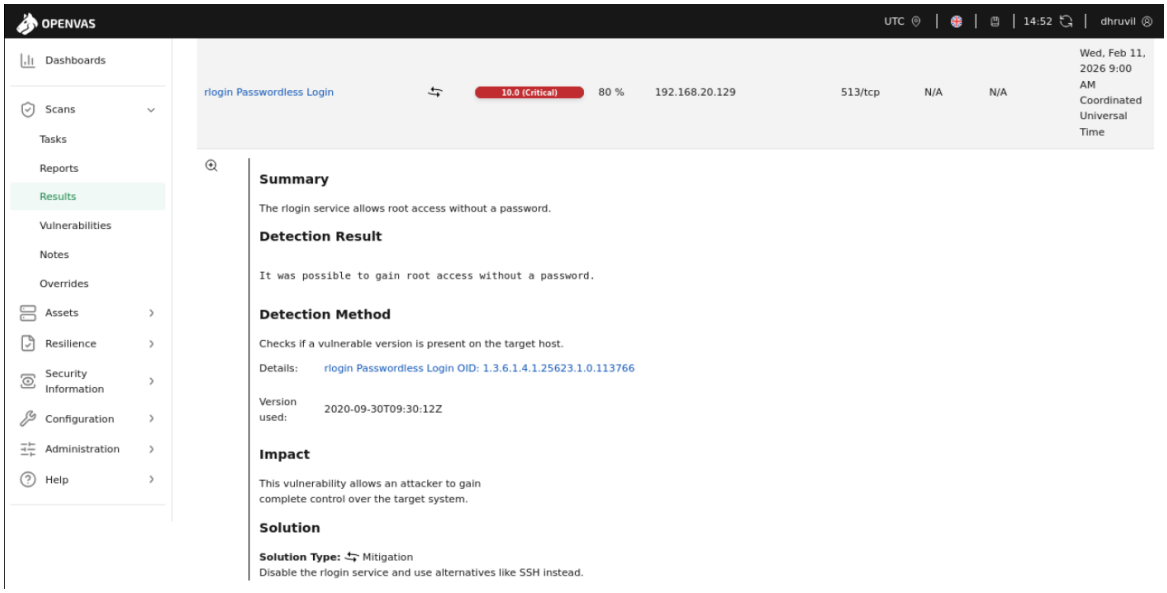


Figure 15: Critical vulnerability details identified by OpenVAS (Passwordless rlogin)

6.3 False Positive Validation

I validated key findings manually to ensure scan accuracy. I confirmed that the rlogin service was active on port 513 and that the version information matched scan results.

This validation reduced the risk of reporting false positives.

```
┌──(kali㉿kali)-[~]
└─$ nikto -h http://192.168.20.129
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.20.129
+ Target Hostname:    192.168.20.129
+ Target Port:        80
+ Start Time:         2026-02-11 02:30:27 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: index.php. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /doc/: Directory indexing found.
+ /doc/: The /doc/ directory is browsable. This may be /usr/doc. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0678
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that co
```

