# Cybersecurity Internship Report at FutureInterns

Name: K.KAVI PREETHY

G-MAIL:kavipreethy03@gmail.com

Task 1:Web Application Security Testing

# TABLE OF CONTENTS

# TASK 1

# WEB APPLICATION SECURITY TESTING

## Objective

The goal of this task was to conduct security testing on a vulnerable web application (OWASP Juice Shop) to identify and exploit common vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Authentication Flaws. This task provided handson experience in ethical hacking, understanding security loopholes, and suggesting possible mitigation strategies.

## Installation and Setup

Environment Setup:

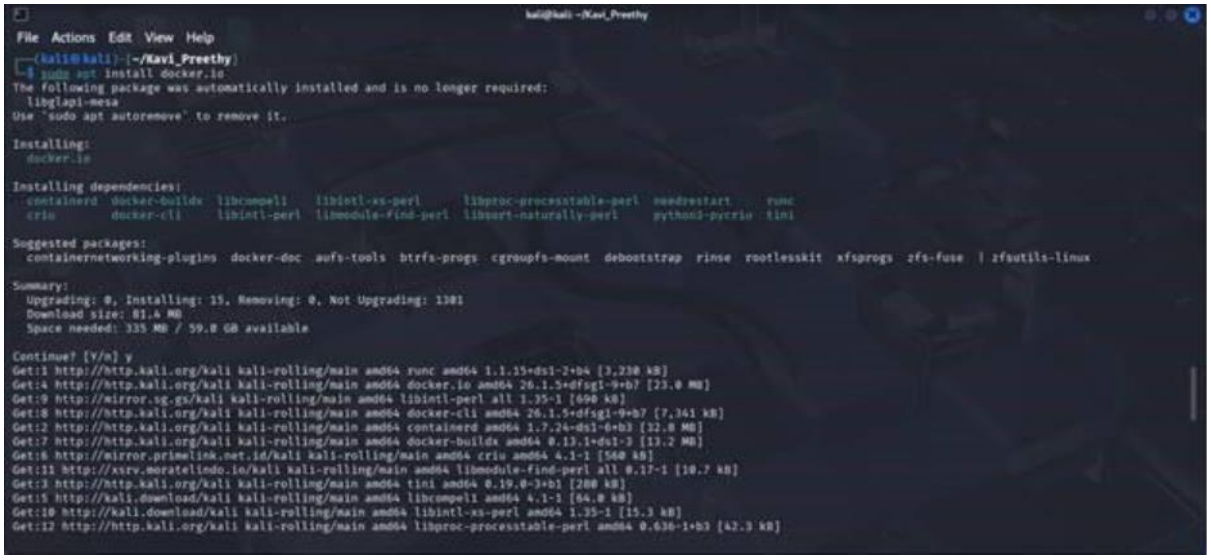
The OWASP Juice Shop application was installed and run using Docker. The following commands were executed:

sudo apt update

This command updates the package list to ensure access to the latest versions of software.

sudo apt install docker.io

Installs Docker, a platform used to run applications in isolated containers.




Figure(1):Docker installation

sudo systemctl start docker

Starts the Docker service so that containers can be run.

sudo docker pull bkimminich/juice-shop

Pulls the official OWASP Juice Shop Docker image from Docker Hub.


Figure(2):Docker image pulled  sudo docker run -d -p

3000:3000 bkimminich/juice-shop

Runs the Juice Shop container in detached mode and maps port 3000 from the container to the host.

```
┌──(kali㉿kali)-[~/Kavi_Preethy]
└─$ sudo docker run -d -p 3000:3000 bkimminich/juice-shop
1ed7965359a260c9cb2591099fc2474423e5a61ad5d0f3ed773027ec5f98081a

┌──(kali㉿kali)-[~/Kavi_Preethy]
└─$
```

Figure(3):Juice shop running on port 3000

The application was then accessed via http://localhost:3000 in a browser.

