**Web Application Vulnerability Assessment Report**  
**Target Application**: Damn Vulnerable Web Application (DVWA)  
**Host OS**: Metasploitable 2  
**Report Prepared By**: Ankit Roy  
**Date**: 11.07.2025  
**Tools Used**: OWASP ZAP, Burp Suite

**📑 Table of Contents**

1. Executive Summary
2. Project Overview
3. Scope of Assessment
4. Testing Methodology
5. Summary of Findings
6. Detailed Vulnerabilities
7. OWASP Top 10 Mapping
8. Conclusion and Recommendations

**🧾 Executive Summary**

This report presents the results of a vulnerability assessment of **DVWA** hosted on **Metasploitable 2**. The objective was to simulate real-world attacks using ethical hacking tools to identify critical security flaws and recommend solutions. Tools used include **Burp Suite**, **OWASP ZAP** aligned with the **OWASP Top 10** framework.

**2. Project Overview**

This hands-on cybersecurity internship project simulates a real-world scenario of testing a startup’s vulnerable web app. It provides insights into how attackers exploit web vulnerabilities and how to defend against them. Findings from DVWA are documented in this professional report.

**Skills Developed**:

* Web App Vulnerability Testing
* Report Writing & Documentation
* OWASP Top 10 Familiarity
* Penetration Testing Basics
* Risk Classification & Mitigation

**3. Scope of Assessment**

* **Tested App**: Damn Vulnerable Web Application (DVWA)
* **Host**: Metasploitable 2 (Local VM)
* **Tools Used**:
  + OWASP ZAP
  + Burp Suite
* **Test Categories**:
  + SQL Injection
  + Cross-Site Scripting (XSS)
  + Cross-Site Request Forgery (CSRF)
  + Command Injection
  + Weak Authentication & Sessions

**4. Testing Methodology**

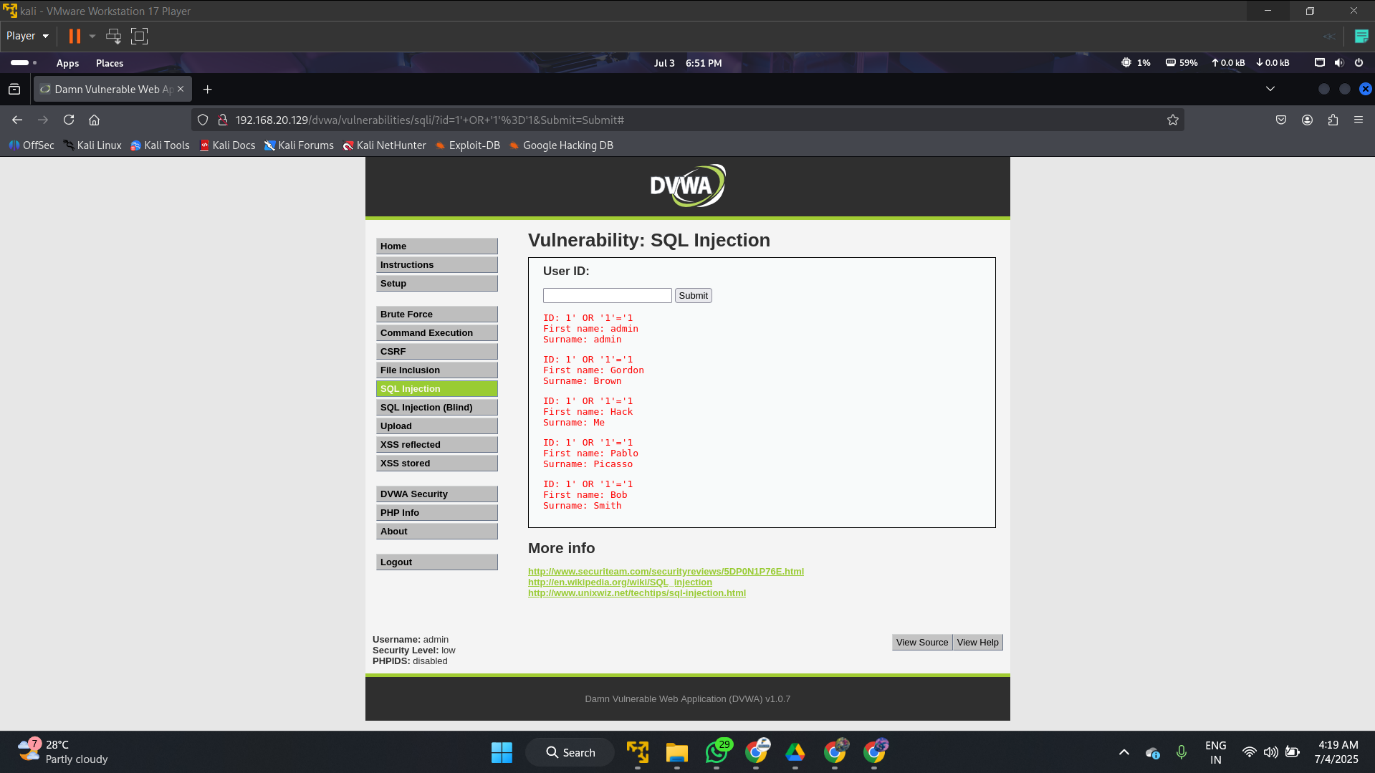
| **Phase** | **Description** |
| --- | --- |
| **Scanning** | Identified potential flaws using ZAP and Burp Suite |
| **Exploitation** | Manually triggered and verified vulnerabilities |
| **Post-Exploitation** | Assessed impact of each flaw |
| **Reporting** | Documented findings with screenshots and remediation steps |

