Penetration Testing Report: OWASP Juice Shop (Local Instance)

Prepared by: Maxwell Bosiako Antwi

Date: 4th May, 2025

Environment: Local Kali Linux with OWASP Juice Shop via Docker

Testing Tools Used: Docker, Burp Suite, Nmap, Firefox, Kali Linux utilities

Executive Summary

This report documents the results of a penetration test conducted against a locally hosted instance of OWASP Juice Shop. The goal of the test was to explore and exploit known vulnerabilities, simulating real-world attack scenarios. Multiple critical vulnerabilities were discovered, including SQL Injection, Stored and Reflected XSS, File Upload Vulnerability, and Directory Traversal. These findings underscore the importance of proper input validation, access controls, and secure configuration practices in modern web applications.

Environment Setup

- Installed Docker on Kali Linux
- Pulled and ran OWASP Juice Shop Docker image:

```
sudo docker pull bkimminich/juice-shop
sudo docker run --rm -p 3000:3000 bkimminich/juice-shop
```

- Verified service availability on port 3000 via Nmap and this port was actively listening:

```
nmap -p 3000 127.0.0.1
```

Detailed Findings

1. SQL Injection (Authentication Bypass)

Vulnerability Type: Injection

Severity: High

Vector: Login form

Payload Used: `' OR '1'='1`

Description:
The login functionality was susceptible to SQL injection. After intercepting the login request using Burp Suite, a payload (`' OR '1'='1`) was injected into both username and password. While this did not initially succeed due to front-end validation, manual modification of the request in Burp Repeater allowed bypassing the authentication controls, successfully logging in as an administrator with the payload (Email: `' OR 1=1-- and` Password: `'`).

Impact:

- Unauthorized access to administrative functionalities
- Potential for data exfiltration or privilege escalation

Recommendation:

- Use parameterized queries (e.g., prepared statements)
- Implement strict server-side input validation
- Employ a Web Application Firewall (WAF) for query anomaly detection

2. Stored Cross-Site Scripting (XSS)

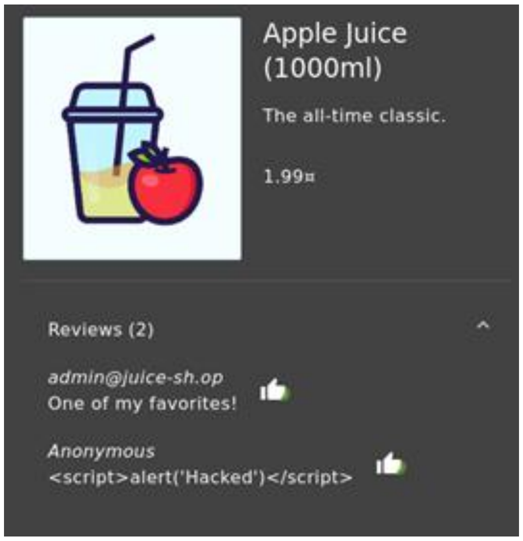
Vulnerability Type: Stored XSS

Severity: High

Vector: Product Review comment field

Payload: `<script>alert("Hacked")</script>`

Description:
Malicious JavaScript was injected into a product review. This script was stored server-side and executed when the product page was reloaded. This affects every user who views the page, making it highly dangerous.



admin@juice-sh.op
One of my favorites!