## 1.SQL Injection - Bypass Login Authentication

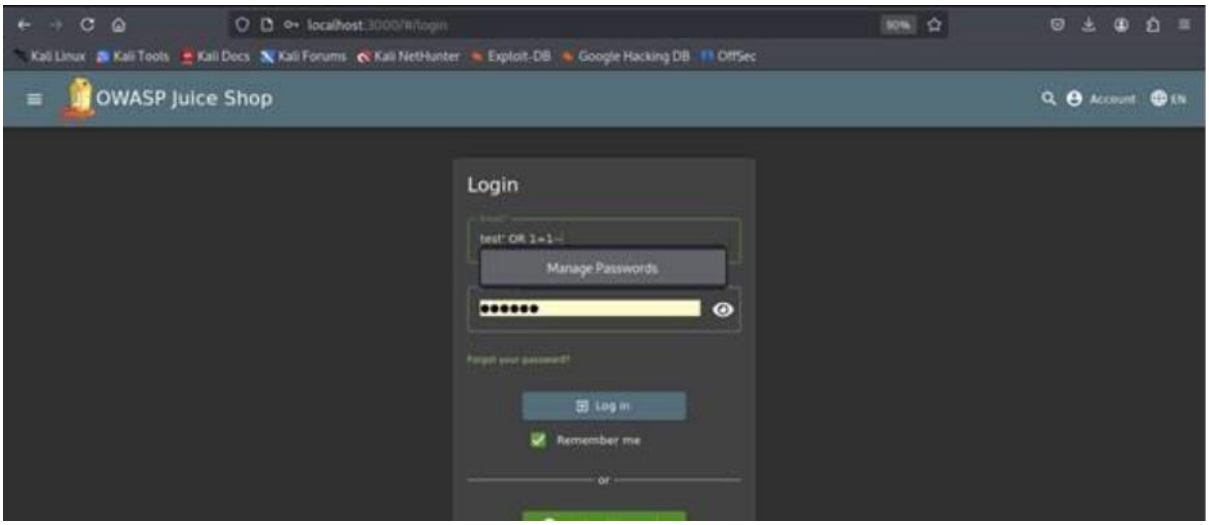
Steps:

- Launched Burp Suite and configured the browser to route traffic through Burp's proxy.
- Navigated to the login page of Juice Shop.
- Entered the SQLi payload:

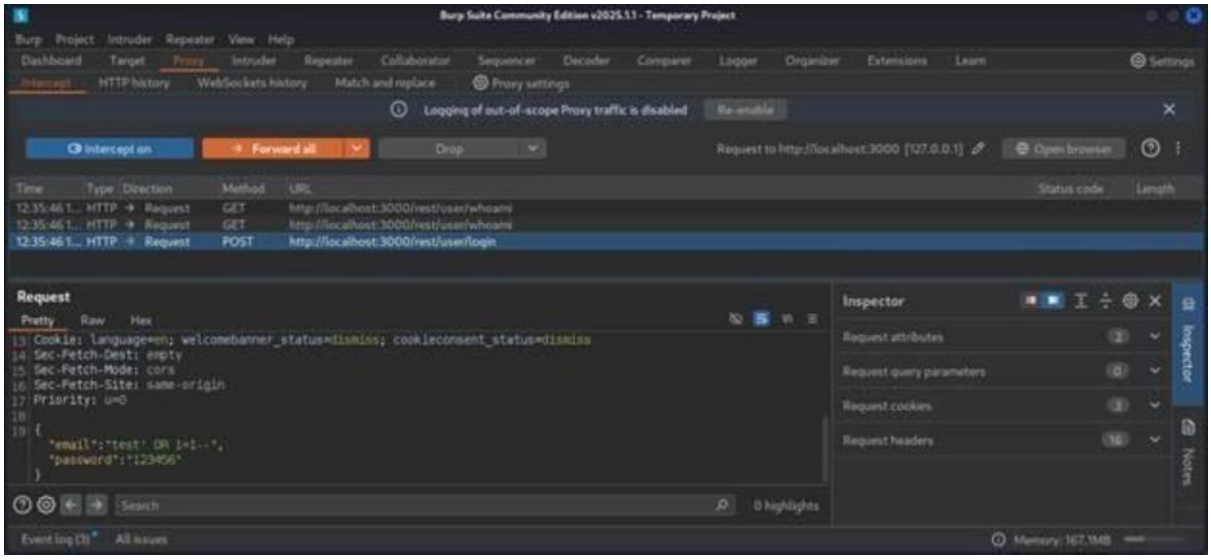
  Email: ' OR 1=1--
  Password: anything

- Captured the HTTP request in Burp Suite and forwarded it.
- Observed that the application logged in successfully without valid credentials.

