(VAPT) Report week -3

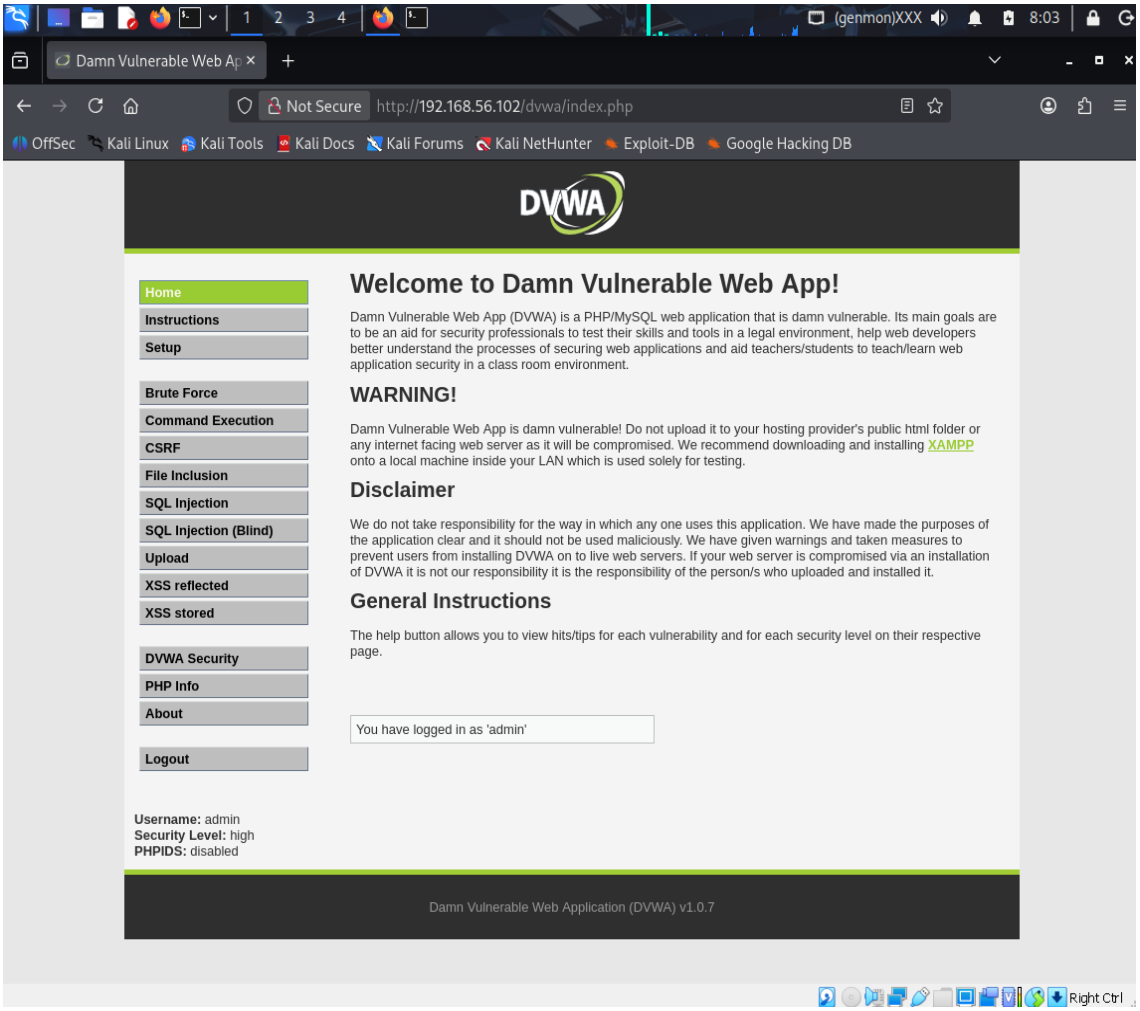
1. Assessment Overview

This assessment involved performing a full Vulnerability Assessment and Penetration Testing (VAPT) exercise against a deliberately vulnerable lab environment consisting of DVWA and Metasploitable. The objective was to identify, exploit, and document security weaknesses across web application and network layers. Multiple high-impact vulnerabilities were discovered, including SQL Injection, Stored Cross-Site Scripting (XSS), weak authentication mechanisms, and remote code execution through vulnerable services. Successful exploitation resulted in unauthorized database access, credential disclosure, persistent client-side attacks, and full root-level system compromise. The findings demonstrate how insecure coding practices and outdated services can be chained together to achieve complete system compromise. Remediation steps are provided to mitigate identified risks and improve the overall security posture.

