

Cybersecurity Internship Report at FutureInterns

Name: K.KAVI PREETHY

G-MAIL:kavipreethy03@gmail.com

Task 1:Web Application Security Testing

TABLE OF CONTENTS

S NO	CONTENTS	PAGE NO
1	Objective	2
2	Installation and setup	2
3	SQL Injection - Bypass Login Authentication	4
4	Cross-Site Scripting (XSS)	7
5	Authentication Flaws	11

6	Mitigation Strategies	14
7	Conclusion	14

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 hands-on experience in ethical hacking, understanding security loopholes, and suggesting possible mitigation strategies.

Installation and Setup

Environment Setup:

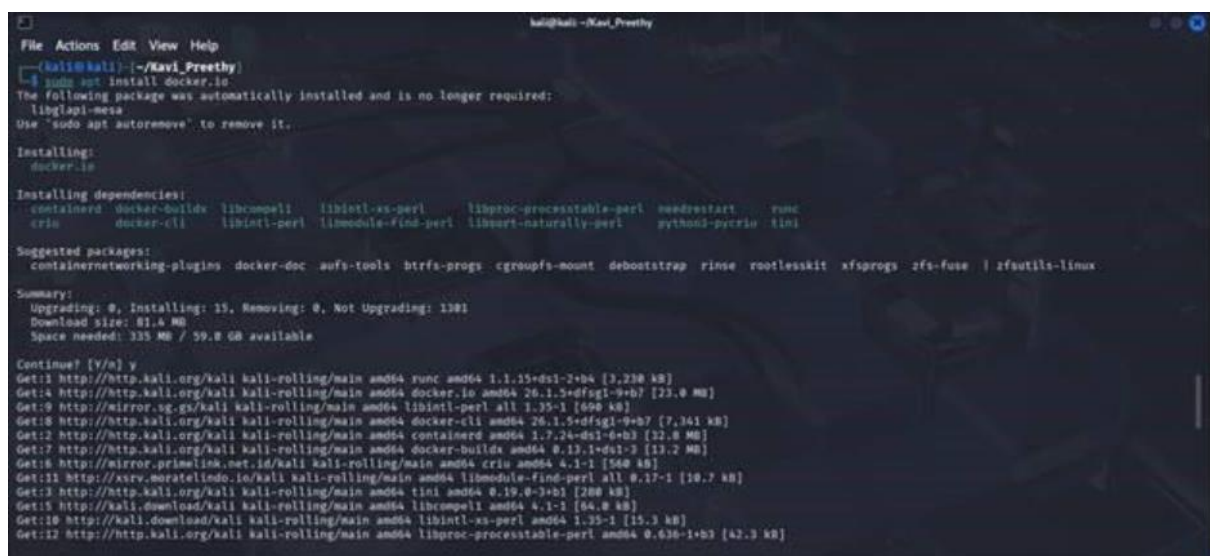
The OWASP Juice Shop application was installed and run using Docker. The following commands were executed:

`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.



```
kali@kali: ~/Kali_Preethy
File Actions Edit View Help
(kali@kali)~$ sudo apt install docker.io
The following package was automatically installed and is no longer required:
libglapi-mesa
Use 'sudo apt autoremove' to remove it.

Installing:
docker.io

Installing dependencies:
containerd docker-buildx libcompell libintl-xs-perl libproc-processtable-perl needrestart runc
crio docker-cli libintl-perl libmodule-find-perl libsort-naturally-perl python3-pycriso tini

Suggested packages:
containernetworking-plugins docker-doc aufs-tools btrfs-progs cgroupfs-mount debootstrap rinse rootlesskit xfsprogs zfs-fuse | zfsutils-linux

Summary:
Upgrading: 0, Installing: 15, Removing: 0, Not Upgrading: 1381
Download size: 81.6 MB
Space needed: 335 MB / 59.8 GB available

Continue? [Y/n] y
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.15-dsl-2+b4 [3.238 kB]
Get:4 http://http.kali.org/kali kali-rolling/main amd64 docker.io amd64 26.1.3+dfsg1-9+b7 [23.0 MB]
Get:9 http://mirror.sg.gs/kali kali-rolling/main amd64 libintl-perl all 1.35-1 [690 kB]
Get:8 http://http.kali.org/kali kali-rolling/main amd64 docker-cli amd64 26.1.5+dfsg1-9+b7 [7.341 kB]
Get:12 http://http.kali.org/kali kali-rolling/main amd64 containerd amd64 1.7.24-dsl-0+b3 [32.8 MB]
Get:7 http://http.kali.org/kali kali-rolling/main amd64 docker-buildx amd64 0.13.1-dsl-3 [13.2 MB]
Get:6 http://mirror.primeink.net.id/kali kali-rolling/main amd64 crio amd64 4.1-1 [560 kB]
Get:11 http://xsr.v.moratelindo.io/kali kali-rolling/main amd64 libmodule-find-perl all 0.17-1 [10.7 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 tini amd64 0.19.0-3+b1 [288 kB]
Get:15 http://kali.download/kali kali-rolling/main amd64 libcompell amd64 4.1-1 [64.8 kB]
Get:10 http://kali.download/kali kali-rolling/main amd64 libintl-xs-perl amd64 1.35-1 [15.3 kB]
Get:13 http://http.kali.org/kali kali-rolling/main amd64 libproc-processtable-perl amd64 0.636-1+b3 [42.3 kB]
```