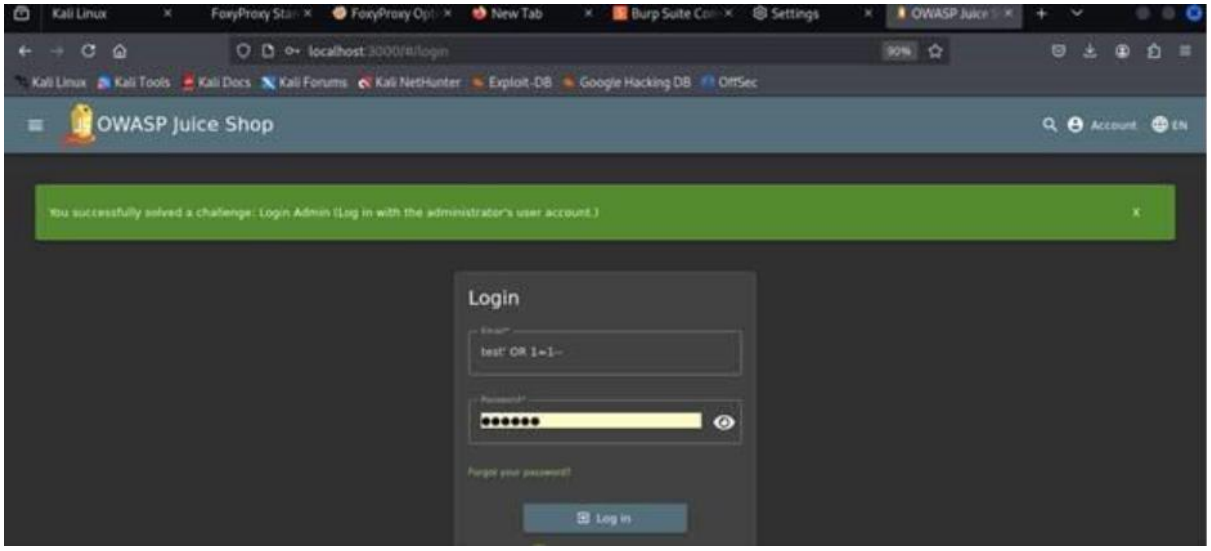
# Screenshot




Figure(4):Login page.




