Task -1 WEB APPLICATION SECURITY TESTING

Confidential Security Assessment Report

Client: OWASP Juice Shop

Assessor: Rohit Pawar

Date: November 6, 2025

Target: http://localhost:3000

Classification: Confidential

---

1. Executive Summary

A security assessment was conducted on the OWASP Juice Shop application. The test identified four major vulnerabilities, including one Critical, two High, and one Medium risk finding.

The findings demonstrate significant weaknesses in authentication, session management, and input validation. The most critical issue is an Authentication Bypass via SQL Injection, which allows an attacker to gain full administrator access with no prior knowledge.

Other high-impact issues include Cross-Site Request Forgery (CSRF) and Sensitive Data Exposure, which allow for user account takeover. Immediate remediation is required to protect the application and its users.

---

2. Risk Rating Methodology

This report uses a standard risk model to classify vulnerabilities based on their potential Impact and the Likelihood of an attacker exploiting them.

- **Critical:** High Impact, High Likelihood. (Requires immediate remediation).
- **High:** High Impact, Medium Likelihood OR Medium Impact, High Likelihood. (Requires prompt remediation).
- **Medium:** Medium Impact, Medium Likelihood. (Remediation should be scheduled).
- **Low:** Low Impact, Low Likelihood. (Remediation as resources permit).

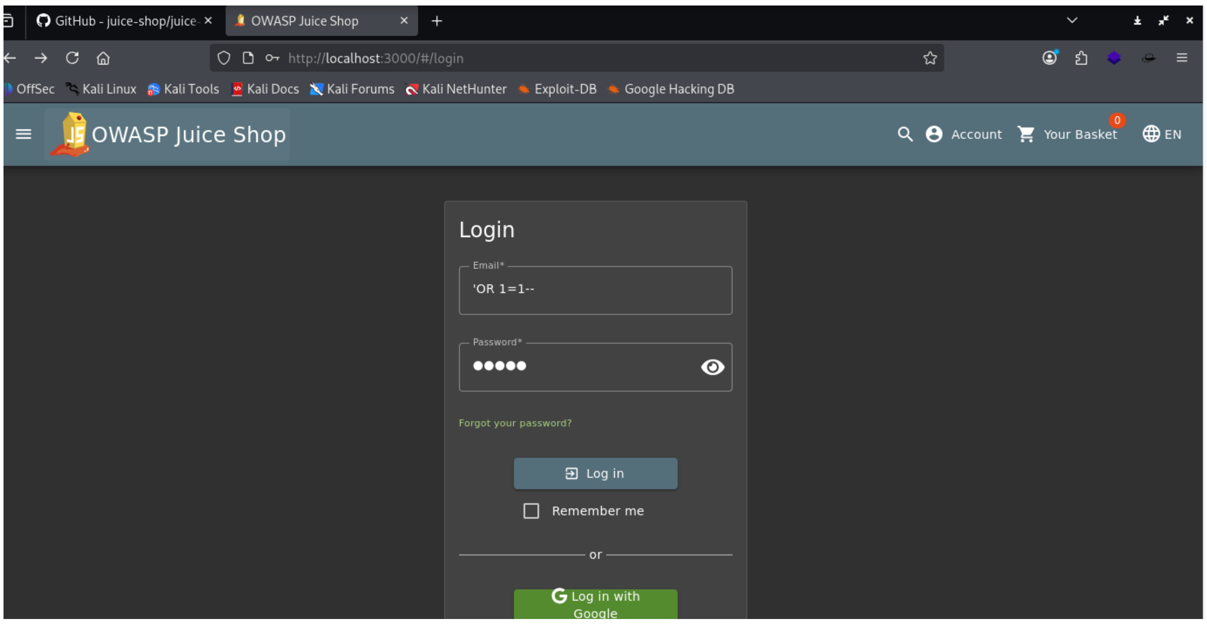
3. Technical Findings

This section details the vulnerabilities identified, their risk ratings, and step-by-step proof of concept.

