Web Application Security Assessment Report

Application: OWASP Juice Shop

Prepared For: Project Portfolio

Prepared By: Sumanth Y

Date of Assessment: October 16, 2025

---

1. Executive Summary

A comprehensive security assessment of the OWASP Juice Shop web application was performed to identify and document security vulnerabilities. The methodology involved a combination of automated scanning with OWASP ZAP and manual penetration testing using Burp Suite.

The assessment uncovered multiple vulnerabilities, including one **Critical** risk, two **High** risk, and several **Medium** risk configuration weaknesses. The most severe finding is an **Authentication Bypass via SQL Injection**, which permits a complete takeover of the administrator account without credentials. Other significant findings include **Cross-Site Request Forgery (CSRF)**, allowing for account defacement, and **Reflected Cross-Site Scripting (XSS)**, which can lead to session hijacking.

The automated scan corroborated the SQL Injection finding and identified several medium-risk security misconfigurations, such as missing security headers and the use of a vulnerable JavaScript library. This report details these findings and provides actionable recommendations for remediation. The immediate priority should be to address the critical SQL Injection flaw.

---

2. Scope and Methodology

2.1 Scope

The scope of this assessment was the OWASP Juice Shop web application hosted in a local test environment.

- **URL:** `http://localhost:3000`

2.2 Methodology

A gray-box testing approach was utilized, where testing was conducted with the knowledge of a standard user.

- **Automated Analysis:** The application was scanned using OWASP ZAP (version 2.16.1) to identify common vulnerabilities and misconfigurations.
- **Manual Analysis:** Manual penetration testing was performed using Burp Suite to identify vulnerabilities requiring contextual understanding, such as authentication flaws, complex injection, and business logic issues.
- **Framework:** Findings are categorized and rated based on their potential impact, drawing from the OWASP Top 10 framework.