Anonymous
<script>alert('Hacked')</script>

Impact:

- Theft of session cookies or user credentials
- Drive-by malware delivery
- UI defacement

Recommendation:

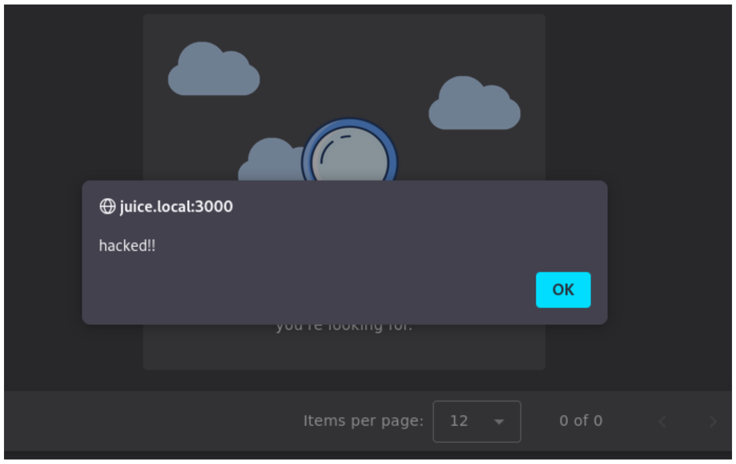
- Sanitize user inputs and encode output (especially in HTML contexts)
- Use frameworks that auto-escape outputs (e.g., React, Angular)
- Implement Content Security Policy (CSP)

3. Reflected Cross-Site Scripting (XSS)

Vulnerability Type: Reflected XSS
Severity: Medium
Vector: Search bar
Payload: `<img src=x onerror=alert("hacked!!")>`

Description:
The search feature failed to sanitize user input in URL parameters, leading to the execution of

arbitrary JavaScript. This vulnerability could be used in phishing campaigns by sending a malicious link to users.


juice.local:3000

hacked!!

Impact:

- User session hijacking
- Redirection to malicious sites

Recommendation:

- Encode all query string values on output
- Implement input validation for query parameters
- Use CSP to mitigate script execution risks

4. File Upload Vulnerability

Vulnerability Type: Insecure File Upload
Severity: High
Vector: Complaint file upload form
Observation: Renaming a `.der` file to `.zip` bypassed the content-type check

Description:

The application validates only the file extension during upload. By changing a `.der` file's extension to `.zip`, the system accepted the upload without verifying the MIME type or file content.

Impact:

- Remote Code Execution (RCE) if files are executed or processed server-side
- Malicious file hosting or web shell planting

Recommendation:

- Enforce server-side MIME type and content checks
- Use file scanning tools (e.g., ClamAV) on upload
- Store uploaded files outside the web root

5. Directory Traversal

Vulnerability Type: Path Traversal
Severity: High
Vector: Order confirmation print/download functionality
Payload: `ftp/legal.md`

Description:
After gaining admin access and completing an order, the print functionality was intercepted using Burp Suite. The `filename` parameter was manipulated to access files outside the web root. Access to `ftp/legal.md` was achieved, confirming the vulnerability.

Impact:

- Unauthorized access to sensitive files and configurations
- Information disclosure that could support further exploitation

Recommendation:

- Sanitize and validate file paths on the server

- Use whitelisting for allowable paths or file names
- Disable direct file access if not needed

Risk Matrix

Vulnerability	Risk Level	Exploitability	Business Impact
SQL Injection	High	Easy	Critical
Stored XSS	High	Moderate	High
Reflected XSS	Medium	Easy	Medium
File Upload Bypass	High	Moderate	Critical
Directory Traversal	High	Moderate	High

Recommendations Summary

Area	Recommended Action
Input Handling	Sanitize, validate, and encode all inputs and outputs
Authentication & Access	Implement least privilege and secure session management
File Upload	Validate MIME types, restrict file types, and scan uploads
Logging & Monitoring	Enable detailed logs for sensitive actions, and alert on anomalies
Patching & Hardening	Keep server software updated, and isolate critical services or paths
Security Headers	Implement CSP, X-Content-Type-Options, and X-Frame-Options

Conclusion

This test confirms the presence of **multiple high-severity vulnerabilities** within the OWASP Juice Shop application. While this environment is designed for training and education, the same types of vulnerabilities exist in many real-world applications. The techniques demonstrated here should guide developers and security teams in **understanding how attackers think** and in **hardening their own applications**.