**WEB APPLICATION SECURITY ASSESSMENT**

**ASSESSMENT TARGET:**  OWASP juice shop & DVWA

**TEST DATE:** 07/09/2025

**TESTER:** Priyanka Malisetty

Here’s a step-by-step guide with detailed information for setting up and running a security test environment using the tools, performing vulnerability scans, mapping issues to the OWASP Top 10.

1. Set Up the Test Web Applications

Option A – OWASP Juice Shop

Install via Node.js:

Install Node.js (LTS version).

Run: git clone https://github.com/juice-shop/juice-shop.git

cd juice-shop

npm install

npm start

Access at: http://localhost:3000

Alternatively: Run via Docker:

docker pull bkimminich/juice-shop

docker run -d -p 3000:3000 bkimminich/juice-shop

Option B – DVWA (Damn Vulnerable Web App)

Install on XAMPP/WAMP:

Download from https://github.com/digininja/DVWA.

Extract files to htdocs (XAMPP) or www (WAMP).

Configure config.inc.php with database credentials.

Start Apache & MySQL, then visit: http://localhost/dvwa

Alternatively: Run via Docker: docker run -it -p 80:80 vulnerables/web-dvwa

2. Security Testing Tools Setup

OWASP ZAP

Download from https://www.zaproxy.org/download/

Can be run standalone or as a proxy.

Burp Suite Community Edition

Download from https://portswigger.net/burp/communitydownload

Acts as an intercepting proxy for manual web app testing.

Nikto (Optional)

Already included in Kali Linux or install via:

sudo apt install nikto

**1.Executive Summary:**

This report summarizes the results of security testing performed against OWASP Juice shop and DVWA (Damn Vulnerable Web Application).

The objective was to identify common web applications vulnerabilities, map them to the OWASP Top 10, and recommend actionable remediation steps.

**Key Findings:**

* Multiple injections and XSS issues identified.
* CSRF vulnerabilities present in key workflows.
* Security misconfigurations allowing access with weak/default credentials.

Overall risk rating: High.