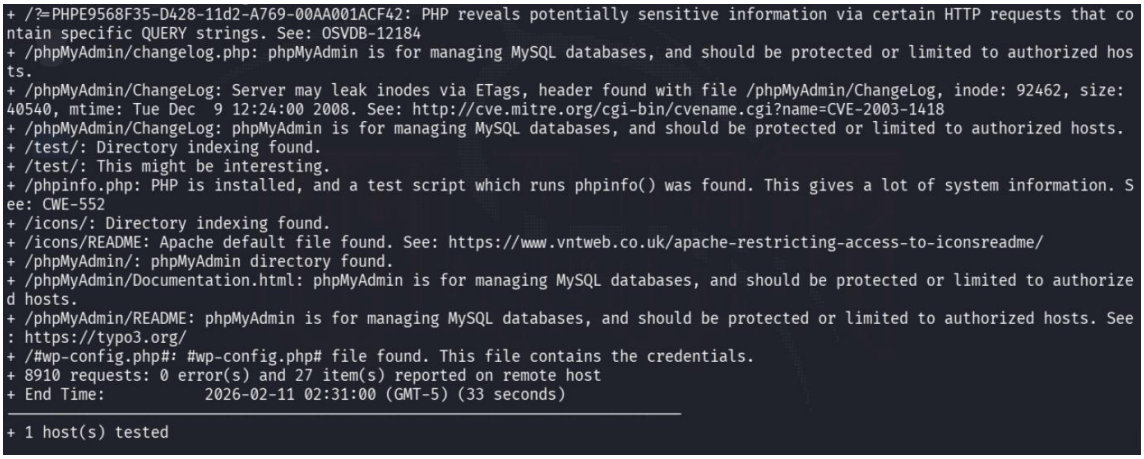


Figure 16-17: Nikto web vulnerability scan results validating misconfigurations and outdated components

7. Phase 3 – Exploitation (Metasploitable2)

7.1 Target Selection for Exploitation

After analyzing scan results, I identified Apache Tomcat 5.5 running on port 8180 as a high-value target. The service exposed the Tomcat Manager interface, which allowed administrative functionality.

I confirmed access to the Tomcat Manager page via:
http://192.168.20.129:8180/manager/html

I authenticated using default credentials (tomcat:tomcat), which confirmed the presence of weak authentication controls.

7.2 Exploitation Using Metasploit

I launched the Metasploit Framework and selected the following module:
exploit/multi/http/tomcat_mgr_deploy

I configured the exploit with the following parameters:

- RHOSTS: 192.168.20.129
- RPORT: 8180
- HttpUsername: tomcat
- HttpPassword: tomcat
- LHOST: 192.168.20.128
- LPORT: 4444

I used a reverse TCP Meterpreter payload to establish remote command execution.

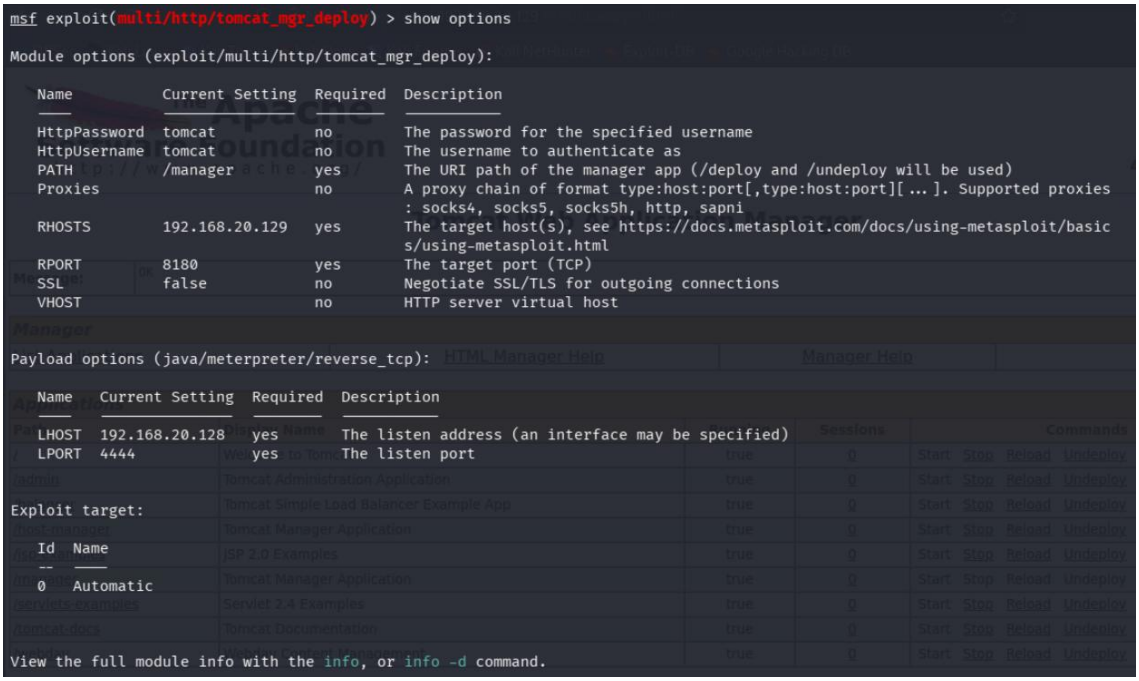


Figure 18: Metasploit configuration for Tomcat Manager exploit

7.3 Successful Exploitation

After executing the exploit, I successfully uploaded a malicious WAR file and triggered remote code execution. Metasploit established a Meterpreter session between Kali Linux and the target system.

