1. Introduction

OWASP Juice Shop is a deliberately insecure web application created to teach web security through hands-on, realistic exercises. It intentionally contains a wide variety of vulnerabilities — from classic SQL injection and XSS to authentication weaknesses and insecure direct object references — mapped closely to the OWASP Top 10. Because it's gamified with flags and challenges, Juice Shop is ideal for both beginners learning the basics of web security and experienced practitioners practicing advanced exploitation techniques.

This report documents a structured penetration test performed against an instance of OWASP Juice Shop hosted on the TryHackMe platform. The goal was not only to capture challenge flags but to treat the exercise as a mini-audit: identify weaknesses, exploit them responsibly in the controlled lab environment, verify impact, and recommend practical remediations. The exercises covered reconnaissance, client-side analysis, manual exploitation using Burp Suite and browser developer tools, and limited automated testing (Burp Intruder) where appropriate.

Beyond technical verification, this assignment focuses on the real-world relevance of each vulnerability: how it could be abused in a production environment, what sensitive data or functionality is at risk, and which development or configuration changes would most effectively reduce that risk. The findings and mitigation suggestions that follow are therefore written with both developers and security reviewers in mind.

2. Methodology

The testing was carried out using:

- Kali Linux (attacker machine)
- Burp Suite (intercepting and modifying HTTP requests)
- TryHackMe VPN connection (for access to the lab environment)
- Browser Dev Tools (for analyzing JavaScript and client-side logic)

General approach followed:

1. Reconnaissance – browsing the application and intercepting requests.
2. Exploitation – performing SQLi, XSS, brute-force, and directory traversal.
3. Verification – confirming vulnerabilities by obtaining flags.
4. Documentation – recording exploitation steps and remediation.

3. Findings & Exploitation

Task 1:Open for Business!

Question 1: Deploy the VM attached to this task to get started! You can access this machine by using your browser-based machine, or if you're connected through OpenVPN.

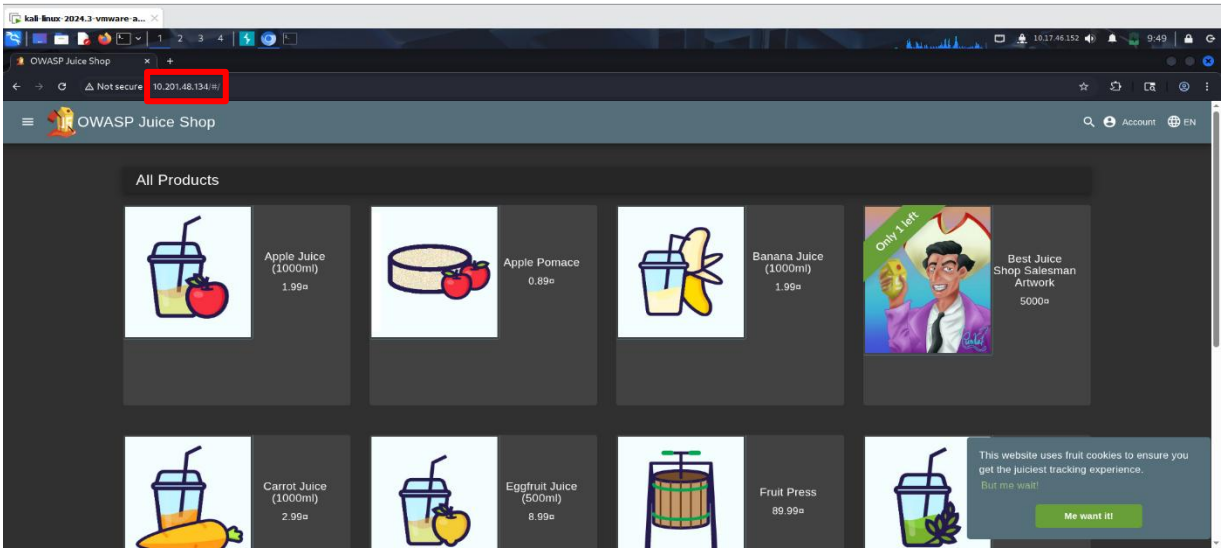
Step 1: Download the VPN .ovpn file from the Access Machine button on the TryHackMe room page.

Step 2: In Kali run:

sudo openvpn --config /path/to/your-file.ovpn

When it shows Initialization Sequence Completed, we're connected.

Question 2: Once the machine has loaded, access it by copying and pasting its IP into your browser; if you're using the browser-based machine, paste the machines IP into a browser on that machine.

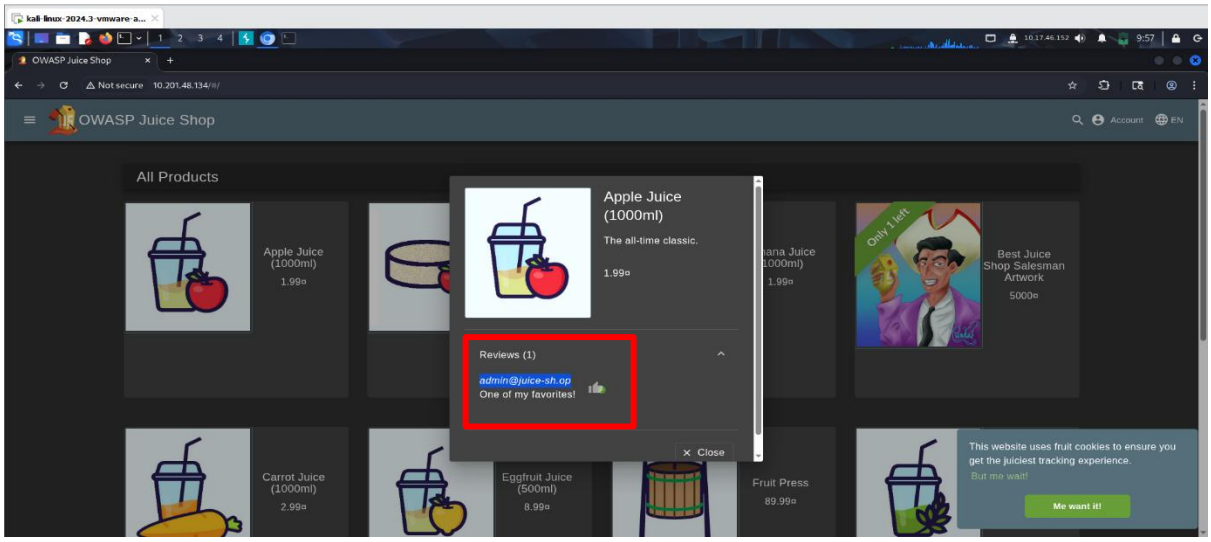


Step 1: Get the **Target IP** from the TryHackMe room.

Step 2: Open Burp → Proxy → **Open Browser** and enter **http://10.201.48.134**. The site will load through Burp so we can inspect and modify requests/responses.

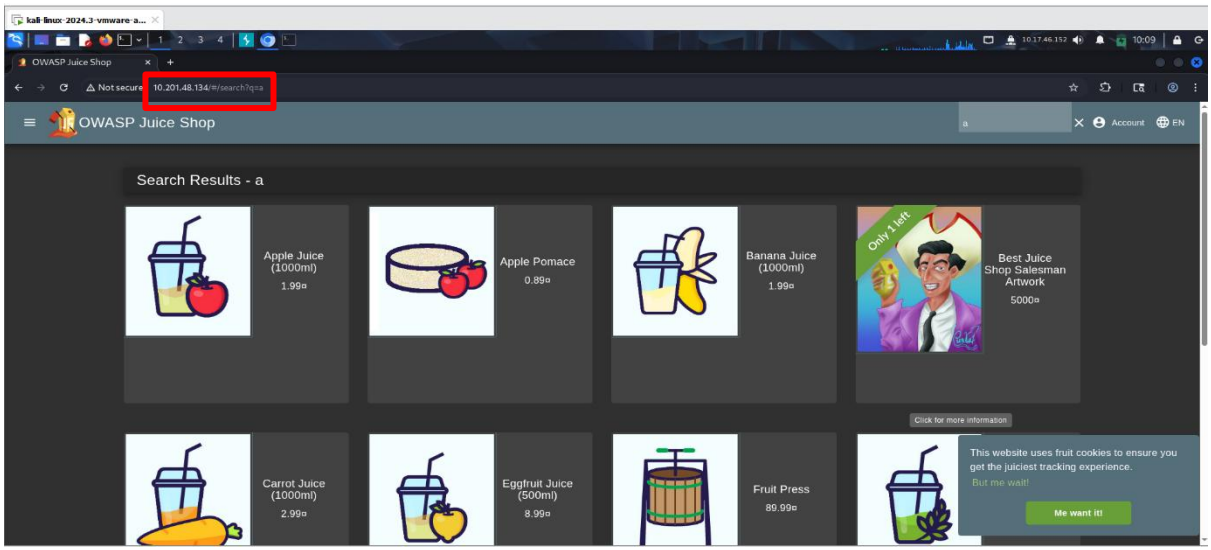
Task 2: Let's go on an adventure!

Question 1: What's the Administrator's email address?



Step 1: click on the apple juice product.
Step 2: Scroll to the Reviews section — the admin email appears there (info leakage).

Question 2: What parameter is used for searching?



Step 1: Click the magnifying glass to open the search bar.

Step 2: Type any text (e.g., a) and press Enter.

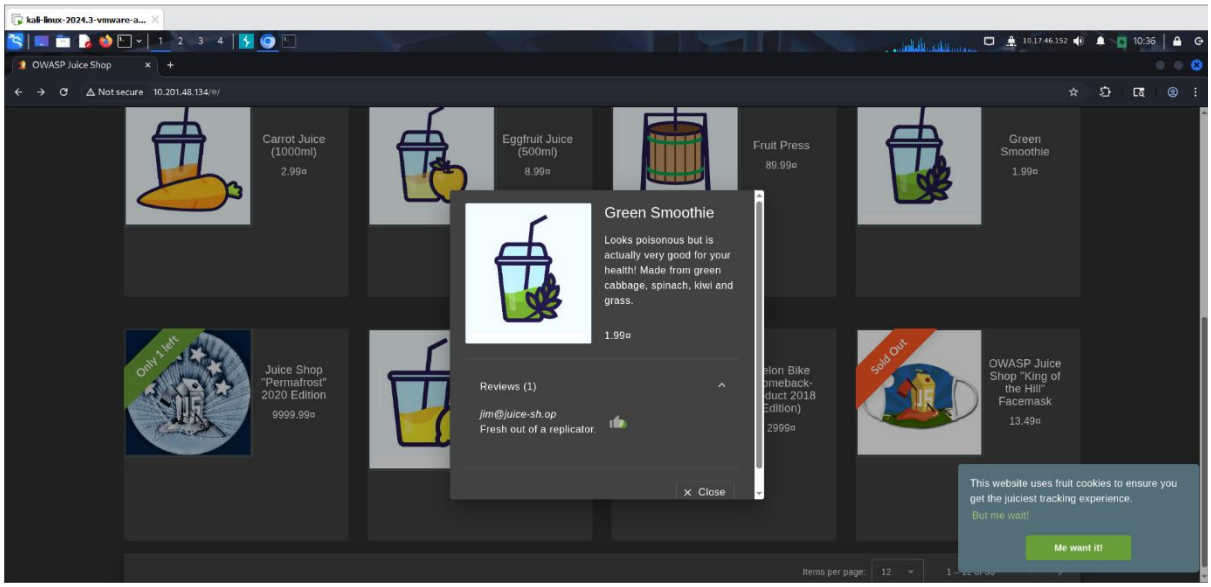
Step 3: Observe the URL change to:

http://10.201.48.134/#/search?q=a

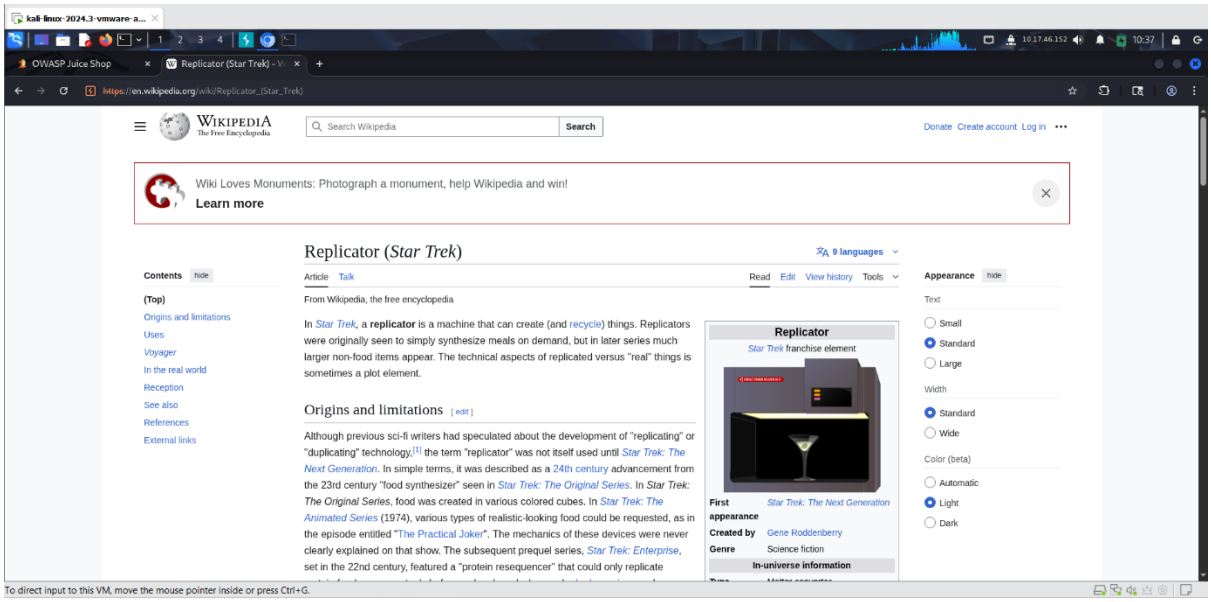
Step 4: The part after ? is the search parameter.

Answer: q

Question 3: What show does Jim reference in his review?



Step 1: Click on the Green Smoothie product and check the reviews. One review mentions "replicator".

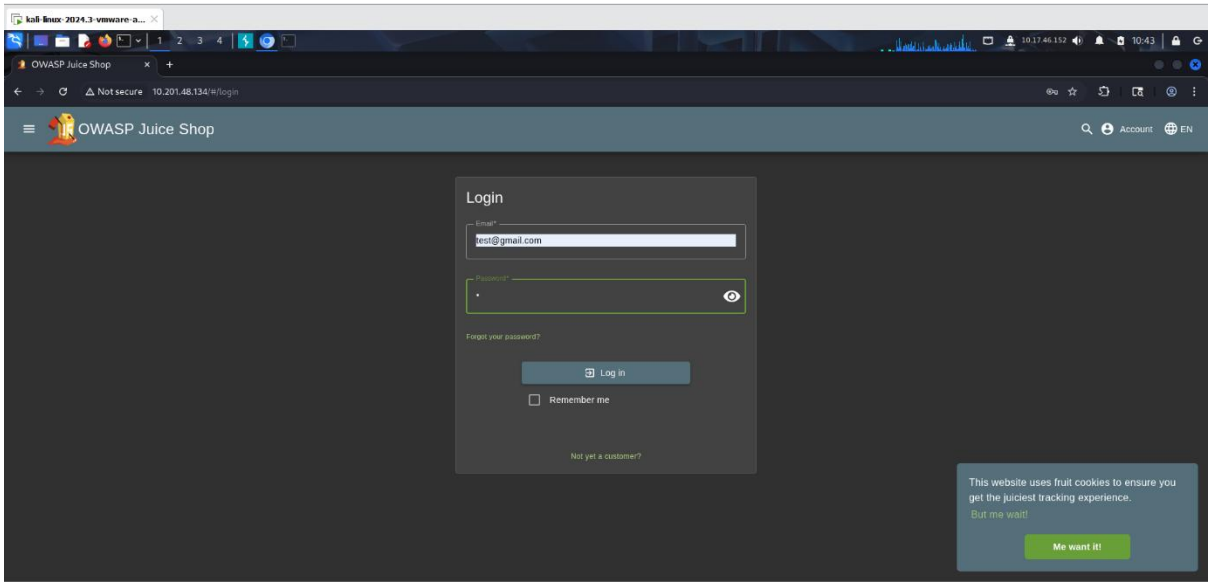


Step 2: Search "replicator" on Google — it shows that it is related to the famous movie series Star Trek.

