TITLE : WEB APPLICATION SECURITY TESTING

Intern Name : Pulugu Sirivarshini

Tools Used:

DVWA(Damn Vulnerable Web Application), XAMPP, Burp Suite, Chrome Dev Tools.

Introduction

This report documents the results of Task 1: Web Application Security Testing as part of the Cyber Security Internship at Future Interns. The goal was to identify and exploit common web vulnerabilities using DVWA (Damn Vulnerable Web Application) in a controlled local environment.

The testing focused on:

- SQL Injection
- Reflected XSS
- Brute Force Authentication
- Session Management Flaws

DVWA was configured with Low Security, and tools such as Burp Suite, XAMPP, and browser DevTools were used to analyse and validate vulnerabilities.

Each vulnerability is presented with payloads, technical findings, and recommended mitigations.

Environment Setup

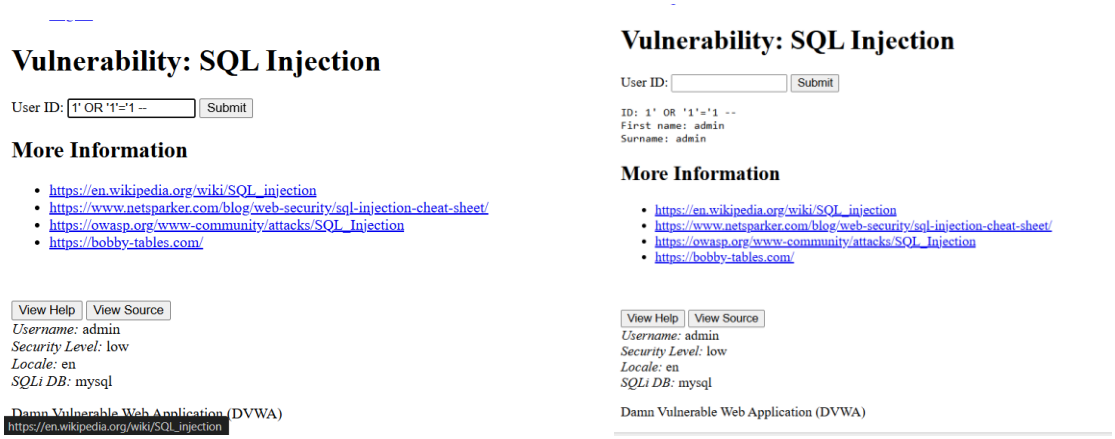
The following tools and configurations were used to conduct the tests:

- **DVWA**: Installed locally via XAMPP with Low Security level enabled
- **XAMPP**: Used to host the web server (Apache) and database (MySQL)
- **Burp Suite**: Used for intercepting and modifying HTTP requests
- **Browser Developer Tools**: Used to inspect cookies and session behaviour

All tests were performed locally on http://localhost/dvwa.

SQL Injection

- **Payload Used:** 1' OR '1'='1 --
- **Result:** Admin user data was retrieved without valid credentials, confirming a successful SQL injection on the user_id parameter.

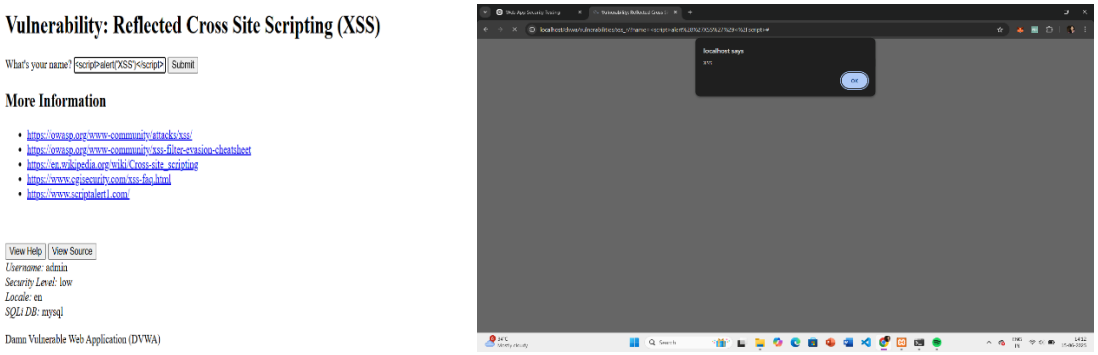


screenshot showing the SQLi result

- **Impact:** Unauthorized data access and potential full database compromise
- **Mitigation:** Use prepared statements or parameterized queries to prevent injection.

Reflected XSS

- **Payload Used:** <script>alert('XSS')</script>
- **Result:** A JavaScript alert was triggered in the browser, confirming that the input was executed as code.



Insert screenshot showing alert popup

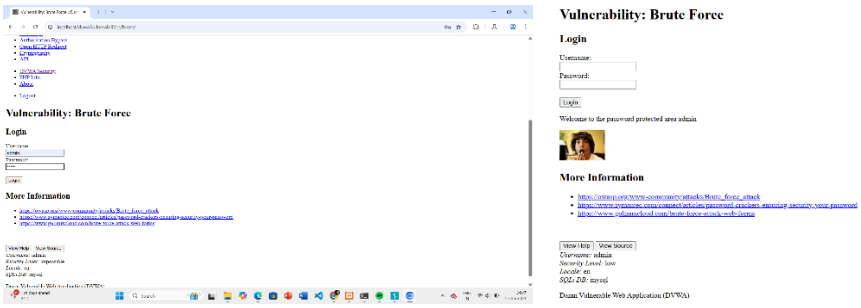
- **Impact:** Could allow session theft, defacement, or phishing attacks
- **Mitigation:** Sanitize user input using functions like htmlspecialchars() to neutralize HTML/JS code.