The console displayed:
Meterpreter session 1 opened

This confirmed full remote access to the target machine.

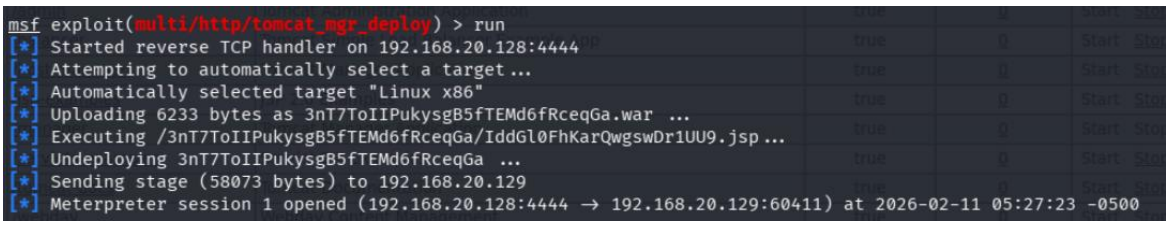


Figure 19: Successful deployment and Meterpreter session establishment

7.4 Access Verification

Inside the Meterpreter session, I executed:

```
sysinfo
getuid
```

The system information confirmed that I gained access to the Linux target system. The user context demonstrated successful authentication bypass and remote control capability.

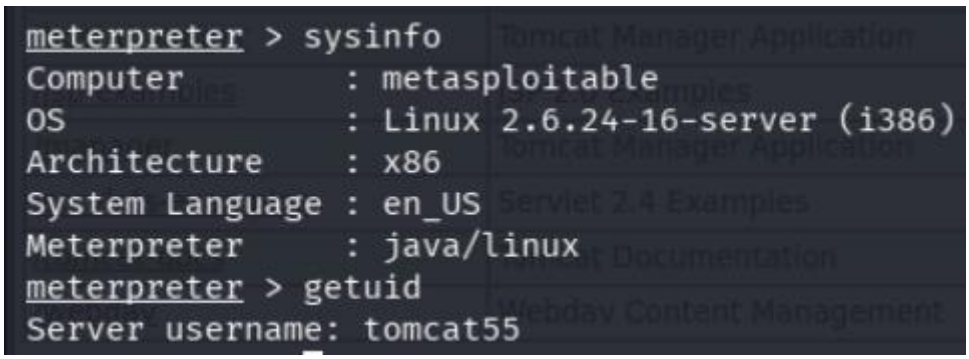


Figure 20: Meterpreter session showing system information and user context

7.5 Impact Analysis

This vulnerability allowed me to:

- Upload malicious files
- Execute arbitrary commands
- Gain remote shell access
- Potentially escalate privileges
- Move laterally within a network

An attacker exploiting this vulnerability could fully compromise the server and access sensitive information.

Severity: Critical (Remote Code Execution)

---

8. Phase 4: Post-Exploitation

8.1 Privilege Validation

After establishing a Meterpreter session, I verified the level of access I obtained. I executed the following commands:

```
getuid
sysinfo
```

These commands allowed me to confirm the user context and system details of the compromised machine. The output demonstrated that I successfully gained remote access and command execution capability on the target system.

8.2 System Enumeration

I initiated a basic system enumeration process to understand the environment further. I accessed the shell using:

`shell`

Then I executed:

`whoami`

This command confirmed the current privilege level within the compromised system.

Figure 21: Switching from Meterpreter to system shell and verifying user access

8.3 Evidence Collection

To simulate forensic awareness and responsible handling of compromised systems, I collected a system file and generated a cryptographic hash.

I executed:

`sha256sum /etc/passwd`

This command generated a SHA256 hash of the system's user file.