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.

```
(kali@kali)~[/Kavi_Preethy]
$ sudo systemctl start docker

(kali@kali)~[/Kavi_Preethy]
$ sudo docker pull bkimminich/juice-shop
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
35d697fe2738: Pull complete
bfb39b82a506: Pull complete
4eff9a62d688: Pull complete
a62778643d56: Pull complete
7c12095b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0ba015eea81d: Pull complete
aaab0a1a13d3: Pull complete
da7818fa955e: Pull complete
d0f7aab3f7d8: Pull complete
d08c3209d929: Pull complete
c1e595f74d52: Pull complete
7fa9cfa885c: Pull complete
5b14f6c9a813: Pull complete
33ce0b1d99fc: Pull complete
fa5e0372ce60: Pull complete
a80b57d21f11: Pull complete
803ff9daac7b: Pull complete
dccc3545c0bd: Pull complete
Digest: sha256:5113ab74c523b6779a4bf25bca021b5cfaw7898b6e88f6545e8bb9e4d33ce6b0
Status: Downloaded newer image for bkimminich/juice-shop:latest
docker.io/bkimminich/juice-shop:latest
```

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.

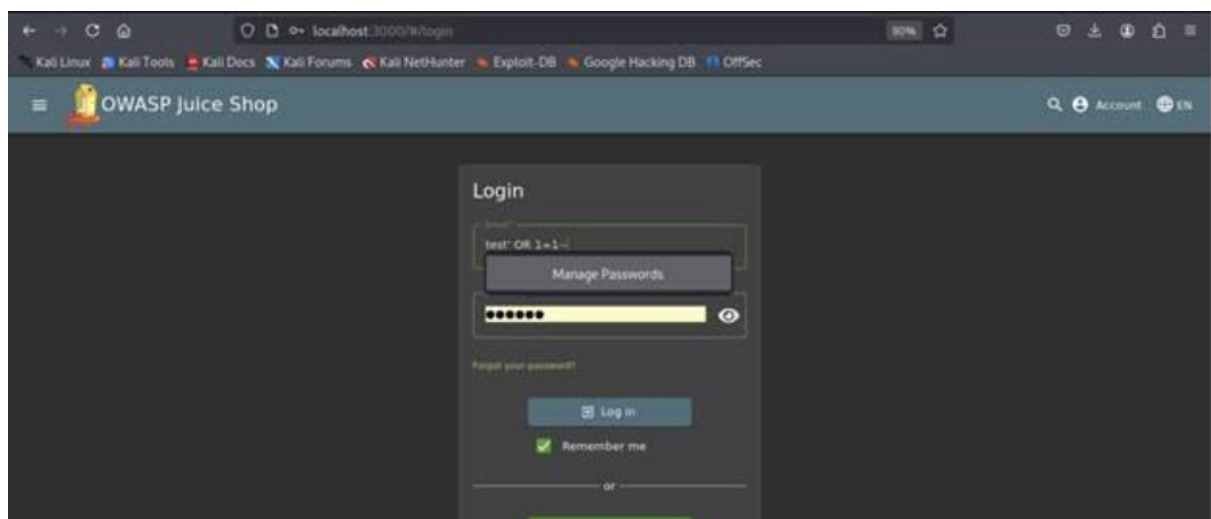
1.SQL Injection - Bypass Login Authentication