2. Methodology (PTES-based)

Target :

- DVWA Web Application
- Metasploitable Virtual Machine



Attacker System:

- Kali Linux

Tools Used:

- Burp Suite
- sqlmap
- Metasploit Framework
- Wireshark
- Firefox Developer Tools

```
msf > ifconfig
[*] exec: ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.101  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::9ae1:645b:e805:e209  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d1:f8:5d  txqueuelen 1000  (Ethernet)
        RX packets 4  bytes 1830 (1.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 3730 (3.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 51  bytes 3612 (3.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 51  bytes 3612 (3.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

msf > exit

┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d1:f8:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute eth0
       valid_lft 450sec preferred_lft 450sec
    inet6 fe80::9ae1:645b:e805:e209/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

3. Attack Surface Mapping

Stage 1: Stored Cross-Site Scripting (XSS) – Initial Foothold

The attack began with the identification of a **Stored XSS vulnerability** within DVWA. Unlike reflected XSS, this payload was permanently stored in the backend database and automatically executed whenever a victim accessed the affected page.

- Malicious JavaScript was injected into an input field (e.g., comments or messages).

- The script executed in the context of authenticated users, including administrators.
- The payload was designed to steal **session cookies** using document.cookie.
- Stolen cookies were exfiltrated to the attacker-controlled system.

Impact:

Persistent XSS allowed long-term access to authenticated sessions without needing credentials, effectively bypassing authentication controls.

Stage 2: Session Hijacking – Privilege Escalation

Using the stolen session cookie:

- The attacker impersonated a logged-in user.
- Higher-privileged areas of the application became accessible.
- Security mechanisms such as login forms and CAPTCHA were bypassed entirely.

Impact:

This transformed a client-side vulnerability into **server-side privilege escalation**, granting access equivalent to legitimate users.

Stage 3: SQL Injection – Data Exfiltration

With authenticated access, the attacker exploited **SQL Injection vulnerabilities** in DVWA modules.

- Injectable parameters were identified in URL/query fields.
- SQL payloads were used to:
    - Enumerate database names and tables
    - Dump user credential tables
    - Extract usernames, password hashes, and configuration data

Impact:

Sensitive backend data was compromised, exposing credentials that could be reused across services.

Stage 4: Credential Reuse & Weak Configuration Abuse

Extracted credentials and configuration details revealed:

- Weak or reused passwords
- Services running with default or insecure configurations
- Internal service exposure not intended for public access

These weaknesses allowed lateral movement from the web layer to backend services.

Impact:

The attack transitioned from **application-level compromise to system-level access**.

Stage 5: Exploitation of Outdated Network Services

The compromised environment exposed legacy services including **VSFTPD** and **Samba**, both running vulnerable versions.

- Known exploits were identified using service enumeration.
- Metasploit modules were used to exploit these services.
- Remote Code Execution (RCE) was achieved.
- Privilege escalation led to **root-level access**.