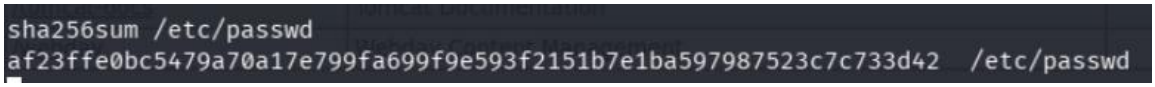


Figure 22: SHA256 hash generated for /etc/passwd file as evidence

8.4 Evidence Documentation

I documented the collected evidence as follows:

Item	Description	Collected By	Date	Hash Value
/etc/passwd	System user file	me	11-02-2026	af23ffe0bc5479a70a17e799fa699f9e593f2151b7e1ba597987523c7c733d42

This process demonstrates proper evidence handling and chain-of-custody awareness during post-exploitation.

8.5 Impact Assessment

Post-exploitation confirmed that an attacker could:

- Execute arbitrary commands
- Access sensitive system files
- Maintain persistence
- Extract confidential data

This phase validated the severity of the vulnerability and demonstrated real-world compromise potential.

---

9. Phase 5 – Capstone: SQL Injection Exploitation (DVWA)

9.1 Target Overview

I conducted web application testing against Damn Vulnerable Web Application (DVWA) hosted locally on:
http://localhost/dvwa

I configured the application security level to **LOW** to simulate a vulnerable production scenario and to validate SQL Injection exploitation.

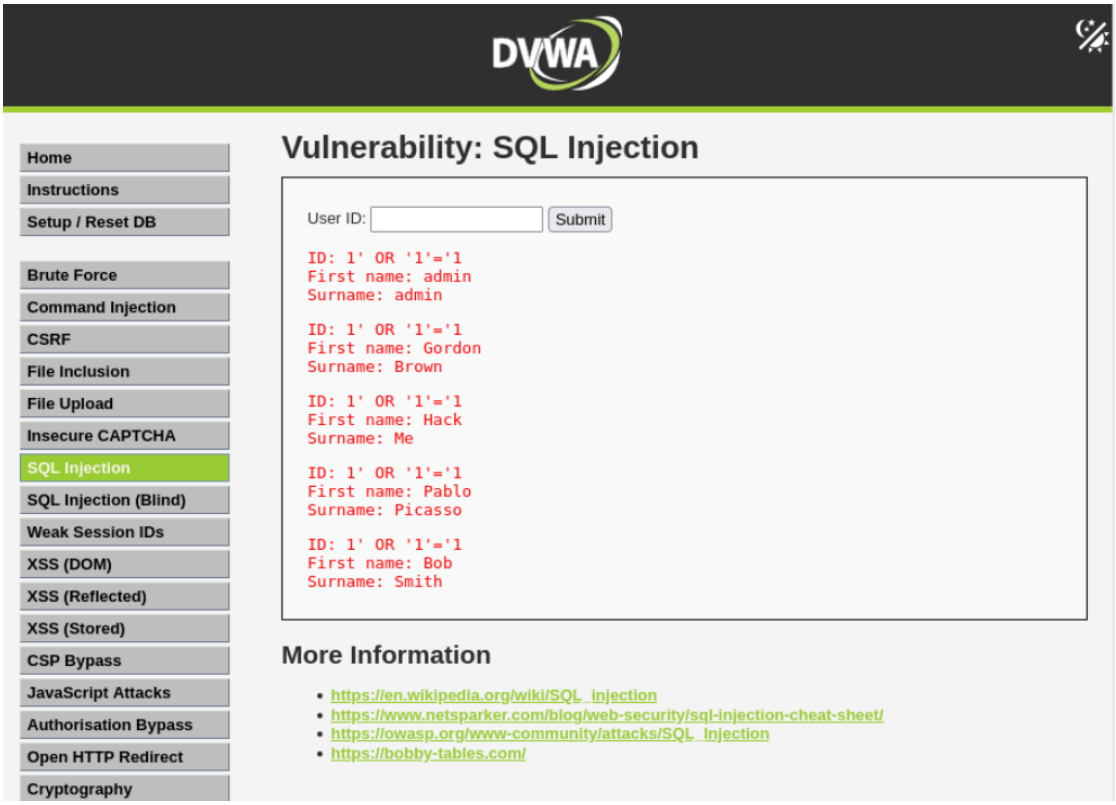


Figure 23: Manual SQL Injection payload confirming vulnerability

9.2 Manual SQL Injection Confirmation

I navigated to:
Vulnerabilities → SQL Injection

I entered the following payload into the User ID field:

```
1' OR '1'='1
```

The application returned multiple user records instead of a single record, which confirmed the presence of SQL Injection.