**5. Summary of Findings**

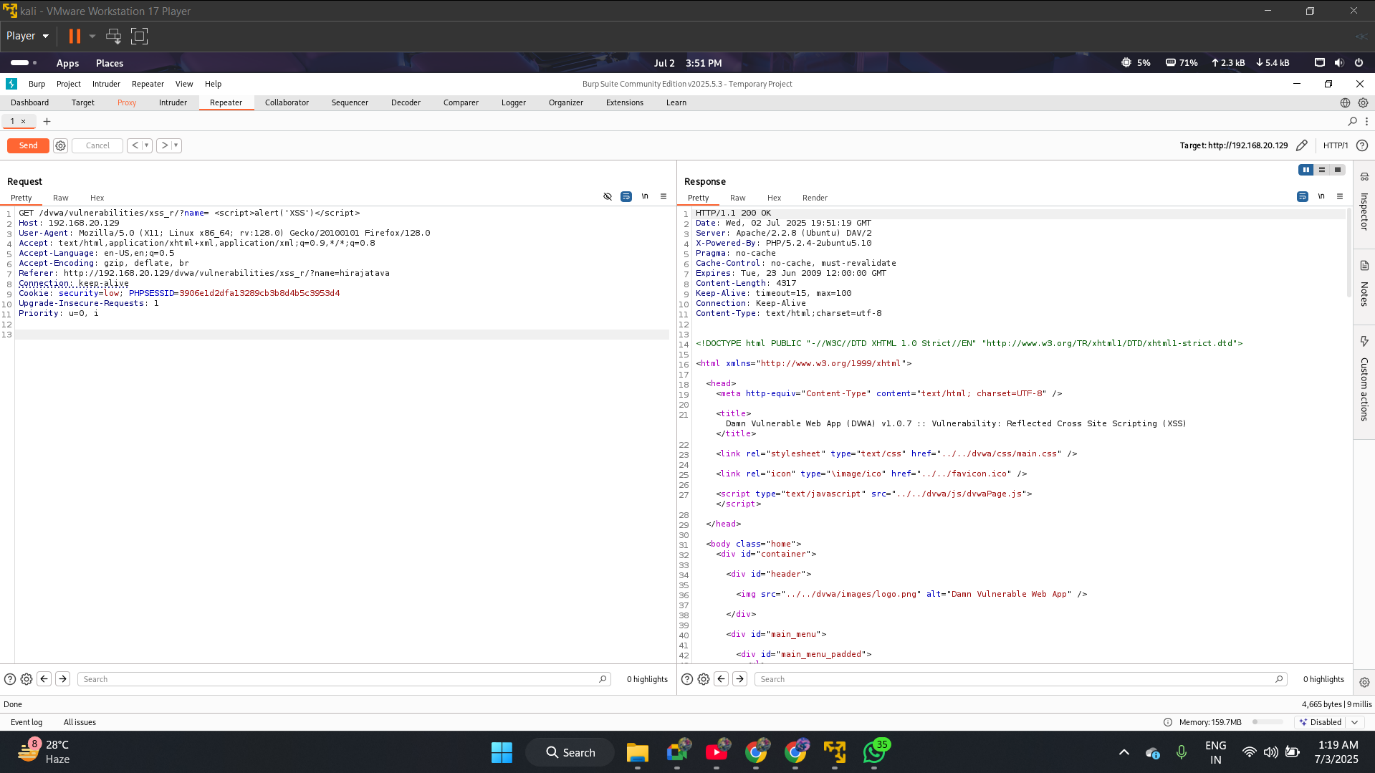
| **#** | **Vulnerability** | **Risk Level** | **Tool Used** | **OWASP Category** |
| --- | --- | --- | --- | --- |
| 1 | SQL Injection | High | Burp Suite | A1 – Injection |
| 2 | Reflected XSS | Medium | ZAP | A7 – XSS |
| 3 | Stored XSS | High | Manual | A7 – XSS |
| 4 | CSRF | Medium | Manual | A8 – CSRF |
| 5 | Command Injection | High | Burp Suite | A1 – Injection |
| 6 | Insecure File Upload | High | Manual | A5 – Access Control |
| 7 | Weak Authentication | High | Manual | A2 – Broken Auth |