Answer: Star Trek

Task 3: Inject the juice

Question 1: Log into the administrator account!

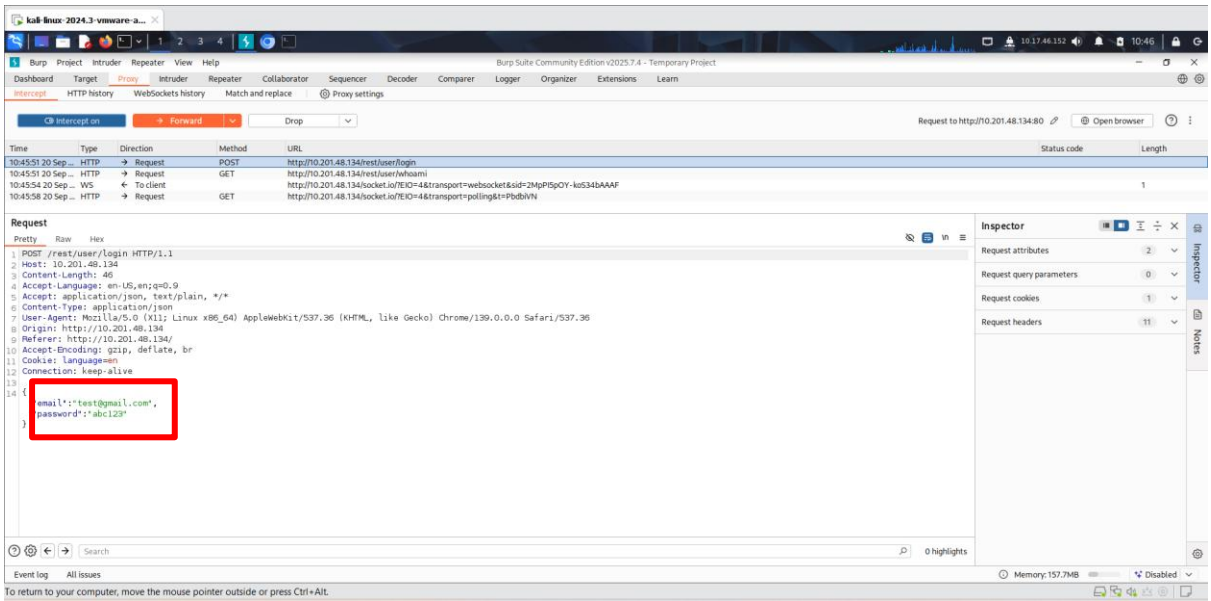


Step 1: Click on the Account section — the login page appears.

Step 2: Enter any random text in Email and Password fields (e.g., test@gmail.com and a).

Step 3: Open Burp Suite and go to Proxy → Intercept.

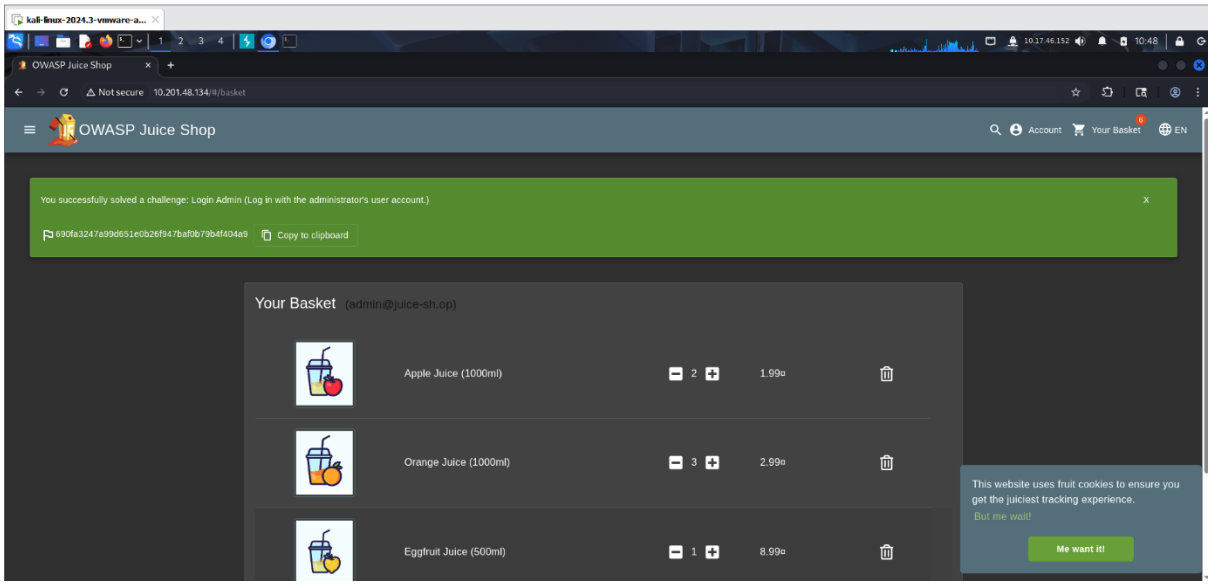
Step 4: Click the Login button on the webpage — Burp will intercept the request.



Step 5: In the intercepted request, replace the Email value with the SQL injection payload: **' or 1=1--**

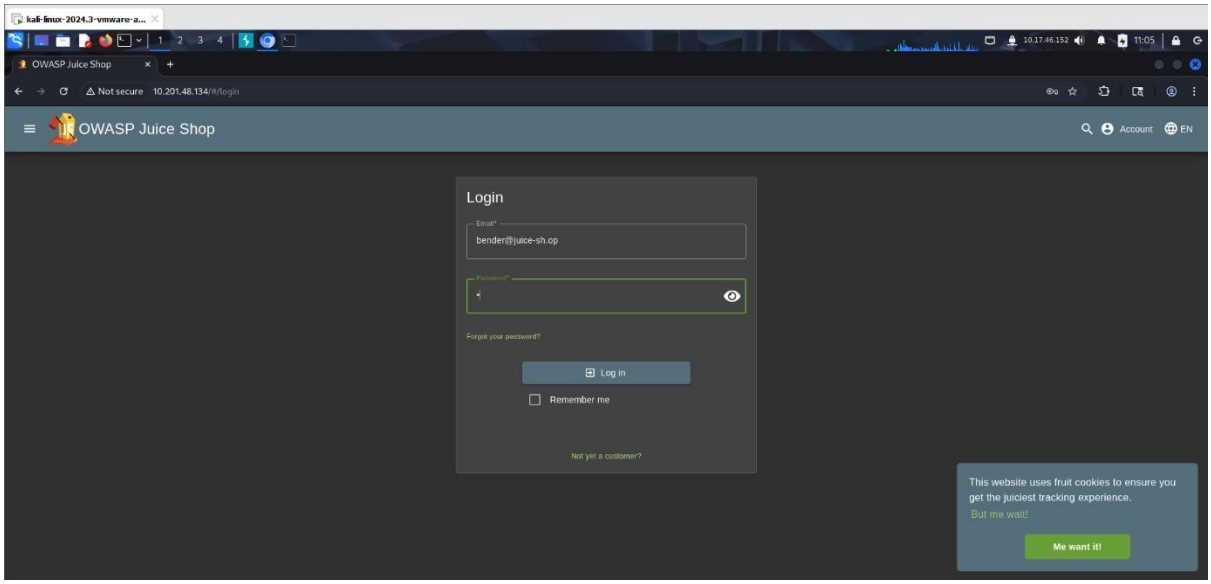
Leave the password as it is

Step 6: Forward the request in Burp. If successful, the login bypasses authentication and we can access the account to capture the flag.



Why: The payload ' or 1=1-- manipulates the SQL query to always return true, bypassing authentication. Burp intercepts the request so we can modify it before it reaches the server.

Question 2: Log into the Bender account!



Step 1: Open the Login page and enter:

Email: bender@juice-sh.op

Password: a

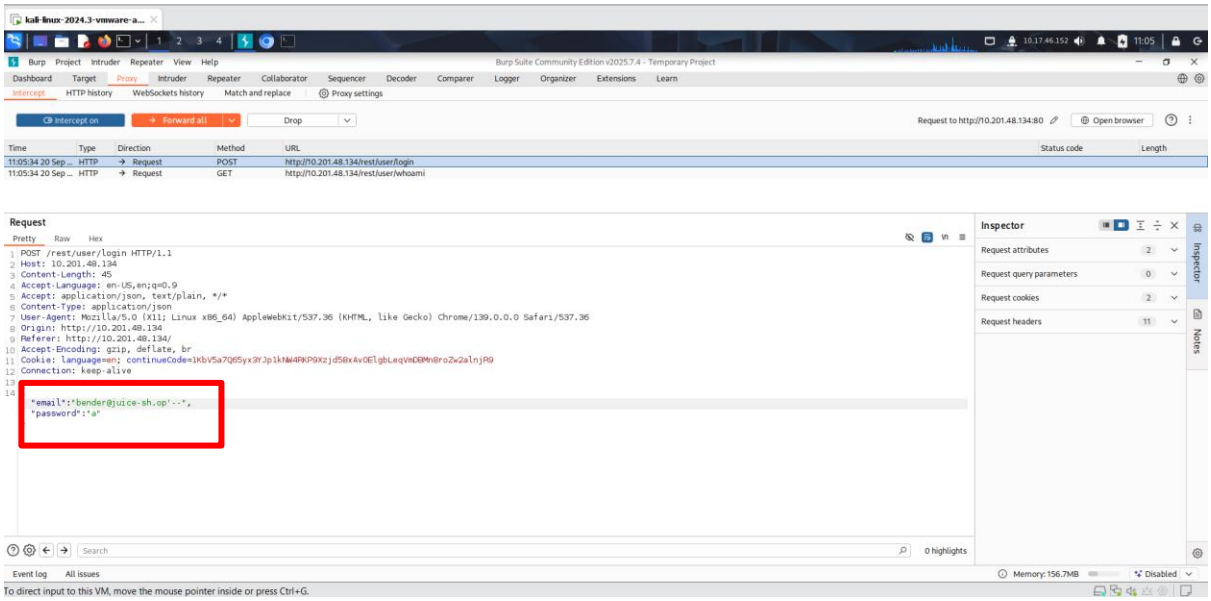
Step 2: Open Burp Suite and go to Proxy → Intercept.

Step 3: Click the Login button on the webpage — Burp will intercept the request.

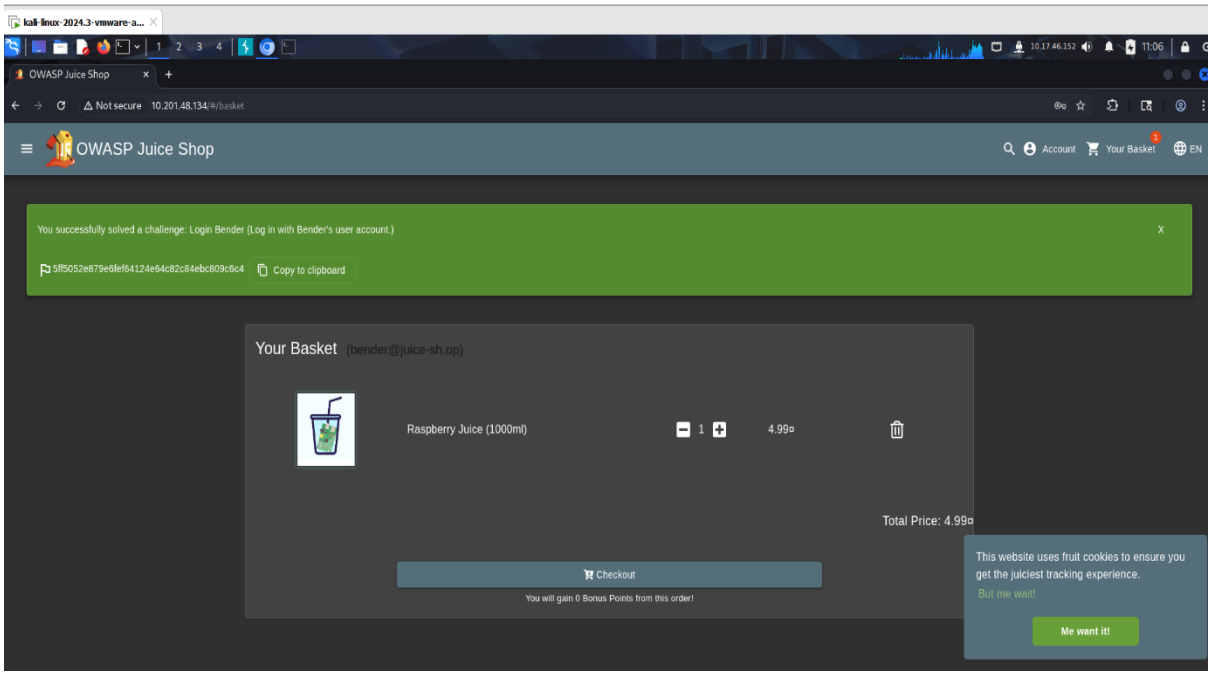
Step 4: In the intercepted request, modify the Email field to:

bender@juice-sh.op'--

Leave the password unchanged.



Step 5: Forward the request in Burp. If successful, the page will log in and display the flag.



Why: Adding '-- comments out the rest of the SQL query, bypassing password verification and allowing login.

