# Web Application Vulnerability Assessment Report -DVWA

**Intern Name**: Adebowale Emmanuel Okikiola
**Internship ID**: FIT/JUL25/CS2572
**Date: August 2025**

**Introduction**

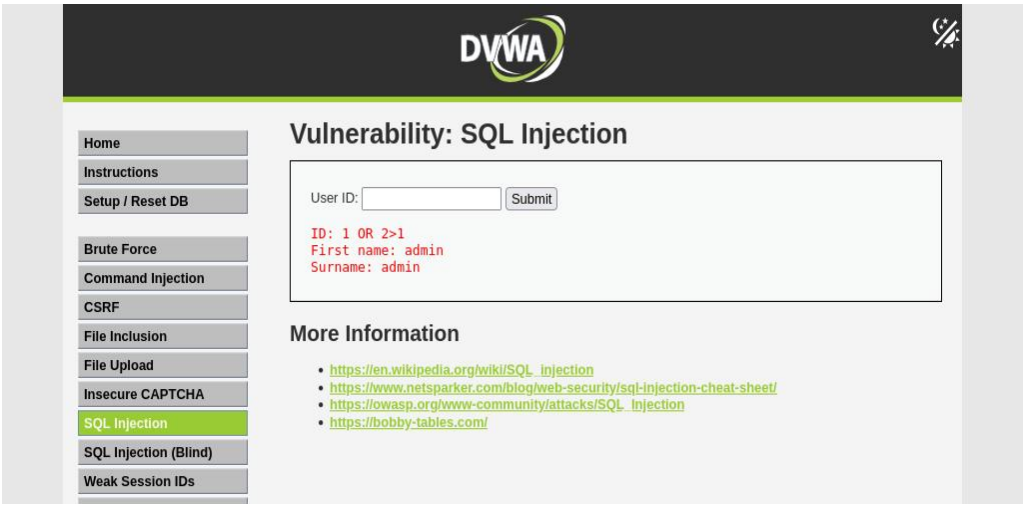
This report documents the results of a web application vulnerability assessment conducted on the Damn vulnerable web application ( DVWA) hosted in a controlled lab. The objective was to identify common OWASP vulnerabilities, understand their exploitation, and recommendation remediation measures.

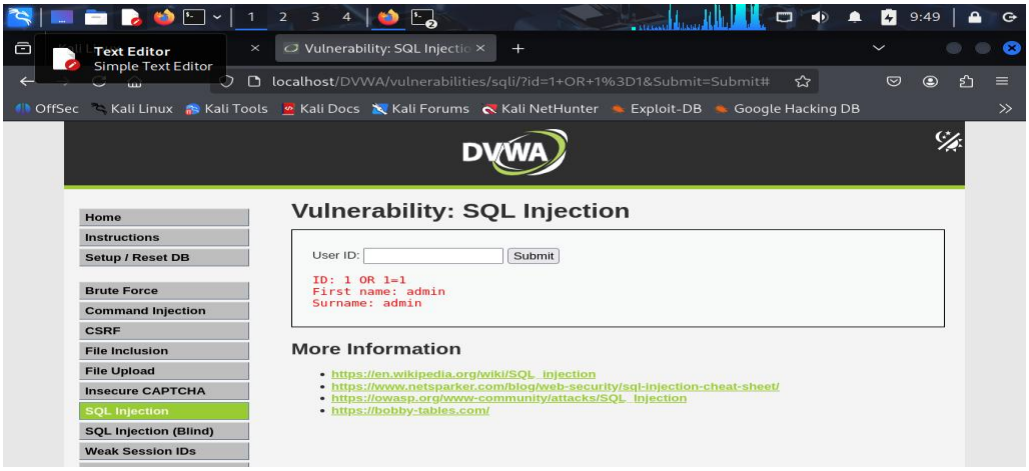
**Methodology**
**Environment**: Kali Linux VM, DVWA, Apache2, MariaDB
**Tools:** DVWA, Firefox Browser, Custom Scripts
**Approach:** Tested multiple OWASP vulnerabilities under the DVWA security level set to low and partially Medium.