Finding 3.1: SQL Injection - Authentication Bypass

- **Risk Rating:** Critical
- **OWASP Top 10:** A07:2021 - Identification and Authentication Failures
- **Description:** The application's login form is vulnerable to SQL Injection. An attacker can input a specially crafted SQL query into the email field to bypass all authentication checks and log in as any user, including the administrator.
- **Proof of Concept (PoC):**
    1. The attacker navigates to the /login page (SQL.png).
    2. In the Email field, the payload `'OR 1=1--` is entered.
    3. Any value is entered in the Password field.
    4. Upon clicking "Log in", the attacker is successfully logged in as the administrator

- **Screenshots:**

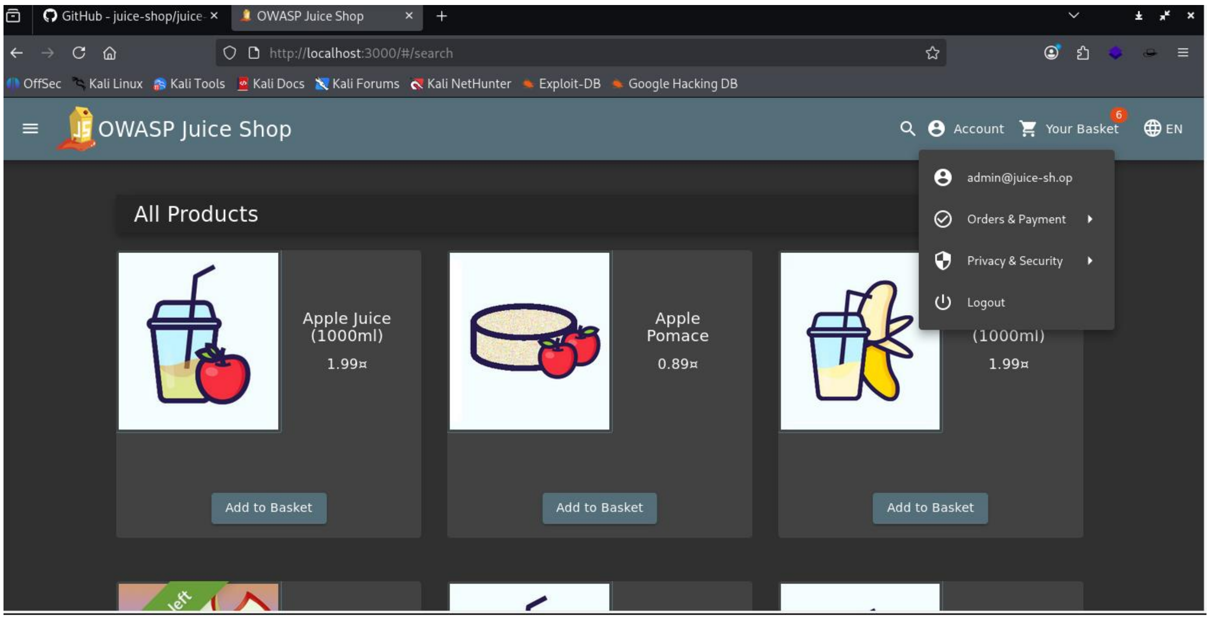


http://localhost:3000/#/login

Login

Email*: 'OR 1=1--

Password*: ●●●●●

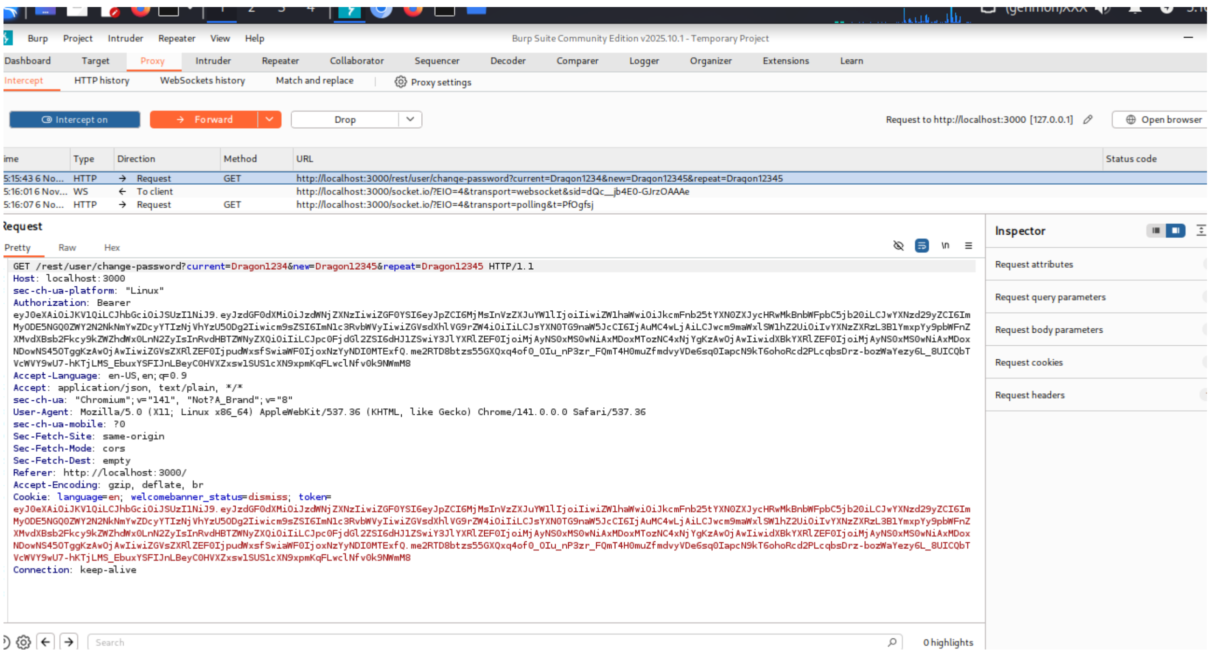


- **Impact:** Full, unauthorized administrative access to the application, leading to a complete system compromise.

Finding 3.2: Cross-Site Request Forgery (CSRF)

- **Risk Rating:** High
- **OWASP Top 10: A01:2021 - Broken Access Control**
- **Description:** The password change function lacks anti-CSRF tokens. An anti-CSRF token is a unique, unpredictable value that verifies a request was intentionally made by the user. Its absence allows an attacker to force a logged-in user to perform actions they did not intend.
- **Proof of Concept (PoC):**
  1. As seen in the Burp Suite request (TP4.png), the password change request is a simple GET request.
  2. The request parameters (current, new, repeat) are the only values needed.
  3. There is no unique anti-CSRF token present to validate the request's origin.

- **Screenshots:**


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.10.1 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://localhost:3000 [127.0.0.1]
Open browser
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Pretty Raw Hex
Search
0 highlights

```
ime            Type   Direction    Method   URL                                                                                             Status code
5:15:43 6 No...  HTTP   → Request     GET      http://localhost:3000/rest/user/change-password?current=Dragon1234&new=Dragon12345&repeat=Dragon12345
5:16:01 6 Nov... WS     ← To client            http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=dQc__jb4E0-GJrzOAAAe
5:16:07 6 No...  HTTP   → Request     GET      http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PfOgfsj
```

