**ZAP Security Scan Report**

**Generated by: ZAP by Checkmarx**

**Scan Date: Tue, 7 Jan 2025, at 10:14:57**

**ZAP Version: 2.15.0**

**Report Generated for:** [**http://localhost:3000**](http://localhost:3000)

**Security Assessment Report**

**Target**: OWASP Juice Shop (<http://localhost:3000>)

**Date**: January 20, 2025

# **1. Overview**

This report documents a detailed security assessment conducted on the **OWASP Juice Shop**, a purposely vulnerable application hosted on localhost:3000. The evaluation utilized the **ZAP by Checkmarx** and **Nikto** tools to uncover vulnerabilities and potential misconfigurations in the application’s design and deployment.

The OWASP Juice Shop serves as a valuable learning resource for understanding web application security but contains several vulnerabilities that must be addressed in real-world scenarios to protect sensitive information and prevent exploitation.

# **2. Objectives**

The primary objectives of this security assessment are:

1. To identify vulnerabilities in the OWASP Juice Shop application and its underlying web server.
2. To analyze issues related to authentication, data exposure, misconfigured headers, and sensitive file accessibility.
3. To recommend best practices for mitigating identified risks and enhancing application security.

# **3. Findings**

## **3.1. General Observations**

* **Host Information**:
  + **IP Address**: 127.0.0.1
  + **Port**: 3000
  + **Server Banner**: No banner retrieved (reduces information leakage but may limit specific vulnerability identification).
* **Headers**:
  + The **Access-Control-Allow-Origin** header is set to \*, which permits unrestricted cross-origin requests.
  + The **X-Content-Type-Options** header is missing, increasing the risk of MIME-type sniffing attacks.

## **3.2. Key Vulnerabilities**

### **3.2.1. High Risk**

1. **Cloud Metadata Potentially Exposed**
   * **Description**: The endpoint /latest/meta-data/ exposes sensitive cloud metadata, which could be exploited for unauthorized access to cloud resources.
   * **Recommendation**: Ensure this endpoint is not publicly accessible. Configure firewalls or access controls to restrict access.

### **3.2.2. Medium Risk**

1. **Content Security Policy (CSP) Header Not Set**
   * **Description**: The application does not implement a CSP header, making it vulnerable to Cross-Site Scripting (XSS) attacks.
   * **Recommendation**: Define and enforce a CSP header to restrict resource loading and prevent script injection attacks.
2. **Session ID in URL Rewrite**
   * **Description**: Session IDs are included in the URL for requests to the WebSocket endpoint (/socket.io/). This practice exposes session tokens, potentially leading to session hijacking.
   * **Recommendation**: Use secure, HttpOnly cookies to store session IDs and avoid passing them in URLs.
3. **Cross-Domain Misconfiguration**
   * **Description**: The application’s CORS policy allows unrestricted cross-origin requests, which attackers can exploit to steal session data or perform unauthorized actions.
   * **Recommendation**: Restrict CORS to trusted origins only.
4. **Missing Anti-Clickjacking Header**
   * **Description**: The X-Frame-Options header is not set, leaving the application vulnerable to clickjacking attacks.
   * **Recommendation**: Set the X-Frame-Options header to DENY or SAMEORIGIN.

### **3.2.3. Low Risk**

1. **Private IP Disclosure**
   * **Description**: Internal/private IP addresses are exposed in the endpoint /rest/admin/application-configuration.
   * **Recommendation**: Restrict API endpoint access and ensure sensitive data like private IPs are not exposed in responses.
2. **Cross-Domain JavaScript Source File Inclusion**
   * **Description**: The application allows cross-domain JavaScript inclusion, which could be exploited to load malicious scripts.
   * **Recommendation**: Ensure that JavaScript files are sourced only from trusted domains.
3. **Timestamp Disclosure in Public Files**
   * **Description**: Unix timestamps are exposed in main.js, which may aid attackers in constructing timeline-based attacks.
   * **Recommendation**: Avoid including unnecessary timestamps in public files.

### **3.2.4. Informational**

1. **Suspicious Comments in Source Code**
   * **Description**: Comments in main.js may reveal sensitive information or implementation details.
   * **Recommendation**: Review and remove comments that disclose application logic or sensitive details.
2. **Modern Web Application Detection**

* **Description**: The application’s modern framework and design could make it a target for specific vulnerabilities.
* **Recommendation**: Regularly update dependencies and perform thorough security testing of frameworks and libraries.

# **4. Recommendations**

## **4.1. Server Hardening**

* Secure sensitive files such as /backup.cer, /dump.jks, and /database.pem. Remove files that are not essential for application functionality.
* Restrict access to directories listed in robots.txt and use authentication mechanisms for critical paths.

## **4.2. Header Configuration**

* Add missing security headers:
  + **X-Content-Type-Options: nosniff**
  + **Content-Security-Policy** to define trusted sources.
  + **Strict-Transport-Security** to enforce HTTPS.

## **4.3. CORS and Session Management**

* Implement a restrictive CORS policy, allowing requests only from trusted origins.
* Use HttpOnly and Secure cookies for session management and avoid exposing session IDs in URLs.

## **4.4. Regular Testing and Maintenance**

* Perform regular security scans using tools like Nikto, OWASP ZAP, and Burp Suite to identify emerging threats.
* Keep all frameworks, libraries, and dependencies up to date to mitigate known vulnerabilities.

# **5. Conclusion**

The OWASP Juice Shop at http://localhost:3000 demonstrates vulnerabilities typical of modern web applications, including missing security headers, sensitive data exposure, and session management flaws.

By implementing the recommendations outlined in this report, the application’s security posture can be significantly enhanced, reducing the likelihood of exploitation and ensuring it remains a valuable educational tool.

# **6. References**

1. OWASP Secure Headers Project
2. [CWE-530: Exposure of Sensitive Information](https://cwe.mitre.org/data/definitions/530.html)
3. [MDN Content-Security-Policy](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy)
4. Nikto Documentation