Figure(5): Intercepted Request in Burp Suite



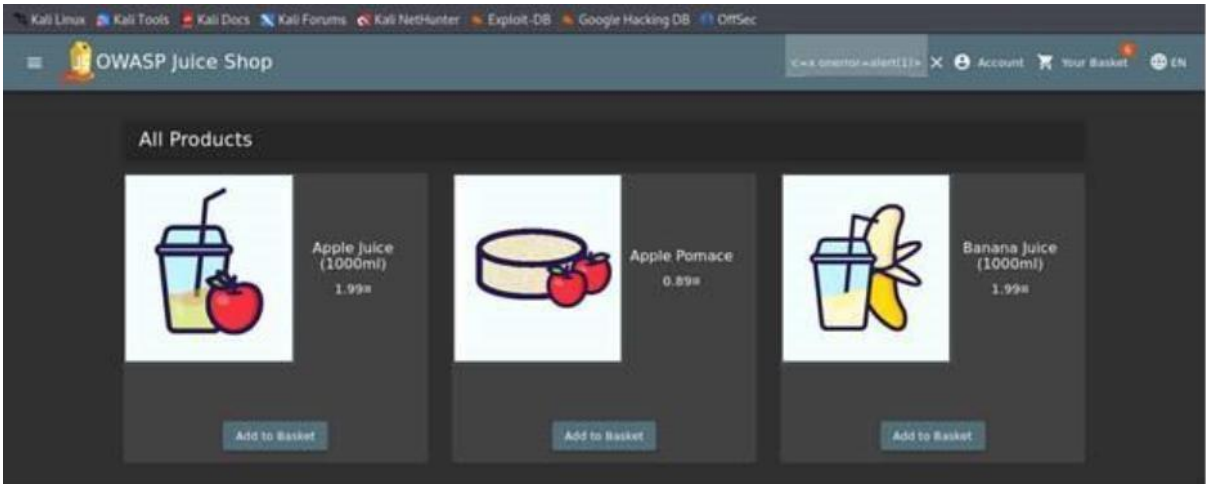

Figure(6):Successful login using SQL injection

- Vulnerability Confirmed: SQL Injection allowed bypassing authentication

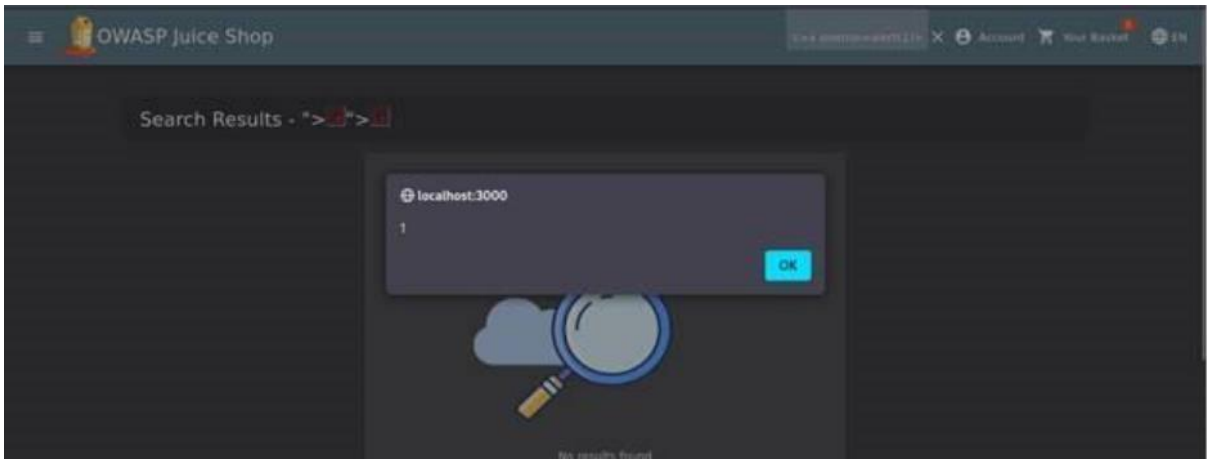
## 2.Cross-Site Scripting (XSS)

- Navigated to the search bar and entered the following payload: <script>alert('XSS')</script>
- A JavaScript alert was triggered – confirming reflected XSS.
- Navigated to the Feedback section.
- Submitted the same script in the comment field.
- Used Burp Suite to monitor the HTTP request and response.