---

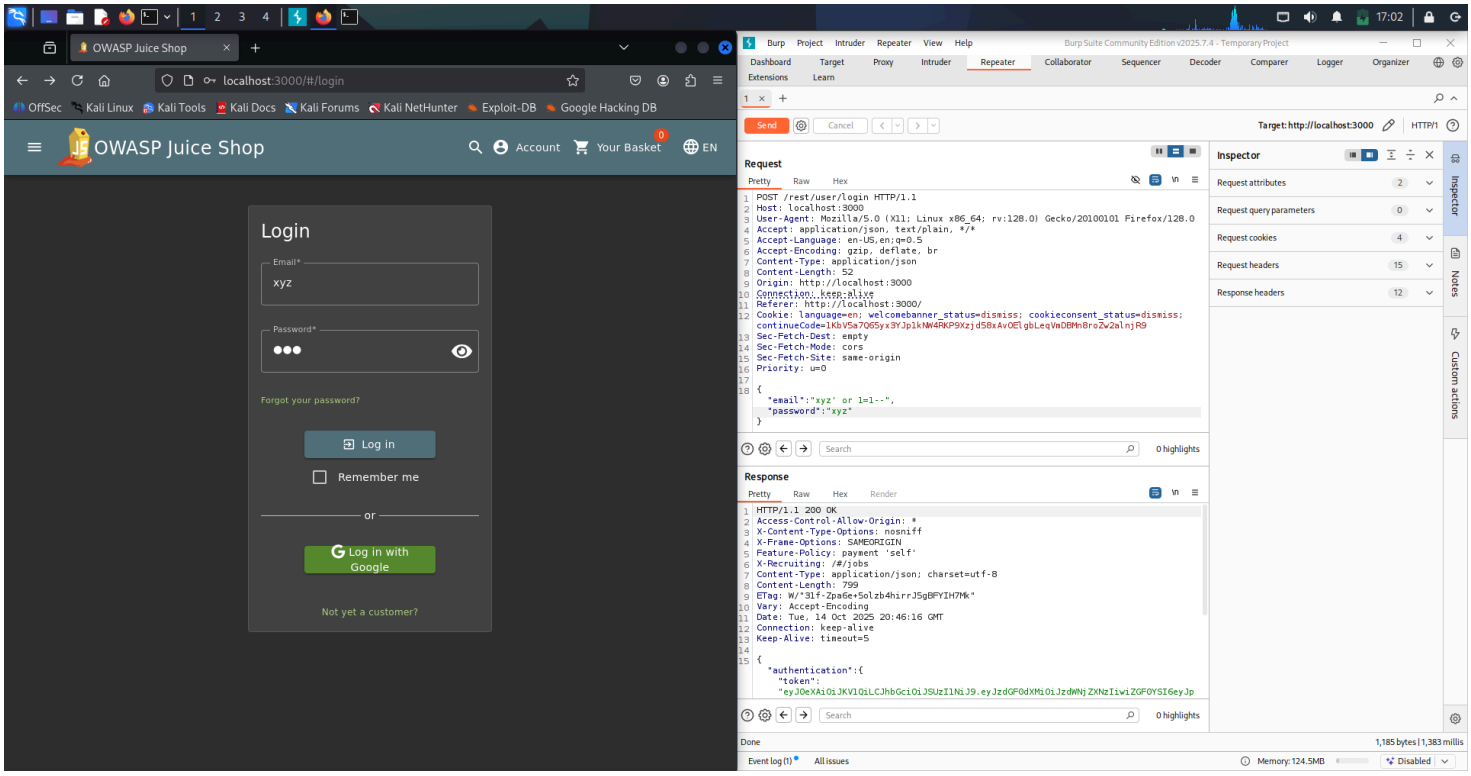
3. Findings and Recommendations

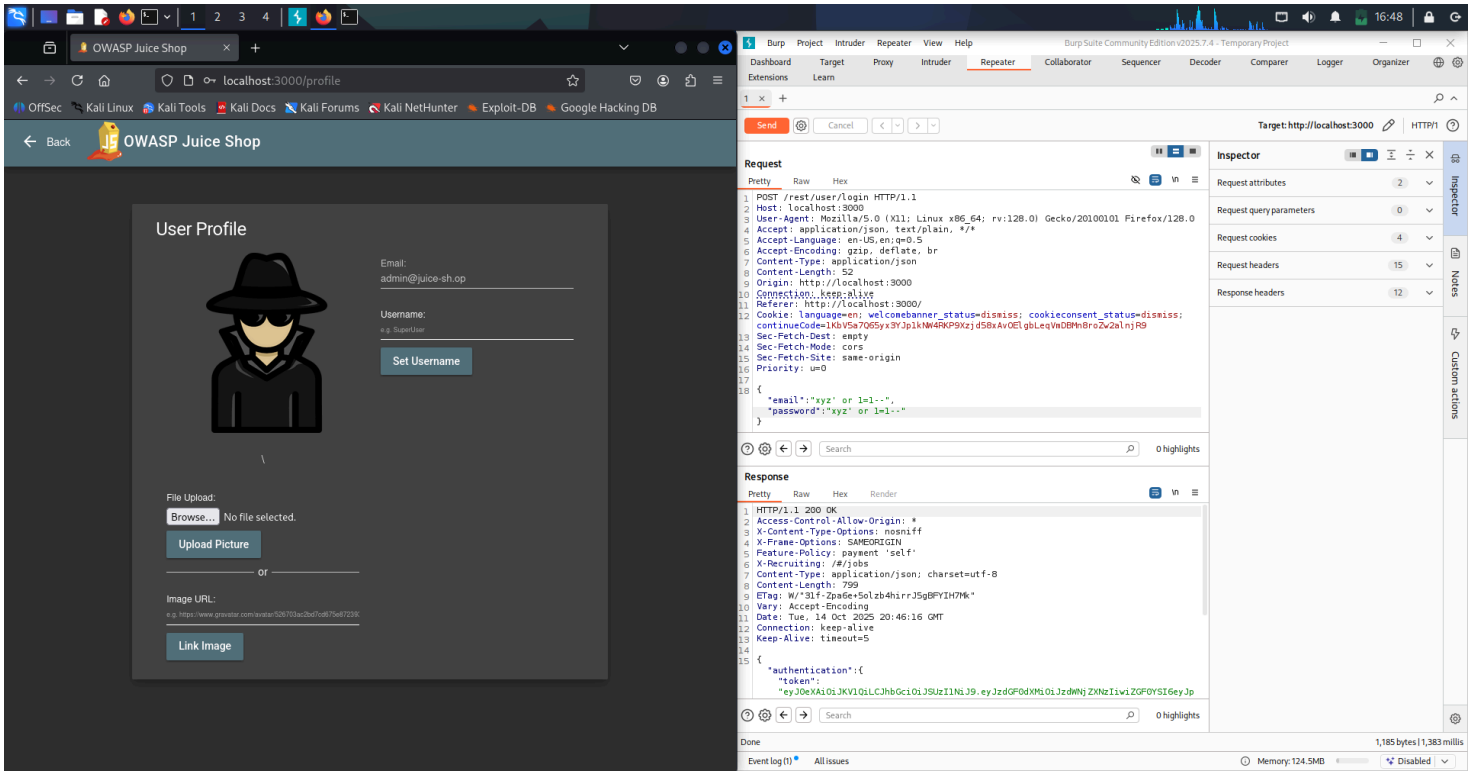
The following section details the vulnerabilities identified during the assessment, ordered by severity.