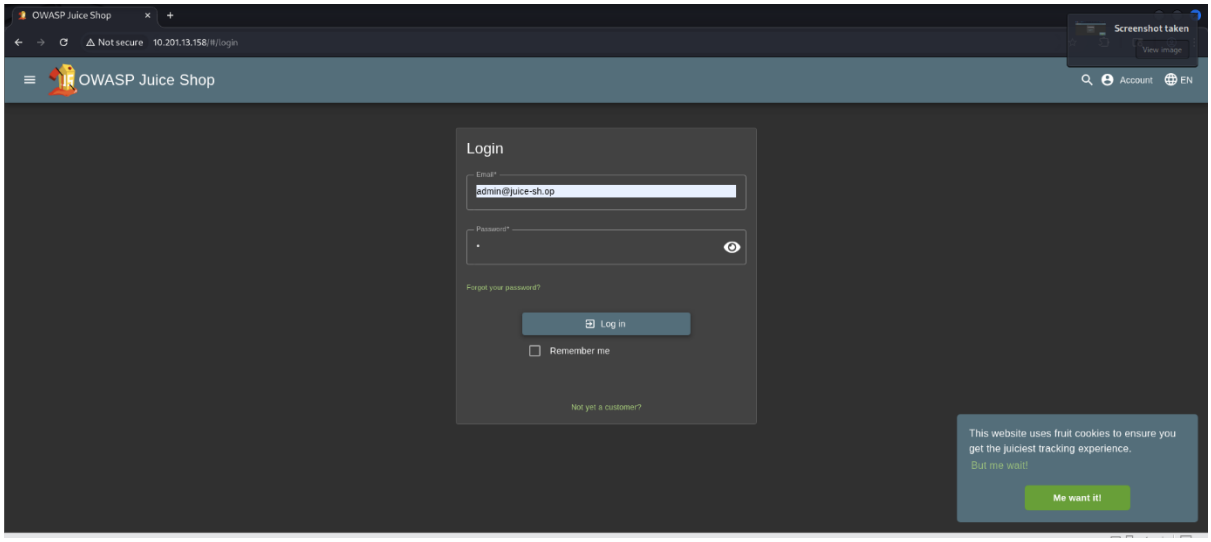
Task 4: Who broke my lock?!

Question 1: Brute force the Administrator account's password!

Step1: Open the Login page and enter:

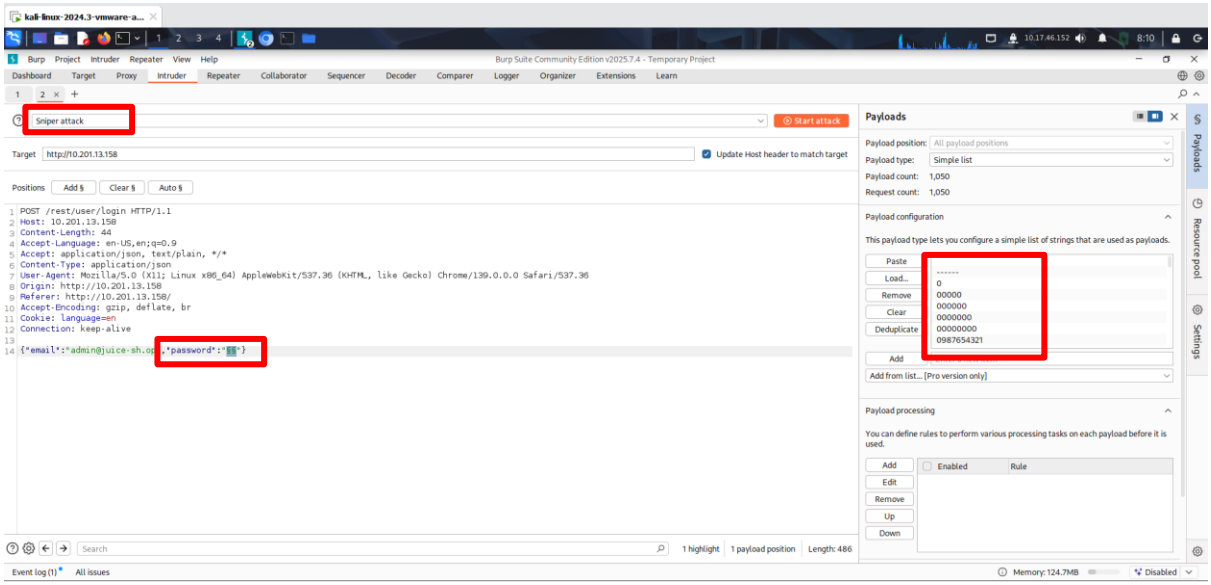
Email: admin@juice-sh.op

Password: a



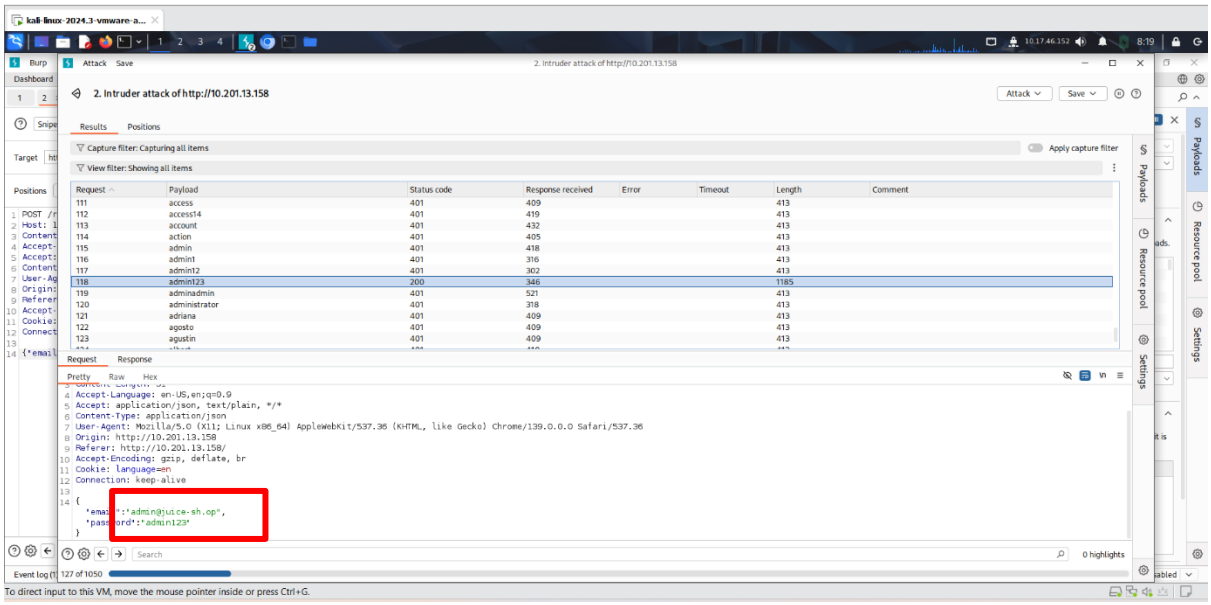
Step 2: Open Burp Suite and Intercept the login request.

Step 3: Send the intercepted request to Intruder and set the attack type to Sniper.

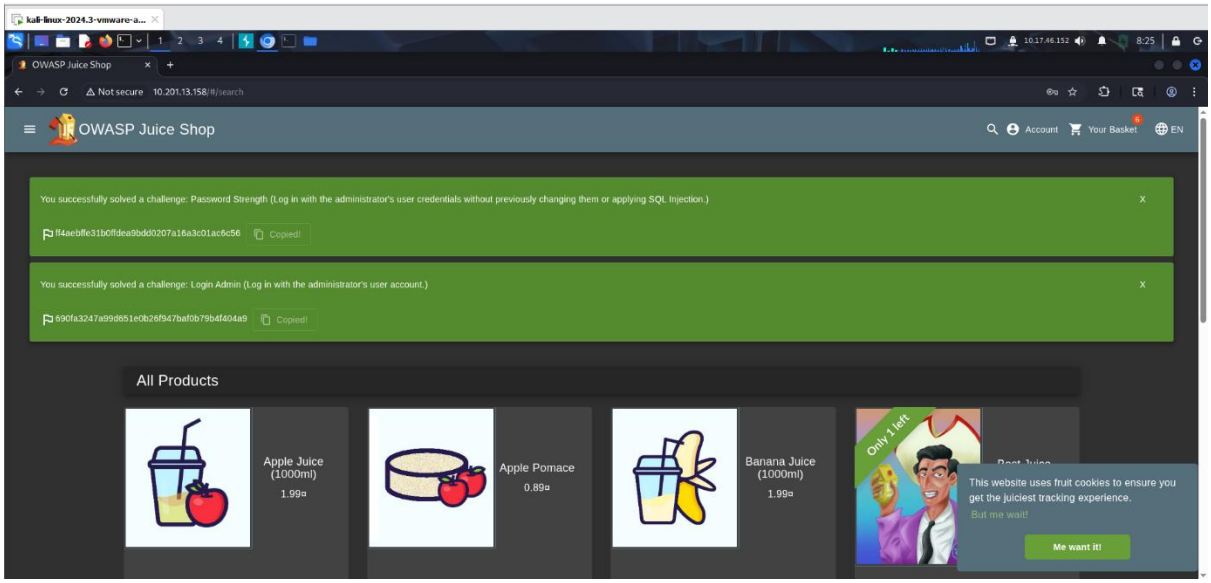


Step 4: Place the § marker on the Password field and load the payload list best1050.txt. Start the attack.

Step 5: After some time, the correct password is found: admin123



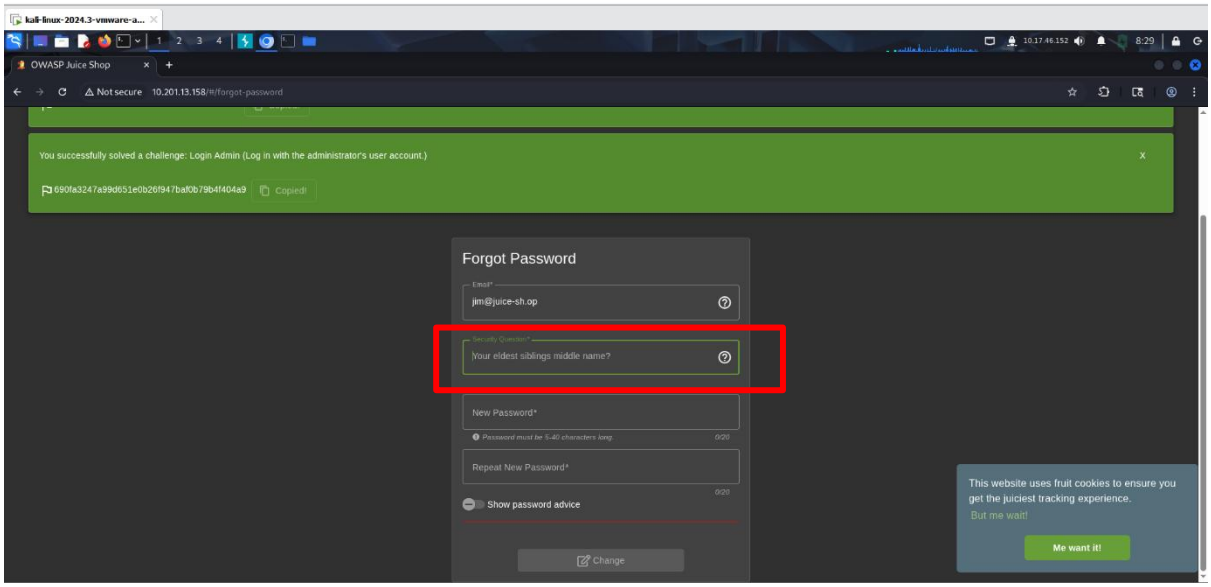
Step 6: Use this password to log in as admin and capture the flag.



Why: Sniper attack in Burp tests one parameter at a time with a list of possible values. Here, it brute-forces the password until it finds the correct one.

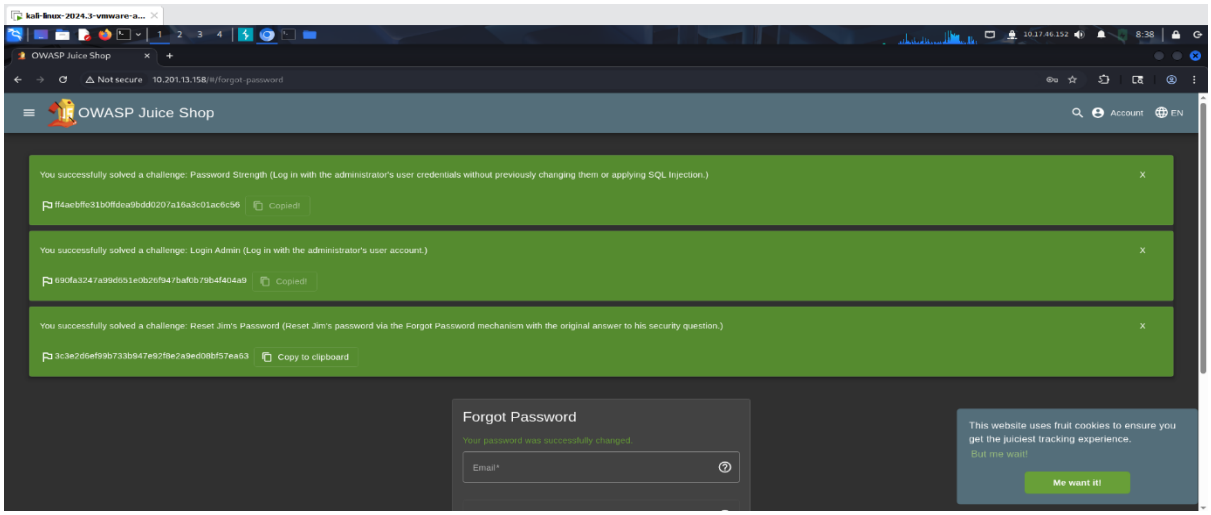
Question 2: Reset Jim's password!

Step 1: Go to the Forget Password page and enter the email: jim@juice-sh.op. The security question appears: Your eldest sibling's middle name?



Step 2: Research online (e.g., Google) about Jim from Star Trek to find the eldest sibling's middle name — it is Samuel.

Step 3: Enter Samuel in the security question and set a new password, e.g., Admin@123.



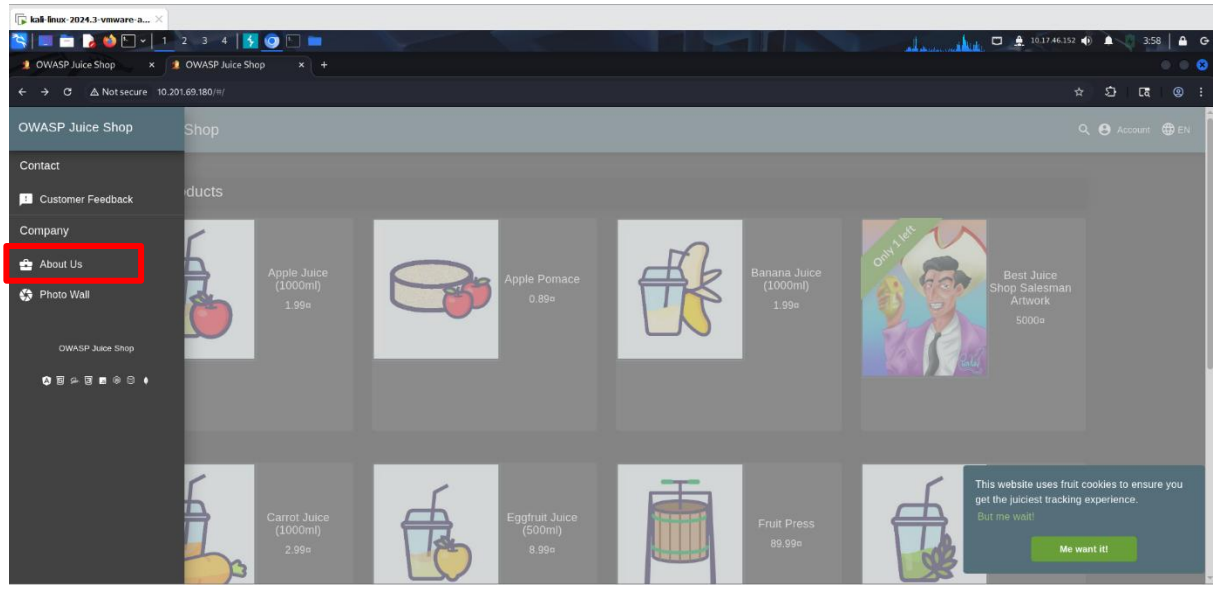
Step 4: Submit the form. The password is successfully changed, and the flag is revealed.

