**Penetration Testing Report**

**Engagement Date:**

**21/01/2025**

**Application: OWASP Juice Shop**

**Host: localhost:3000**

# **1. Executive Summary**

This report documents the findings from a penetration testing engagement on the OWASP Juice Shop application, hosted on localhost:3000. The assessment combined both manual testing and the use of automated tools to uncover potential vulnerabilities. These tools included Nikto, ZAP Proxy, Nmap, Metasploit, and Hydra. The testing focused on identifying common security flaws such as misconfigurations, weak passwords, exposure of sensitive information, and vulnerabilities that could be exploited to gain unauthorized access or perform attacks.

## **Key Findings:**

* **Critical**:
  + **Cloud Metadata Exposure:** Sensitive cloud metadata was exposed, risking unauthorized access to cloud resources.
* **High**:
  + **Missing Content Security Policy (CSP):** Absence of a CSP header, making the application vulnerable to XSS attacks.
  + **Session ID in URL Rewrite:** Session IDs exposed in URLs for WebSocket connections increase the risk of session hijacking.
  + **Insecure CORS Configuration:** Improper CORS configuration allows unauthorized access to application resources from malicious domains.
  + **Missing Anti-Clickjacking Header:** The absence of the X-Frame-Options header exposes the app to clickjacking attacks.
* **Medium**:
  + **Private IP Disclosure:** Internal/private IP addresses exposed in error messages or response headers.
  + **Brute-Force Vulnerability:** Weak credentials allowed successful brute-force attacks against the 'admin' account using common passwords.
* **Low**:
  + **Cross-Domain JavaScript Source File Inclusion:** Potential risk for loading scripts from malicious or untrusted domains.
  + **Timestamp Disclosure:** Unix timestamps visible in publicly accessible files could assist attackers in time-based attacks.
* **Informational**:
  + **Suspicious Comments in Source Code:** The presence of detailed comments in the source code may inadvertently disclose sensitive implementation details.
  + **Modern Web Application Detection:** The use of modern web frameworks could potentially introduce new attack vectors.

# **2. Methodology**

The testing was carried out using a combination of automated tools, manual techniques, and vulnerability databases. Below is the detailed methodology followed:

## **Tools Used:**

* **Nikto:** A web server scanner to identify misconfigurations and vulnerabilities.
* **ZAP Proxy:** A comprehensive tool for both manual and automated testing of web application security.
* **Nmap:** A network scanner to identify open ports and services running on the target system.
* **Metasploit:** A framework used for targeted exploitation of vulnerabilities.
* **Hydra:** A brute-force tool used to test weak credentials on the login page.

## **Techniques Applied:**

* **Vulnerability Scanning:** Automated scans using Nikto and ZAP Proxy were conducted to find common vulnerabilities like outdated software, missing headers, etc.
* **Manual Testing:** Interactive use of ZAP Proxy was employed to identify vulnerabilities like Cross-Site Scripting (XSS) and misconfigurations.
* **Service Enumeration:** Nmap was used to identify open ports, services, and any potential services that could be attacked.
* **Targeted Exploitation:** Metasploit was utilized to exploit discovered vulnerabilities and assess the system’s response to these exploits.
* **Brute-Force Testing:** Hydra was deployed to test the application’s resilience to brute-force attacks using common password lists.

# **3. Findings**

## **3.1. Nikto Scan Results:**

* **Missing Security Headers:** The application lacked several security headers such as X-Content-Type-Options, increasing the likelihood of MIME-sniffing attacks.
* **Sensitive File Exposure:** Backup files such as backup.cer and dump.jks were accessible, which could expose sensitive data such as encryption keys or configurations.
* **Directory Listing in Robots.txt:** The robots.txt file exposed unnecessary directory paths, potentially revealing sensitive endpoints to attackers.