Finding 1: Authentication Bypass via SQL Injection

- **Risk Rating: Critical**
- **Description:** The application's login function is vulnerable to SQL Injection. The `email` parameter in the login request does not properly sanitize user input, allowing an attacker to manipulate the underlying SQL query to bypass the password check.
- **Impact:** This vulnerability can be exploited to gain unauthorized access to any user account, including the administrator's, without a valid password. This leads to a complete compromise of the application, its data, and all user accounts.
- **Steps to Reproduce:**
    1. Intercept the `POST /rest/user/login` request using Burp Suite.
    2. In the JSON request body, modify the `email` parameter to the payload: `' or 1=1--`.
    3. Forward the request. The server responds with a valid authentication token for the administrator account.

- **Evidence:**

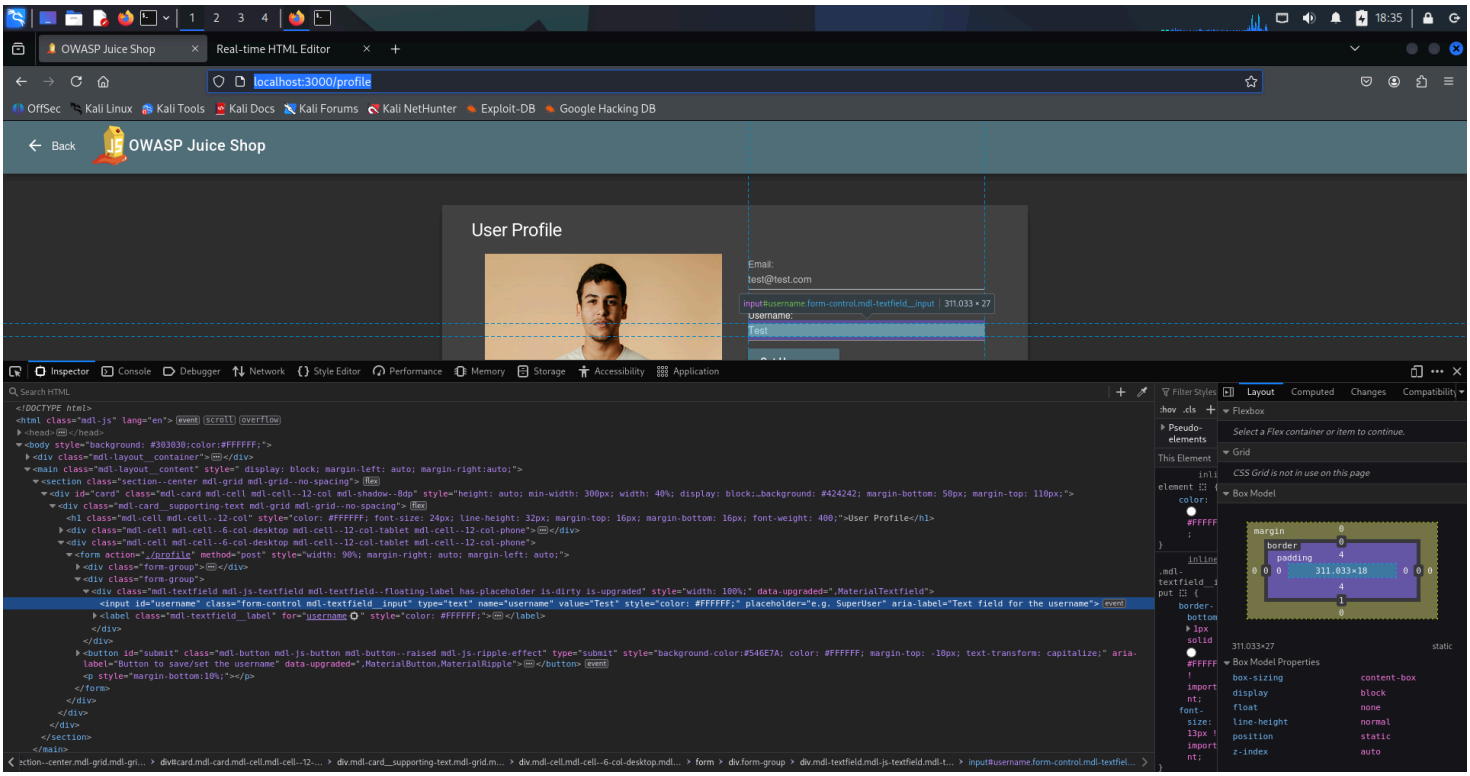

OWASP Juice Shop
localhost:3000/#/login
Login
Email*
xyz
Password*
Forgot your password?
Log in
Remember me
or
Log in with Google
Not yet a customer?
Request
POST /rest/user/login HTTP/1.1
{
"email":"xyz' or 1=1--",
"password":"xyz"
}
Response
HTTP/1.1 200 OK
{
"authentication":{
"token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJp