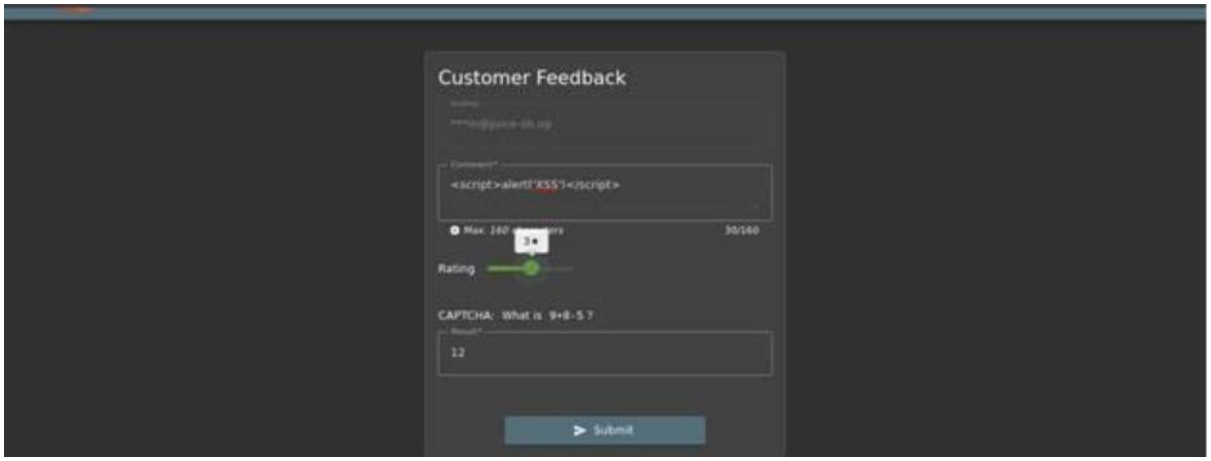
## Screenshots



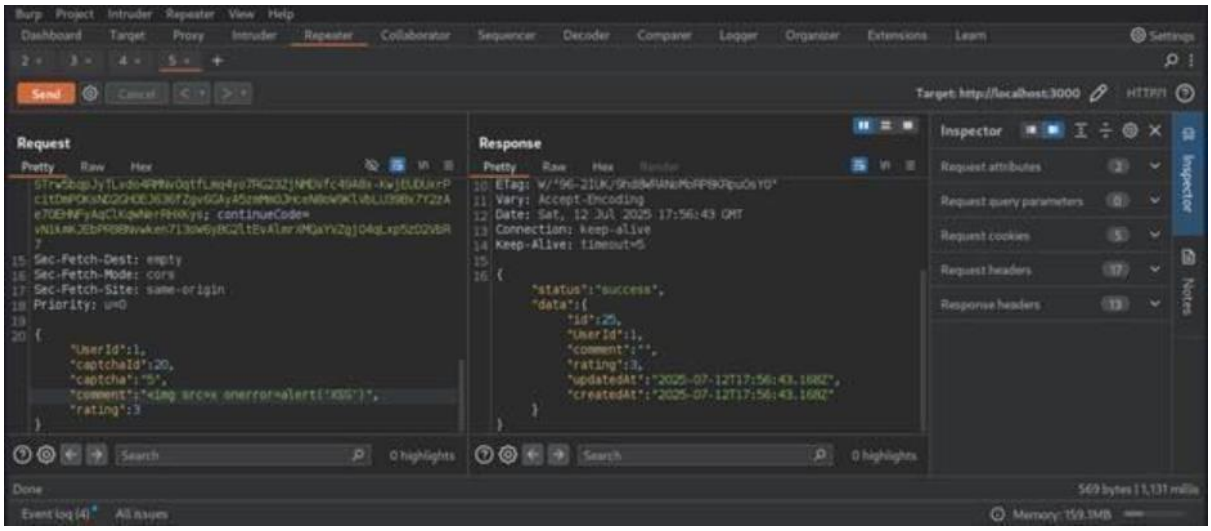

Figure(7):Payload in search bar



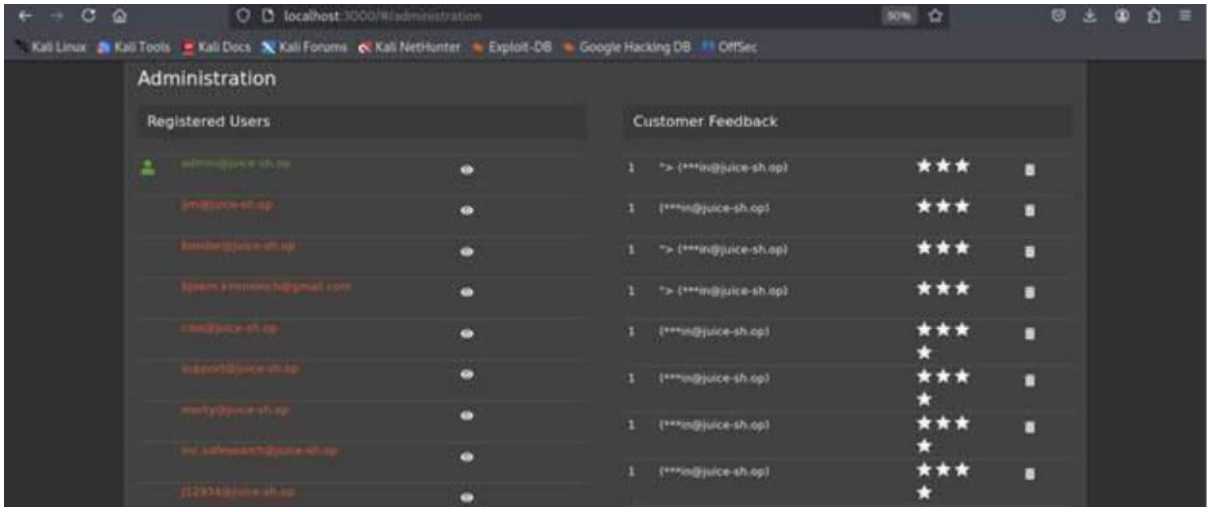

Figure(8):Alert Triggered(Reflected XSS)