Why: Answering the security question correctly allows password reset and access to the account.

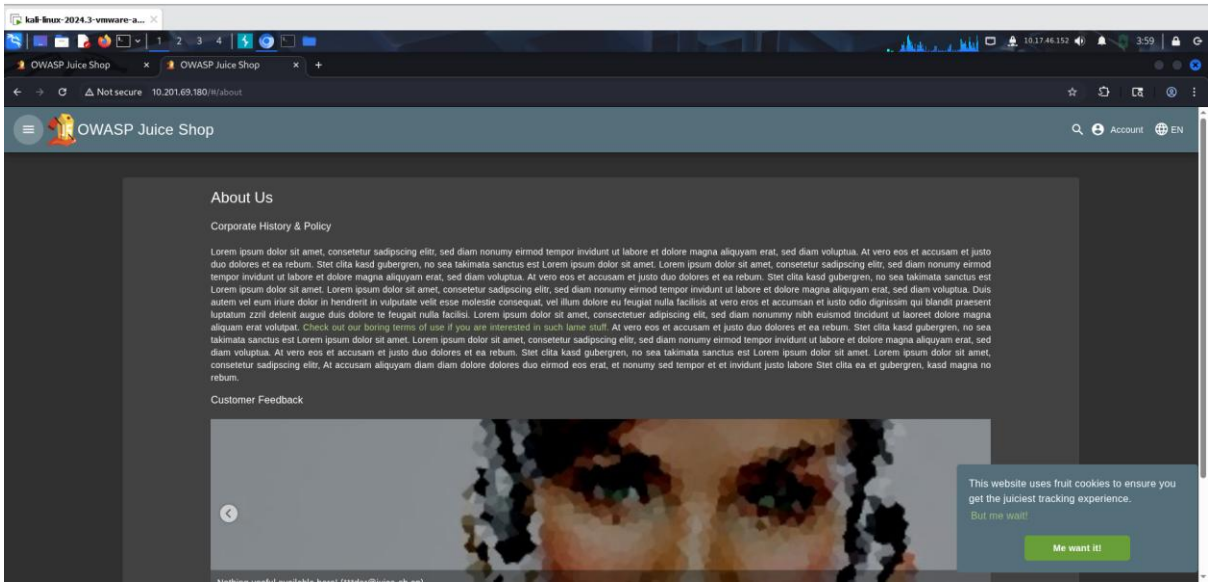
Task 5: AH! Don't look!

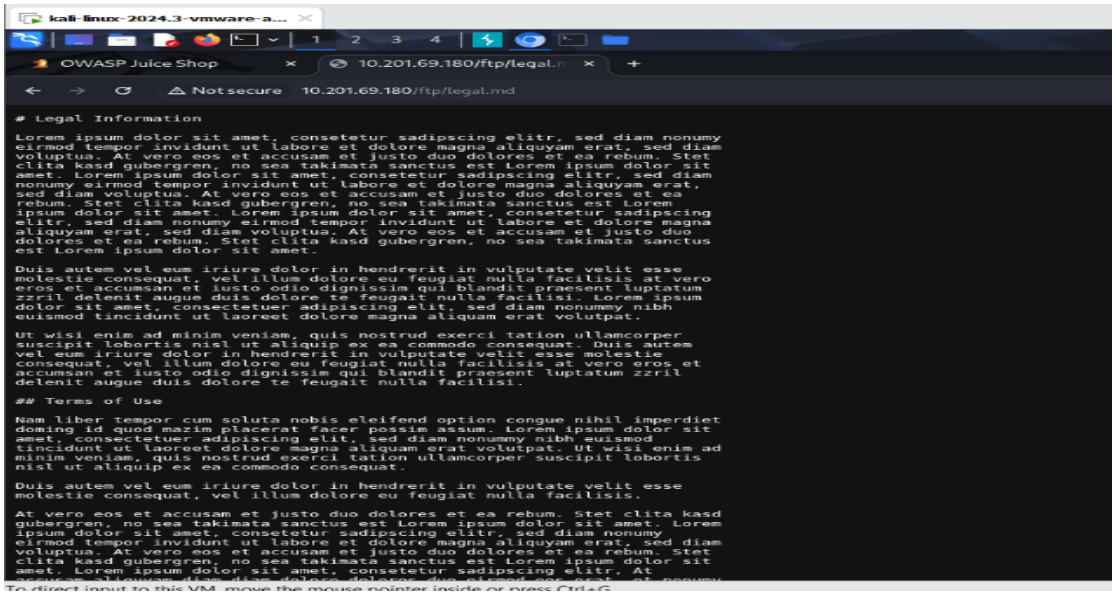
Question 1: Access the Confidential Document!

Step 1: Navigate to the about us page.

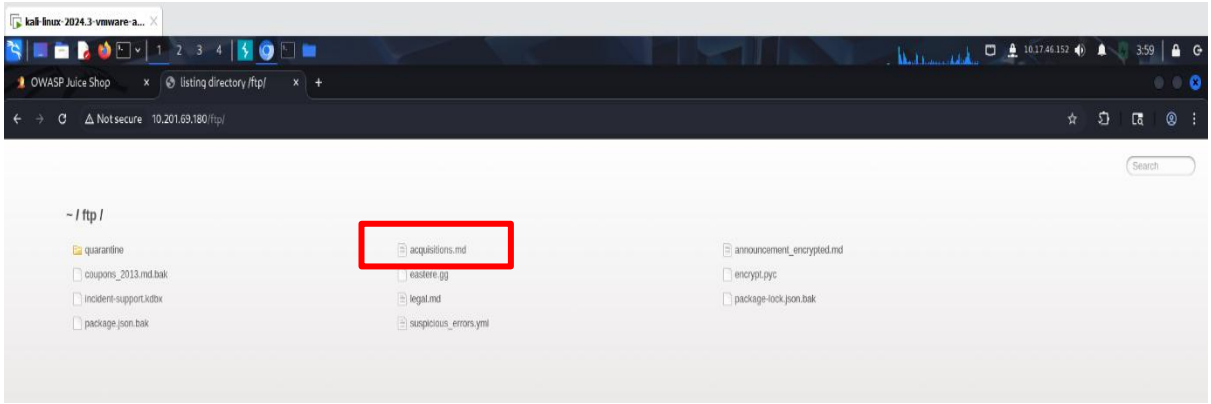


Step 2: There we find a link of http://10.201.69.180/ftp/legal.md. Then Navigating to that **/ftp/** directory reveals that it is exposed to the public.

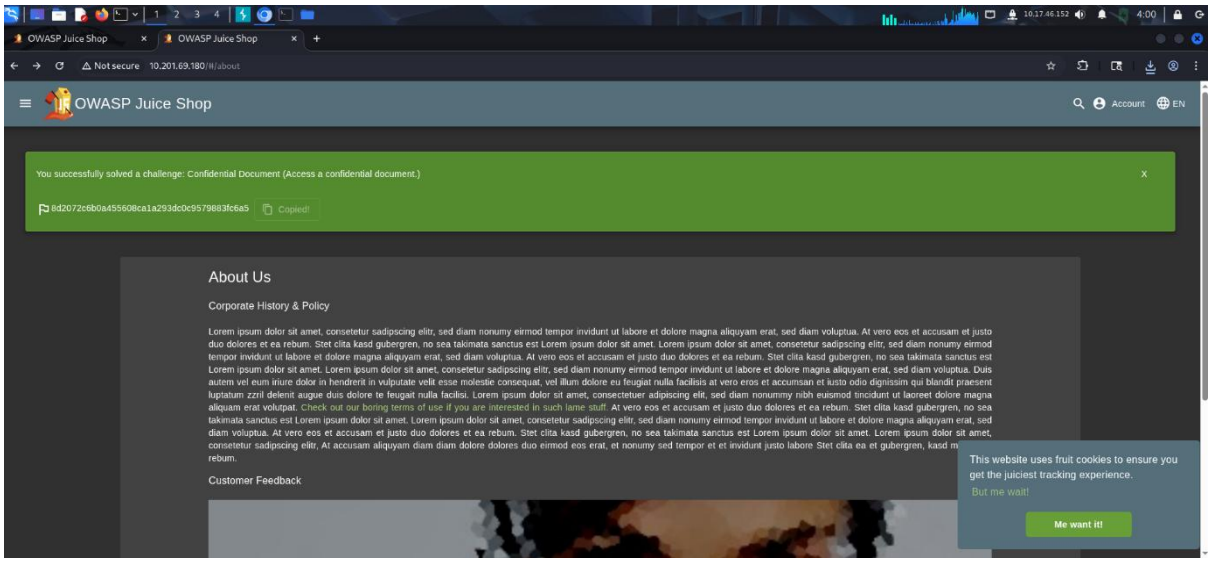




Step 3: the ftp page will show this files where we download the **acquisitions.md** and save it.

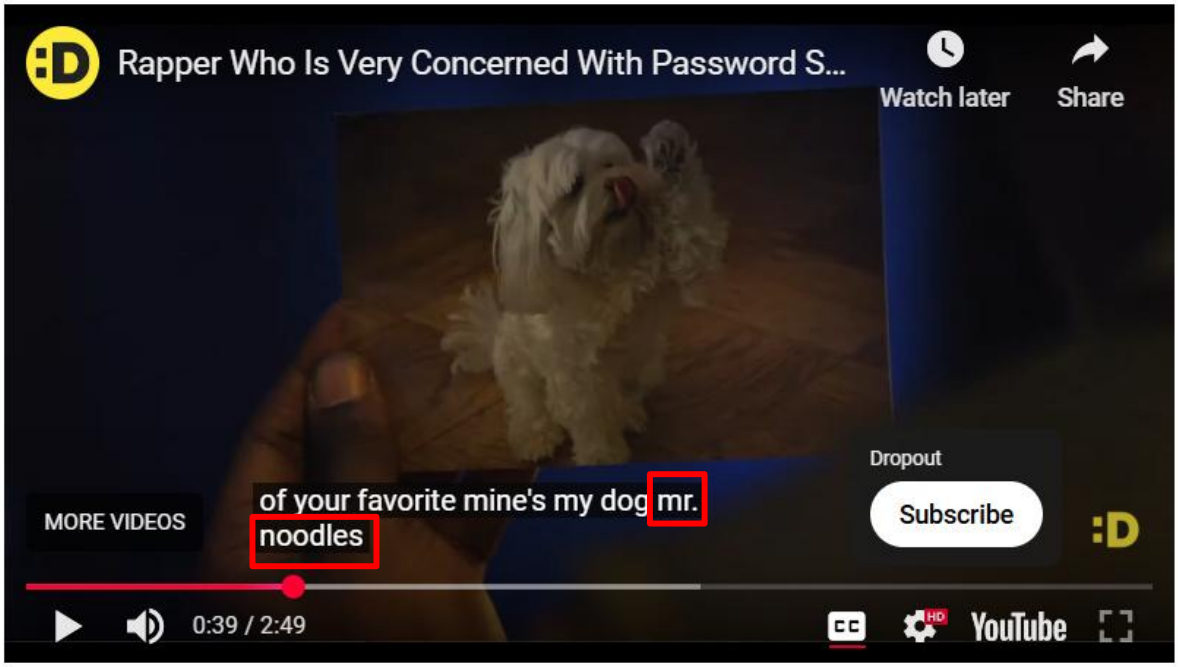


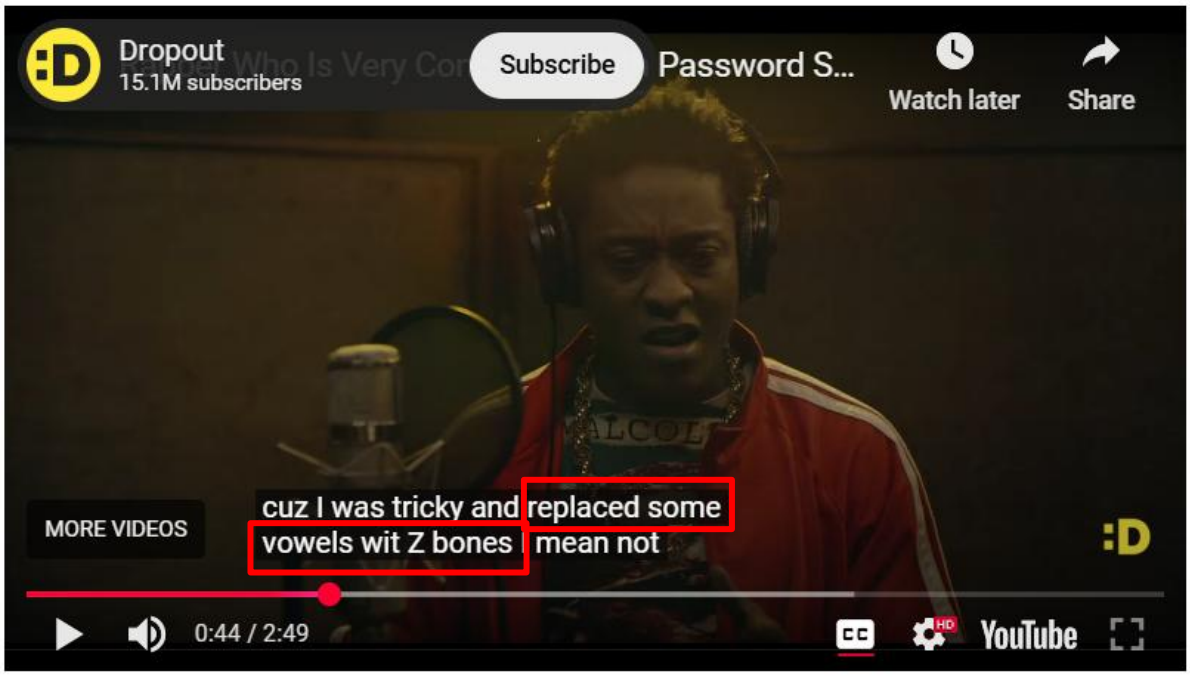
Step 4: After saving the file we get the flag.



Question 2: Log into MC SafeSearch's account!

