Web Application Security Testing Report

Intern Name: K. Baby Tejaswi

Project Title: Web Application Security Testing
Tool Used: Damn Vulnerable Web Application (DVWA)
Track id:FUTURE_CS_01

1. Introduction

- **Objective:** Assess DVWA for vulnerabilities, document exploitation methods, and recommend mitigations.
- **Scope:** Focused on Brute Force, SQL Injection (normal & blind), and XSS (DOM-based, Reflected, Stored).
- **Tools Used:** Burp Suite, SQL Map, Foxy Proxy, OWASP ZAP, Kali Linux VM.

2. Methodology

1. **Environment Setup**: DVWA deployed on Kali Linux VM, database configured, and application accessed locally.
2. **Attack Simulation**: Conducted targeted attacks for Brute Force, SQL Injection, and XSS.
3. **Vulnerability Verification**: Confirmed vulnerabilities by analyzing responses and system behavior.
4. **Mapping & Mitigation**: Correlated findings with OWASP Top 10 and proposed remediation strategies.

3. Key Findings

3.1 Brute Force Attack

- **Observation:** Weak authentication allowed automated credential guessing via Burp Suite Intruder.
- **Impact:** Unauthorized access to accounts.

3.2 SQL Injection

- **Normal SQL Injection:** Payload ' OR '1'='1 successfully bypassed authentication.
- **Blind SQL Injection:** SQLMap used to enumerate databases, extract users table, and crack hashed credentials.
- **Impact:** Full database compromise possible.

3.3 Cross-Site Scripting (XSS)

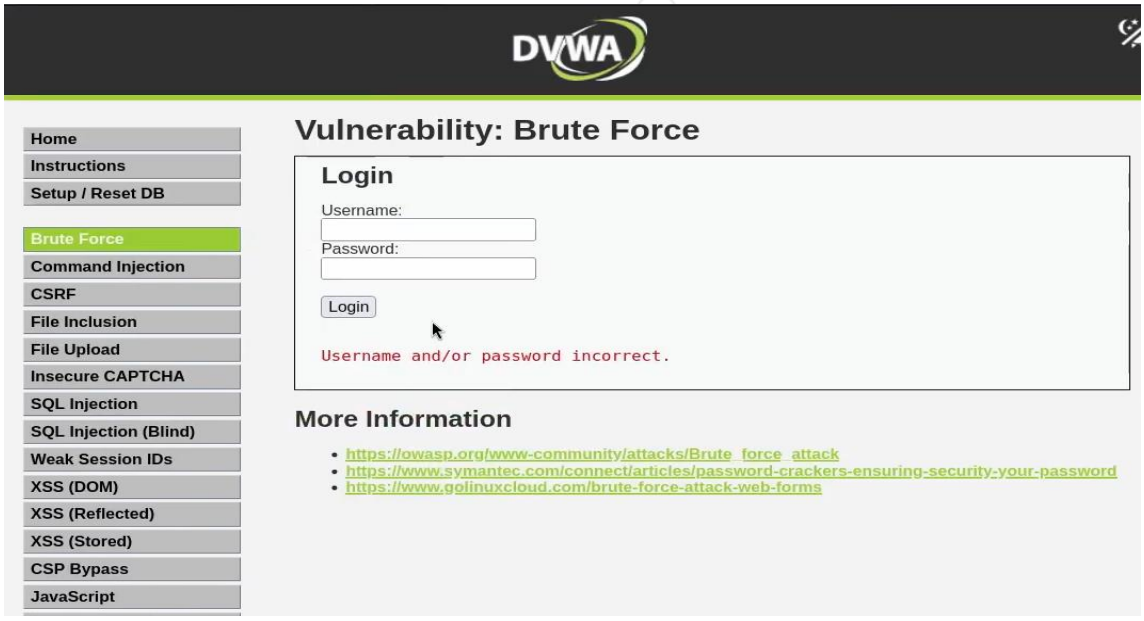
- **DOM-based XSS:** Unsanitized DOM manipulation executed arbitrary JavaScript (alert(document.cookie)).

- **Reflected XSS:** Application echoed unvalidated input, enabling client-side code execution.
- **Stored XSS:** Malicious scripts persisted in guestbook entries and executed in other users' browsers.
- **Impact:** Session hijacking, data theft, and user impersonation.