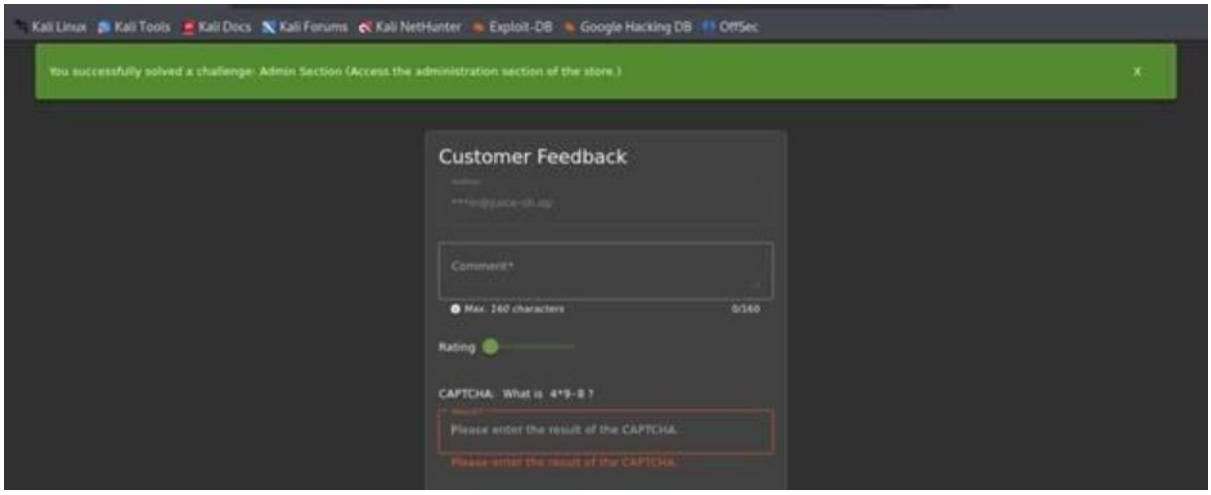

Figure(9):Payload in customer feedback form




Figure(10):Response in burp suite




Figure(11):Administator page




Figure(12):Feedback Reflected with script

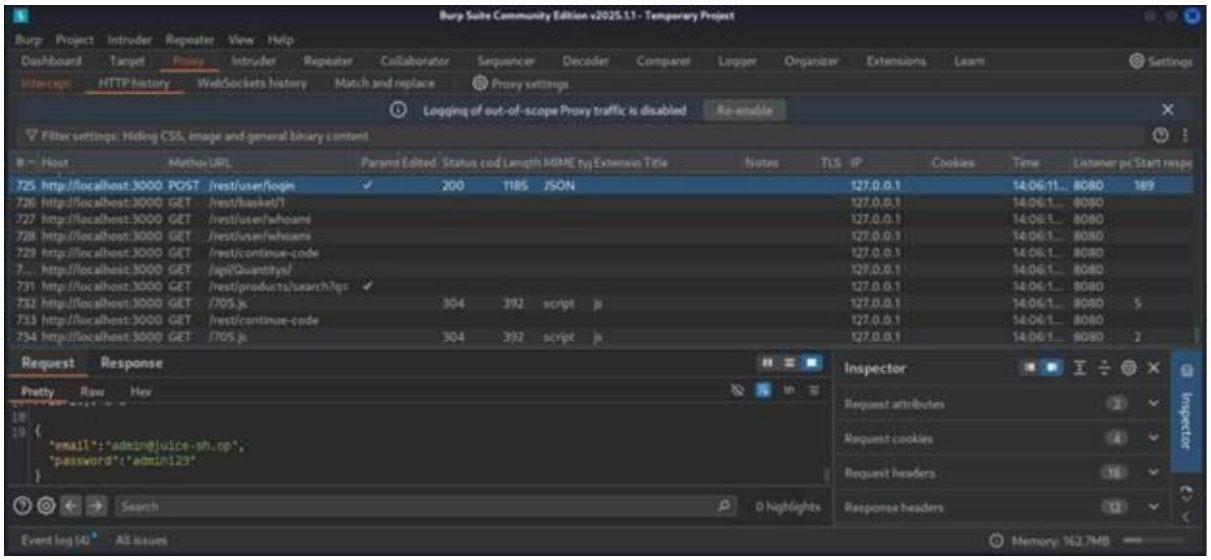
- Vulnerability Confirmed: Both reflected and stored XSS were exploitable.