Steps:

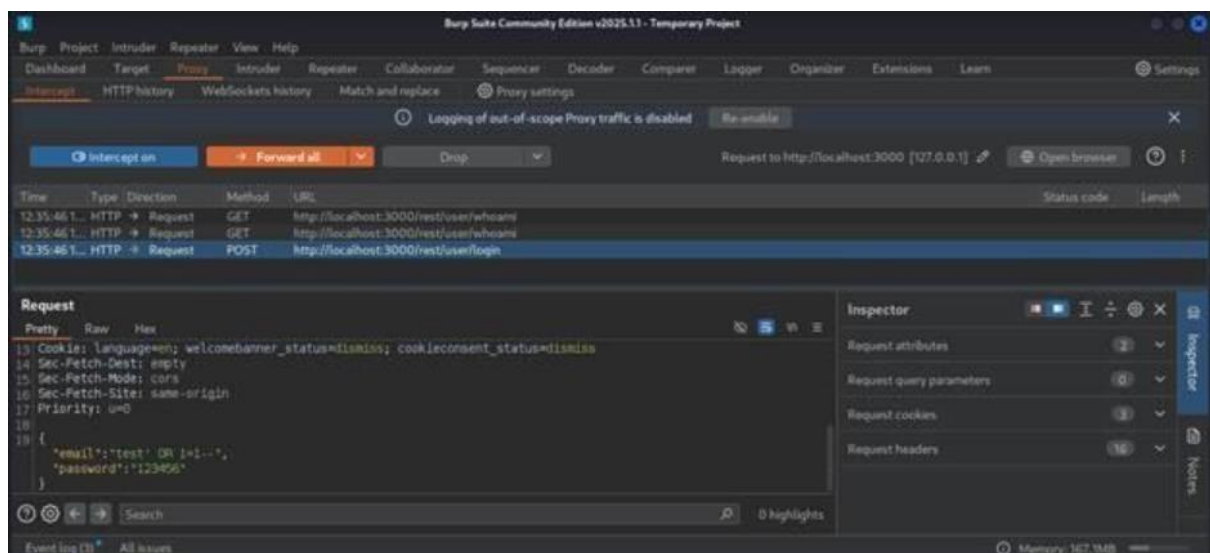
- Launched Burp Suite and configured the browser to route traffic through Burp's proxy.

- 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.

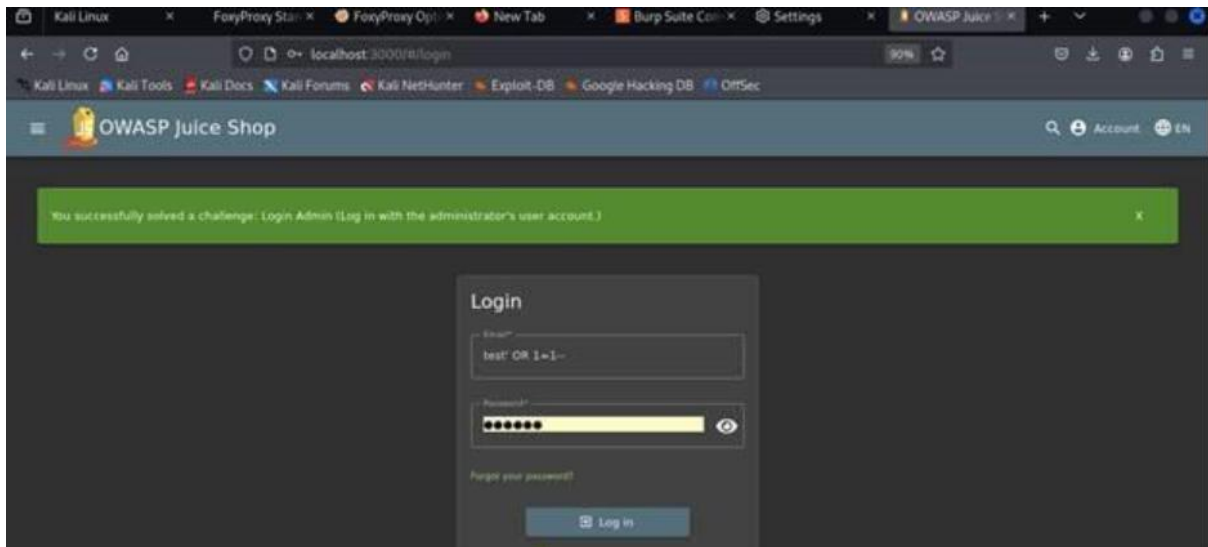
Screenshot



Figure(4):Login page.