4. Individual Vulnerability Analysis

4.1 SQL Injection (Critical)

Description:

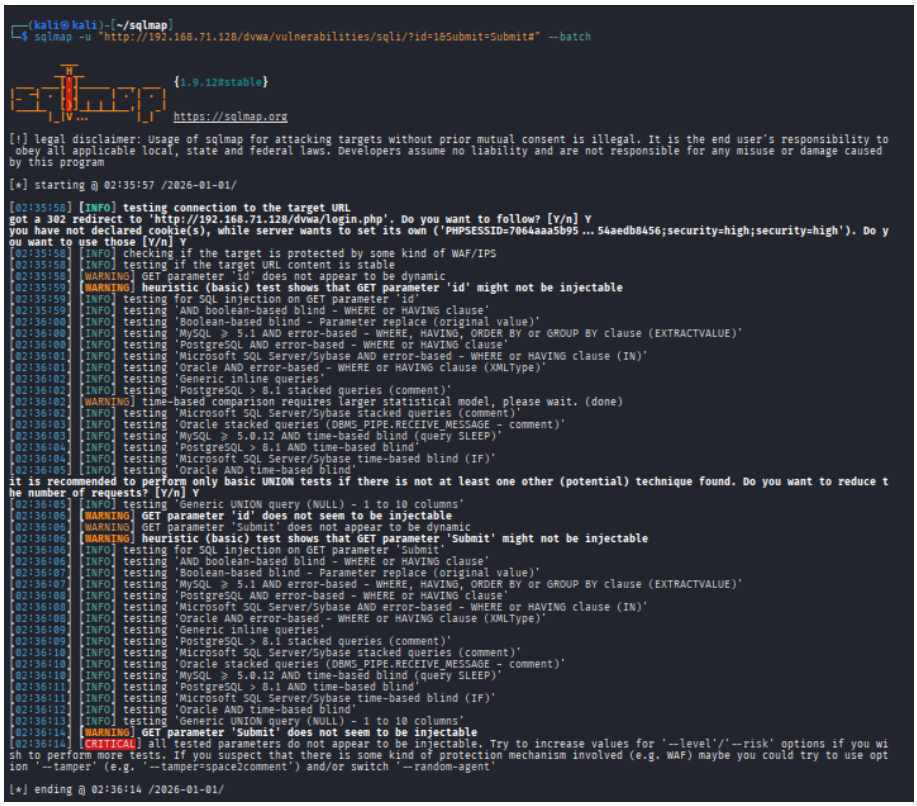
The application failed to properly sanitize user input in SQL queries, allowing attackers to manipulate backend database queries.

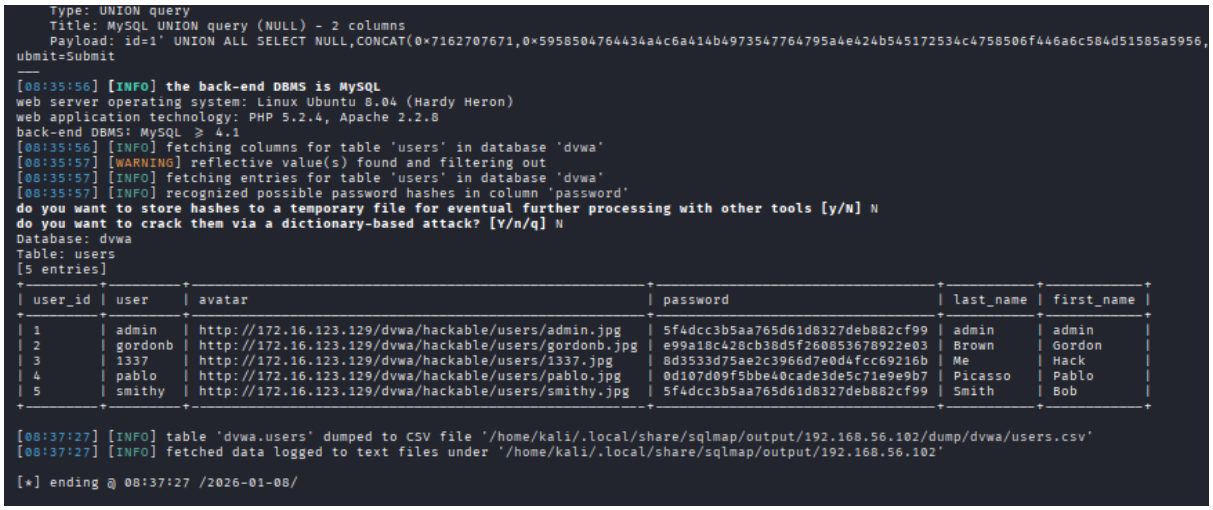
Evidence:

- Manual payload ' OR '1'='1 returned all user records