## 3. Findings

### 3.1 SQL Injection

**Description**: SQL injection allows attackers to manipulate back-end database queries.
**Test payloads**: 1' OR '1'='1' -- and numeric injections.
**Observation**: injection attempts confirmed improper input sanitization.
**Evidence**:

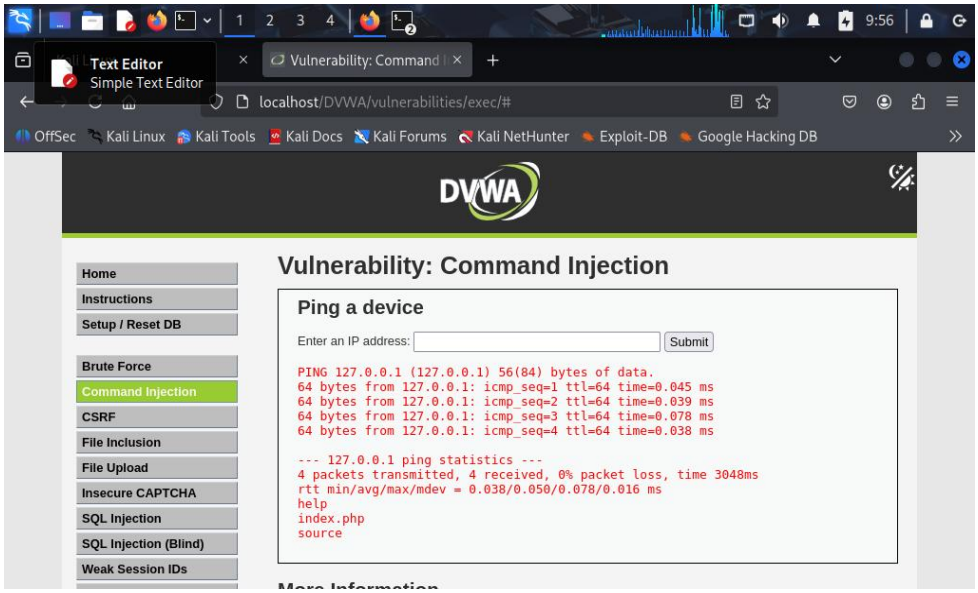


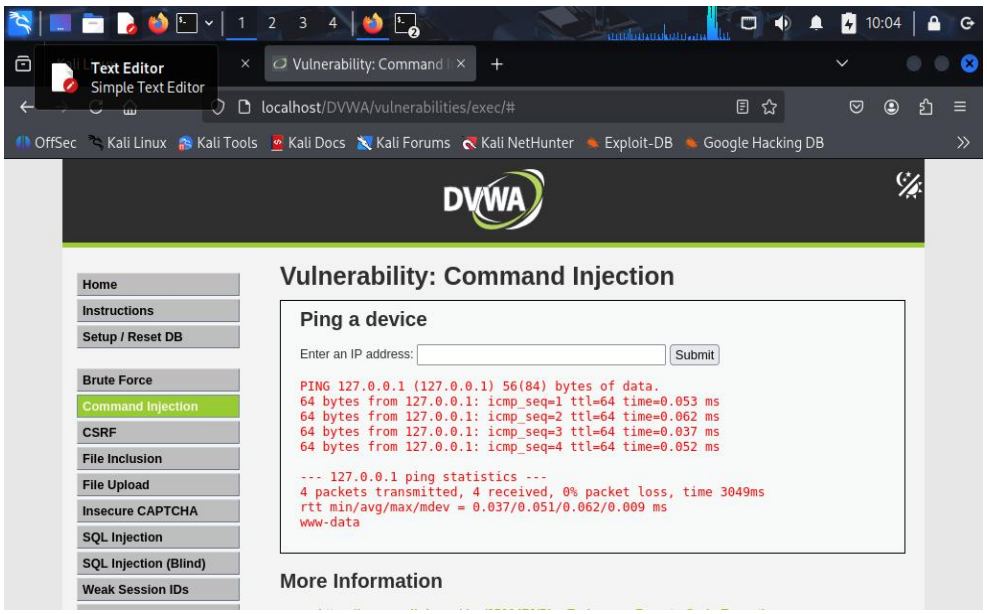




**Severity**: High
**Recommendation**: use parameterized queries, prepared statements, and strict input validation.

**Command injection**

**Description**: Command injection allows execution of systems commands via user input.
**Test payloads**: 127.0.0.1 && ls, 127.0.0.1 && whoami.
Observation: The server executed commands and revealed directory listings and user privileges.
**Evidence:**