Figure(5): Intercepted Request in Burp Suite



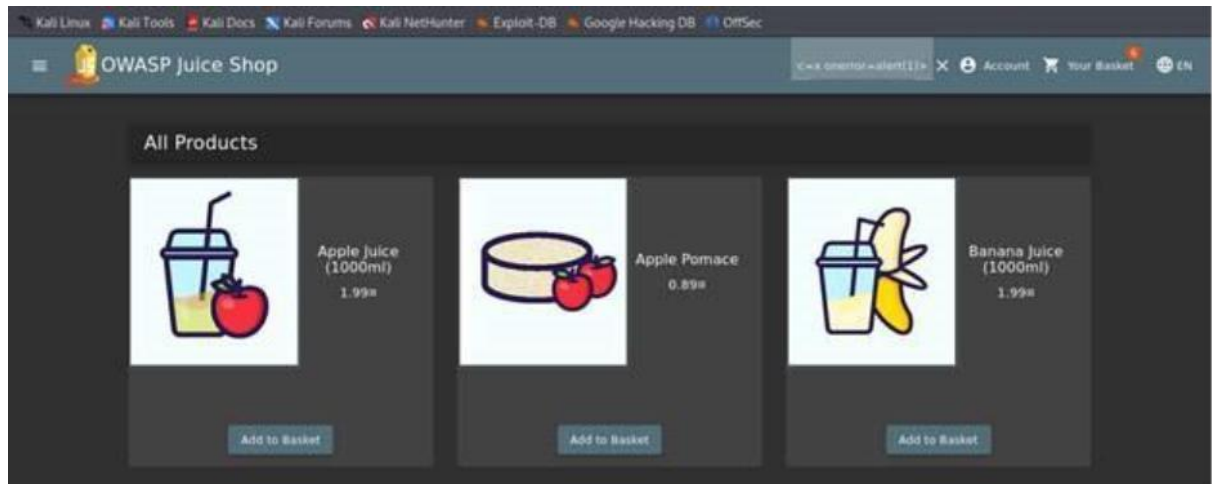
Figure(6):Successful login using SQL injection

- Vulnerability Confirmed: SQL Injection allowed bypassing authentication

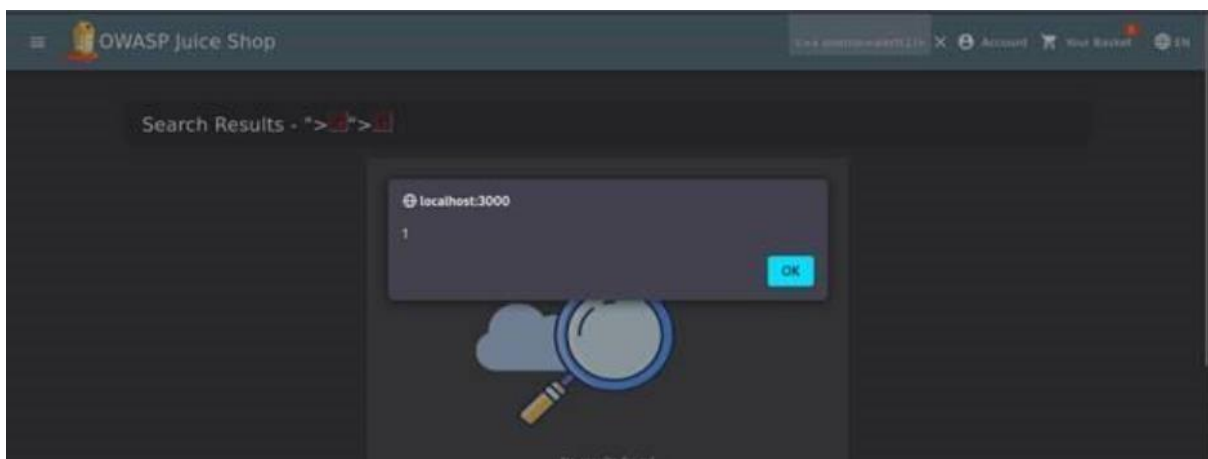
2.Cross-Site Scripting (XSS)

- 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.