## 3. Authentication Flaws

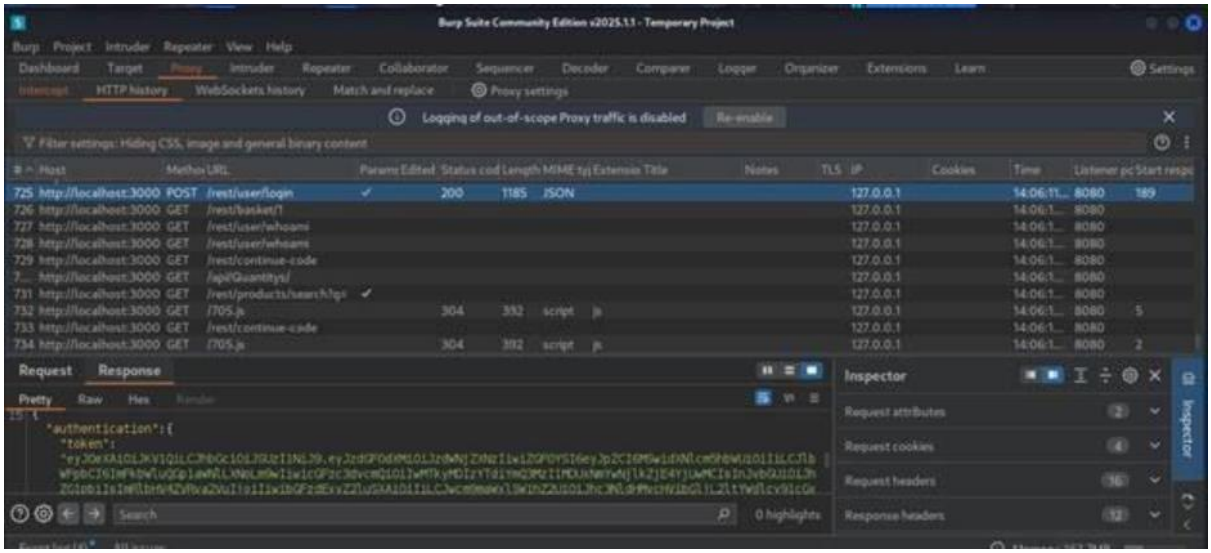
- Logged in with admin credentials.
- Intercepted the HTTP response in Burp Suite.
- Copied the JWT token from the response.
- Opened Developer Tools in Firefox (Ctrl + Shift + I).
- Navigated to Application > Local Storage.
- Added a new item manually: Key: token, Value: [Copied token]
- Refreshed the page and gained admin access.