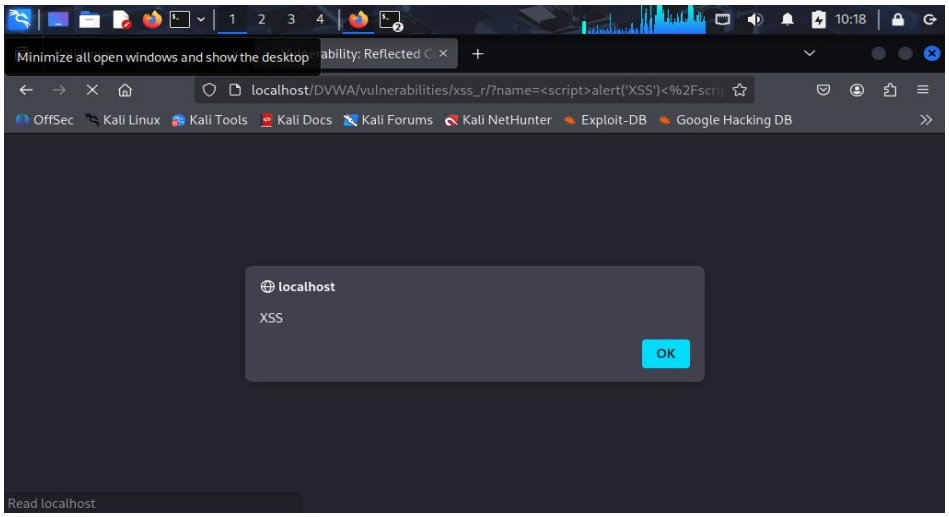



**Severity**: Critical
**Recommendations:** Validate inputs, disable direct command execution, use parameterized functions.

## Cross- site Scripting (XSS)- Reflected

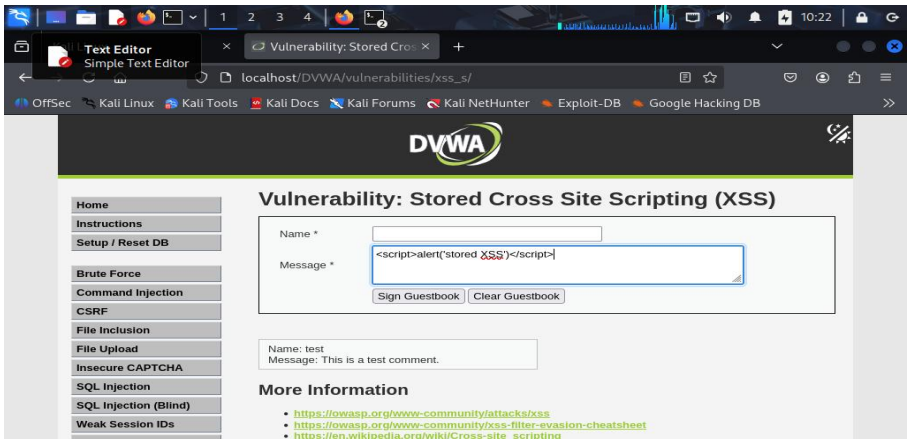
**Description:** Reflected XXS executes malicious scripts in the victim's browser.
**Test payload**: <script>alert('XSS')</script>.
**Observation**: JavaScript alert popup confirmed vulnerability.
**Evidence:**