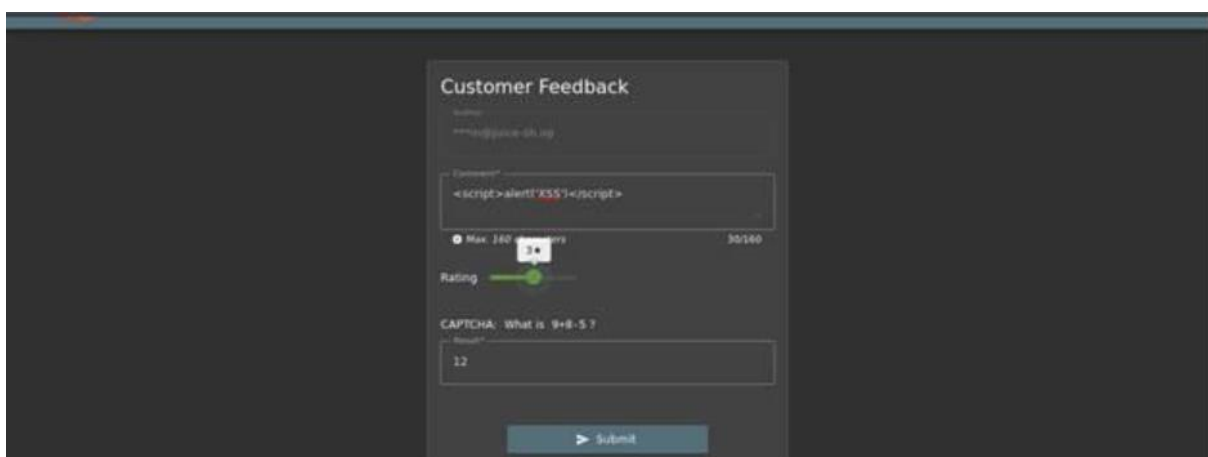
Screenshots



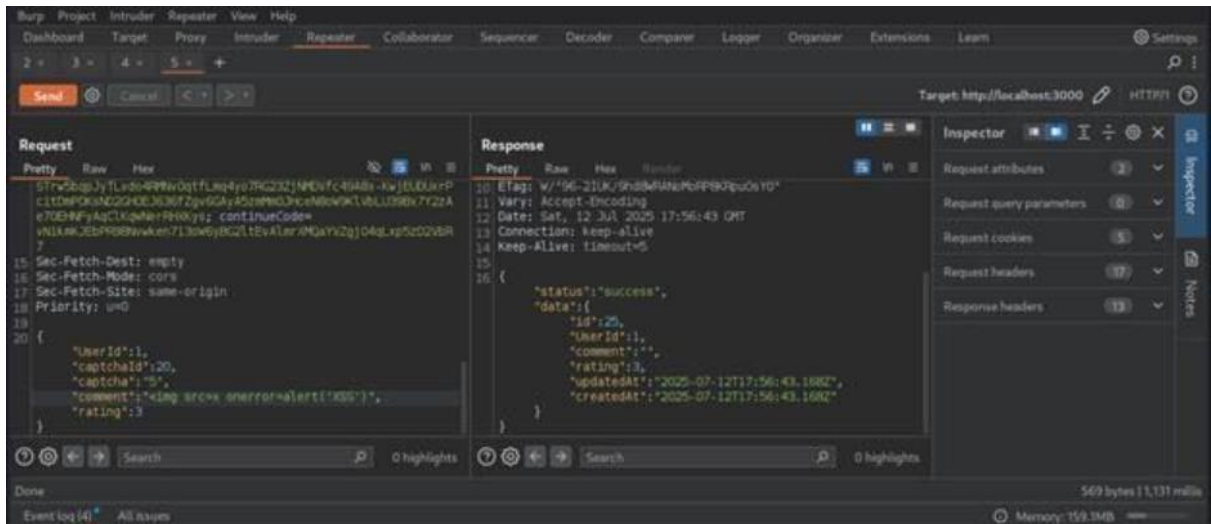
Figure(7):Payload in search bar



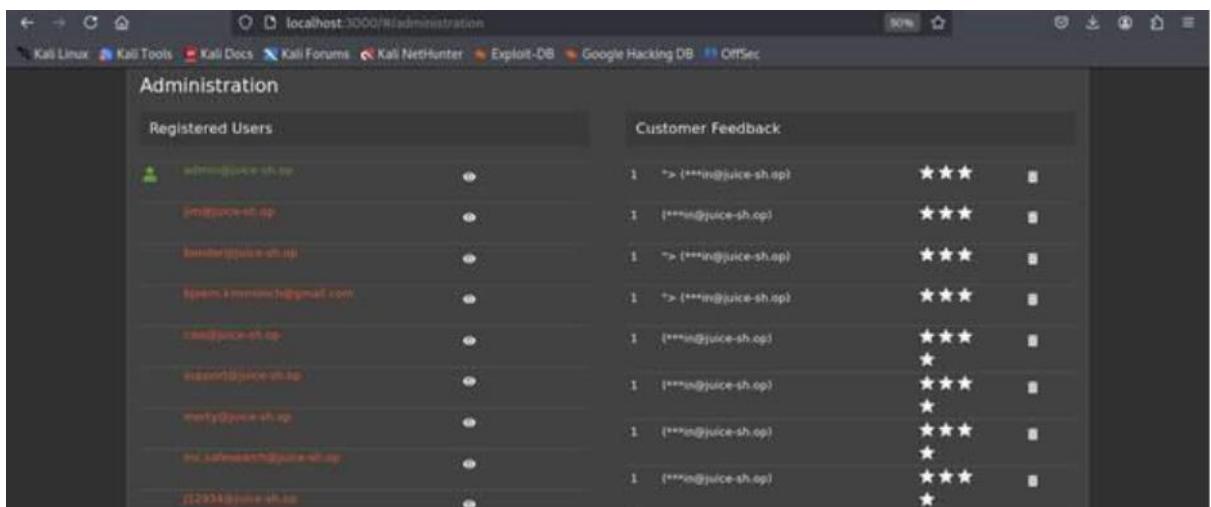