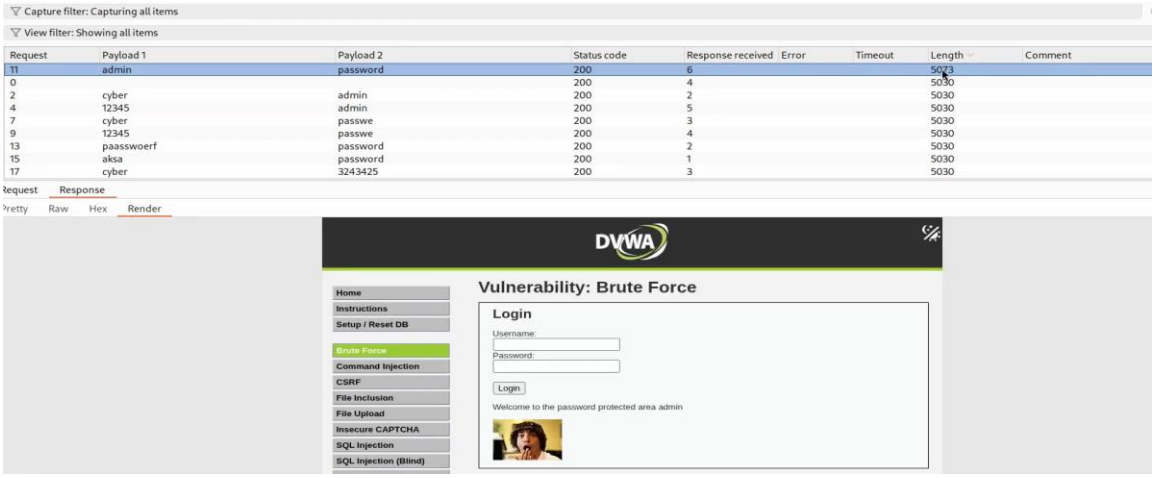
---

4. OWASP Top 10 Mapping

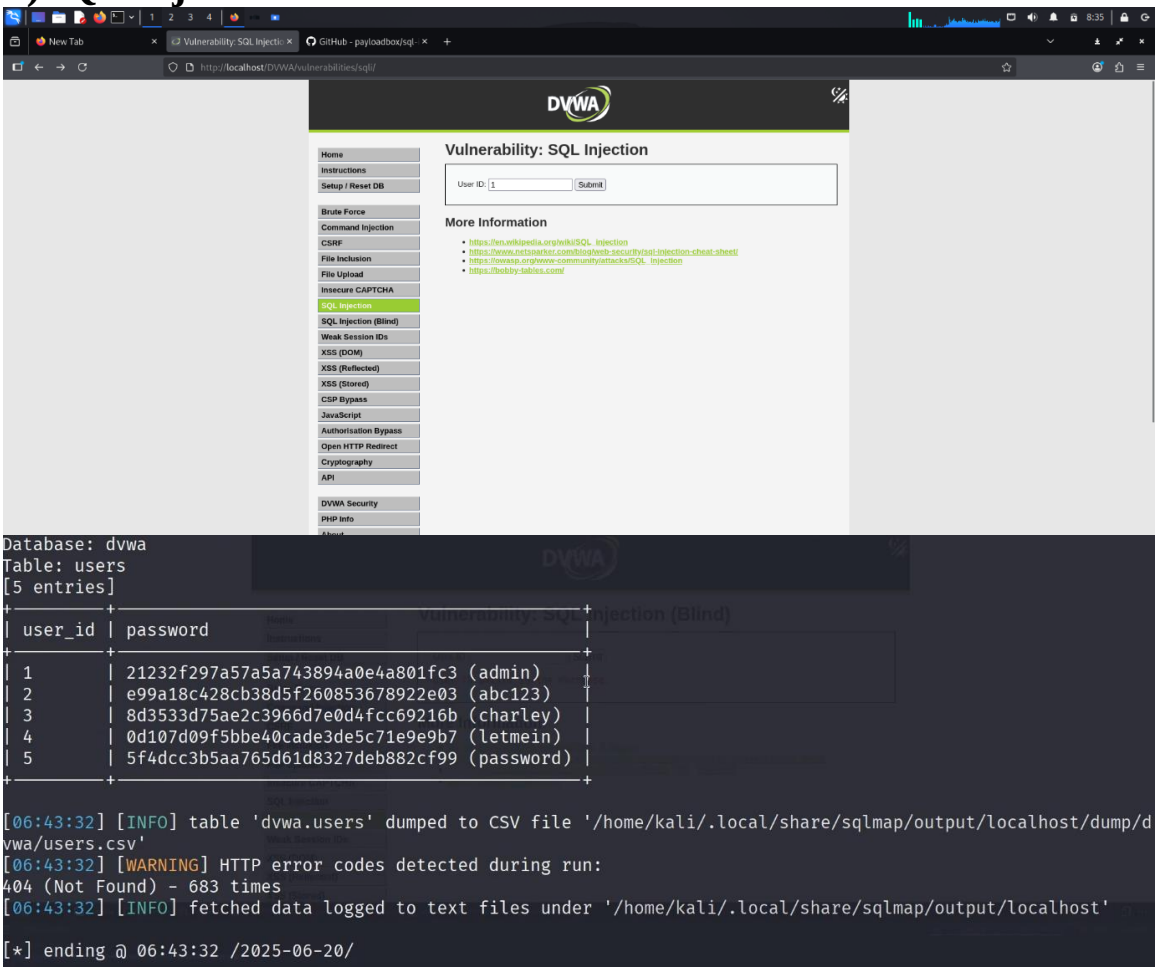
OWASP Category	Finding in DVWA
A1 – Injection	SQL Injection vulnerabilities detected.
A2 – Broken Authentication	Brute force attacks possible without restrictions.
A5 – Broken Access Control	Functions accessible without proper checks.
A6 – Security Misconfiguration	Default insecure configurations observed.
A7 – Cross-Site Scripting	DOM, Reflected, and Stored XSS confirmed.
A9 – Components with Known Vulns	DVWA uses intentionally outdated components.
A10 – Insufficient Logging/Monitoring	No alerts or logs captured during attacks.