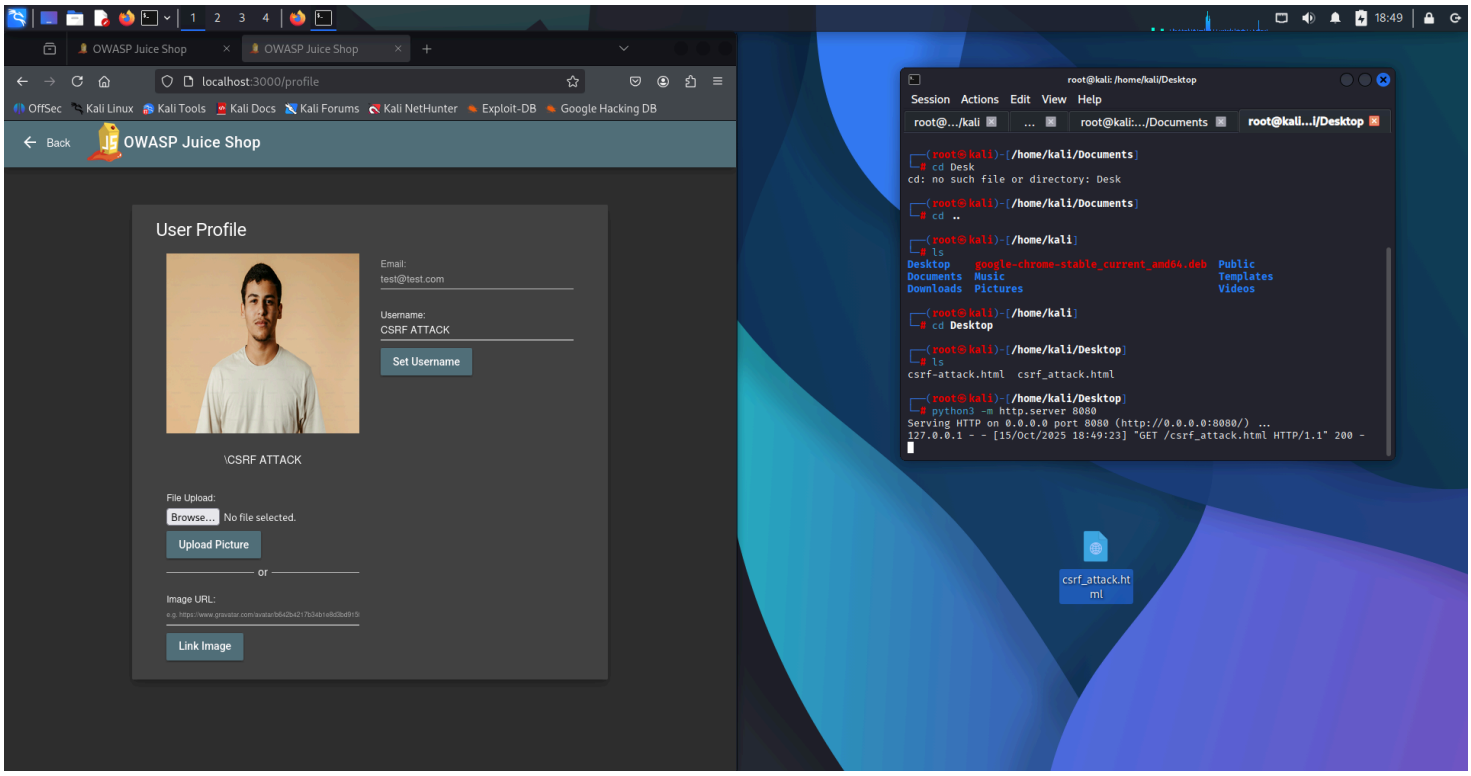

OWASP Juice Shop
localhost:3000/profile
Back
OWASP Juice Shop
User Profile
Email:
admin@juice-sh.op
Username:
Set Username
File Upload:
Browse...
No file selected.
Upload Picture
or
Image URL:
Link Image
Request
POST /rest/user/login HTTP/1.1
{
"email":"xyz' or 1=1--",
"password":"xyz' or 1=1--"
}
Response
