Security Testing Report

Target Application: OWASP Juice Shop
Testing Type: Web Application Security Testing
Tools Used: OWASP ZAP 2.17.0
Tester Role: QA / Security Intern
Testing Date: 4th February 2026

1. Introduction

This report documents the results of security testing performed on the **OWASP Juice Shop** web application. The purpose of this testing was to identify common web application security vulnerabilities in alignment with the **OWASP Top 10** and to assess the application's security posture using both **passive scanning** and **manual testing techniques**.

OWASP Juice Shop is an intentionally vulnerable application designed for security learning and testing.

2. Scope of Testing

In Scope

- Input validation testing
- Cross-Site Scripting (XSS) testing
- Authentication and session handling checks
- Security configuration review
- Passive vulnerability scanning

Out of Scope

- Denial of Service (DoS)
- Advanced exploitation
- Source code review

3. Tools and Environment

Tool	Purpose
OWASP ZAP 2.17.0	Intercepting proxy, passive scanning
Browser (ZAP-configured)	Manual interaction

Tool	Purpose
Test URL	OWASP Juice Shop demo application

4. Testing Methodology

The following approach was used:

1. **Manual Exploration** of the application using OWASP ZAP's browser
2. **Passive Scanning** to identify configuration and header-related issues
3. **Manual Input Testing** using common payloads
4. **Authentication & Session Testing**
5. **Documentation of Findings** with severity and remediation suggestions

5. Vulnerability Findings

🔴 Vulnerability ID: SEC-001

Title: Missing Security Headers & Configuration Vulnerabilities
Category: Security Misconfiguration
Severity: Medium (Systemic Issues)

Description:
The application lacks multiple security headers and exhibits several configuration vulnerabilities, including:

- Missing Content Security Policy (CSP) Header
- Missing Strict-Transport-Security Header
- Missing X-Content-Type-Options Header
- Missing Anti-clickjacking Header
- Cross-Domain Misconfiguration (Systemic)
- Cross-Domain JavaScript Source File Inclusion (2 instances)
- Private IP Disclosure
- Timestamp Disclosure

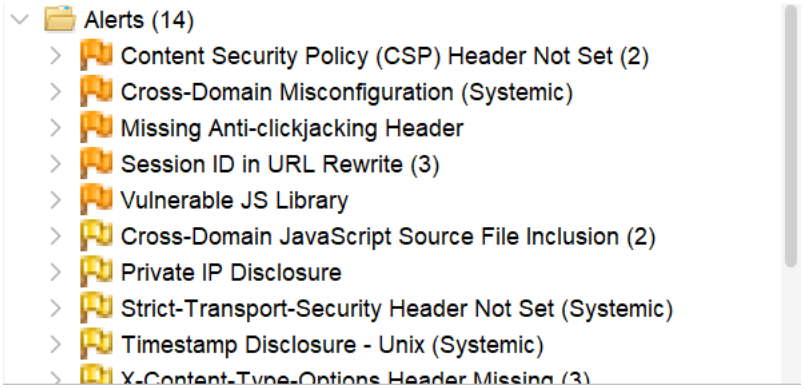
These misconfigurations increase the risk of client-side attacks, data leakage, and UI redressing.

Steps to Reproduce:

1. Perform a passive scan using OWASP ZAP.
2. Navigate to the Alerts tab.

3. Observe the listed configuration vulnerabilities.

Evidence:



Suggested Fix:
Configure the server to include the following HTTP headers:

- Content-Security-Policy: ...
- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY
- Ensure no sensitive internal IP addresses are disclosed.
- Validate and restrict cross-domain resource inclusions.

🔴 **Vulnerability ID: SEC-002**

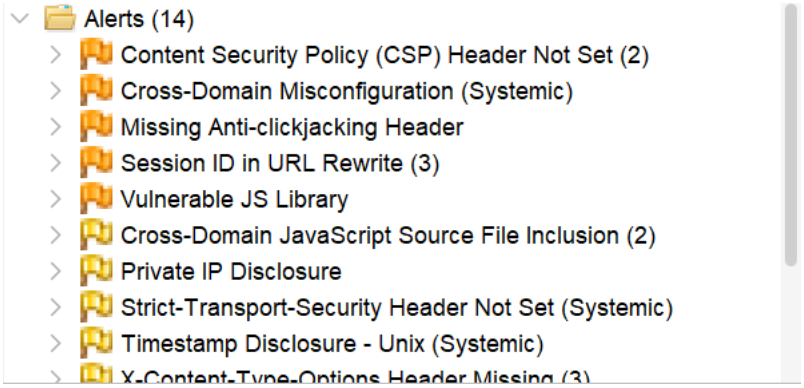
Title: Session ID in URL Rewrite
Category: Broken Access Control / Session Management
Severity: Medium

Description:
Session identifiers are being exposed in the URL via URL rewriting in at least 3 instances. This can lead to session fixation, session hijacking, and leakage of session tokens via logs or Referer headers.