6. Detailed Vulnerabilities

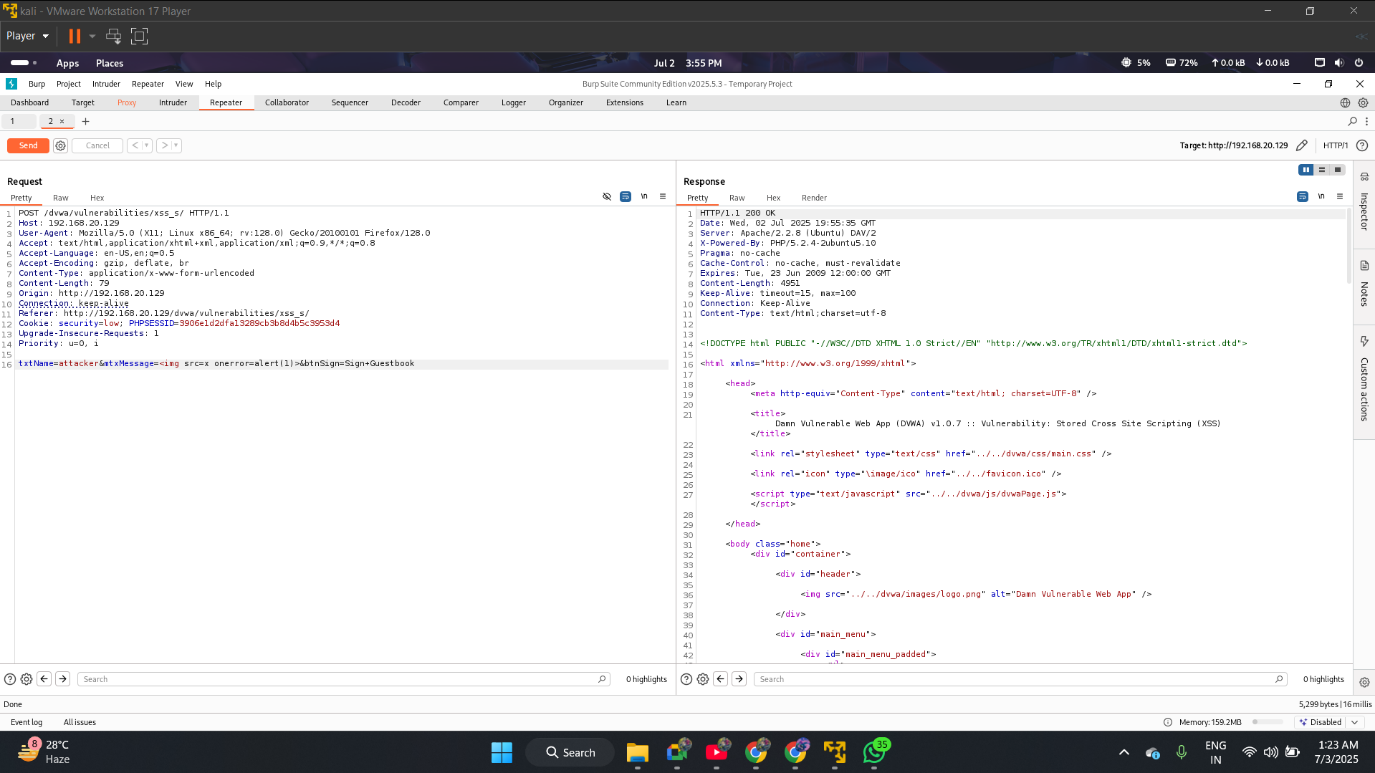
**6.1 SQL Injection**

* **Description**: Login form input not sanitized, allowing SQL manipulation.
* **Risk**: Low
* **Payload Used**: ' OR '1'='1
* **Impact**: Bypass auth, dump DB
* ****
* **Mitigation**:
  + Use parameterized queries
  + Apply input validation
  + Avoid dynamic SQL construction