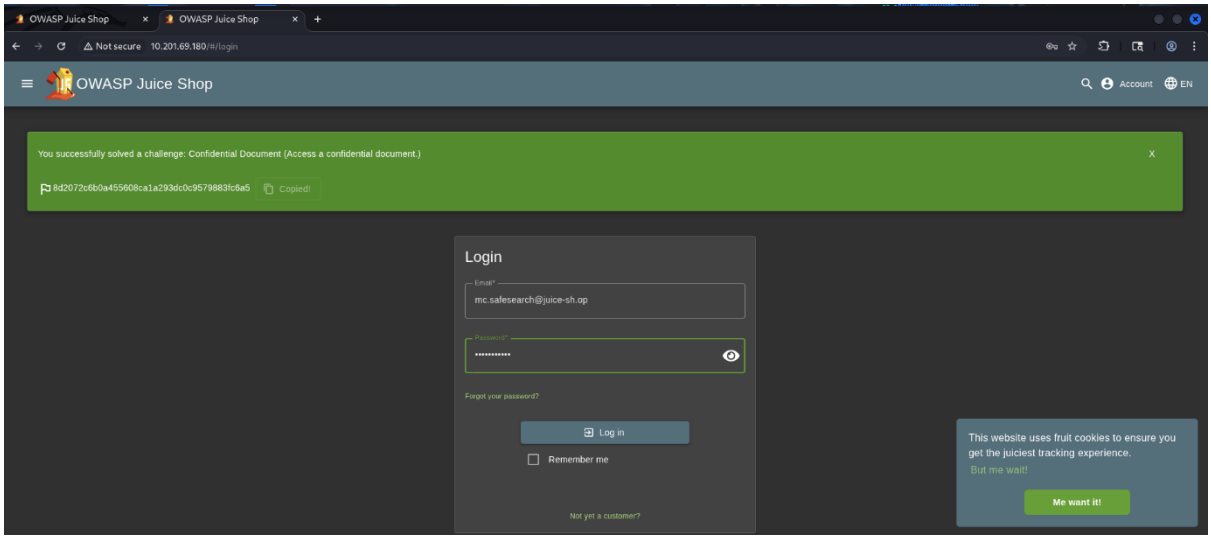
Step 1: In this task we have to login to the mc safesearch's account. For password he given a hint on the song. Which is mentioned in below image



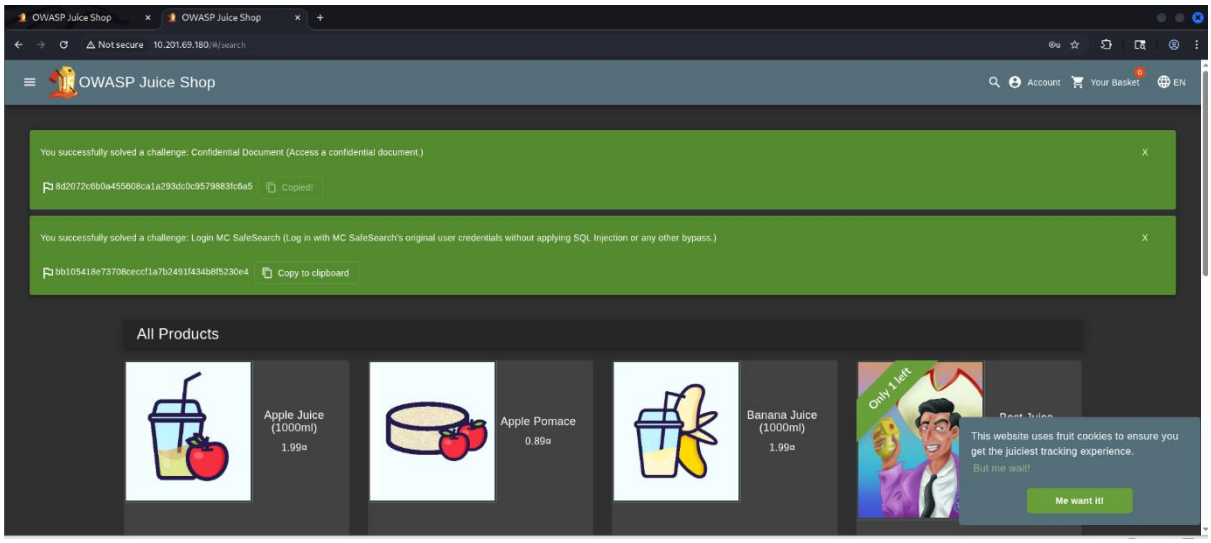


In this he mentioned that his dog name is the password and he replaced vowels with zeros. Which make password as "**Mr. N00dles**".

Step 2: now we use *mc.safesearch@juice-sh.op* as email id and password as "**Mr. N00dles**". And try to login the account.

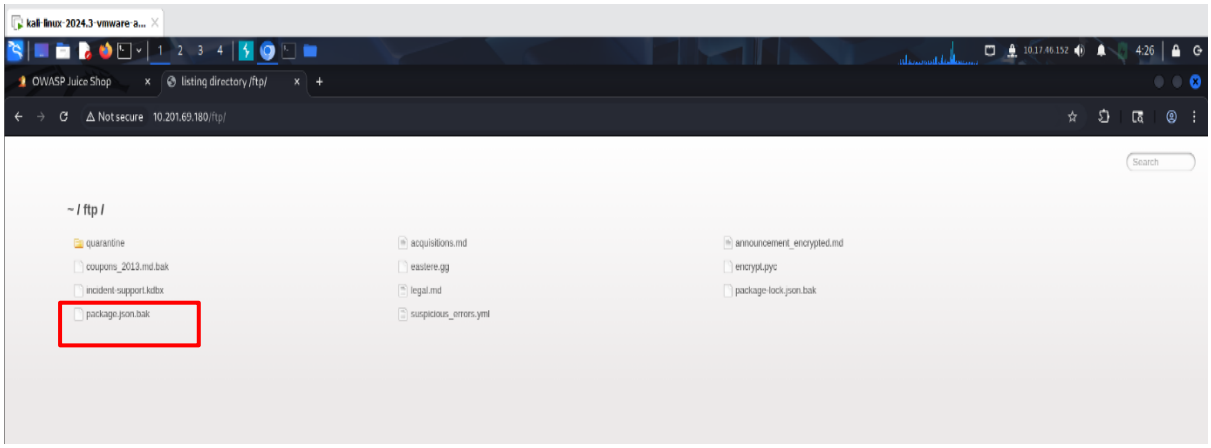


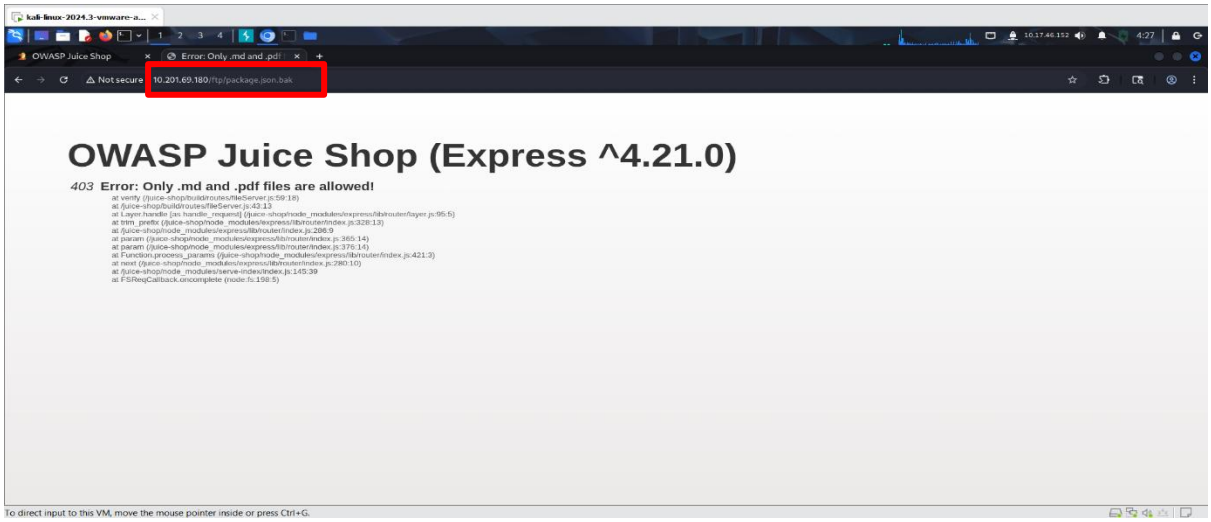
Step 3: Using these credentials we successfully login to the account and get the flag.



Question 3: Download the Backup file!

Step 1: We will now go back to the http://10.201.69.180/ftp/ folder and try to download **package.json.bak**. But it seems we are met with a 403 which says that only .md and .pdf files can be downloaded.

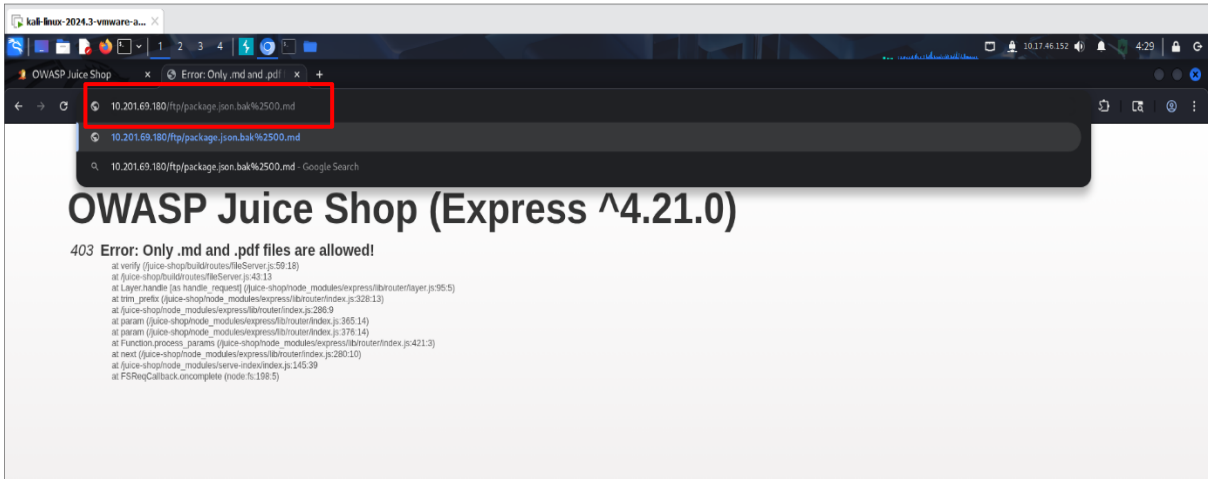




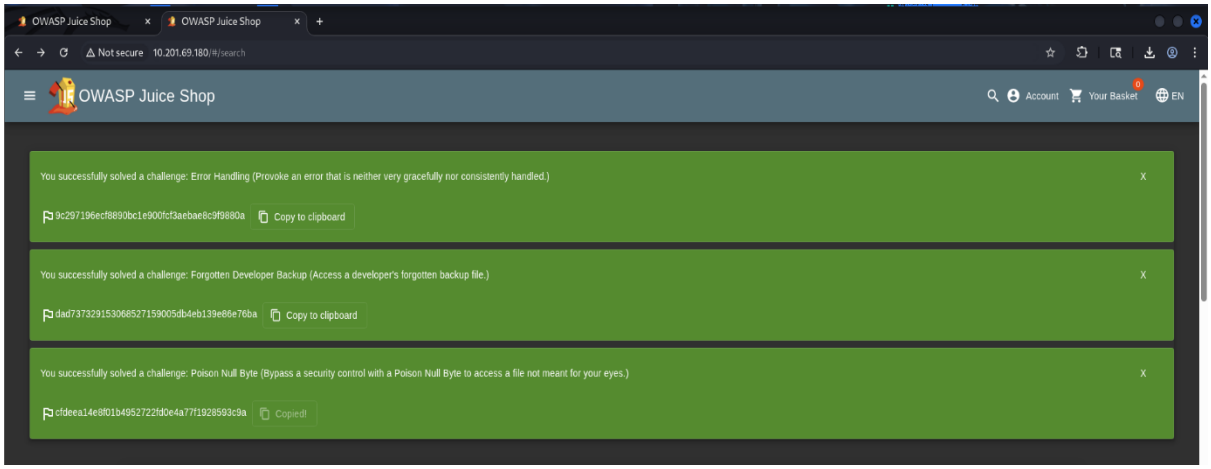
Step 2: To get around this, we will use a character bypass called "**Poison Null Byte**". A Poison Null Byte looks like this: ***%00***.

Note: as we can download it using the url, we will need to encode this into a url encoded format.

The Poison Null Byte will now look like this: ***%2500***. Adding this and then a **.md** to the end will bypass the 403 error!



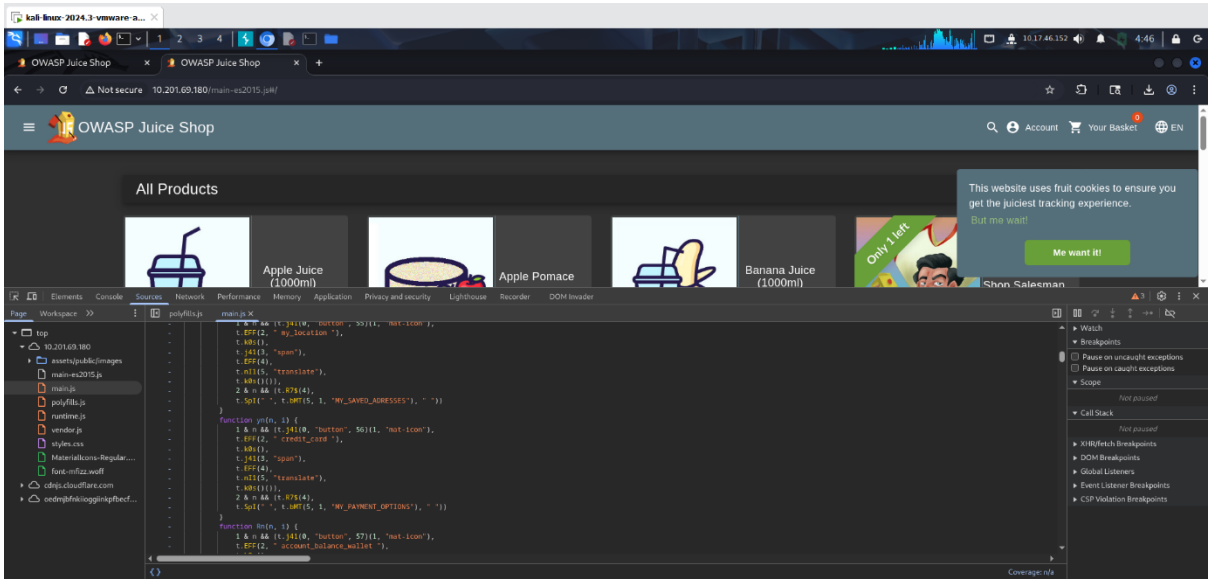
Step 3: after that we can get the flag.



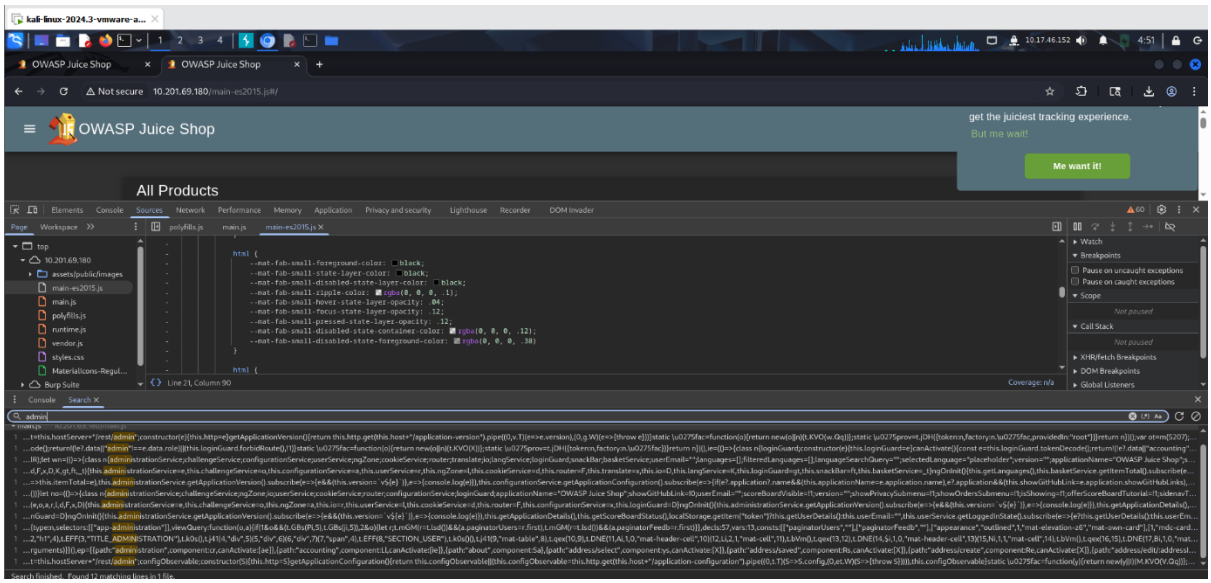
Task 6: Who's flying this thing?