Steps to Reproduce:

1. Browse the application while intercepting traffic with ZAP.
2. Observe session tokens appended in URLs.
3. Refer to ZAP alerts for "Session ID in URL Rewrite."

Evidence:



Suggested Fix:

- Use HTTP cookies for session management.
- Ensure session tokens are not passed in URLs.
- Implement secure session handling with HttpOnly and Secure flags.

🔴 Vulnerability ID: SEC-003

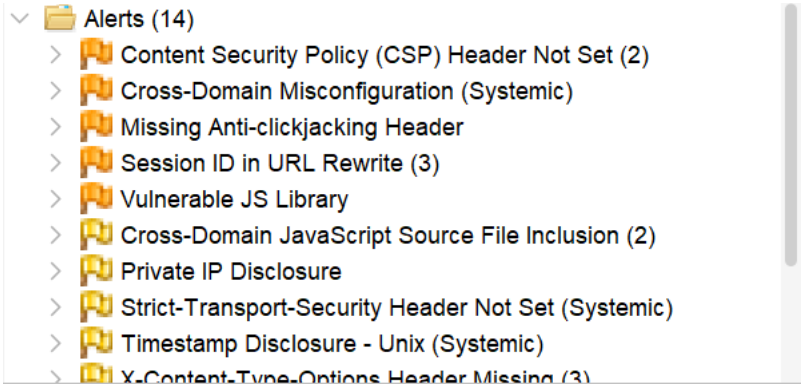
Title: Vulnerable JavaScript Library
Category: Using Components with Known Vulnerabilities
Severity: Medium

Description:
The application uses an outdated or vulnerable JavaScript library, which may expose the application to known exploits.

Steps to Reproduce:

1. Review the source code or ZAP alerts.
2. Identify the vulnerable library version.

Evidence:



Suggested Fix:

- Identify and update the JavaScript library to a secure version.

- Regularly audit and update third-party dependencies.

🔴 **Vulnerability ID: SEC-004**

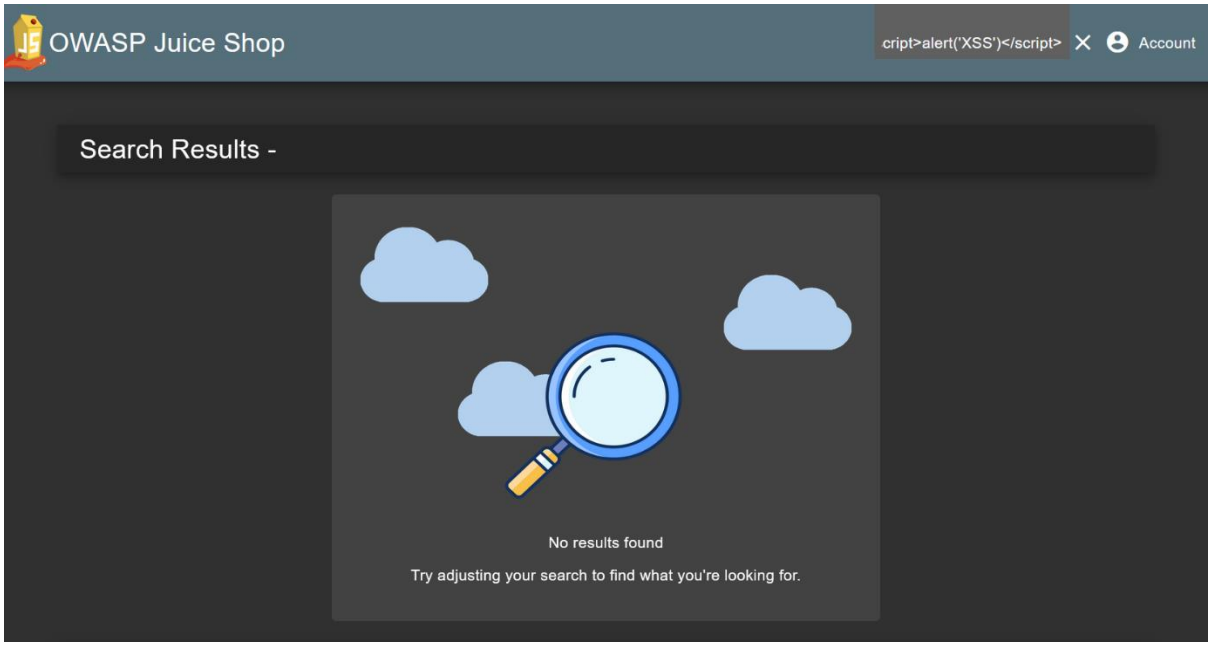
Title: Reflected Cross-Site Scripting (XSS)
Category: Cross-Site Scripting (XSS)
Severity: High

Description:
User input is directly reflected in the application's response without proper encoding or validation, allowing execution of arbitrary JavaScript.