Figure(8):Alert Triggered(Reflected XSS)



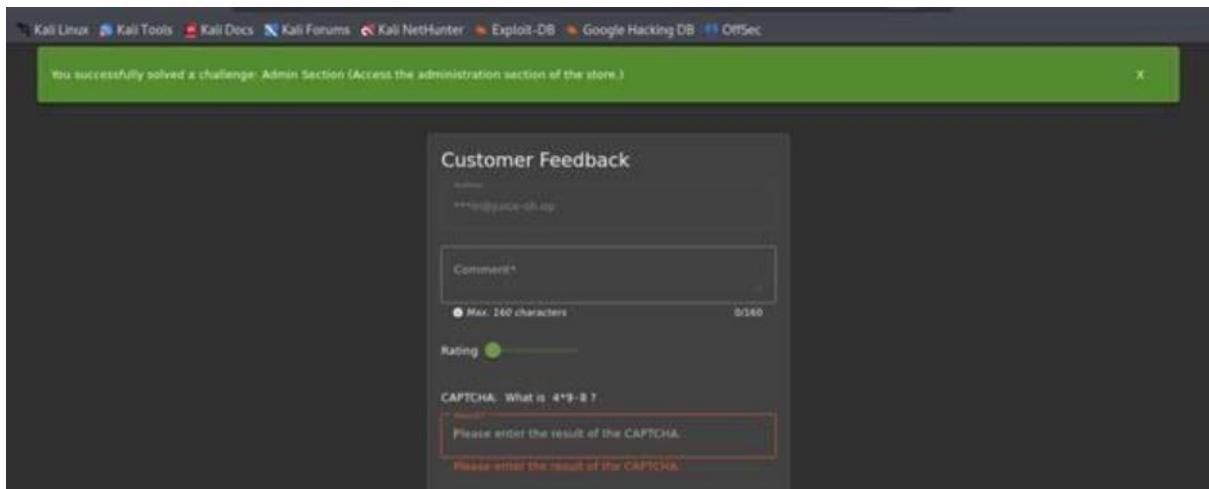
Figure(9):Payload in customer feedback form



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.