9.3 Automated Exploitation Using sqlmap

After confirming the vulnerability manually, I automated exploitation using sqlmap.
I executed:

```
sqlmap -u
“http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit”\
--cookie=”security=low; PHPSESSID=<session_id>” \
--dbs
```

sqlmap identified the parameter id as vulnerable to both:

- Time-based blind SQL Injection

- UNION-based SQL Injection

It confirmed that the backend database management system was MySQL (MariaDB fork).

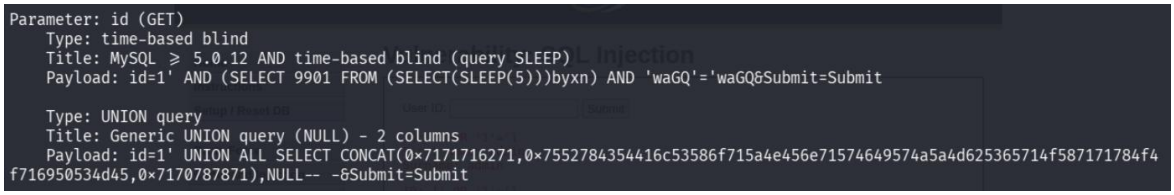


Figure 24: sqlmap confirming SQL Injection vulnerability types

9.4 Database Enumeration

sqlmap successfully enumerated available databases:

- dvwa
- information_schema

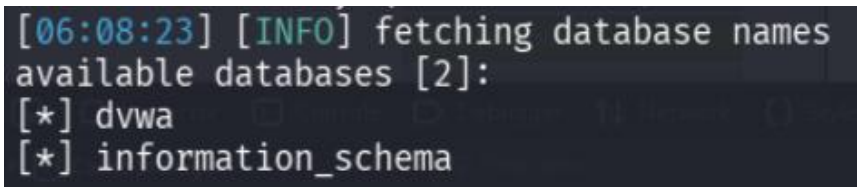


Figure 25: sqlmap enumerating available databases

9.5 Table Enumeration

I then enumerated tables within the dvwa database:

- access_log
- guestbook
- security_log
- users

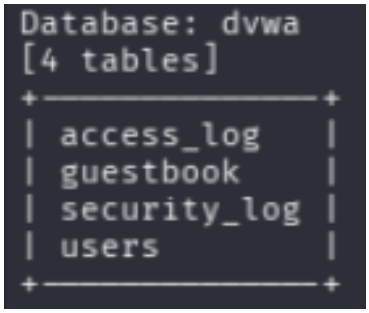


Figure 26: sqlmap listing tables in dvwa database

9.6 Data Extraction

I extracted data from the users table using:

```
Sqlmap -D dvwa -T users –dump
```

sqlmap successfully retrieved:

- Usernames

- Password hashes
- Roles
- Login metadata

It also cracked password hashes using a dictionary-based attack and revealed plaintext credentials such as:

- admin → password
- gordonb → abc123
- pablo → letmein

Figure 27: sqlmap dumping user table including cracked password hashes

9.7 Impact Analysis

This vulnerability allowed me to:

- Extract the entire user database
- Retrieve administrative credentials
- Crack password hashes
- Gain full application-level compromise

An attacker could leverage these credentials to:

- Perform account takeover
- Escalate privileges
- Access sensitive business data
- Pivot to internal systems

Severity: **Critical** (Database compromise and credential exposure)

---

10. Risk Assessment and Severity Classification

10.1 Risk Evaluation Approach

I evaluated each identified vulnerability based on severity, exploitability, and potential business impact. I prioritized vulnerabilities using CVSS scoring principles and practical exploitation results.