- sqlmap successfully enumerated tables and dumped user credentials

Impact:

- Unauthorized access to sensitive data
- Credential disclosure





4.2 Stored Cross-Site Scripting (High)

Description:

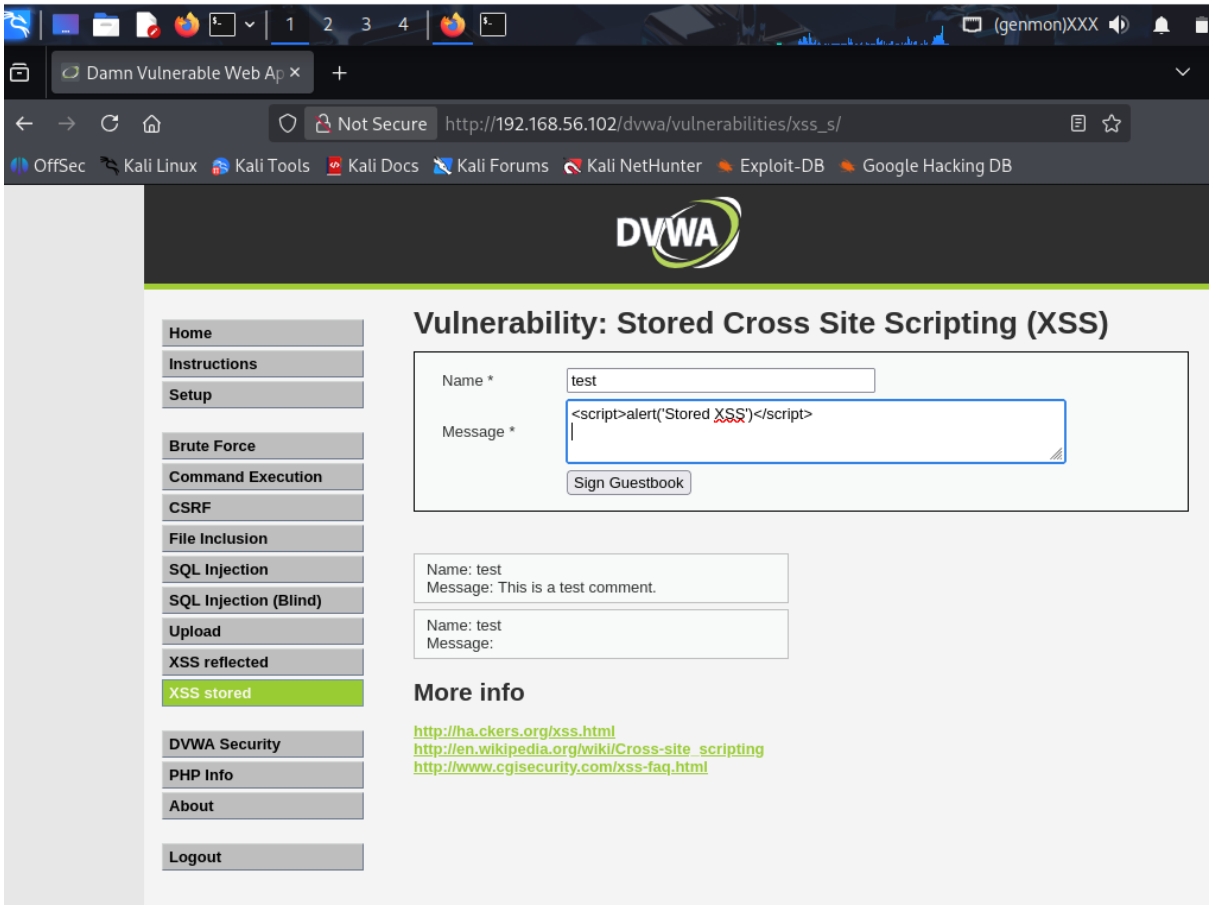
User input was stored and rendered without proper output encoding, allowing persistent JavaScript execution.