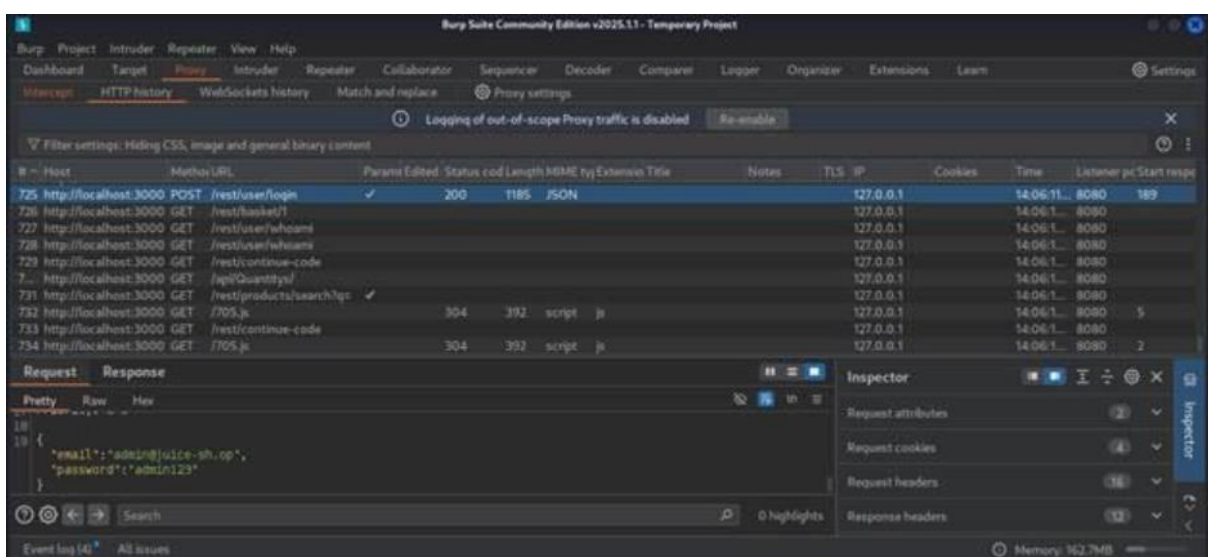
3. Authentication Flaws

- 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.