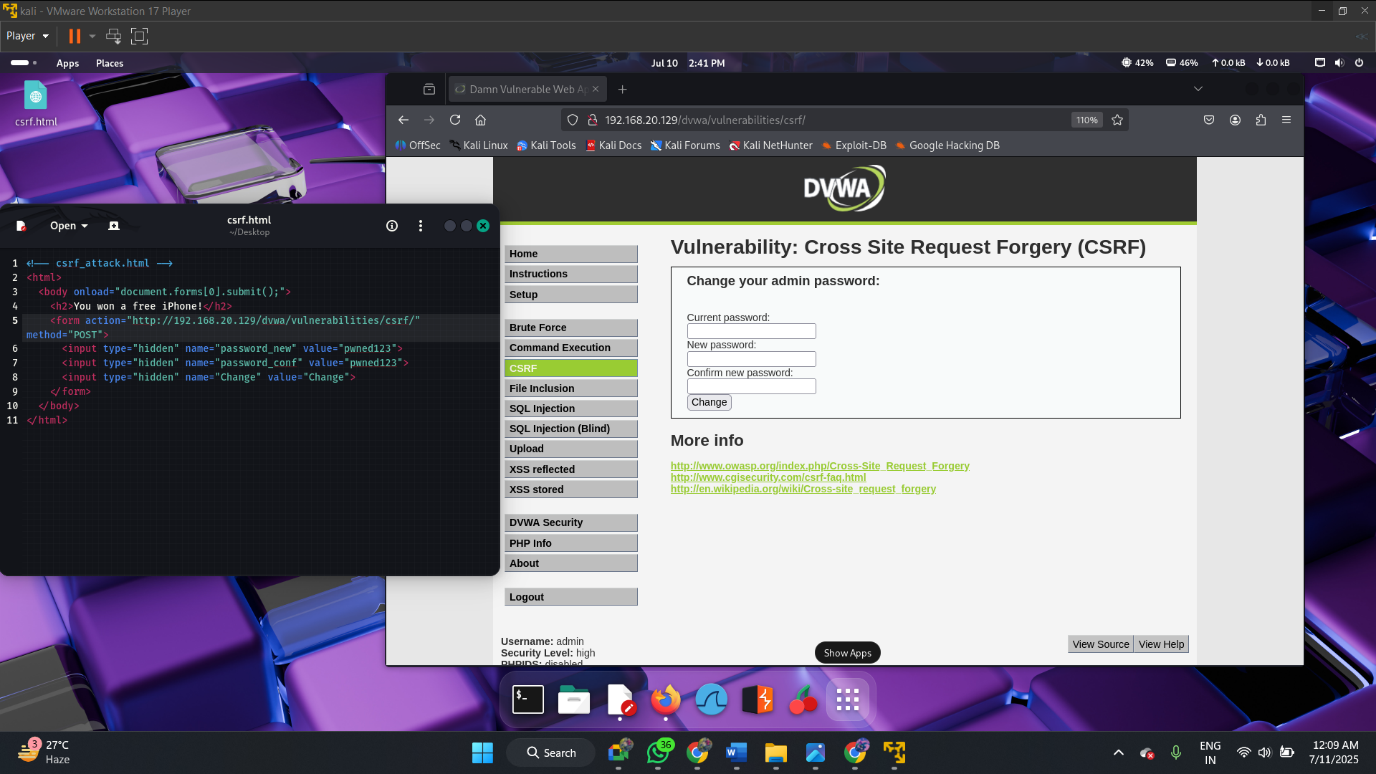