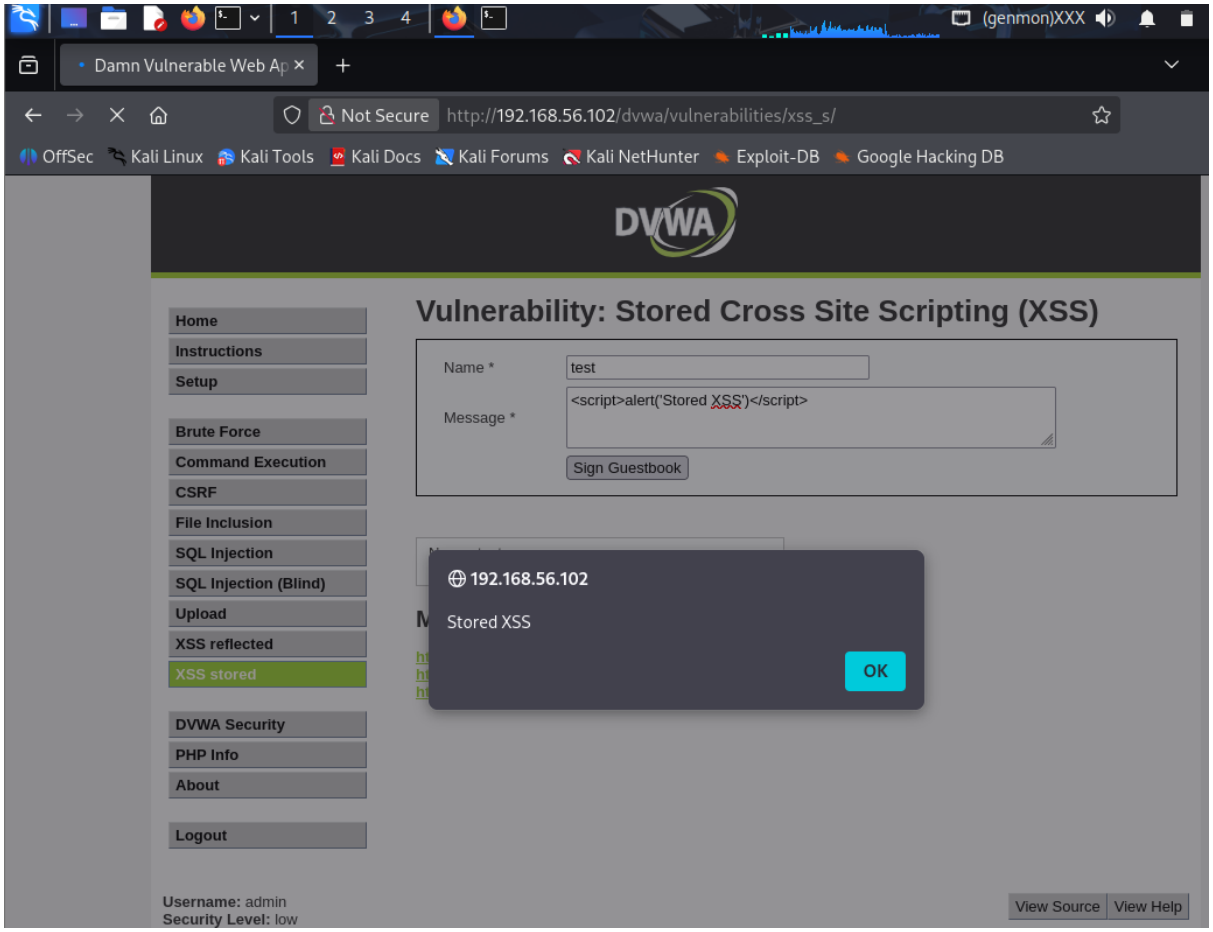
Payload Used:

<script>alert('Stored XSS')</script>

Impact:

- Session hijacking
- Persistent client-side exploitation


Damn Vulnerable Web Ap
Not Secure
http://192.168.56.102/dvwa/vulnerabilities/xss_s/
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
test
Message *
<script>alert('Stored XSS')</script>
Sign Guestbook
Name: test
Message: This is a test comment.