Steps to Reproduce:

1. Navigate to a search or input field.
2. Enter the payload:
   <script>alert('XSS')</script>
3. Submit the input and observe the alert popup.

Evidence:



Suggested Fix:

- Implement strict output encoding (HTML, JavaScript contexts).
- Validate and sanitize all user inputs on the server side.
- Consider implementing a Content Security Policy (CSP).

🟡 **Vulnerability ID: SEC-005**

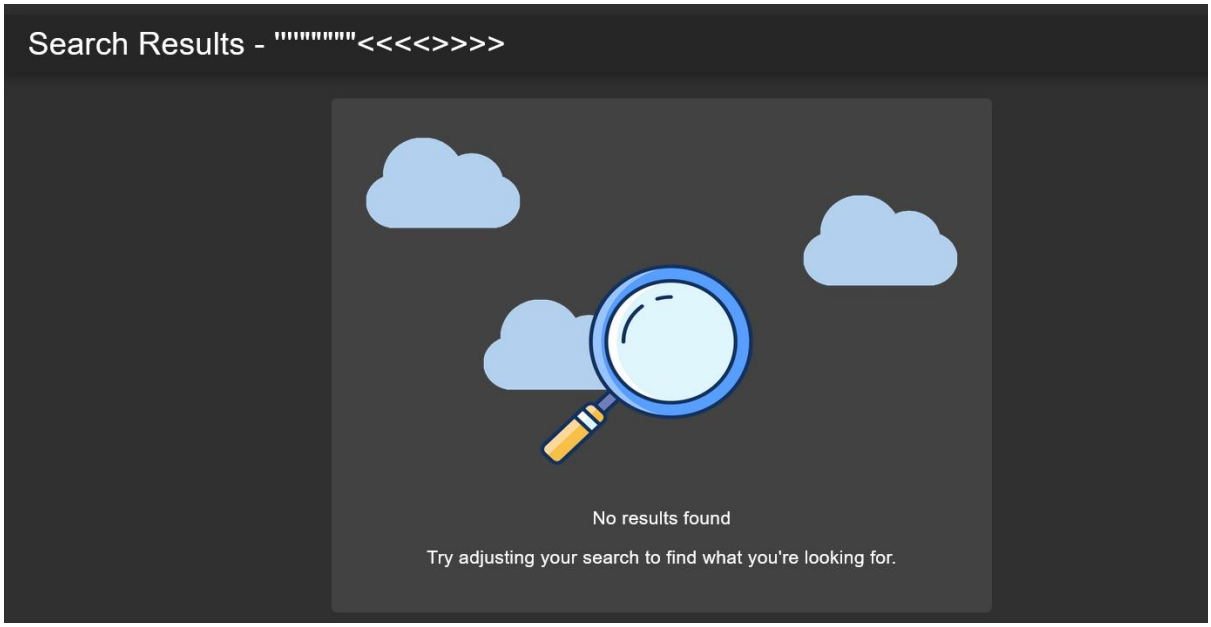
Title: Missing Input Validation in Search Functionality
Category: Input Validation
Severity: Medium

Description:
The search functionality does not properly validate or sanitize user input, allowing injection of script tags and special characters.

Steps to Reproduce:

1. Use the search bar.
2. Enter malicious or malformed input such as:
   """"<<>> or <script>alert('XSS')</script>
3. Observe that the input is reflected without sanitization.

Evidence:



Suggested Fix:

- Implement server-side input validation and sanitization.
- Use allow-list validation for expected characters.
- Apply contextual output encoding.

6. Authentication & Session Testing

Login / Logout Testing

- Logout invalidated session successfully

- Back button did not restore authenticated access

Result: Secure

Session Timeout Testing

- Session remained active after inactivity period

Result: Weak session timeout enforcement

Severity: Medium
Recommendation: Implement session expiration after defined idle time.

7. Summary of Findings

ID	Vulnerability	Severity
SEC-001	Missing Security Headers	Low
SEC-002	Cookie without HttpOnly	Medium
SEC-003	Weak Input Validation	Medium
SEC-004	Cross-Site Scripting (XSS)	High

8. Conclusion

The security assessment of the OWASP Juice Shop application identified multiple vulnerabilities ranging from **Low to High severity**. The most critical issues involved **Cross-Site Scripting (XSS)** and **session security weaknesses**. While the application serves as a learning platform, these findings highlight common real-world web application security risks.

Implementing the recommended fixes would significantly improve the overall security posture.

9. Recommendations

- Enforce strict input validation
- Implement proper output encoding
- Enable secure HTTP headers
- Secure cookies with HttpOnly and Secure flags
- Apply session timeout controls