Request

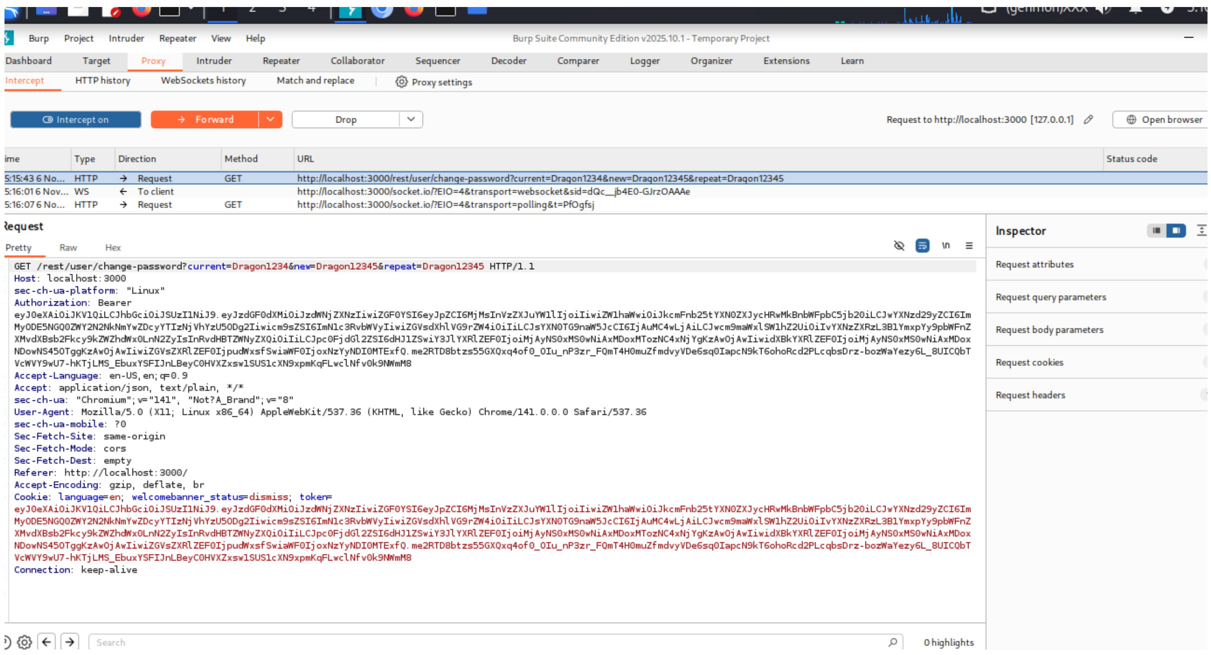
```
GET /rest/user/change-password?current=Dragon1234&new=Dragon12345&repeat=Dragon12345 HTTP/1.1
Host: localhost:3000
sec-ch-ua-platform: "Linux"
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJkcmFnb25tYXN0ZXJycHRwMkBnbWFpbC5jb20iLCJwYXNzd29yZCI6Im
MyODE5NGQ0ZWY2N2NkNmYwZDcyYTIzNjVhYzU5ODg2Iiwicm9sZSI6ImN1c3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRzL3B1YmxpYy9pbWFnZ
XMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDoxMTozNC4xNjYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDox
NDowNS450TggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzYyNDI0MTExfQ.me2RTD8btzs55GXQxq4of0_OIu_nP3zr_FQmT4H0muZfmdvyVDe6sq0IapcN9kT6ohoRcd2PLcqbsDrz-bozWaYezy6L_8UICQbT
VcWVY9wU7-hKTjLMS_EbuxYSFIJnLBeyC0HVXZxsw1SUS1cXN9xpmKqFLwclNfv0k9NWmM8
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
sec-ch-ua-mobile: ?0
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJkcmFnb25tYXN0ZXJycHRwMkBnbWFpbC5jb20iLCJwYXNzd29yZCI6Im
MyODE5NGQ0ZWY2N2NkNmYwZDcyYTIzNjVhYzU5ODg2Iiwicm9sZSI6ImN1c3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRzL3B1YmxpYy9pbWFnZ
XMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDoxMTozNC4xNjYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDox
NDowNS450TggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzYyNDI0MTExfQ.me2RTD8btzs55GXQxq4of0_OIu_nP3zr_FQmT4H0muZfmdvyVDe6sq0IapcN9kT6ohoRcd2PLcqbsDrz-bozWaYezy6L_8UICQbT
VcWVY9wU7-hKTjLMS_EbuxYSFIJnLBeyC0HVXZxsw1SUS1cXN9xpmKqFLwclNfv0k9NWmM8
Connection: keep-alive
```

- **Impact:** An attacker can create a malicious webpage (e.g., with a hidden image link) that, when visited by a logged-in victim, will change their password to one chosen by the attacker, leading to full account takeover.

Finding 3.3: Sensitive Data Exposure in URL (GET Request)

- **Risk Rating:** High
- **OWASP Top 10: A02:2021 - Cryptographic Failures**
- **Description:** The application transmits the user's current and new password using the GET HTTP method during a password change. This places the sensitive credentials directly in the URL in plain text.
- **Proof of Concept (PoC):**
    1. As seen in the Burp Suite request (TP4.png), the full request is: GET /user/change-password?current=Dragon1234&new=Dragon12345&repeat=Dragon12345
    2. The user's current password (Dragon1234) and new password (Dragon12345) are visible in the request.