Name: test
Message:
More info
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html



4.3 API Authorization & Data Exposure Issues (High)

(Modern Web / API Layer)

4.3.1 Insecure Direct Object Reference (IDOR) via APIs

Description:

API endpoints exposed object identifiers without validating user authorization, allowing attackers to access or modify other users' data.

Evidence:

- Manipulation of object IDs in API requests returned unauthorized user data
- No ownership validation performed server-side

Impact:

- Unauthorized data access
- Horizontal privilege escalation

4.3.2 Authentication & Authorization Bypass in APIs

Description:

Certain API endpoints failed to enforce authentication and authorization checks consistently.

Evidence:

- API endpoints responded with valid data despite missing or invalid authentication tokens
- Authorization enforced only at the client-side

Impact:

- Unauthorized access to protected resources
- Potential account takeover

4.3.3 Excessive Data Exposure via APIs

Description:

API responses returned more data than necessary for intended functionality.

Evidence:

- Full user objects returned instead of minimal required fields
- Exposure of internal IDs, email addresses, and role information

Impact:

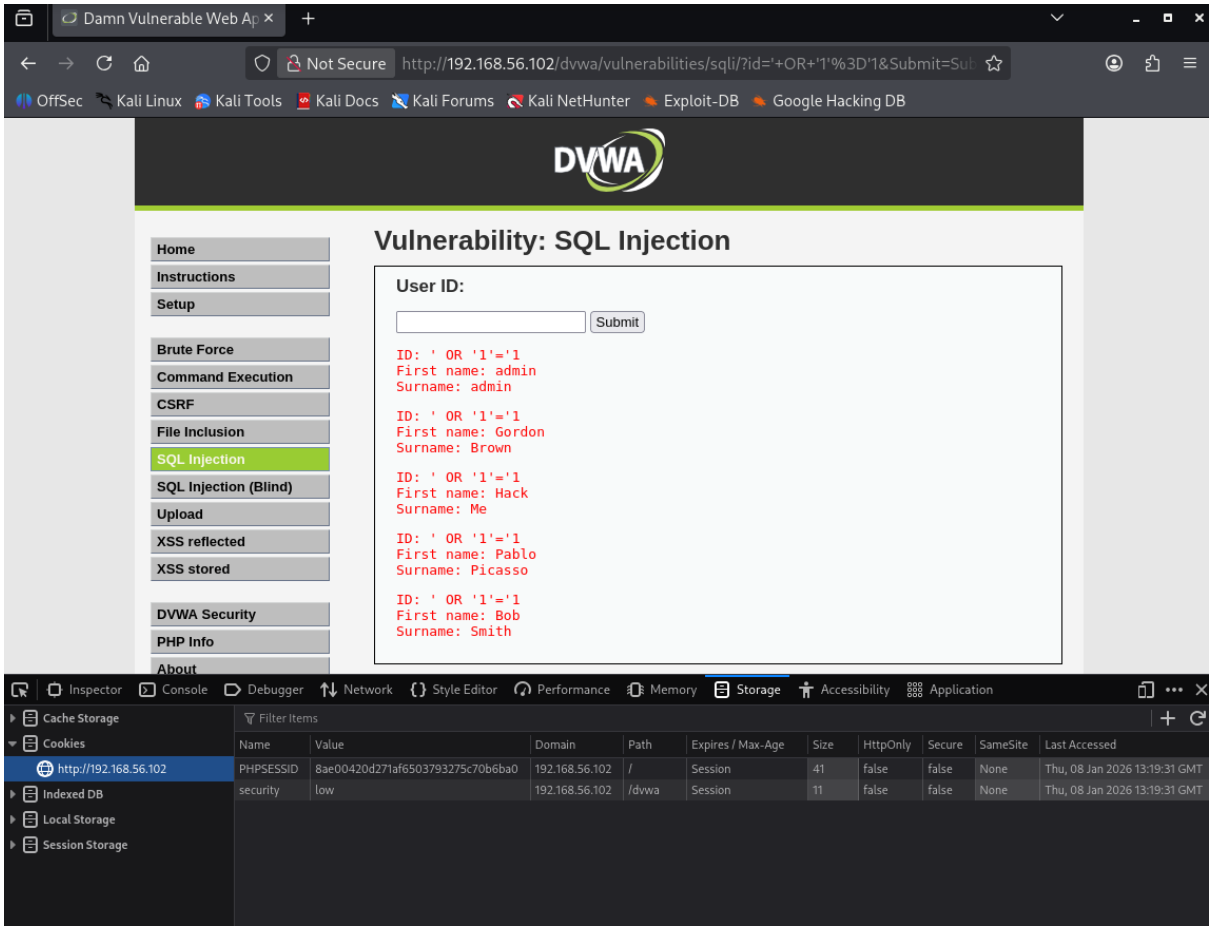
- Sensitive information disclosure
- Increased attack surface for chained attacks

5. Chained Exploitation Scenario

Stage	Attack Vector	Tool Used	Impact	Success Status
001	SQL Injection	sqlmap	192.168.56.102	Success
002	Stored XSS	Manual	192.168.56.102	Success
003	VSFTPD Backdoor RCE	Metasploit	192.168.56.102	Root Access