Question 1: Access the administration page!

Step 1: Open Chrome → **Developer Tools** → **Sources**.



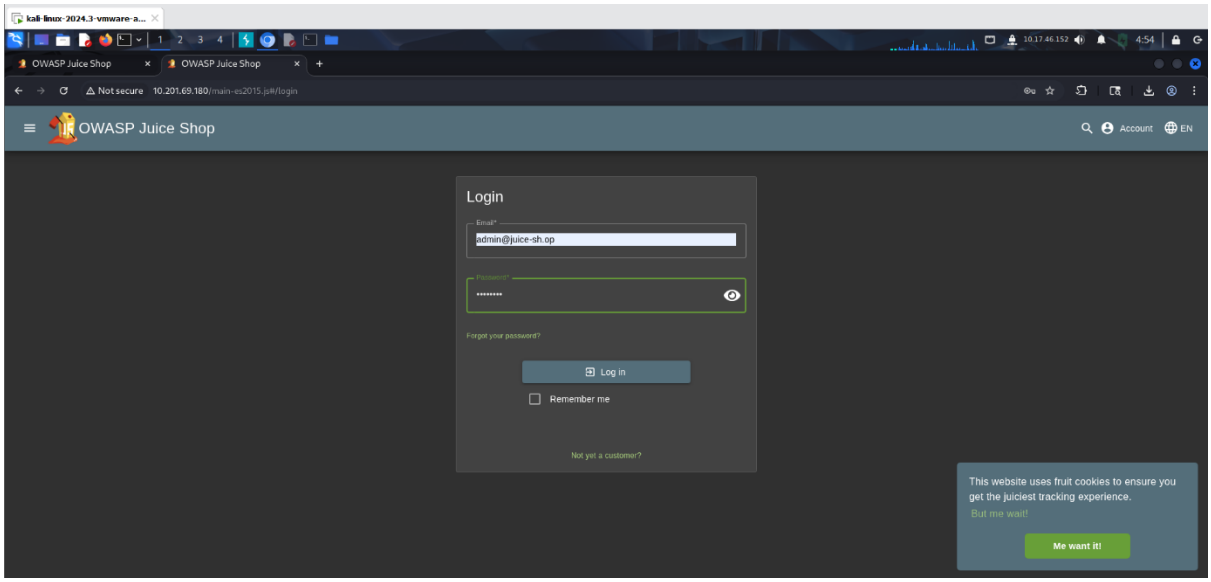
Step 2: Refresh the target page so the debugger/sources panel populates with the site's files.In the file list, find and open main-es2015.js.

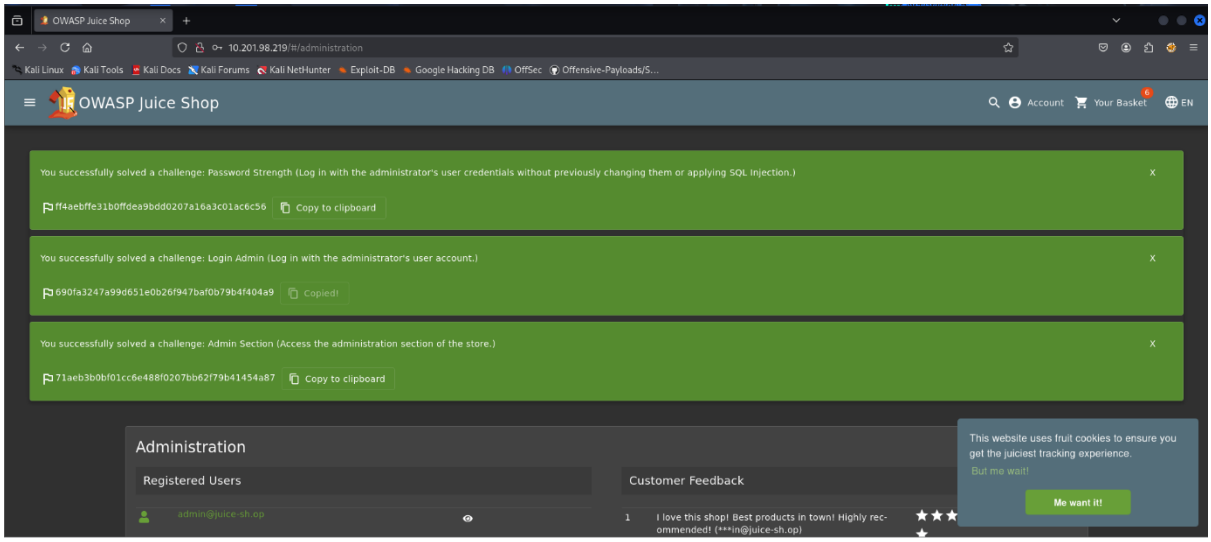


Step 3: Click the **{ }** button at the bottom to format the file into readable code. Use the search box (Ctrl/Cmd+F) and search for the term admin.

Step 4: Inspect the matches and locate the entry that reads path: administration (we can also see related about/path entries nearby). Note the implied route /#/administration from that path value.

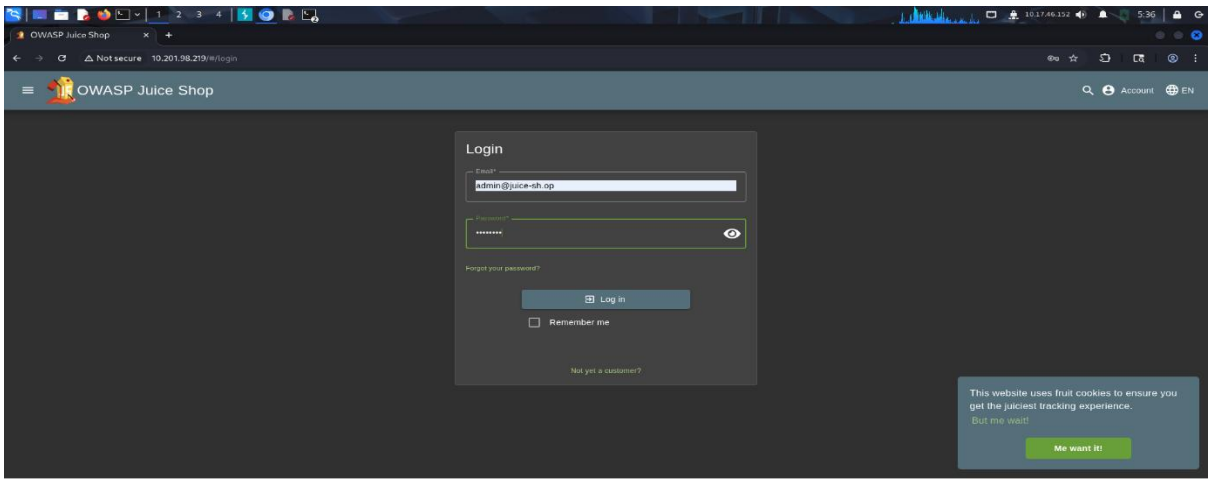
Step 5: After login. Try visiting http://10.201.98.219/#/administration in browser.





Question 2: View another user's shopping basket!

Step 1: Log in as **admin** and click **Your Basket** in the app.

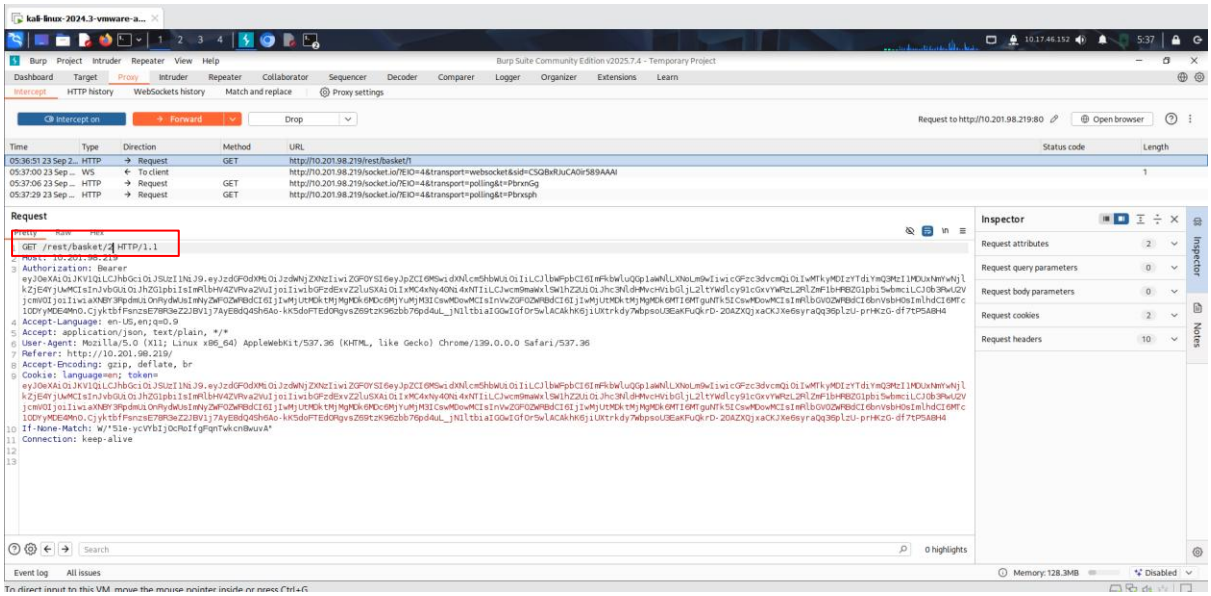


Step 2: Burp Suite is running and Proxy → Intercept is **On**.

Step 3: Click through/refresh the basket page and **forward** intercepted requests until we see a request like:

GET /rest/basket/1 HTTP/1.1

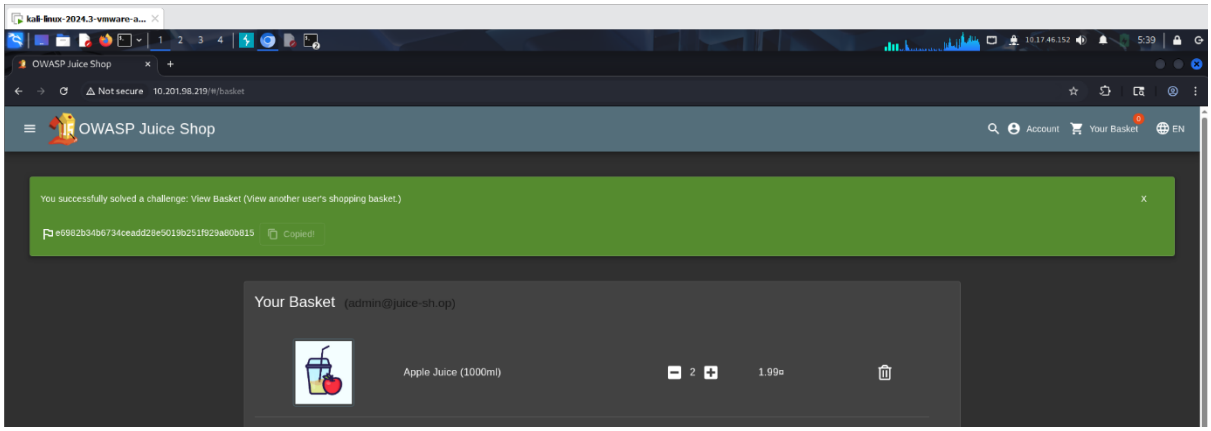
(this is the request that reads the basket for user id 1).



Step 3: Right-click the intercepted request → **Change Request** (or view it in the Intercept tab) and edit the path: replace /rest/basket/1 with /rest/basket/2. Forward the modified request to the server.

Step 4: Inspect the server response (in Burp's Response tab or in browser) — it should return the basket contents for **UserID 2**.

Step 5: we get the flag after that.



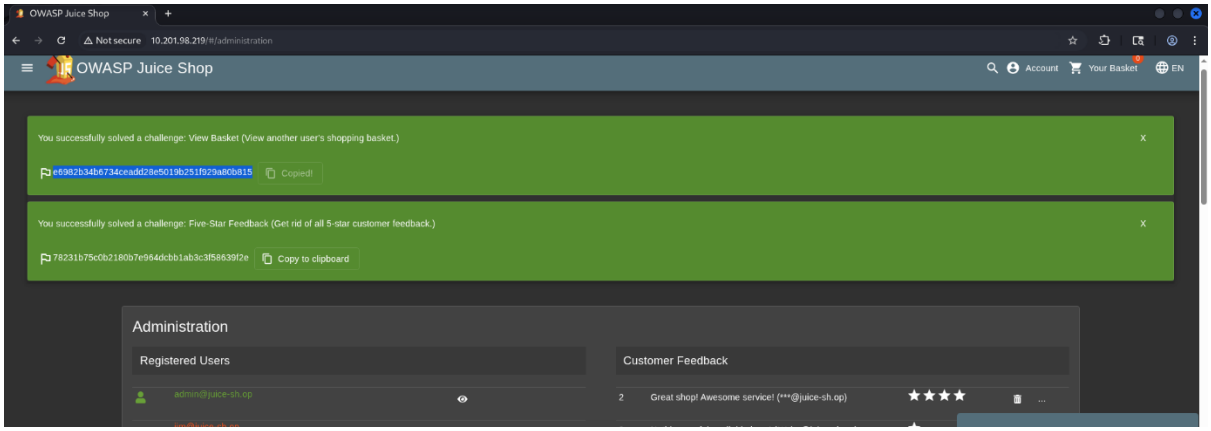
Why: the app exposes a numeric resource ID in the request path; by changing that ID we're requesting a different user's basket.

Question 3: Remove all 5-star reviews!

Step 1: Open browser and go to http://10.201.98.219/#/administration. Ensure we are logged in as **admin**.

Step 2: On the **Administration** page, find the **Reviews** section or reviews table/list. Locate the review that has **5 stars**.

Step 3: Click the **bin / trash** icon next to that 5-star review. After this we find the flag once page get reloaded.



Why: The bin icon issues a delete request for that review record; removing it can trigger the app to reveal state/flags in this lab.