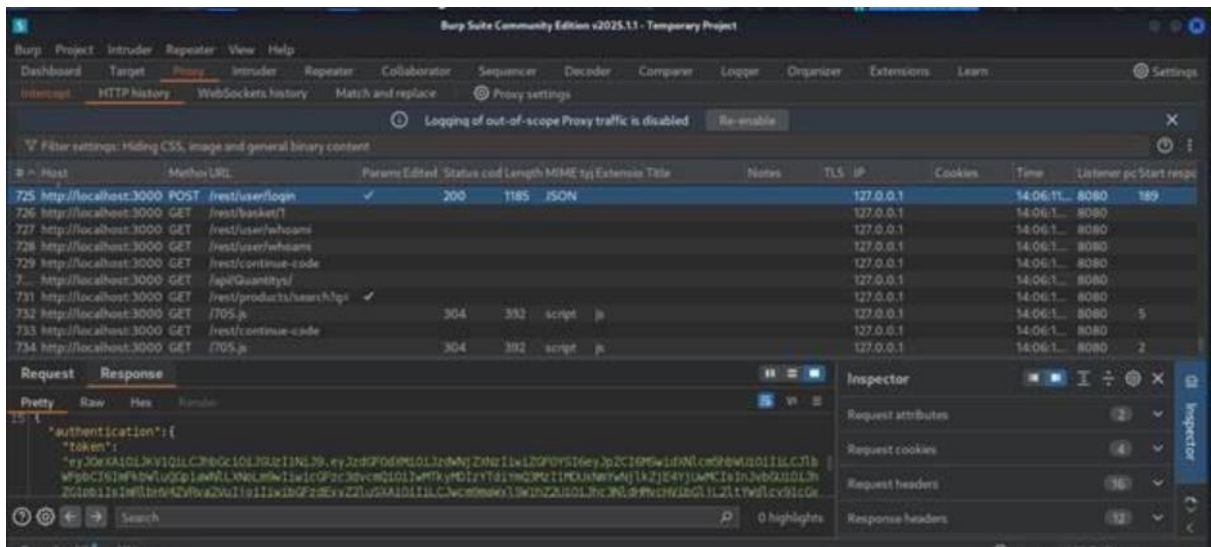
Screenshot



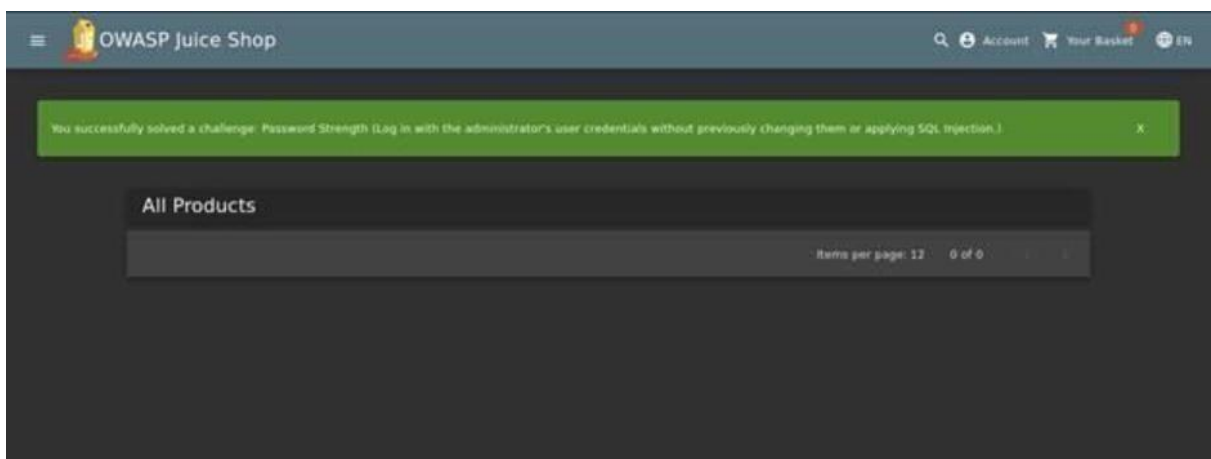
Figure(13):Admin Login Attempt



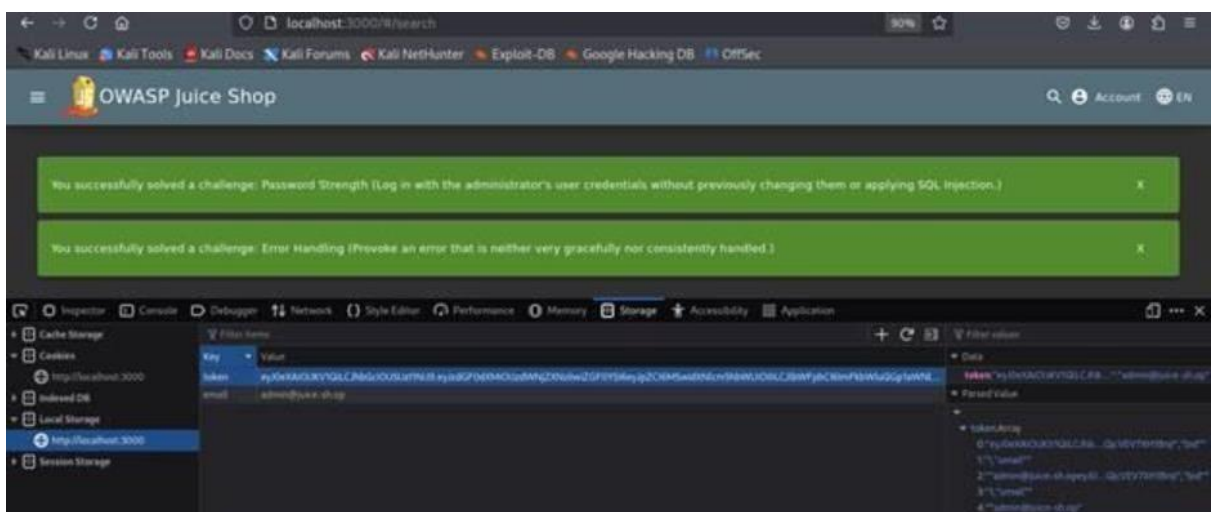
Figure(14):Burp suite Request of admin



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.