HTTP/1.1 200 OK
{
"authentication":{
"token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJp

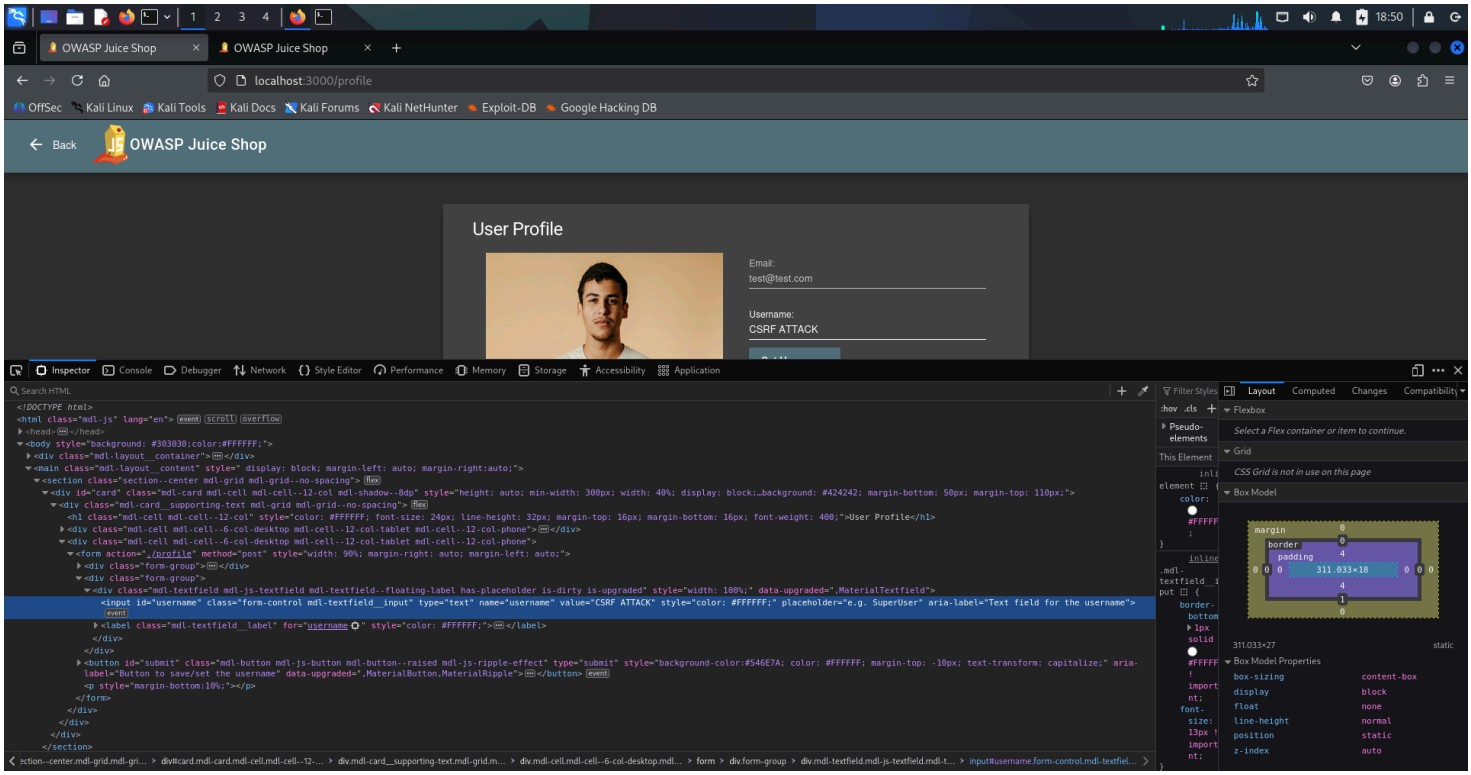
- **Remediation:** Implement **parameterized queries** (also known as prepared statements) for all database interactions. This ensures that user input is always treated as data and never as executable code.

