TASK 1: Web Application Security Testing Report

Name: Deepta Chakravarty

Track Code: FUTURE_CS_01

Domain: Cyber Security

Internship Provider: Future Interns

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

Submission Type: Task Report

AIM

To conduct ethical penetration testing on a deliberately vulnerable web application (DVWA) and identify common vulnerabilities including **SQL Injection**, **Reflected Cross-Site Scripting (XSS)**, and **Stored XSS** using professional security testing tools and techniques.

TOOLS USED

- **DVWA (Damn Vulnerable Web Application):** A testing platform designed for practicing web application security.
- **Burp Suite Community Edition:** A web vulnerability scanner and intercepting proxy.
- **Google Chrome Browser:** For interacting with the DVWA frontend.
- **Kali Linux:** Penetration testing OS used to host and run DVWA locally.

VULNERABILITIES TESTED

1. SQL Injection

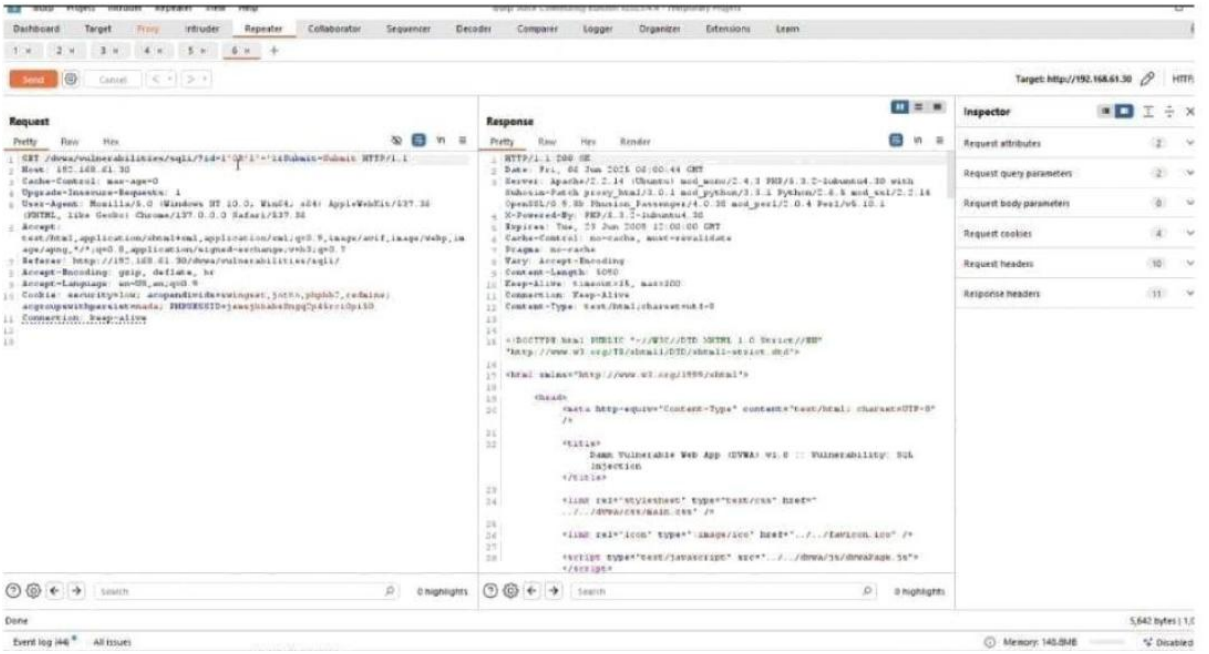
- **Test Input:** ' OR '1'='1
- **Target Module:** DVWA → SQL Injection