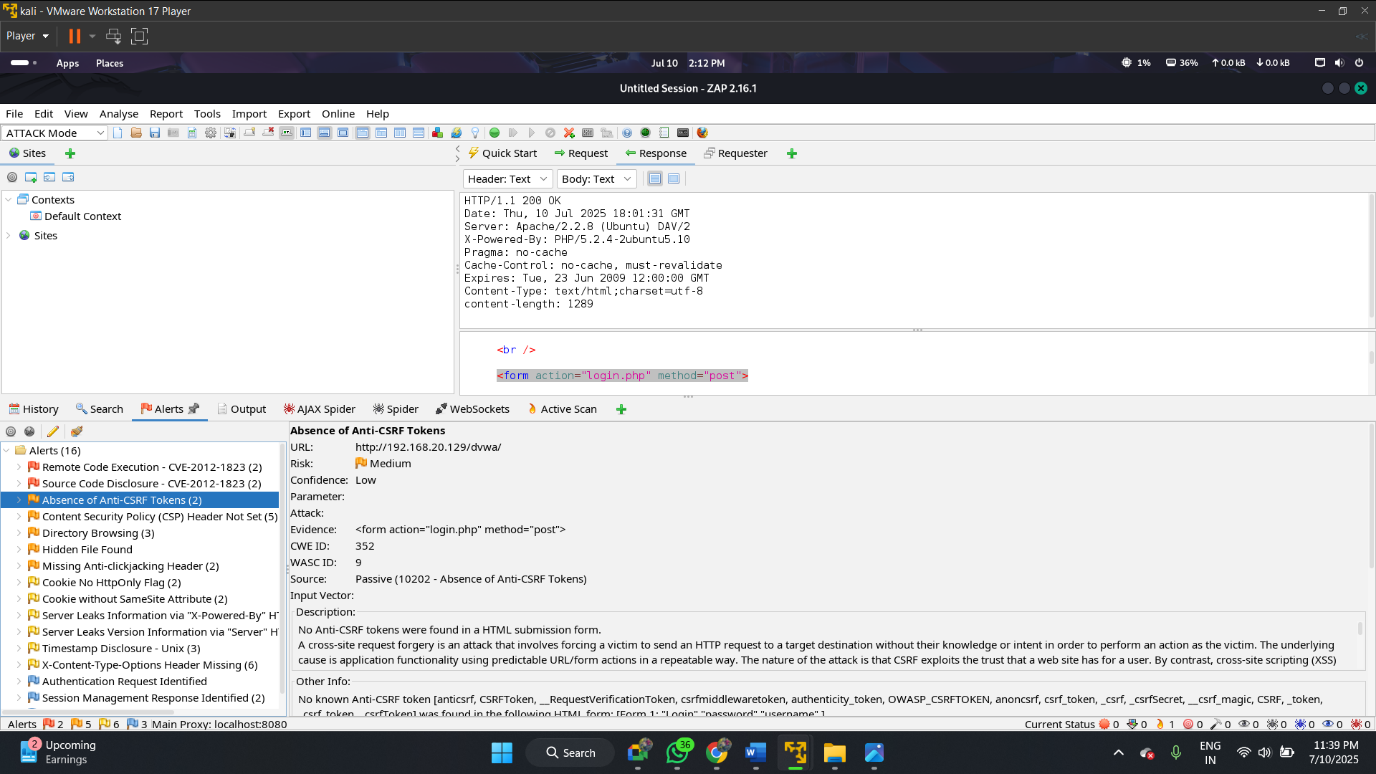
**6.2 Reflected XSS**