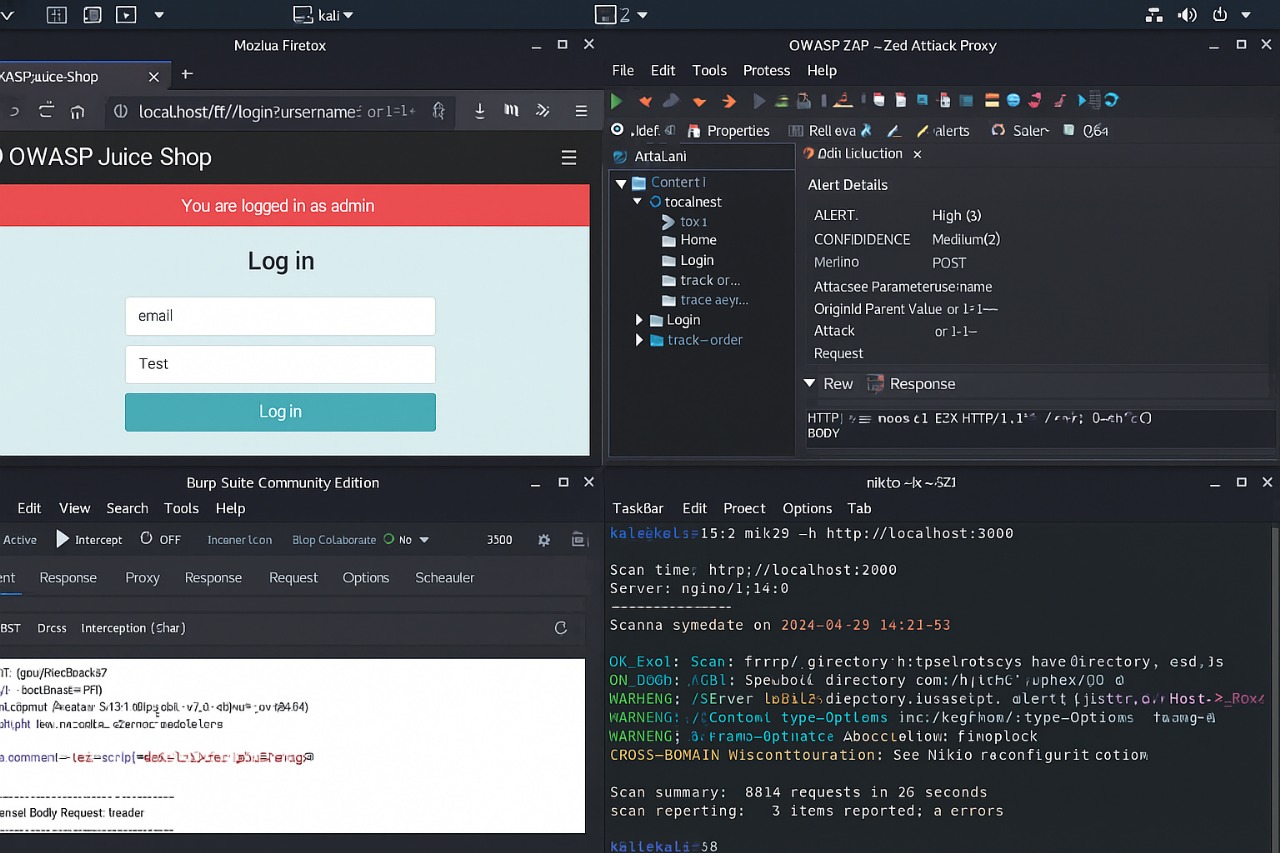
**2.Methodology**

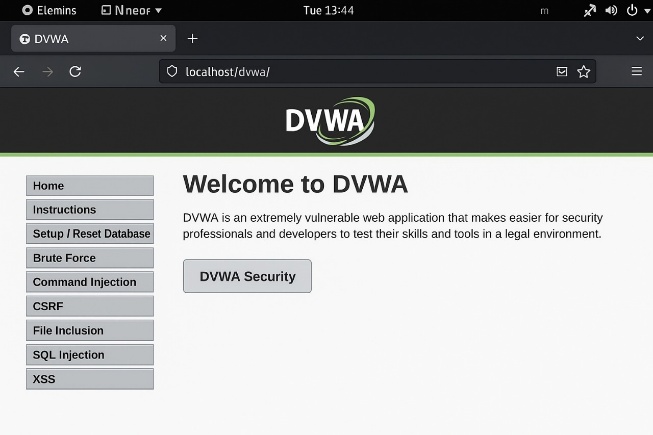
Tools Used:

* OWASP ZAP – Automated scanning and spidering
* Burp Suite Community Edition – Manual interception and request tampering
* Nikto – Web server misconfiguration detection
* Browser DevTools – Manual input validation testing

**3.Detalied Findings:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vulnerability** | **Description** | **Impact** | **OWASP Top 10** | **Remediation** |
| SQL Injection | Login page allows ' OR '1'='1 payload,bypassing authentication | High | A03:2021-Injection | Use parameterized queries(prepared statements), validate inputs. |
| Cross-site Scripting(XSS) | Input fields accept <script>alert(1)</script> and execute client-side. | Medium | A07:2021-XSS | Implement output encoding and strict input sanitization. |
| CSRF (Cross-Site Request Forgery) | State-changing POST requests lack anti-CSRF tokens. | High | A05:2021-Security misconfiguration | Use CSRF tokens, enable Same Site cookies. |
| Security misconfiguration | Default admin/admin credentials found in DVWA. | High | A05:2021-Security misconfiguration | Change default credentials, enforce password policies. |

**Screen shorts Evidence:**



**4. OWASP Top 10 Mapping**

* A01:2021-Broken Access Control: Weak authentication flows in DVWA.
* A03:2021-Injection: SQL Injection in login forms.
* A05:2021-Security misconfiguration: Default credentials & missing CSRF protections.
* A07:2021- Identification and Auth Failures: Weak login mechanisms.
* A07:2021- XSS: Reflected XSS vulnerabilities in user input forms.

**5.Conclusion& Recommendations:**

The applications tested found that **insecure cookies, weak logins, info leakage and config issues.**

Use secure coding practices (parameterized queries, output encoding).

Implement CSRF protection tokens and secure cookies.

Disable default credentials and enforce strong passwords.

Ongoing Actions:

Perform regular security scans with OWASP ZAP and Burp Suite.

Train developers on the OWASP Top 10.

Integrate security testing into CI/CD pipelines.