## **3.2. ZAP Proxy Scan Results:**

* **Cloud Metadata Exposure:** A vulnerability was discovered where sensitive cloud metadata was exposed. This could provide attackers with detailed information regarding the cloud infrastructure and its components.
* **Missing Content Security Policy (CSP) Header:** The absence of a strict CSP header was found, making the app vulnerable to XSS and data injection attacks.
* **Session ID in URL:** WebSocket connections transmitted session IDs in the URL, which increases the risk of session hijacking, as attackers could intercept these IDs.
* **Insecure CORS Configuration:** The Cross-Origin Resource Sharing (CORS) policy allowed requests from any origin, which could lead to unauthorized access by malicious third-party websites.
* **Missing Anti-Clickjacking Header:** The absence of the X-Frame-Options header made the app vulnerable to clickjacking attacks.
* **Private IP Disclosure:** Internal IP addresses were exposed, revealing potential targets within the network.
* **Cross-Domain JavaScript Source File Inclusion:** The application was found to be vulnerable to including JavaScript files from external sources, which could be leveraged to execute malicious code.
* **Timestamp Disclosure:** Unix timestamps were embedded in public-facing files, which could help attackers with timing attacks.

## **3.3. Nmap Scan Results:**

* **Open Port 3000/tcp:** The service was running on port 3000, which was identified as open.
* **Server Banner Information:** The server’s banner revealed details about the underlying application and web server software, which could aid attackers in identifying potential vulnerabilities.

## **3.4. Metasploit Framework:**

* **Confirmed HTTP Version and Server Information:** Metasploit was used to retrieve server details and confirm the HTTP version being used, providing further insight into possible exploits.
* **Exploitable Vulnerabilities:** Based on the server information obtained, no active vulnerabilities were confirmed but several could be explored further with a targeted attack.

## **3.5. Hydra Brute-Force Attack Results:**

* **Successful Brute-Force Attack on Admin Account:** Hydra successfully brute-forced the admin user account with several weak passwords:
  + 123456
  + password
  + 12345678
  + 1234
* **Implication:** This highlights weak password policies, as attackers could gain unauthorized access to critical accounts.

# **4. Recommendations**

Based on the findings, the following remediation actions are recommended to improve the security posture of the OWASP Juice Shop application:

## **4.1. Implement and Enforce Content Security Policy (CSP):**

* Define and enforce a strong Content Security Policy (CSP) to prevent XSS attacks and reduce the risk of malicious code execution.

## **4.2. Secure Session Management:**

* Ensure session cookies are configured with HttpOnly and Secure attributes.
* Avoid passing session IDs via URL, and instead use secure, HTTP-only cookies for session management.
* Implement additional protections against session fixation and session hijacking.

## **4.3. Configure CORS Properly:**

* Restrict Cross-Origin Resource Sharing (CORS) to only trusted origins to prevent unauthorized access from malicious websites.

## **4.4. Address Cloud Metadata Exposure:**

* Securely configure cloud metadata and ensure that sensitive data is not exposed publicly. Implement cloud access controls and security monitoring.

## **4.5. Remove or Secure Sensitive Files:**

* Delete unnecessary files like backup.cer and dump.jks. If such files are needed, ensure they are properly secured with access restrictions and encryption.

## **4.6. Implement Anti-Clickjacking Measures:**

* Set the X-Frame-Options header to DENY or SAMEORIGIN to prevent clickjacking attacks.

## **4.7. Remove Insecure Comments:**

* Review and sanitize all comments in the source code to ensure that no sensitive information or internal implementation details are exposed to the public.

## **4.8. Strengthen Password Policies:**

* Implement stronger password policies, including a minimum length, complexity requirements, and password expiration.
* Enforce account lockout policies to protect against brute-force attacks.
* Consider adding multi-factor authentication (MFA) for an extra layer of security.

## **4.9. Regular Security Audits:**

* Conduct regular penetration tests and vulnerability scans to detect and address emerging threats and vulnerabilities.

## **4.10. Keep Software Updated:**

* Regularly update the web application, web frameworks, and all dependencies to patch known vulnerabilities and security issues.

# **5. Disclaimer**

This penetration testing report provides findings based on the tests conducted within the scope of the engagement. It is not an exhaustive assessment and may not cover every vulnerability in the system. Further testing and regular assessments are recommended to continuously improve the security posture of the application.