Finding 2: Cross-Site Request Forgery (CSRF) on Username Update

- **Risk Rating: High**
- **Description:** The "Set Username" function on the user profile page is vulnerable to Cross-Site Request Forgery. The request to change the username lacks a unique anti-CSRF token, meaning the application cannot verify if the action was intentionally performed by the user.
- **Impact:** An attacker can create a malicious webpage that forces a logged-in victim's browser to submit a request to change their username to an attacker-controlled value. This can be used for account defacement and impersonation.
- **Steps to Reproduce:**
    1. Create a malicious HTML page with an auto-submitting form targeting the username update endpoint.
    2. Host the page on a local server.
    3. When a logged-in user visits the malicious page, their username is changed without their consent.
- **Evidence:**



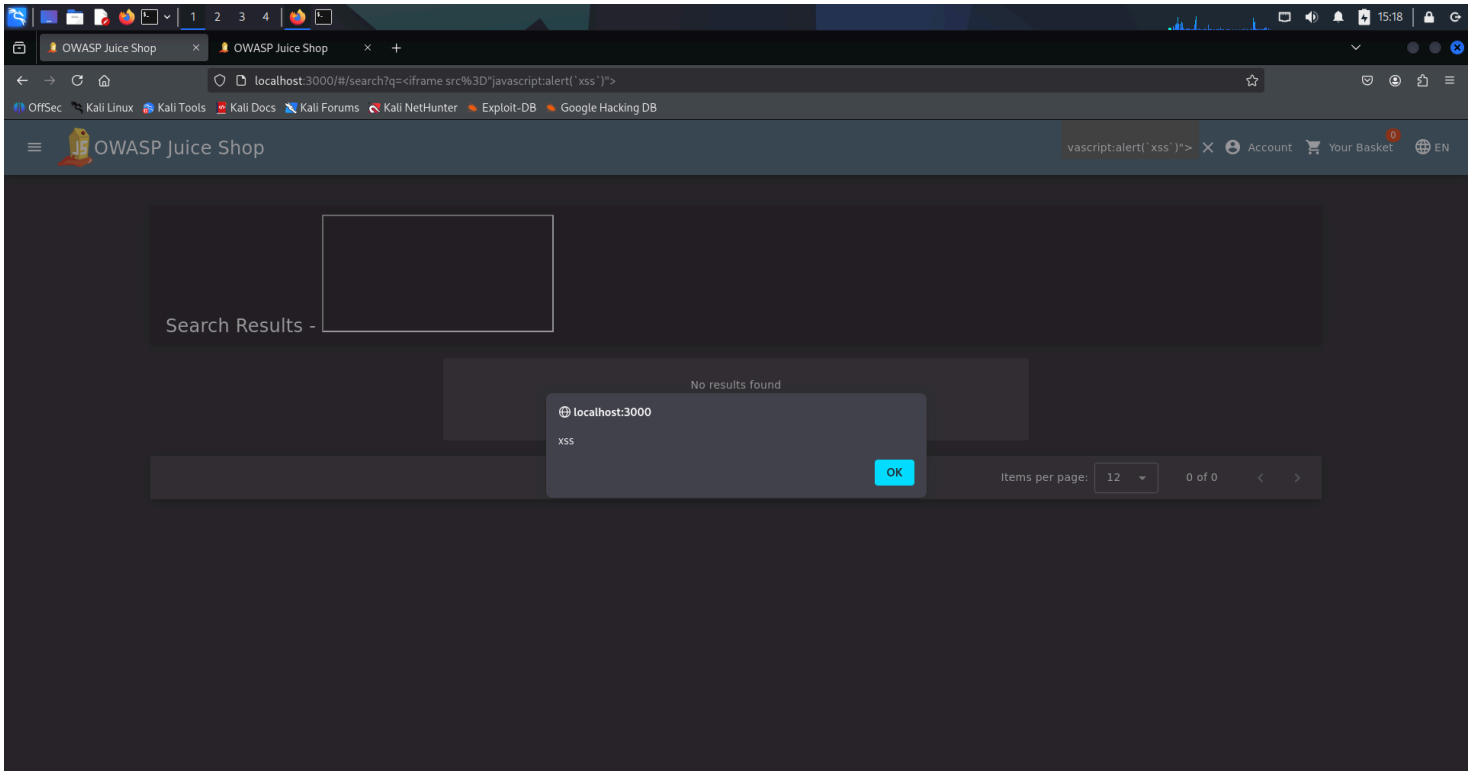




- **Remediation:** Implement a synchronized token pattern (anti-CSRF tokens). The server should generate a unique, unpredictable token for each user session and embed it in forms. This token must be validated upon every state-changing request.

Finding 3: Reflected Cross-Site Scripting (XSS)

- **Risk Rating: High**
- **Description:** The application is vulnerable to Reflected XSS in the order tracking page. User input from the `id` URL parameter is rendered directly into the page's HTML without proper output encoding.
- **Impact:** An attacker can craft a malicious link containing a JavaScript payload. If a victim clicks this link, the script will execute in their browser, allowing the attacker to hijack their session, perform unauthorized actions, or redirect them to malicious sites.
- **Steps to Reproduce:**
    1. Navigate to the following URL: `http://localhost:3000/#/search?q=<iframe src="javascript:alert('xss')">`.
    2. A JavaScript alert box will appear, confirming script execution.
- **Evidence:**



- **Remediation:** Implement context-aware output encoding for all user-supplied data before it is rendered in the HTML.

Finding 4: Insecure Security Header Configuration

- **Risk Rating: Medium**
- **Description:** The automated scan revealed that the application is missing several important HTTP security headers.
    - **Content Security Policy (CSP) Header Not Set:** The lack of a CSP makes the application more susceptible to XSS and data injection attacks.
    - **Missing Anti-clickjacking Header:** The absence of an `X-Frame-Options` header or `frame-ancestors` CSP directive exposes the application to Clickjacking attacks.
    - **X-Content-Type-Options Header Missing:** This allows older browsers to perform MIME-sniffing, which could lead to security issues.
- **Remediation:** Configure the web server to include these HTTP headers in all responses to enforce browser-level security policies.

Finding 5: Use of Component with Known Vulnerabilities

- **Risk Rating: Medium**
- **Description:** The application uses jQuery version 2.2.4, a JavaScript library with multiple known public vulnerabilities. These vulnerabilities include Cross-Site Scripting, as documented in CVE-2020-11022 and CVE-2015-9251.
- **Remediation:** Update the jQuery library to the latest stable version to mitigate all known vulnerabilities associated with the current version.

---

4. Conclusion

The security assessment of the OWASP Juice Shop application identified one critical, two high, and several medium-risk vulnerabilities. The findings demonstrate significant security gaps, particularly the SQL Injection flaw that allows for a complete system compromise. It is strongly recommended that the development team prioritize fixing the vulnerabilities detailed in this report, starting with the critical and high-risk issues, to protect the application and its users.