- **Screenshots:**


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.10.1 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://localhost:3000 [127.0.0.1]
Open browser
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
Pretty Raw Hex
Search
0 highlights

```
ime              Type  Direction    Method  URL                                                                                         Status code
5:15:43 6 No...  HTTP  → Request    GET     http://localhost:3000/rest/user/change-password?current=Dragon1234&new=Dragon12345&repeat=Dragon12345
5:16:01 6 Nov... WS    ← To client          http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=dQc__jb4E0-GJrzOAAAe
5:16:07 6 No...  HTTP  → Request    GET     http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PfOgfsj
```

Request

```
GET /rest/user/change-password?current=Dragon1234&new=Dragon12345&repeat=Dragon12345 HTTP/1.1
Host: localhost:3000
sec-ch-ua-platform: "Linux"
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJkcmFnb25tYXN0ZXJycHRwMkBnbWFpbC5jb20iLCJwYXNzd29yZCI6Im
MyODE5NGQ0ZWY2N2NkNmYwZDcyYTIzNjVhYzU5ODg2Iiwicm9sZSI6ImNlc3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRzL3B1YmxpYy9pbWFnZ
XMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDoxMTozNC4xNjYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDox
NDowNS450TggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzYyNDI0MTExfQ.me2RTD8btzs55GXQxq4of0_OIu_nP3zr_FQmT4H0muZfmdvyVDe6sq0IapcN9kT6ohoRcd2PLcqbsDrz-bozWaYezy6L_8UICQbT
VcWVY9wU7-hKTjLMS_EbuxYSFIJnLBeyC0HVXZxsw1SUS1cXN9xpmKqFLwclNfv0k9NWmM8
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
sec-ch-ua-mobile: ?0
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjMsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJkcmFnb25tYXN0ZXJycHRwMkBnbWFpbC5jb20iLCJwYXNzd29yZCI6Im
MyODE5NGQ0ZWY2N2NkNmYwZDcyYTIzNjVhYzU5ODg2Iiwicm9sZSI6ImNlc3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiLCJwcm9maWxlSW1hZ2UiOiIvYXNzZXRzL3B1YmxpYy9pbWFnZ
XMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDoxMTozNC4xNjYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyNS0xMS0wNiAxMDox
NDowNS450TggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNzYyNDI0MTExfQ.me2RTD8btzs55GXQxq4of0_OIu_nP3zr_FQmT4H0muZfmdvyVDe6sq0IapcN9kT6ohoRcd2PLcqbsDrz-bozWaYezy6L_8UICQbT
VcWVY9wU7-hKTjLMS_EbuxYSFIJnLBeyC0HVXZxsw1SUS1cXN9xpmKqFLwclNfv0k9NWmM8
Connection: keep-alive
```

- **Impact:** Passwords are saved in the user's browser history, the web server's access logs, and any proxy server logs. This is a severe information leak, as anyone with access to these logs can see the user's credentials.