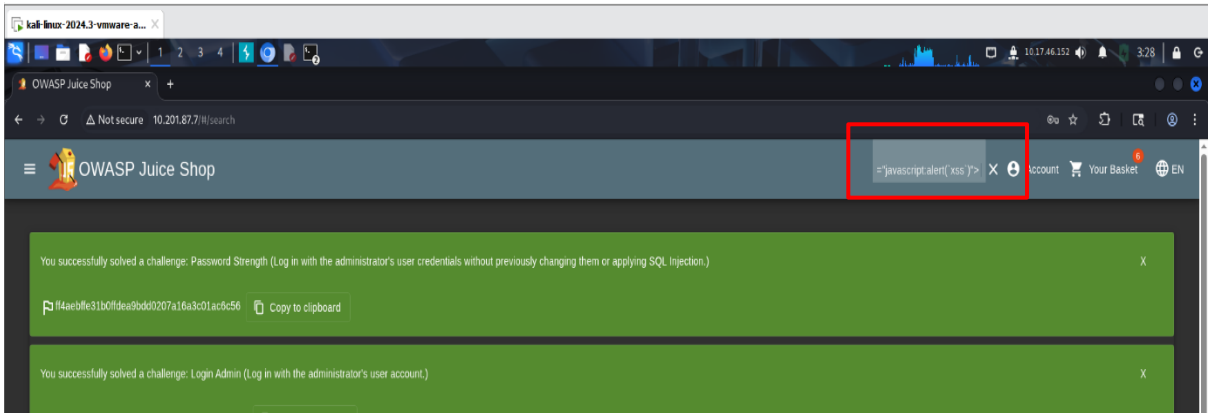
Task 7: Where did that come from?

Question 1: Perform a DOM XSS!

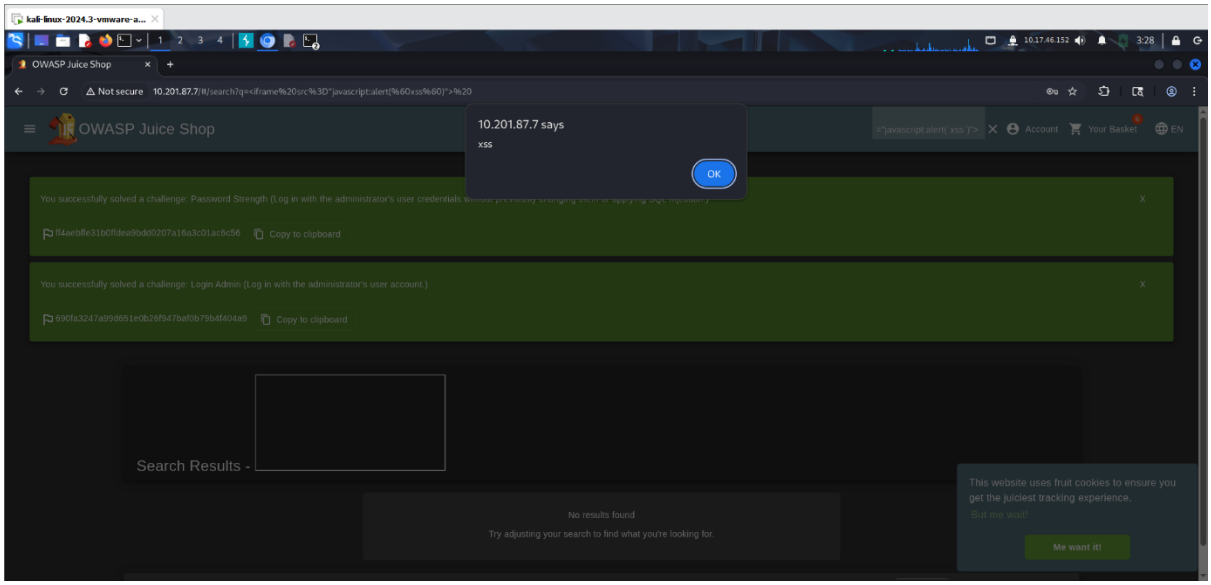
Step 1: Open the web application and locate the **search bar**.

Step 2: Input the following payload into the search bar:

<iframe src="javascript:alert(`xss`)">



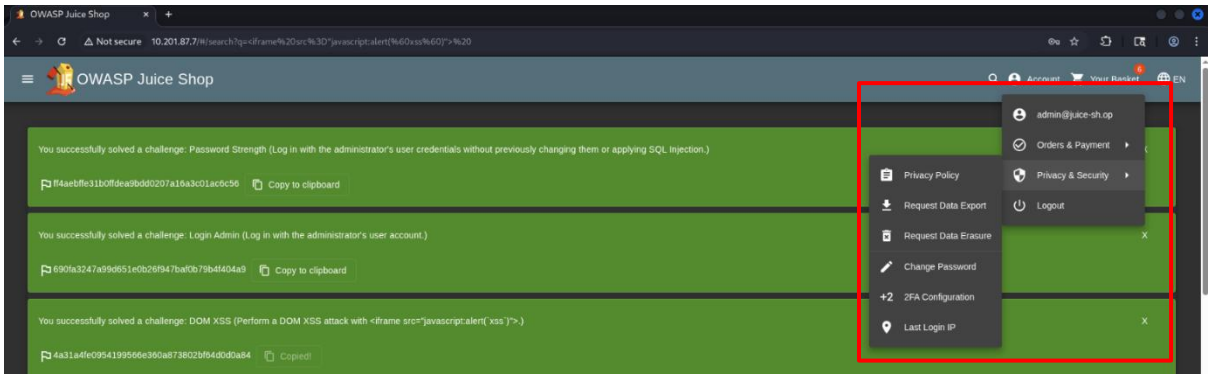
Step 3: Press **Enter** or submit the search. An alert box (xss) should pop up.

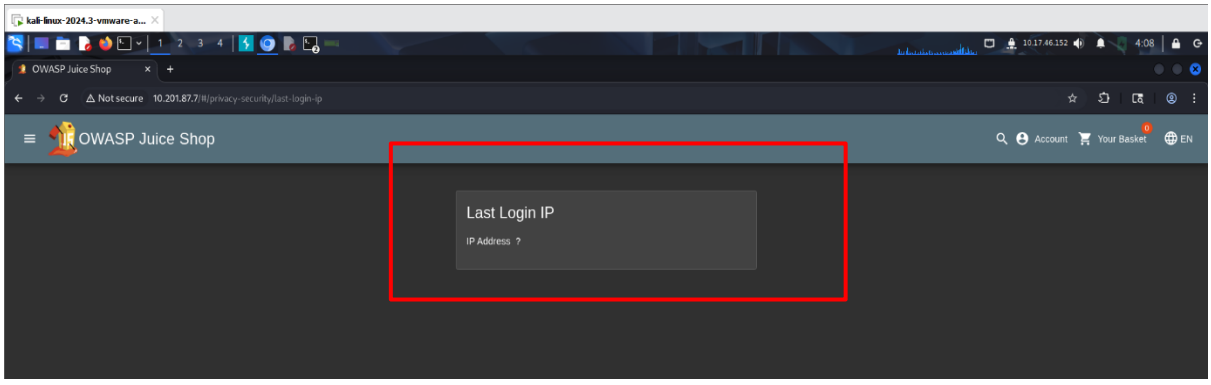


Question 2: Perform a persistent XSS!

Step 1: Login to the admin account.

Step 2: Navigate to the **Last Login IP** page. We should see the last IP address (e.g., 0.0.0.0 or 10.x.x.x).

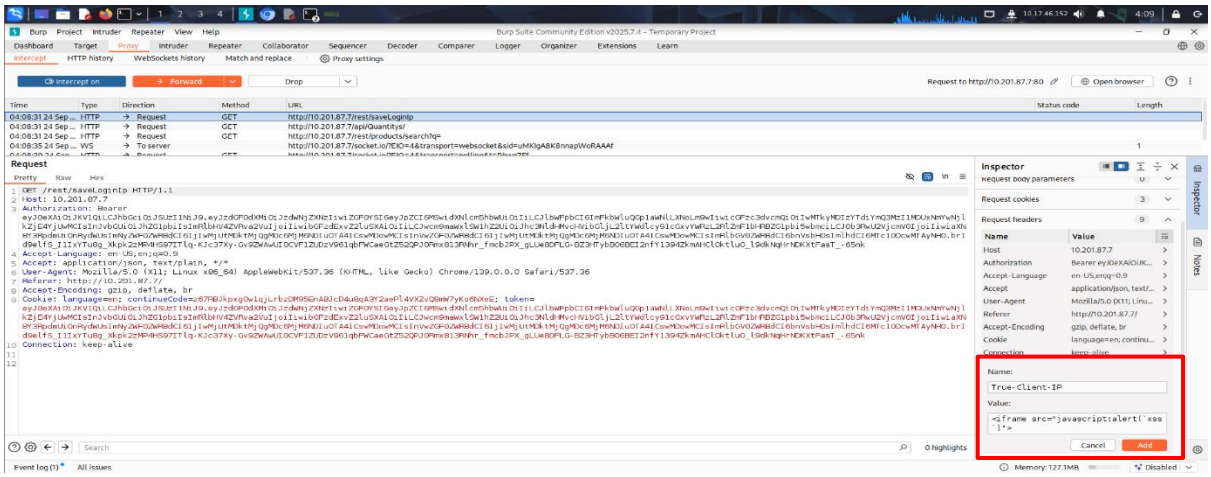


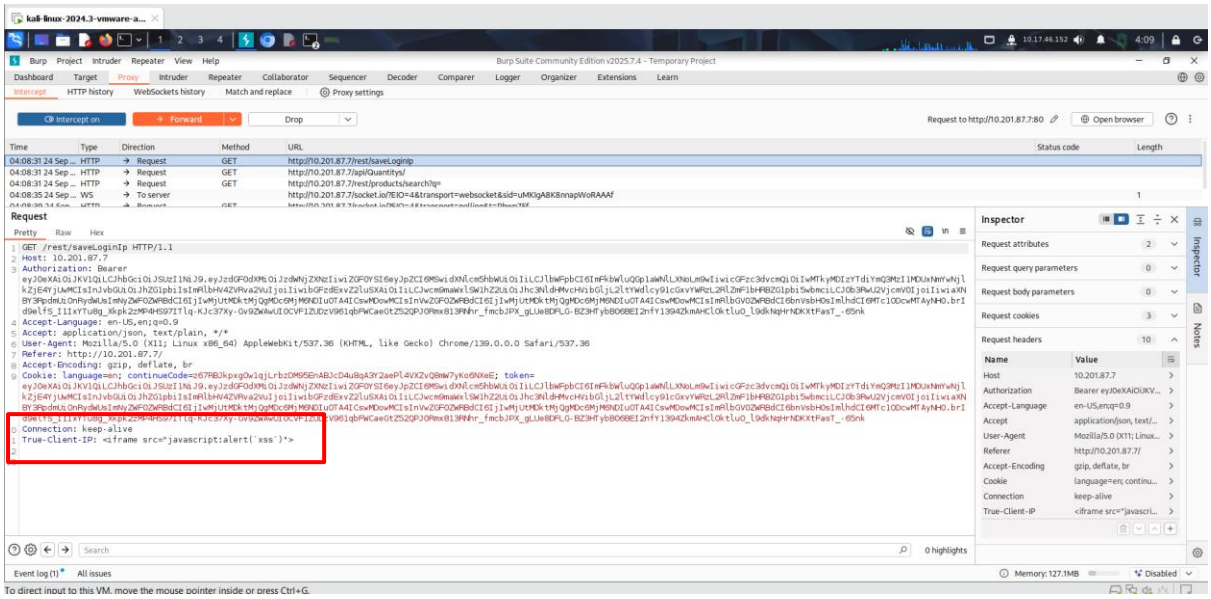


Step 3: Logout to trigger a new login record — this is needed so the server logs a new IP. Make sure **Burp Suite Intercept** is **On** to capture the logout request.

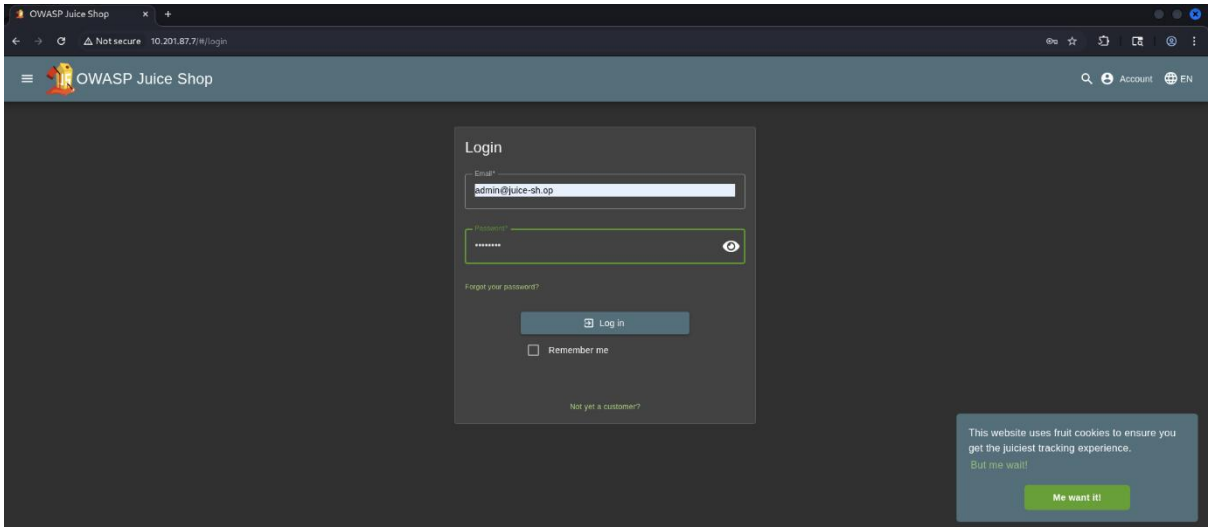
Step 4: In Burp, go to the **Headers** tab of the intercepted request and **add a new header**:

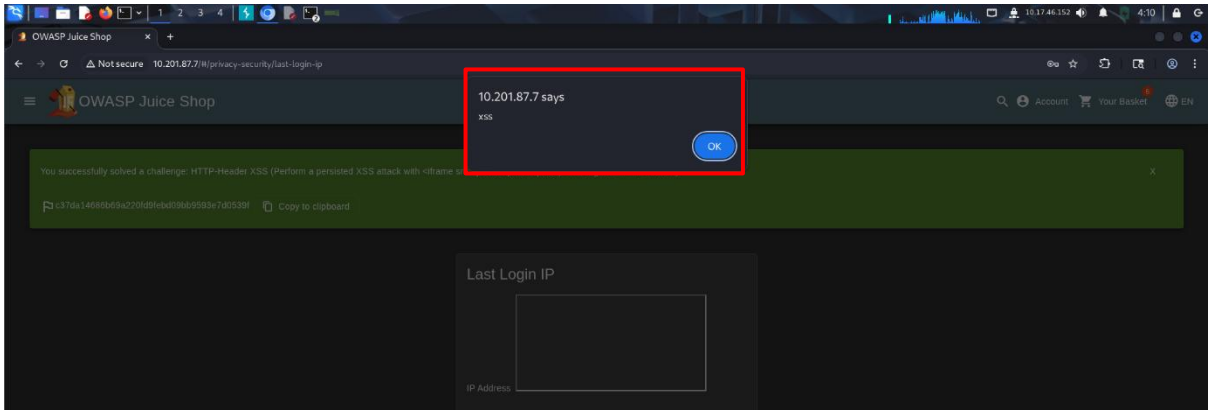
True-Client-IP: <iframe src="javascript:alert(`xss`)">





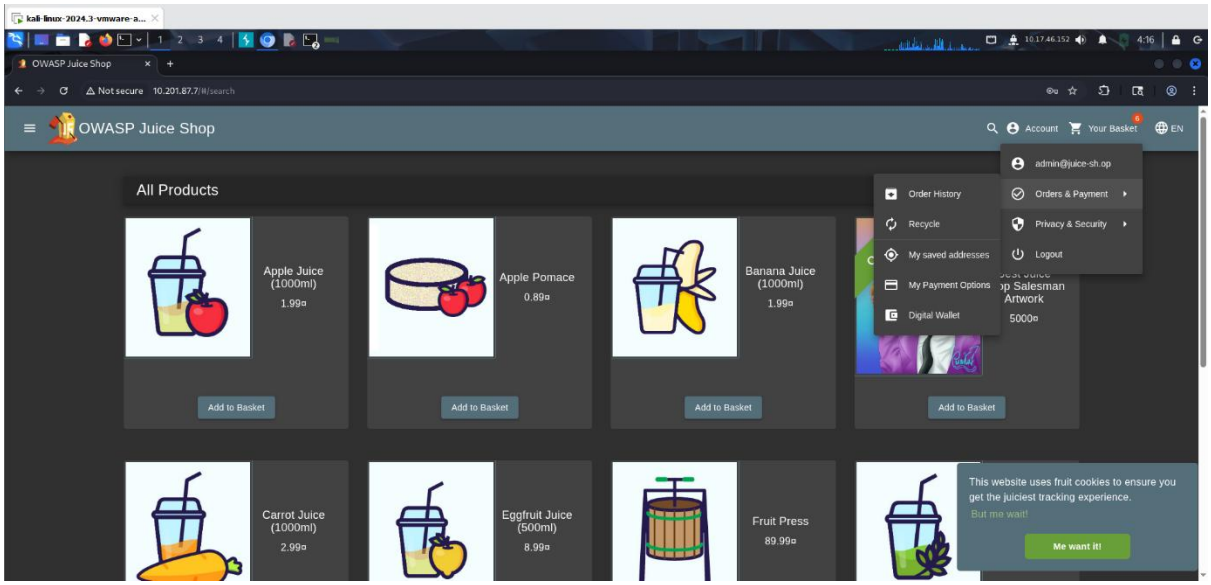
Step 5: Forward the modified request to the server. And Log back into the **admin account** and navigate again to the **Last Login IP** page. The XSS alert (xss) should trigger.