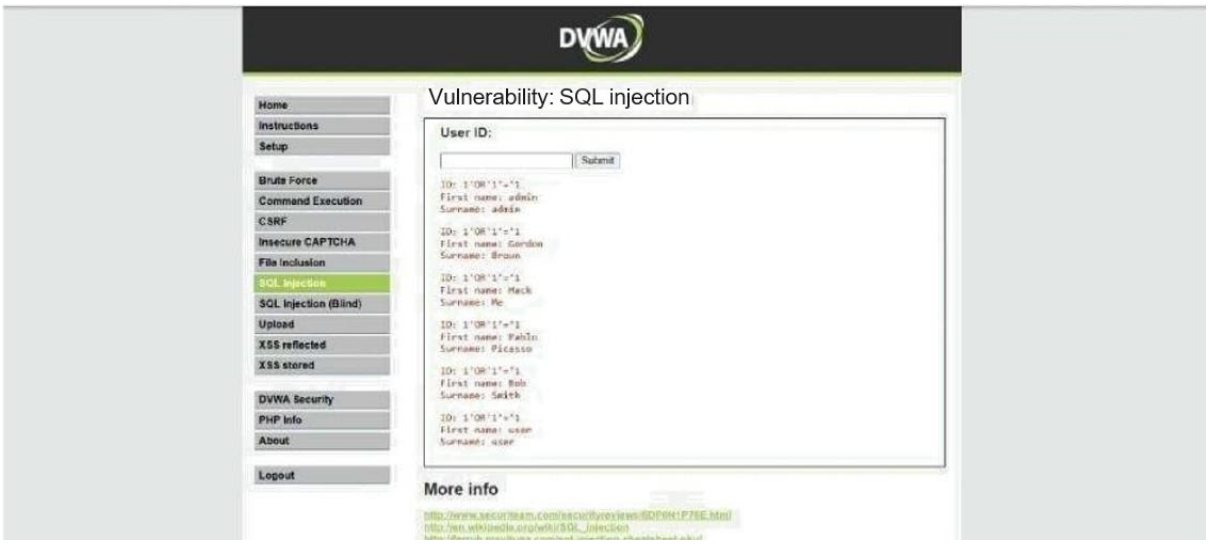
Testing Methodology:

- Intercepted the request using Burp Suite.
- Modified the id parameter in the request with the above SQL payload.
- The server responded with unintended user information, bypassing standard authentication checks.

Result:

- The application is vulnerable to **Classic SQL Injection**.
- User data was extracted directly from the database without authorization.





2. Reflected Cross-Site Scripting (XSS)

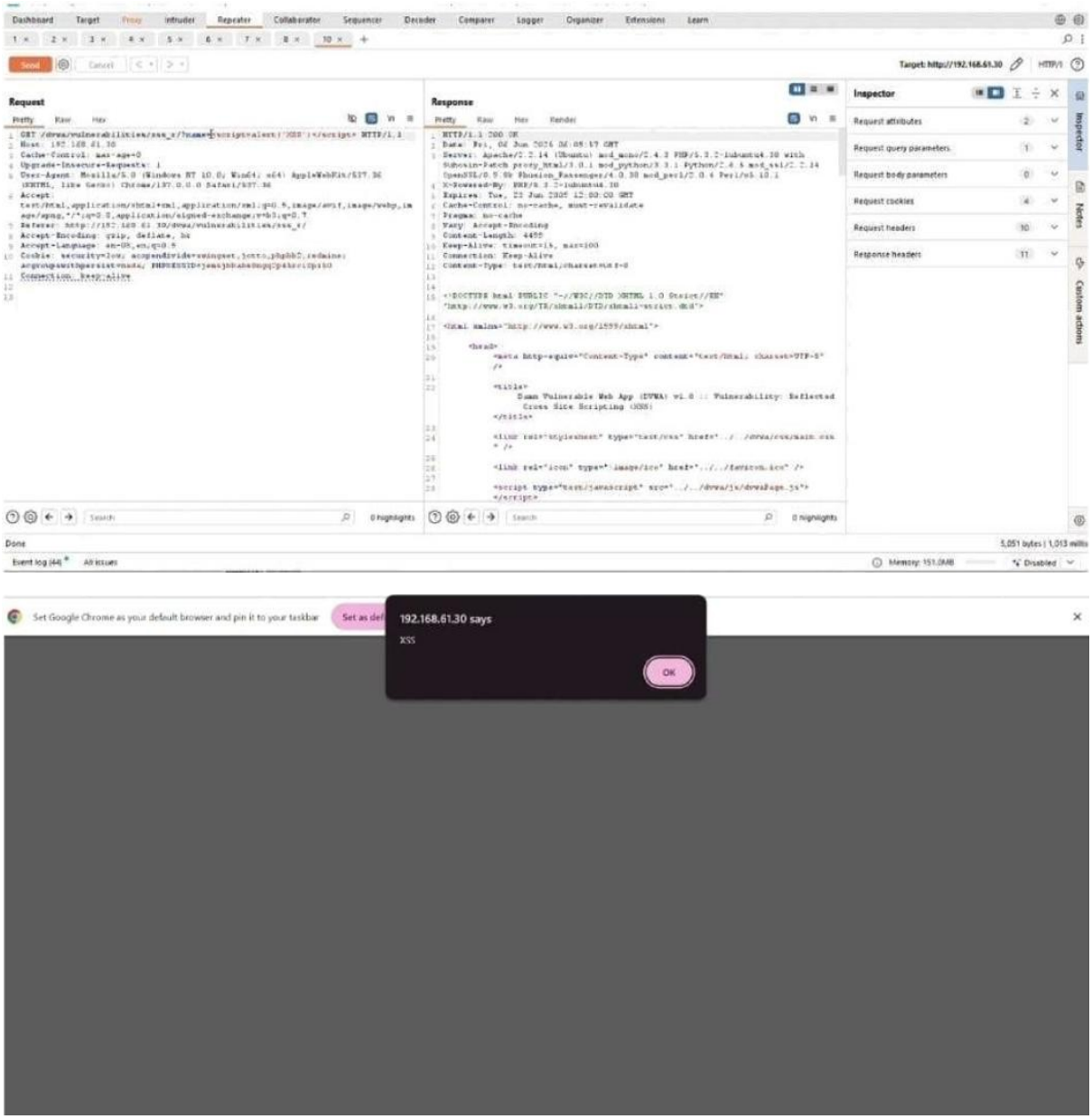
- **Test Input:** <script>alert("XSS")</script>
- **Target Module:** DVWA → XSS (Reflected)