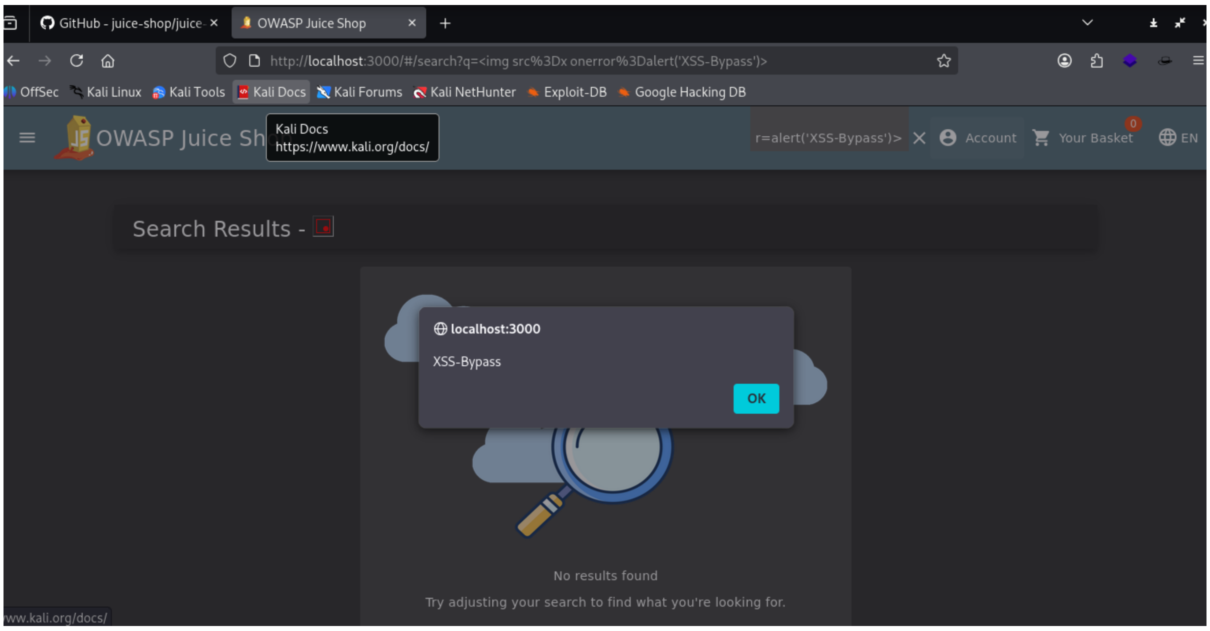
-

Finding 3.4: Reflected Cross-Site Scripting (XSS)

- **Risk Rating:** Medium
- **OWASP Top 10:** A03:2021 - Injection
- **Description:** The search function fails to properly sanitize user-supplied input and reflects it back to the page. This allows an attacker to inject malicious client-side scripts that execute in the victim's browser.
- **Proof of Concept (PoC):**

  1. The search bar was found to block basic <script> tags.
  2. A bypass payload was used: <img src=x onerror=alert('XSS-Bypass')>
  3. This payload was entered into the search bar. The application rendered it, and the browser, failing to load image "x", executed the JavaScript in the onerror event.
  4. An alert box with the text "XSS-Bypass" appeared, confirming the vulnerability.

- **Screenshots:**



```
http://localhost:3000/#/search?q=<img src%3Dx onerror%3Dalert('XSS-Bypass')>
```

Search Results -

localhost:3000

XSS-Bypass

No results found

Try adjusting your search to find what you're looking for.

- **Impact:** An attacker can send a malicious link to a victim. If the victim clicks it, the attacker's script can steal their session cookies, hijack their account, or redirect them to a phishing site.

---

4. Recommendations & Mitigation

1. **For SQL Injection (Critical):**

    ○ **Mitigation:** Implement Parameterized Queries (Prepared Statements). This is the most effective defense. It separates the SQL command from the user data, making it impossible for the user input to be executed as code.

2. **For CSRF (High):**

    ○ **Mitigation:** Implement an Anti-CSRF Token (e.g., a "Synchronizer Token"). A unique, unpredictable token should be generated for each user session and required for all state-changing requests (like changing a password).

3. **For Sensitive Data Exposure (High):**

    ○ **Mitigation:** Use the POST Method for all forms that handle sensitive data or change state. Data must be sent in the request body, not the URL. This should be implemented along with the Anti-CSRF token.

4. **For XSS (Medium):**

    ○ **Mitigation:** Implement Context-Aware Output Encoding. The application must encode all user-supplied data before rendering it in the HTML response. For example, < should be converted to `&lt;` and > to `&gt;`.

5. OWASP Top 10 Compliance Checklist

This checklist maps the identified findings to the 2021 OWASP Top 10 categories.

A01:2021 - Broken Access Control

- **Status:** FAIL
- **Related Findings:** Finding 3.2 (CSRF)

A02:2021 - Cryptographic Failures

- **Status:** FAIL
- **Related Findings:** Finding 3.3 (Sensitive Data in URL)

A03:2021 - Injection

- **Status:** FAIL
- **Related Findings:** Finding 3.4 (XSS)

A04:2021 - Insecure Design

- **Status:** Not Tested

A05:2021 - Security Misconfiguration

- **Status:** Not Tested

A06:2021 - Vulnerable Components

- **Status:** Not Tested

A07:2021 - Ident. & Auth. Failures

- **Status:** FAIL
- **Related Findings:** Finding 3.1 (SQLi Auth. Bypass)

A08:2021 - Software/Data Integrity

- **Status:** Not Tested

A09:2021 - Logging & Monitoring

- **Status:** Not Tested

A10:2021 - SSRF

- **Status:** Not Tested

6. Final Deliverables

This PDF document serves as the primary report. The following supplementary files are provided in a Github archive:

- **OWASP ZAP Report:** (ZAP_Scan_Report.html)
- **Nikto Scan Log:** (Nikto_Report.html)
- **Evidence Files:** All screenshots (.png) used in this report.