penetration testing report

Target: OWASP JUICE SHOP

**Submitted by:** Unaiza Fatima  
**ID**: GxMSyHQoJJCBBbOvaMe7  
**Role:** Cyber Security Intern  
**Duration:** 15-3-2025-15-4-2025  
**Organization**:  Intern Intelligence

**Target Application:** OWASP Juice Shop  
**Tester Name:** Unaiza Fatima   
**Testing Platform:** Kali Linux

**1. Executive Summary**

In this report, I present the results of a penetration test I conducted on OWASP Juice Shop, an intentionally vulnerable web application designed for security training and awareness. My goal was to simulate real-world attack scenarios, identify and exploit potential vulnerabilities, evaluate their impact, and suggest suitable remediation steps.

**Key Findings:**

* Full admin access via credential exposure
* SQL injection allowing unauthorized access
* Multiple instances of broken authentication
* Cross-Site Scripting (XSS) vulnerabilities
* Insecure transport of sensitive data
* Access to confidential files
* Cryptographic flaws
* Cross-Origin Resource Sharing (CORS) misconfigurations

**Overall Risk Rating:** High

**Why It Matters:** Exploiting these vulnerabilities in a real-world environment could lead to complete system compromise, user data leakage, unauthorized access, and reputational damage. This report provides a comprehensive overview of these issues to facilitate effective remediation.

**2. Methodology**

**Testing Phases:**

* **Reconnaissance:** Identifying application structure and potential entry points
* **Scanning:** Automated and manual testing for vulnerabilities
* **Exploitation:** Safely exploiting vulnerabilities to assess risk
* **Reporting:** Documenting findings with evidence and recommendations

**Tools Used:**

* Kali Linux (OS)
* Burp Suite (Web Proxy)
* OWASP ZAP (Vulnerability Scanner)
* Browser Developer Tools
* Manual Testing Techniques

**3. Scope of Testing**

| **Component** | **Details** |
| --- | --- |
| Target Application | OWASP Juice Shop |
| URL | <http://localhost:3000> |
| Environment | Local test environment |

**4. Vulnerability Findings**

**4.1 Admin Credential Disclosure**

* **Severity:** Critical
* **Description:** The admin credentials (\*\*\*\*\*\*@juice-sh.op / 1\*\*\*\*\*) were exposed, enabling direct access to the administrative interface.
* **Impact:** Complete system control, including access to user data, order history, and back-end configurations.
* **Why It Matters:** Attackers gaining admin privileges can manipulate or extract sensitive data, delete user records, or escalate attacks.
* **Remediation:** Enforce strong passwords, implement rate-limiting, and avoid using predictable credentials.

**4.2 SQL Injection**

* **Severity:** High
* **Description:** Unsanitized input fields allowed execution of malicious SQL queries (' OR 1=1--).
* **Impact:** Authentication bypass and potential access to the database.
* **Why It Matters:** SQL injection is a top OWASP vulnerability that can expose all data in the application’s database.
* **Remediation:** Use parameterized queries and ORM frameworks.

**4.3 Broken Authentication – Jim's Password Reset**

* **Severity:** High
* **Description:** The reset mechanism for Jim’s account was vulnerable to manipulation, allowing unauthorized password reset.
* **Impact:** Account takeover, impersonation, and unauthorized access to private data.
* **Remediation:** Use cryptographically strong reset tokens with short expiry.

**4.4 Broken Authentication – Bjorn’s Favorite Pet**

* **Severity:** High
* **Description:** Security questions were easily guessable or brute-forceable.
* **Impact:** Unauthorized recovery of account credentials.
* **Remediation:** Avoid security questions or use MFA instead.

**4.5 Reflected XSS**

* **Severity:** High
* **Description:** Unsanitized user input was reflected in the browser without proper encoding.
* **Payload Example:** ?search=<script>alert("XSS")</script>
* **Impact:** Session hijacking, phishing, or defacement.
* **Remediation:** Properly encode output and validate input.

**4.6 DOM XSS – Privacy Policy Page**

* **Severity:** Medium
* **Description:** JavaScript processed user input insecurely, allowing DOM-based XSS.
* **Impact:** Execution of attacker-controlled scripts in user browsers.
* **Remediation:** Use safe client-side APIs and libraries like DOMPurify.

**4.7 CORS Misconfiguration**

* **Severity:** Medium
* **Description:** Permissive cross-origin resource sharing allowed data to be shared with unauthorized origins.
* **Impact:** Potential leakage of sensitive information to malicious websites.
* **Remediation:** Restrict CORS to known and trusted origins.

**4.8 Plaintext Password over HTTP**

* **Severity:** Critical
* **Description:** Login credentials were transmitted over HTTP without encryption.
* **Impact:** Credentials could be intercepted using packet sniffing tools.
* **Remediation:** Enforce HTTPS across the entire application.

**4.9 Login to Admin Account**

* **Severity:** Critical
* **Description:** Admin account was accessible using known credentials.
* **Impact:** Full administrative control of the application.
* **Remediation:** Implement account lockout policies and use MFA.

**4.10 Confidential Files**

* **Severity:** High
* **Description:** Access to files such as .git, backup files, and logs was unrestricted.
* **Impact:** Leakage of sensitive information including source code and configuration.
* **Remediation:** Block access to sensitive directories/files using web server configuration.

**4.11 Weak Cryptography**

* **Severity:** Medium
* **Description:** Custom cryptographic implementation used for tokens and password storage.
* **Impact:** Vulnerable to reverse engineering and token forgery.
* **Remediation:** Use secure, vetted algorithms such as SHA-256 and bcrypt.

**5. Exploitation Summary**

| **Vulnerability** | **Exploited** | **Impact Summary** |
| --- | --- | --- |
| Admin Login | Yes | Full application control |
| SQL Injection | Yes | Unauthorized login |
| Jim’s Password Reset | Yes | Account takeover |
| Bjorn’s Security Question | Yes | Credential recovery |
| Reflected XSS | Yes | Arbitrary JavaScript execution |
| DOM XSS | Yes | Script execution in browser |
| Confidential Files | Yes | Access to internal resources |
| Plaintext Password | Yes | Credentials intercepted |
| CORS Misconfiguration | Partially | Potential third-party access |
| Weak Cryptography | Partially | Risk of token manipulation |

**6. Risk Assessment**

| **Vulnerability** | **Severity** | **Why It’s Critical** |
| --- | --- | --- |
| Admin Login / Credential Leak | Critical | Complete admin access |
| SQL Injection | High | Database and login compromise |
| Jim's Password Reset | High | Unauthorized control |
| Bjorn’s Pet Question | High | Account access via guessing |
| Reflected XSS | High | User/session hijacking |
| DOM XSS | Medium | Browser-level code execution |
| Plaintext Password via HTTP | Critical | Easy interception of credentials |
| Confidential Files | High | Information leakage |
| CORS Misconfiguration | Medium | Sensitive data leak |
| Weak Cryptography | Medium | Token compromise risk |

**7. Conclusion**

The penetration test on OWASP Juice Shop revealed a variety of critical and high-risk vulnerabilities that could have severe consequences in a production environment. These vulnerabilities underline the importance of secure authentication mechanisms, encrypted communications, proper access control, and secure coding practices. Addressing the identified issues will greatly reduce the attack surface of the application and improve its overall security posture.

**8. Appendix**

* Screenshots of payloads and responses
* Burp Suite and ZAP logs
* Exploited URLs and payloads
* References:
  + OWASP Top 10
  + OWASP Juice Shop GitHub
  + CWE Database
  + Mozilla Developer Network (MDN)