### Screenshot

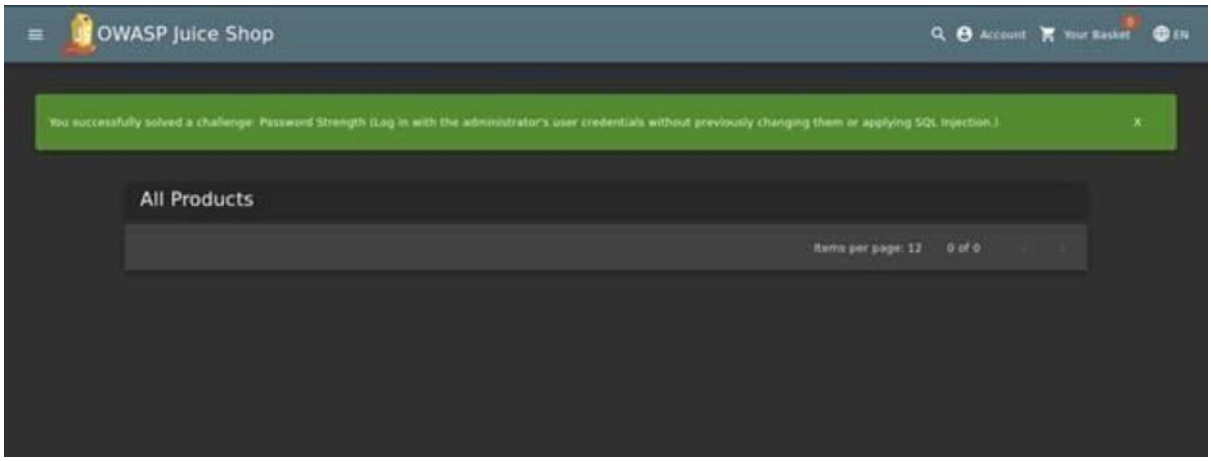

Figure(13):Admin Login Attempt



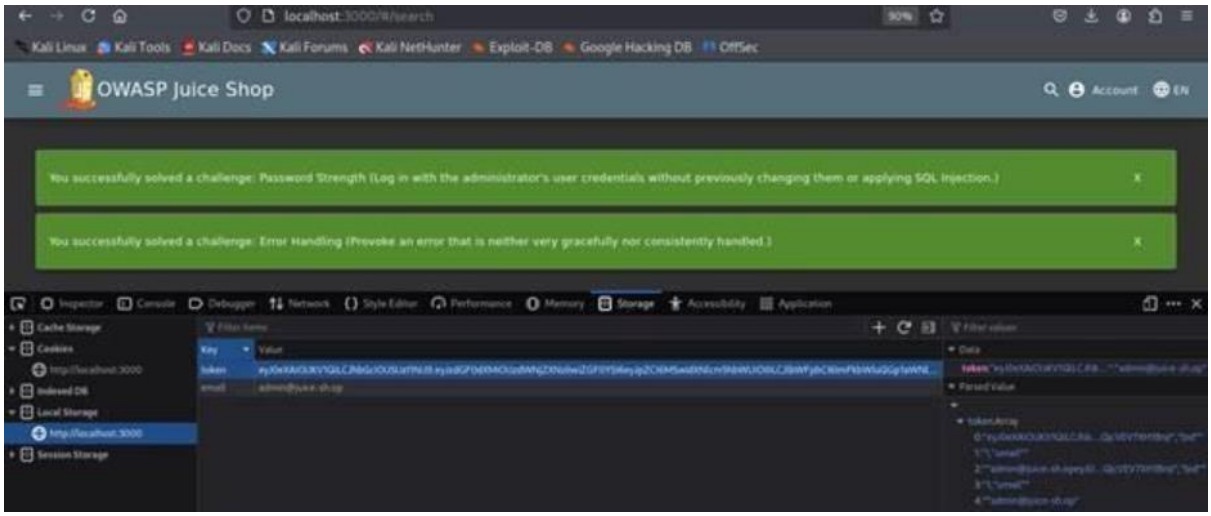

Figure(14):Burp suite Request of admin




Figure(15):Burp suite Response of admin




Figure(16):Admin Panel Accessed




Figure(17):Error handling Verified

- Vulnerability Confirmed: Token-based authentication could be manipulated using stolen tokens.

## Mitigation Strategies

- Input Validation: Sanitize all user inputs on both client and server sides.
- Parameterized Queries: Use prepared statements to prevent SQL Injection.
- Output Encoding: Escape user input before rendering to prevent XSS.
- Secure JWT Handling: Implement secure cookie storage, token expiration, and validation mechanisms.

## Conclusion

Through this internship task, I successfully performed real-world web application penetration testing on OWASP Juice Shop. The vulnerabilities identified include:

- SQL Injection: Enabled login bypass using classic payloads.
- Cross-Site Scripting (XSS): Both reflected and stored XSS were found in the search and feedback modules.
- Authentication Flaws: Token hijacking led to unauthorized admin access.