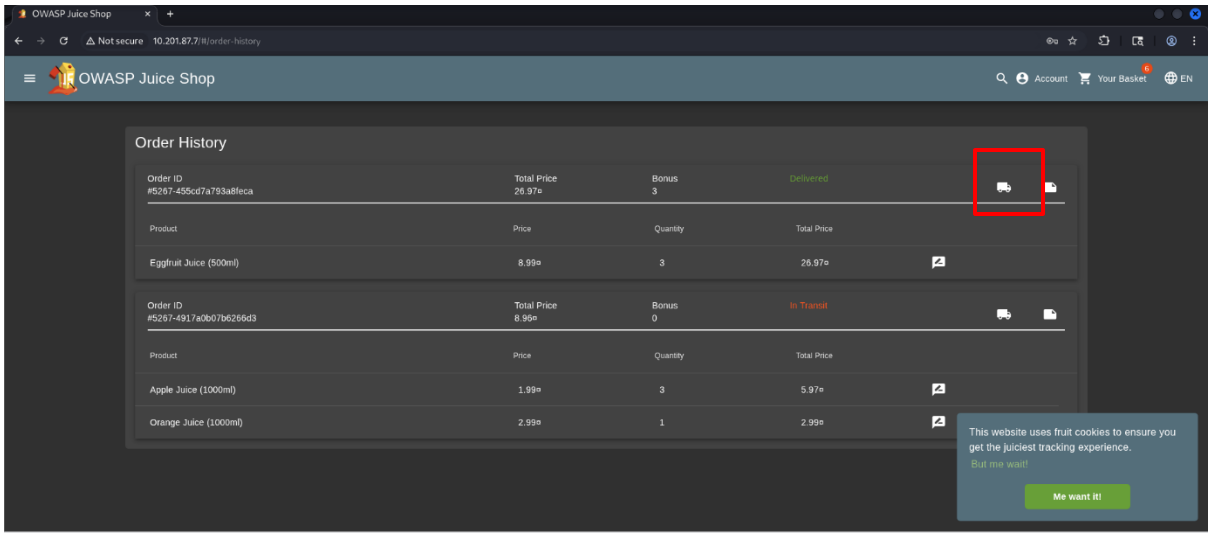


Question 3: Perform a reflected XSS!

Step 1: Login to the admin account. Navigate to the **Order History** page.

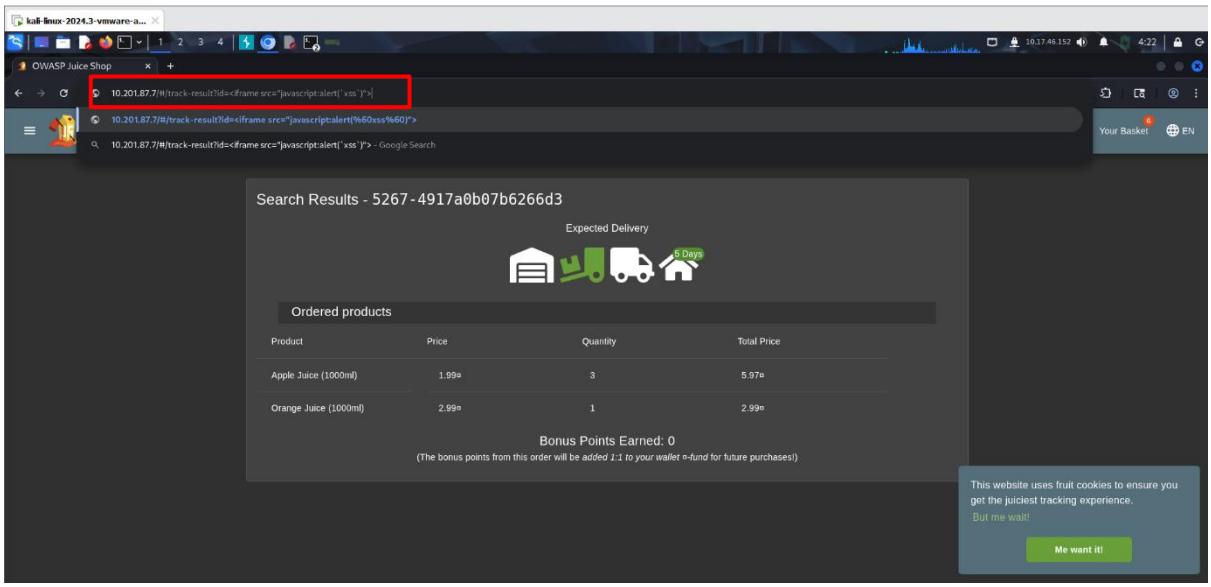


Step 2: Locate an order and click the **Truck** icon to open the **tracking result** page.

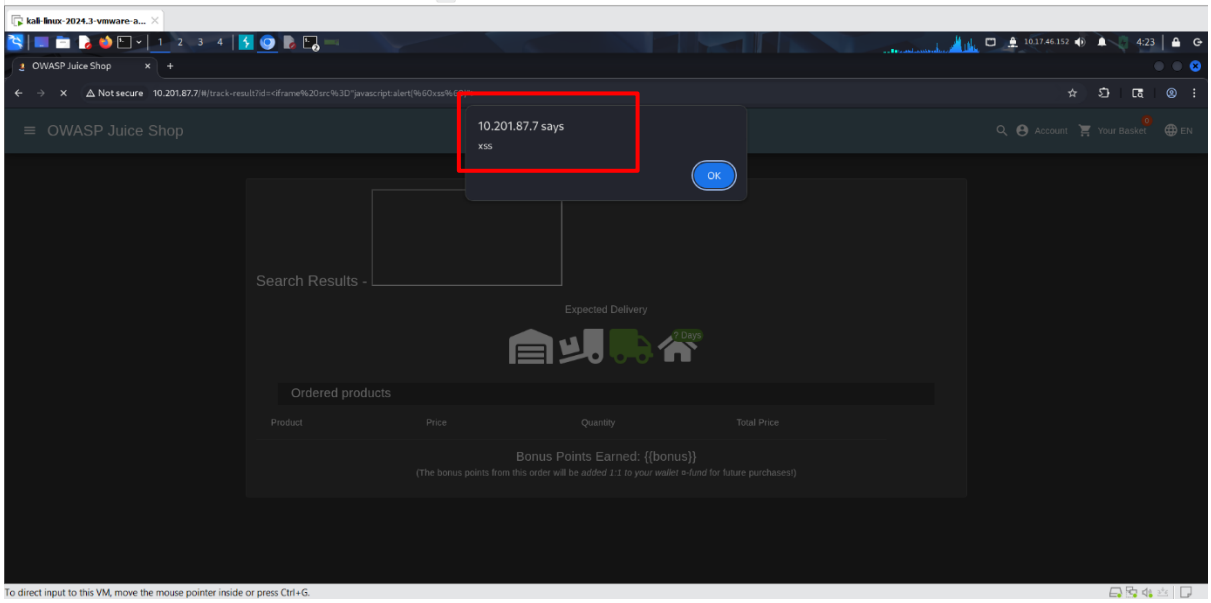


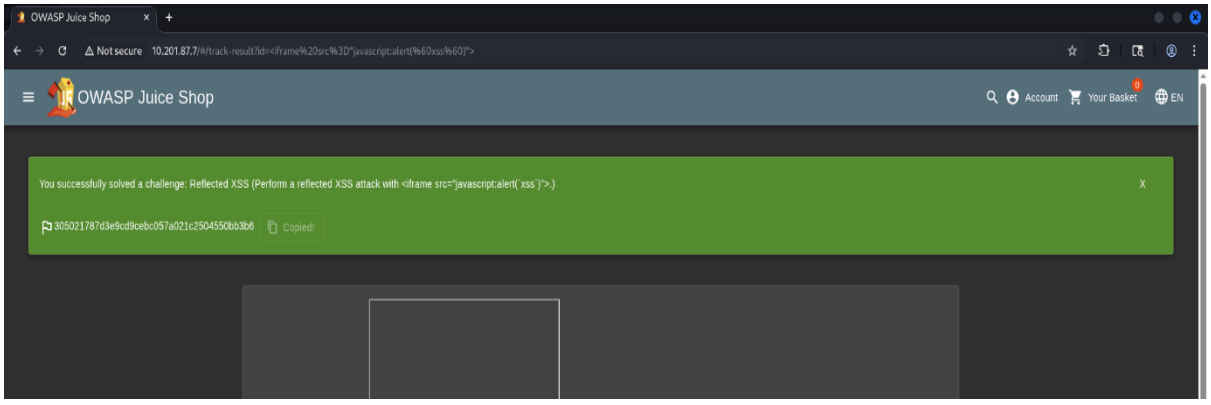
Step 3: Replace the tracking ID in the URL or input field with the XSS payload:

<iframe src="javascript:alert(`xss`)">



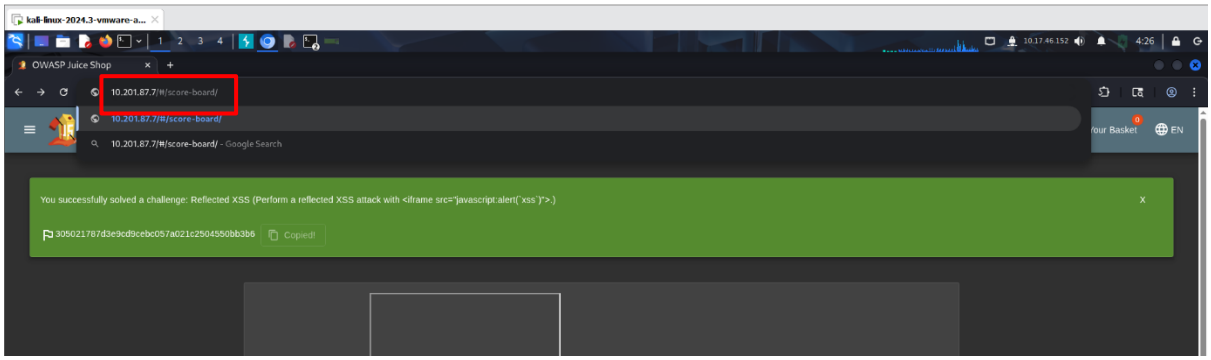
Step 4: Submit the URL/input. And refresh the page — an **alert box** with xss should appear, confirming the XSS vulnerability. Then we get the flag.



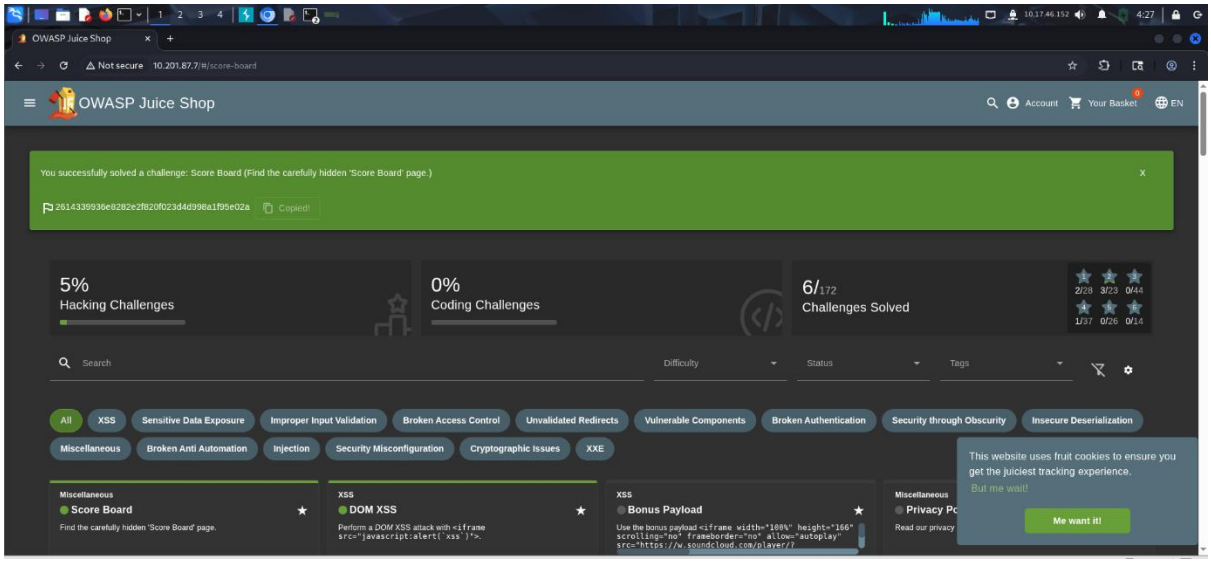


Task 8: Exploration!

Step 1: add this to the url **/#/score-board/** and press enter.



Step 2: It will show the Score Board. And we get the flag.



4. Conclusion

This assessment of OWASP Juice Shop demonstrated several high-risk vulnerabilities, including:

- SQL Injection (authentication bypass)
- Broken Authentication (brute-force, weak recovery)
- Insecure Direct Object Reference (IDOR)
- Cross-Site Scripting (XSS)
- Sensitive Data Exposure via Directory Traversal

The exercise highlights the importance of secure coding practices, especially in line with the OWASP Top 10. By applying input validation, strong authentication, least-privilege access, and secure coding frameworks, most of these flaws can be mitigated effectively.