I considered the following factors:

- Ease of exploitation
- Authentication requirements
- Level of access gained
- Data sensitivity exposure
- Potential for lateral movement

10.2 Critical Findings

SQL Injection – DVWA:

- Impact: Full database compromise
- Data Exposure: User credentials and administrative accounts
- Exploitability: Remote and unauthenticated
- Result: Successful data extraction and password cracking
- Risk Level: Critical

Apache Tomcat Manager Remote Code Execution:

- Impact: Remote shell access
- Exploitability: Weak default credentials
- Result: Successful Meterpreter session
- Risk Level: Critical

10.3 High and Medium Findings

Outdated Services (Metasploitable2)

- Outdated Apache version
- Outdated Samba service
- Exposed database services
- Weak authentication mechanisms
- Risk Level: High to Medium depending on exploitability.

10.4 Likelihood vs Impact Analysis

I evaluated risk based on both likelihood and impact:

Vulnerability	Likelihood	Impact	Risk Level
SQL Injection	High	High	Critical
Tomcat RCE	High	High	Critical
Outdated Apache	Medium	High	High
Weak FTP Configuration	Medium	Medium	Medium

10.5 Overall Risk Summary

The assessment revealed multiple critical vulnerabilities that allow full system compromise and sensitive data exposure. If deployed in a production environment, these vulnerabilities would pose a severe security threat.

Immediate remediation is required to mitigate business and operational risks.

---

11. Remediation and Recommendation

11.1 Web Application Security Improvements

I recommend implementing the following controls:

- Use prepared statements (parameterized queries)
- Avoid dynamic SQL query construction
- Implement strict input validation
- Apply server-side input sanitization
- Use ORM frameworks to abstract database queries
- Enforce proper error handling without exposing SQL errors

These controls will eliminate SQL Injection risks and protect sensitive database information.

11.2 Authentication and Access Control

To prevent unauthorized access, I recommend:

- Remove default credentials immediately
- Enforce strong password policies
- Implement multi-factor authentication (MFA)
- Disable unused services and administrative interfaces
- Restrict administrative panels to internal networks

11.3 Server Hardening

I recommend the following hardening measures:

- Update outdated software (Apache, Tomcat, Samba)
- Apply regular security patches
- Disable unnecessary services (FTP, Telnet, rlogin)
- Restrict database service exposure to internal networks only
- Implement firewall rules to limit open ports

11.4 Monitoring and Logging

To improve detection capability, I recommend:

- Enable centralized logging
- Monitor unusual login attempts
- Implement intrusion detection systems (IDS)
- Regularly review access logs

11.5 Regular Security Testing

I recommend conducting:

- Periodic vulnerability scanning
- Quarterly penetration testing
- Secure code reviews
- Developer security training

Proactive security testing will significantly reduce long-term risk.

---

12. Conclusion

During this engagement, I successfully conducted a comprehensive Vulnerability Assessment and Penetration Testing exercise in a controlled lab environment. I identified multiple critical vulnerabilities across network services and web applications. I validated these findings through controlled exploitation and confirmed their real-world impact.

I demonstrated remote code execution against Metasploitable2 by exploiting weak Tomcat Manager authentication and successfully established a reverse shell. I also exploited a SQL Injection vulnerability in DVWA, extracted the entire user database, and cracked password hashes to reveal plaintext credentials. These results confirmed full system compromise and data exposure risks.

This assessment highlights the importance of secure configuration, regular patch management, strong authentication mechanisms, and secure coding practices. Even simple misconfigurations or input validation failures can lead to severe security breaches.

By implementing the remediation recommendations provided in this report, an organization can significantly reduce its attack surface and strengthen its overall security posture.

---

13. References

I used the following authoritative sources and documentation to guide this assessment:

1. OWASP. (2024): https://owasp.org
2. PTES. (2023): http://www.pentest-standard.org
3. FIRST. (2024): https://www.first.org/cvss
4. National Institute of Standards and Technology. (2024): https://nvd.nist.gov
5. Kali Linux Documentation. (2024): https://www.kali.org/docs/
6. Metasploit Framework Documentation. (2024): https://docs.metasploit.com
7. sqlmap Documentation. (2024): https://sqlmap.org
8. OpenVAS / Greenbone Vulnerability Manager Documentation. (2024): https://greenbone.net
9. Nmap Reference Guide. (2024): https://nmap.org
10. Damn Vulnerable Web Application (DVWA): https://github.com/digininja/DVWA