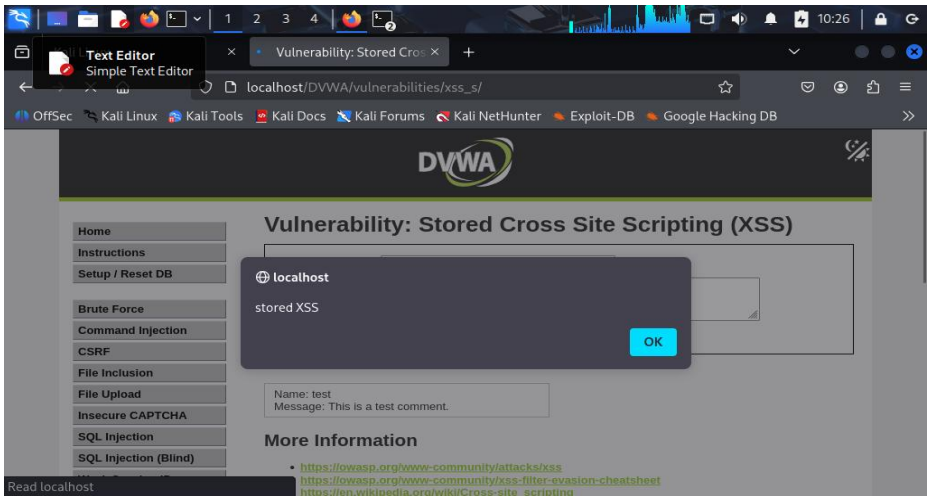



Severity: High
Recommendation: Apply output encoding, sanitize inputs, and implement content security policy (CSP).

## Cross-site Scripting (XSS)- Stored

**Description**: Stored XSS persists malicious scripts in the database.
**Test payload**: <script>alert('stored XSS')</script>.
**Observation**: Script executed on every page load, confirming persistent vulnerability.
**Evidence:**

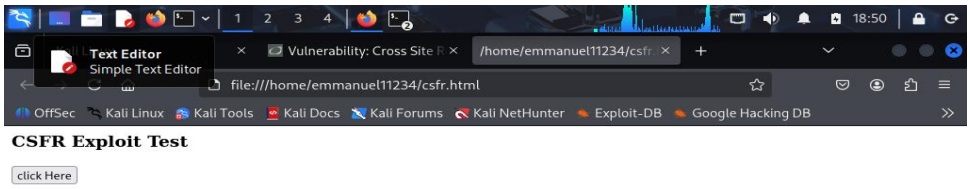


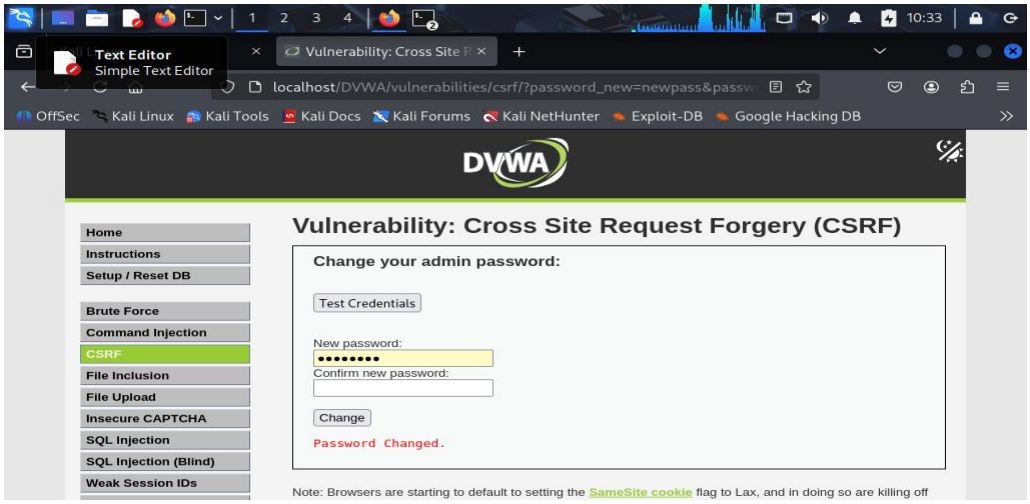




**Severity:** Critical
**Recommendation:** Filter and sanitize all user input before storage.

## Cross-site Request Forgery (CSFR)

**Description:** CSFR forces authenticated users to execute unwanted actions.
**Test:** Created a malicious HTML form (csfr.html) that changed the admin password.
**Observation:** password successfully changed without authentication tokens.
**Evidence:**