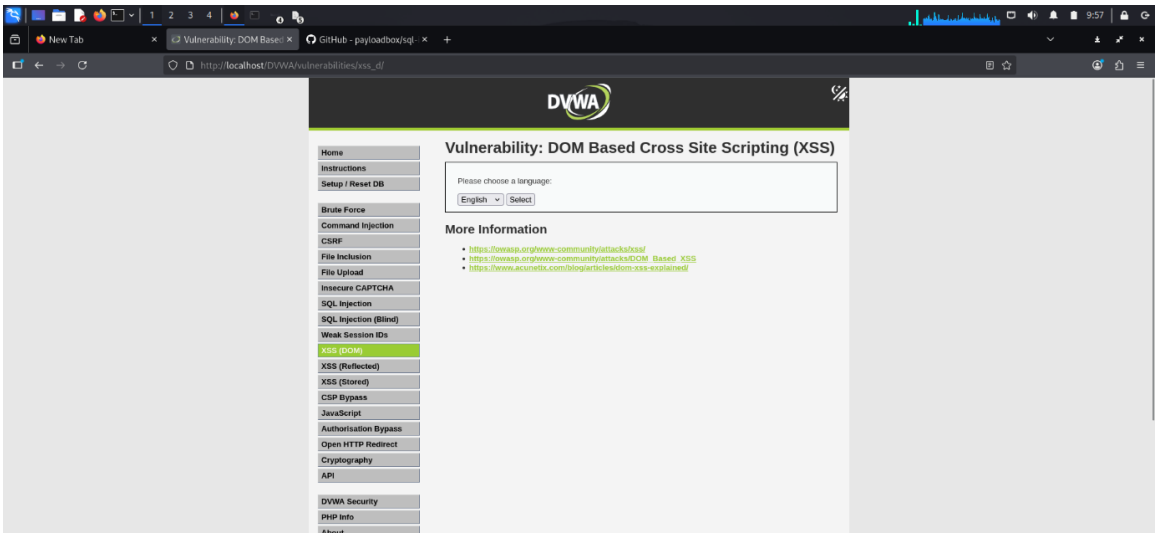
a)Brute force



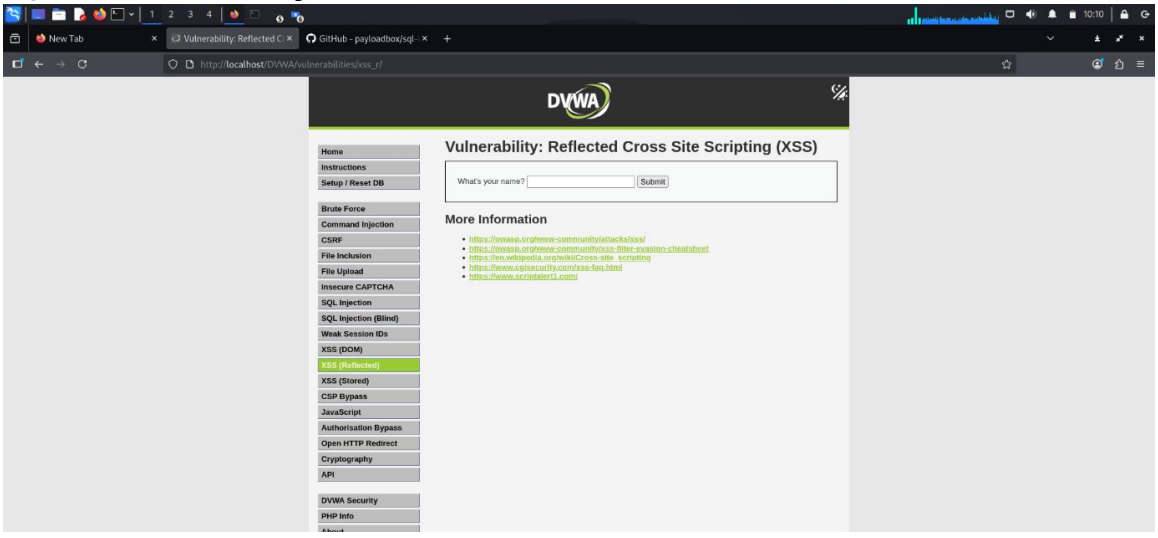
b)SQL injection

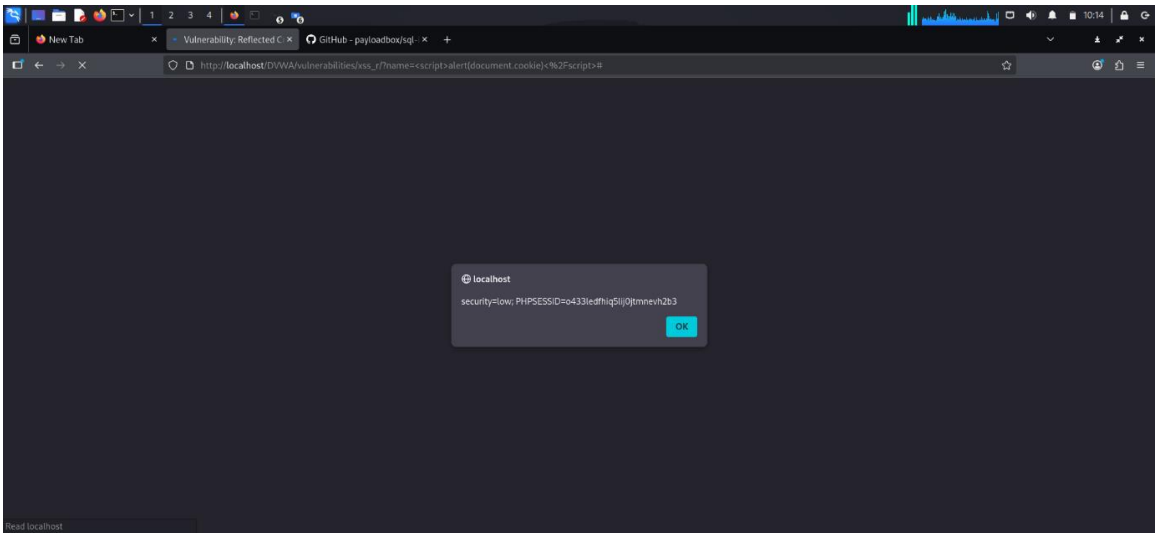


a) Vulnerability: DOM Based XSS

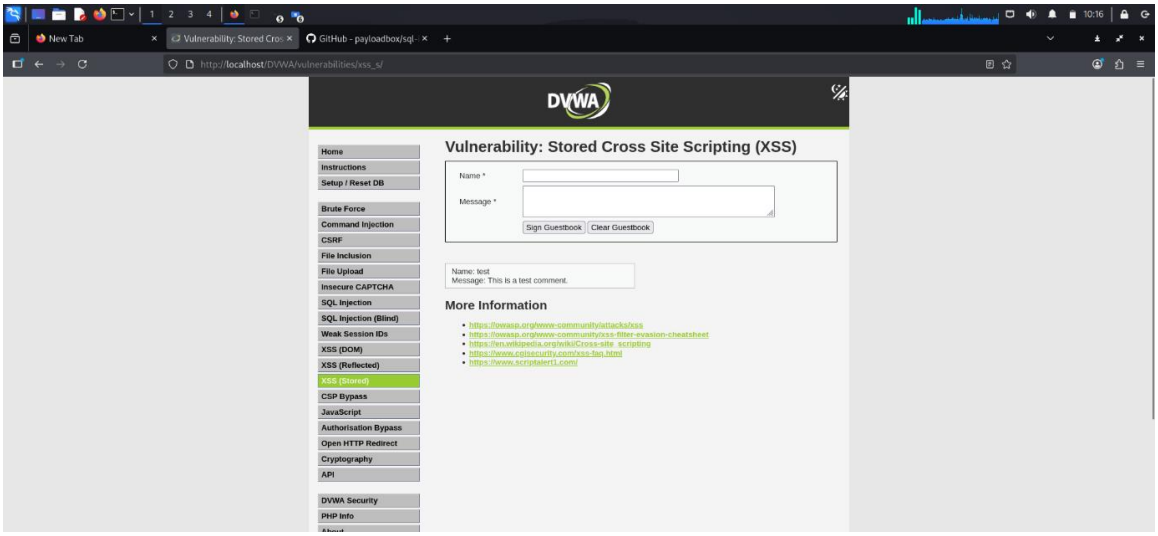


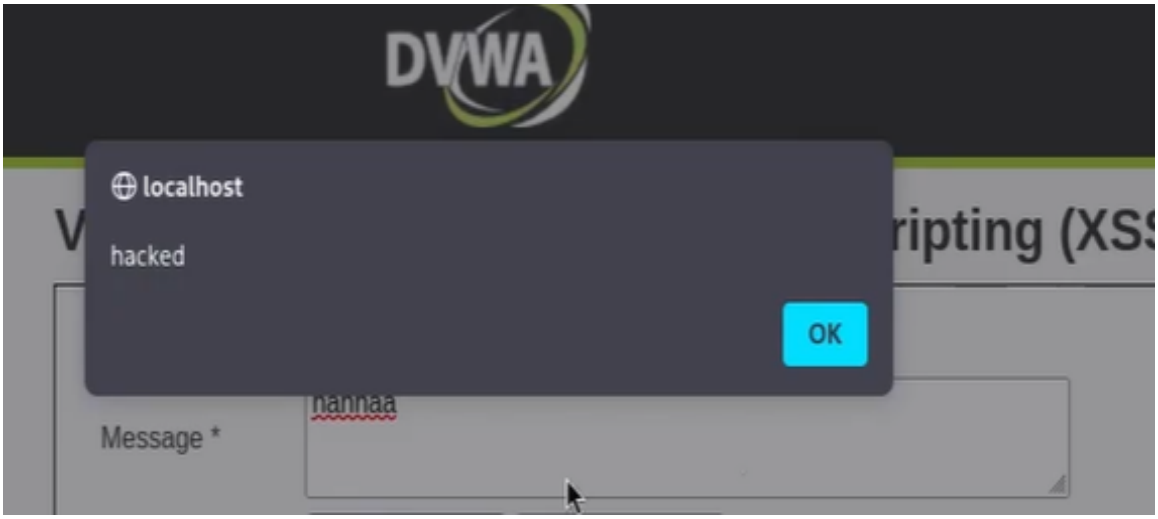
b) Vulnerability: Reflected XSS





c) Vulnerability: Stored XSS





Mitigation Strategies

Brute Force

- **Implement account lockout after failed login attempts:** Lock a user's account temporarily after several unsuccessful login attempts to deter automated guessing attacks.
- **Use CAPTCHA to prevent automated attacks:** Add CAPTCHA challenges on login forms to distinguish human users from bots.
- **Enforce strong password policies:** Require passwords with sufficient length and complexity to make brute-force attempts less effective.

SQL Injection

- **Use prepared statements and parameterized queries:** Always separate SQL code from user input so that attackers cannot inject malicious SQL commands.
- **Validate and sanitize inputs thoroughly:** Check and clean all user-provided data to ensure it meets expected formats before processing.
- **Employ Web Application Firewalls (WAF):** Use a WAF to detect and block suspicious SQL injection patterns before they reach your application.

Cross-Site Scripting (XSS)

- **Apply output encoding:** Convert special characters in user-supplied data into safe representations before displaying them in the browser to prevent script execution.
- **Implement Content Security Policies (CSP):** Set a CSP header to restrict where scripts can be loaded from, reducing the impact of injected code.
- **Validate and sanitize all user inputs:** Rigorously check and clean user input to ensure that harmful scripts are removed or escaped before processing.

5. Conclusion

The DVWA assessment confirmed critical vulnerabilities that reflect real-world security challenges. While DVWA is intentionally insecure, the findings emphasize the importance of:

- Secure coding practices (input validation, parameterized queries).
- Stronger authentication mechanisms (lockouts, CAPTCHAs, password policies).
- Defense-in-depth strategies (WAF, CSP, monitoring).