Brute Force Authentication

Test Credentials:

- Username: admin
- Password: password

- **Result:** Login was successful without any rate limiting, CAPTCHA, or lockout mechanism.
- **Impact:** An attacker can repeatedly guess passwords and gain access.



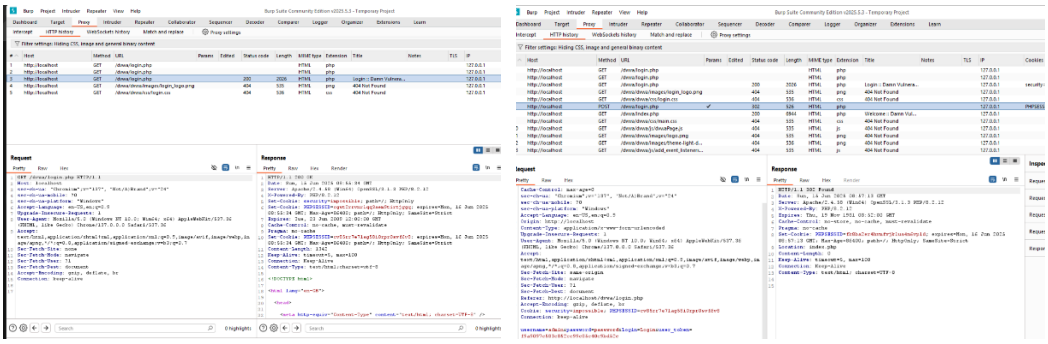
screenshot of successful brute-force login

- **Mitigation:** Implement account lockout, rate limiting, and CAPTCHA after multiple failed login attempts.

Session Fixation

- **Test Performed:** Observed PHPSESSID before and after login using Burp Suite.
- **Result:** Session ID **changed after login**, confirming that DVWA regenerates sessions properly.

Phase	PHPSESSID
Before Login	cv85rr7e71ag58i0rpr0svf6v8
After Login	fk8ba2er4hvmfvjklua4n6vpld

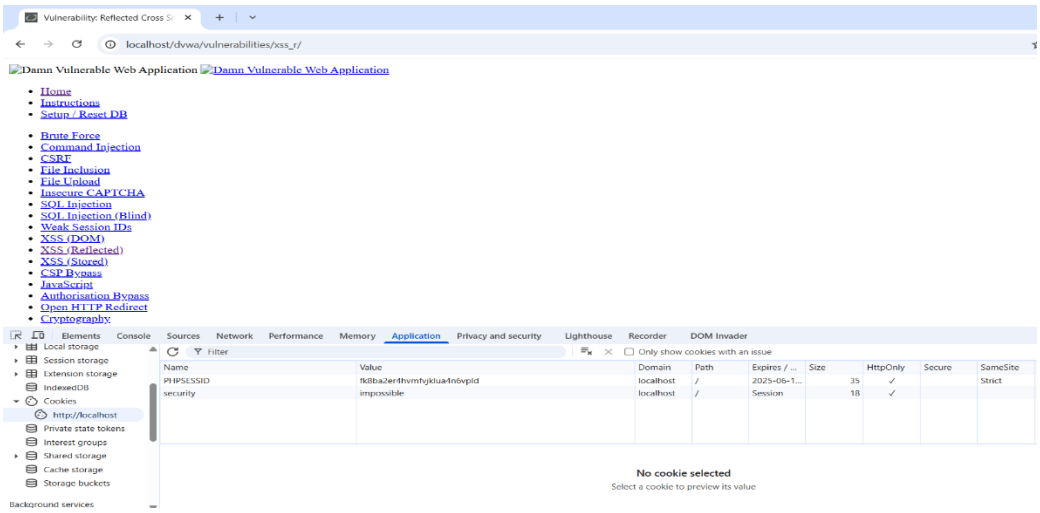


screenshots from Burp HTTP history

- **Conclusion:** DVWA is **not vulnerable** to session fixation.
- **Recommendation:** Continue regenerating session IDs after login for secure session handling.

Weak Session Cookie Flags

- **Cookie Analyzed:** PHPSESSID
- **Observation:**
    - HttpOnly: Enabled
    - Secure: Not Enabled



screenshot from browser DevTools > Application > Cookies

- **Impact:** Without the Secure flag, cookies can be transmitted over unencrypted HTTP, risking session hijacking.
- **Mitigation:** Enable Secure flag and serve the application over HTTPS:

```
ini_set('session.cookie_secure', '1');
```

Summary

The web application security testing task was successfully completed using DVWA configured at a low security level. Key vulnerabilities such as SQL Injection and Reflected Cross-Site Scripting (XSS) were identified and exploited using simple payloads, demonstrating the importance of proper input validation and output encoding. The Brute Force module allowed unrestricted login attempts without rate-limiting or CAPTCHA, indicating weak authentication controls.

Session management was also tested: DVWA was found to regenerate session IDs after login, which means it is not vulnerable to session fixation. However, the session cookies lacked the Secure flag, which could expose session data over unsecured connections.

Each issue discovered was documented with evidence and followed by clear mitigation recommendations based on OWASP best practices. This task provided practical experience in identifying, exploiting, and reporting web application vulnerabilities, reinforcing core concepts of ethical hacking and secure coding.