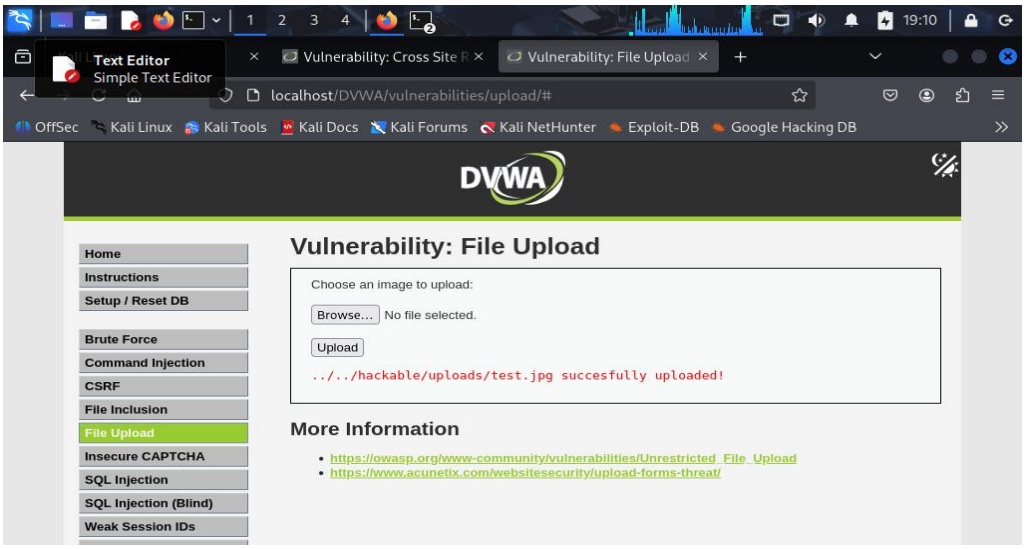


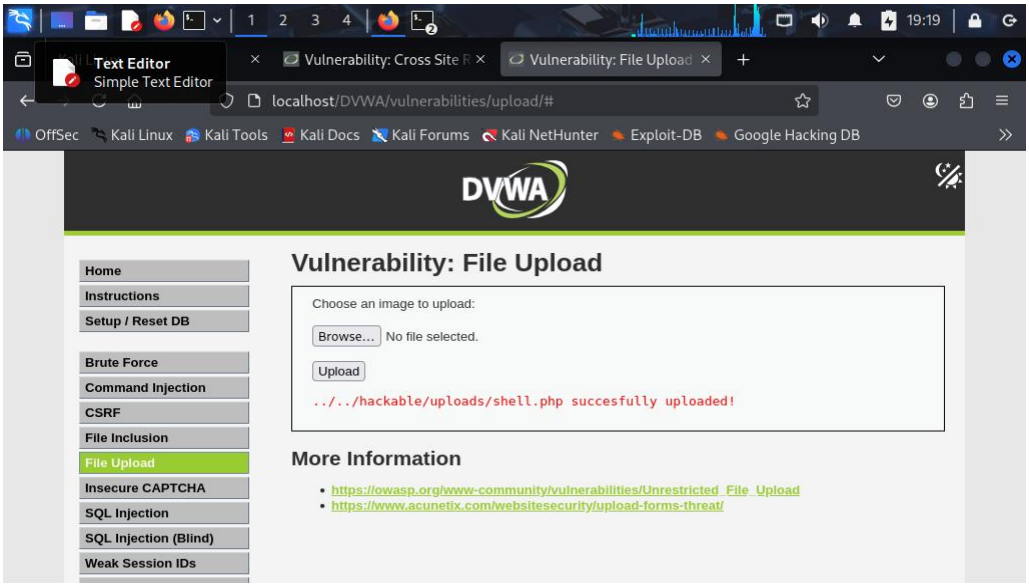




**Severity**: High
**Recommendation:** implement CSFR tokens, verify origin headers, and use Samsite cookies.

## File upload vulnerability

**Description:** Unrestricted file upload allows attackers to upload malicious files.
Test: Uploaded a shell. php script and executed it successfully.
**Observation:** Server executed the uploaded php code, proving remote code execution potential.
**Evidence:**




**Severity**: Critical
**Recommendation**: Restrict allowed file types, validate extensions, and store files outside web root.

## Conclusion

The assessment demonstrated multiple high and critical vulnerabilities within DVWA. These tests highlight the importance of secure coding practices, proper input validation, and security controls. Understanding these exploits equips defenders with better detection and mitigation strategies.

**References**

OWASP Testing guide
DVWA Documentation
Apache & PHP Security Best Practices