* **Description**: Input reflected in URL output without encoding
* **Risk**: Medium
* **Impact**: Cookie theft, redirection
* **Tool Used**: Burp suite
* ****
* **Mitigation**:
  + Encode all output
  + Use Content Security Policy (CSP)
  + Validate input length and characters

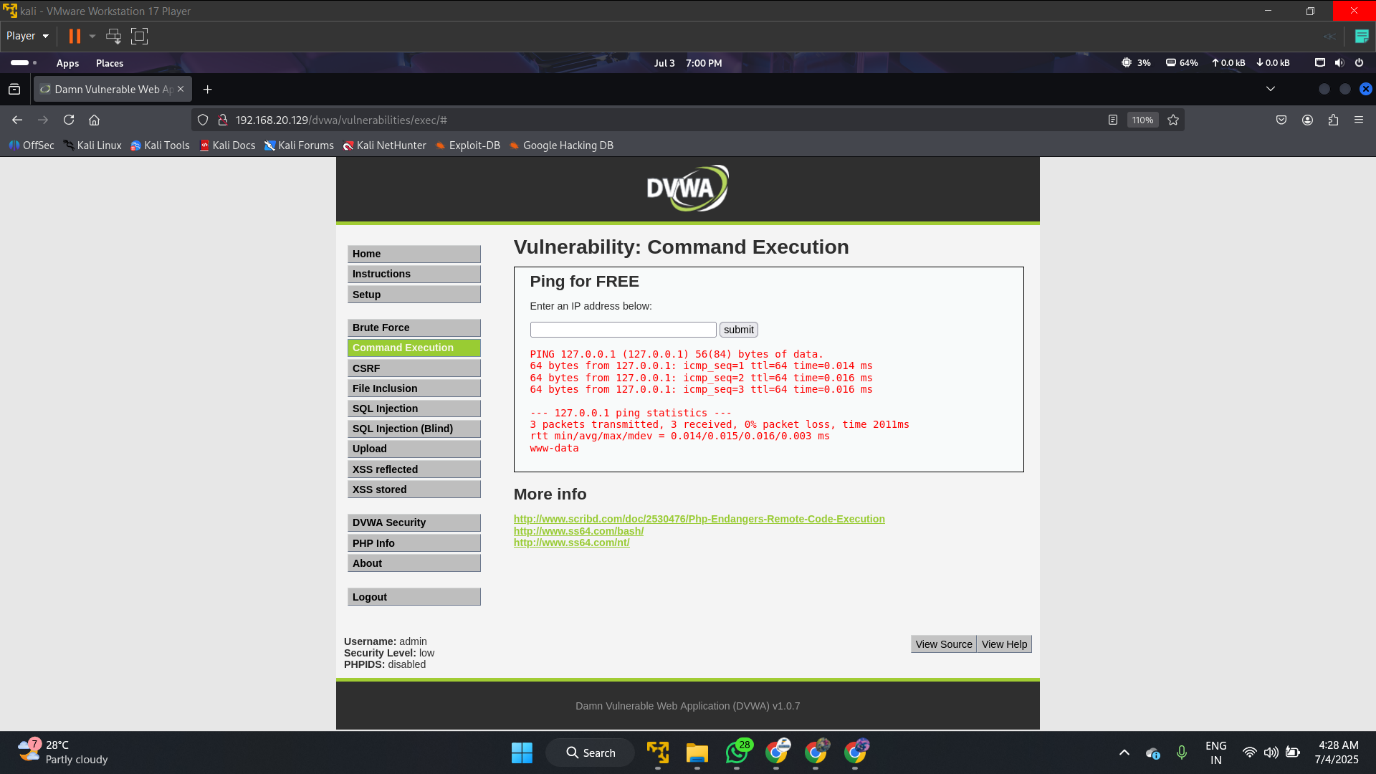
**6.3 Stored XSS**

* **Description**: Malicious script stored in comment section
* **Risk**: Low
* **Impact**: Persistent attack on other users
* **Tool Used**: Burp Suite
* ****
* **Mitigation**:
  + Sanitize database inputs
  + Escape HTML output
  + Use secure libraries like DOMPurify

**6.4 CSRF**

* **Description**: 2 CSRF tokens, allowing forged requests
* **Risk**: Medium
* **Impact**: Force password change via GET
* **Tool Used**: Manual
* ****
* 
* **Mitigation**:
  + Add CSRF tokens
  + Use SameSite cookie attribute
  + Require re-authentication for sensitive actions

**6.5 💻 Command Injection**

* **Description**: User input directly injected into OS command
* **Risk**: Low
* **Impact**: Full system compromise
* **Tool Used**: Manual
* ****
* **Mitigation**:
  + Avoid system commands with user input
  + Use escapeshellarg() or similar
  + Run app under least privilege

**7. OWASP Top 10 Mapping**

| **OWASP Category** | **Detected?** | **Comments** |
| --- | --- | --- |
| A1 – Injection | ✅ | SQL & Command Injection found |
| A2 – Broken Authentication | ✅ | Weak login mechanisms |
| A3 – Sensitive Data Exposure | ❌ | Not found |
| A4 – XML External Entities (XXE) | ❌ | Not applicable in DVWA |
| A5 – Broken Access Control | ✅ | Insecure File Upload |
| A6 – Security Misconfiguration | ⚠️ | Basic default config observed |
| A7 – Cross-Site Scripting (XSS) | ✅ | Both stored and reflected XSS |
| A8 – Insecure Deserialization | ❌ | Not detected |
| A9 – Using Components with Known Vulns | ❌ | Not tested |
| A10 – Insufficient Logging & Monitoring | ❌ | Not evaluated |

**8. Conclusion and Recommendations**

The DVWA application on Metasploitable 2 contains several critical vulnerabilities commonly found in real-world systems. Immediate action is needed to:

* Implement secure coding practices
* Use a Web Application Firewall (WAF)
* Regularly update and patch web components
* Educate developers on OWASP standards