6. Post-Exploitation Observations

During exploitation, sqlmap was customized by explicitly providing session cookies and increasing the detection depth to ensure reliable exploitation of authenticated SQL injection points. This customization allowed accurate database enumeration while maintaining session persistence, demonstrating how minor configuration changes can significantly improve exploitation success in real-world environments.



7. Risk Assessment & Business Impact

7.1 Network Traffic Capture

Network traffic was captured during the exploitation phase using **Wireshark** on the attacker machine. The capture focused specifically on **HTTP traffic** to validate interaction between the attacker system and the DVWA web server at the network level.

A display filter (http) was applied to isolate relevant application-layer traffic. The captured packets clearly demonstrate a successful HTTP request–response exchange between the attacker and target systems during DVWA access.

Capture Details:

- Tool Used: Wireshark
- Interface: eth0
- Display Filter: http

- Attacker IP: 192.168.56.101
- Target IP: 192.168.56.102
- File Name: **http_vaptweek3.pcapng**

7.2 Observed Packets

The following packets were observed in the capture and confirm active communication during testing:

Source IP	Destination IP	Protocol	Information
192.168.56.101	192.168.56.102	HTTP	GET /dvwa/ HTTP/1.1
192.168.56.102	192.168.56.101	HTTP	HTTP/1.1 200 OK (text/html)

These packets confirm that the attacker system successfully accessed the DVWA application and received a valid server response, validating web application interaction during the VAPT exercise.

