Web Application Security Testing – Security Assessment Report

1. Introduction

This report summarizes the vulnerability assessment performed on an open-source, intentionally vulnerable web application. The objective of this project was to identify, analyze, and document potential security weaknesses using ethical hacking tools and OWASP standards. The assessment simulates real-world security testing commonly required by startups, SaaS providers, and e-commerce platforms to ensure a secure web environment.

2. Scope of Assessment

The scope included evaluating the application's core modules, input fields, authentication mechanisms, and request handling. Testing was carried out on platforms such as OWASP Juice Shop and DVWA. The focus areas included:

- SQL Injection (SQLi)
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Broken authentication and session management
- Security misconfigurations

3. Tools Used

The following are the tools used in this project

- Burp Suite (Community Edition) – For manual testing and interception
- OWASP ZAP – For automated scanning and vulnerability detection
- Nikto – For web server scanning
- DVWA & OWASP Juice Shop – Test environments
- Kali Linux (optional) – Penetration testing OS
- Google Docs / MS Word – Documentation and reporting

4. Methodology

The testing methodology followed five major steps:

1. Information Gathering – Understanding the application's structure, endpoints, and functionality.

2. Scanning & Enumeration – Identifying open ports, services, and possible attack vectors.

3. Vulnerability Testing – Executing tests for SQLi, XSS, CSRF, and other OWASP Top 10 vulnerabilities.

4. Analysis – Verifying findings, determining their impact, and mapping them to OWASP categories.

5. Reporting – Documenting vulnerabilities with evidence, severity levels, and remediation steps.

5. Key Findings

The following key vulnerabilities were identified during the assessment:

- **SQL Injection**: Application inputs were not properly sanitized, allowing malicious SQL queries.
- **Reflected XSS**: Certain pages allowed user-controlled scripts to be executed.
- **Weak Authentication**: Password complexity requirements were minimal.
- **Security Misconfigurations**: Application disclosed unnecessary information via error messages.
- **CSRF Vulnerabilities**: Sensitive actions were not protected by CSRF tokens.

6. Impact Assessment

If exploited, the vulnerabilities could lead to severe consequences such as:

- Unauthorized access to sensitive user data

- Account takeover

- Data manipulation and database exposure

- Execution of malicious scripts

- Overall compromise of application integrity

7. Recommendations

Based on the findings, the following remediation steps are recommended:

- Implement input validation and parameterized queries for SQLi prevention

- Encode and sanitize outputs to prevent XSS attacks

- Strengthen authentication mechanisms with strong password policies

- Configure proper error handling and disable debug information

- Add CSRF tokens to sensitive requests

8. Conclusion

The security assessment highlighted multiple vulnerabilities commonly found in modern web applications. By addressing the identified weaknesses and following OWASP security practices, the application can significantly improve its security posture. This project provided hands-on experience in ethical hacking, vulnerability scanning, and professional security reporting.