Testing Methodology:

- Injected the JavaScript payload into a search/query input field.
- The payload was reflected in the HTTP response and executed in the browser.
- A JavaScript alert box popped up, confirming execution.

Result:

- The input was not sanitized or encoded before rendering.
- Reflected XSS vulnerability successfully demonstrated.



3. Stored Cross-Site Scripting (XSS)

- **Test Input:** <script>alert("Stored XSS")</script>
- **Target Module:** DVWA → XSS (Stored)

Testing Methodology:

- Inserted the payload into the comment/guestbook section.
- Payload was stored in the backend and executed automatically upon page reload.

Result:

- Persistent XSS vulnerability confirmed.
- Malicious script is executed on every page load, affecting all users.

FINDINGS SUMMARY

Vulnerability	Description	Result
SQL Injection	Input fields are not properly sanitized; payload bypassed logic and retrieved database content.	**Critical**
Reflected XSS	Script reflected directly in response and executed in the browser.	**High**
Stored XSS	Script stored in the database and persisted across sessions.	**Very High**

SECURITY LEVEL

All tests were conducted with DVWA's **security level set to "Low"**, intentionally exposing the vulnerabilities for educational purposes. This setting allowed exploitation without any security filters or validation mechanisms.

OVERALL OBSERVATIONS

- DVWA effectively demonstrates real-world security flaws.
- Vulnerabilities stemmed from **improper input validation**, **lack of output encoding**, and **absence of secure query practices**.
- Such issues can be leveraged by attackers to:
    - Extract sensitive information.
    - Execute malicious code in users' browsers.
    - Bypass authentication mechanisms.

RECOMMENDATIONS

- Use **parameterized queries** (e.g., PreparedStatement in SQL) to mitigate SQL Injection.
- Implement **Content Security Policy (CSP)** headers to restrict script execution.
- Sanitize and validate all user inputs both client-side and server-side.
- Encode output before rendering in HTML.

- Use **Web Application Firewalls (WAFs)** to detect and block injection attempts.
- Regularly update and patch application components and libraries.

LEARNING OUTCOMES

This task provided hands-on experience in:

- Detecting and exploiting **SQL Injection** and **XSS** vulnerabilities.
- Using **Burp Suite** for intercepting, modifying, and analyzing HTTP requests.
- Understanding the importance of **input validation**, **output encoding**, and **secure coding practices**.
- Practicing ethical hacking techniques in a controlled and legal environment.

CONCLUSION

- This exercise demonstrated the ease with which attackers can exploit insecure applications.
- Successfully bypassed logic, retrieved sensitive data, and injected persistent malicious scripts.
- Reinforced the need for **secure development lifecycles**, **regular security audits**, and **developer education**.
- Improved my practical skills in **vulnerability testing**, **web application analysis**, and **security remediation techniques**.