7.3 Evidence Integrity Verification

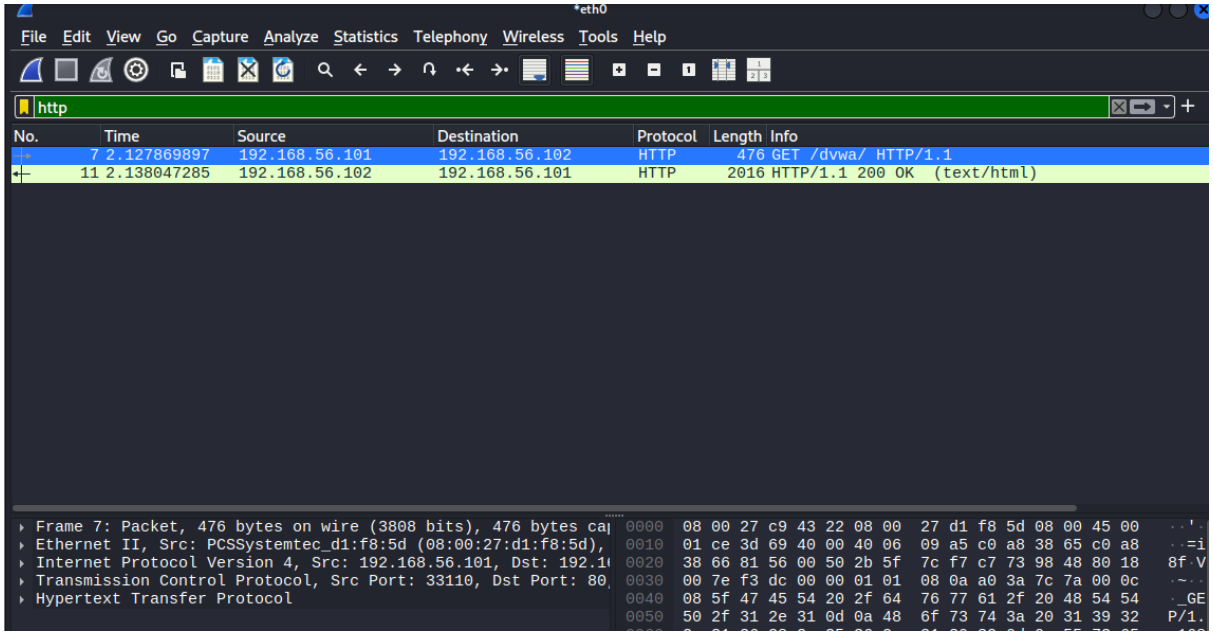
To ensure evidence integrity and maintain proper chain of custody, a cryptographic hash was generated for the captured network traffic file using the SHA-256 hashing algorithm.

Hash Command Used:

sha256sum http_vaptweek3.pcapng

Evidence Hash Record:

Evidence Item: Network Traffic Capture
File Name: http_vaptweek3.pcapng
Hash Algorithm: SHA-256
Hash Value: 313b04ce74fcba29f0bc550e7acd323f55e4b4668eb7372a0c0627a517c5e010



7.4 Evidence Log

Item ID	Evidence Type	Tool	Description	Integrity
E001	Network Capture	Wireshark	HTTP traffic during DVWA exploitation	SHA-256 Verified

7.5 Evidence Collection Summary

Network traffic was successfully captured during the exploitation phase using Wireshark. The observed HTTP packets confirm request–response communication between the attacker and DVWA server. Evidence integrity was preserved through SHA-256 hashing, ensuring authenticity and proper chain-of-custody in accordance with VAPT reporting standards.

8. Remediation Recommendations

- Use prepared statements and parameterized queries to prevent SQL Injection
- Implement proper input validation and output encoding to mitigate XSS
- Disable or upgrade outdated services such as VSFTPD and Samba
- Enforce strong authentication and secure session management
- Apply regular patching and vulnerability scanning
- Enable logging, monitoring, and intrusion detection systems

9. Remediation Strategy

The assessment revealed serious security weaknesses that allow attackers to access sensitive data and take full control of systems. Simple vulnerabilities were chained together to compromise servers completely. These risks could lead to data breaches, downtime, and reputational damage. Immediate remediation and secure development practices are required to reduce exposure.

10. Lessons Learned

This VAPT exercise successfully demonstrated the complete attack lifecycle, from vulnerability discovery to exploitation and post-exploitation. Multiple critical vulnerabilities were identified and exploited, highlighting the importance of defense-in-depth and secure system design. Addressing the findings will significantly improve the security posture of the environment.