

Chapter 15

1. Document vulnerabilities that you are able to successfully exploit on the server. Describe in detail what you did and what level of access you were able to obtain. If you obtain a user account with limited privileges, document whether you were able to escalate the privileges to root. Document each exploit that you are able to successfully launch.

- The purpose of this lab is to gain knowledge about the significance and necessity of carrying out web application penetration testing. I also learned how penetration testers and other cybersecurity experts find security flaws in online applications with the aid of the OWASP Top 10. Additionally, using the OWASP, I learned how to execute vulnerability detection and exploitation on a web application. Doing this, I will incorporate tools such as Burp Suite and OWASP Juice Shop.
- **Tools used:**
- Foxy Proxy Is an add-on for your web browser that lets you set up several profiles for different web proxy configurations, making it easy to switch between proxies without having to manually adjust them.
- Burp Suite Is a proxy-based tool that enables penetration testing professionals to change request messages from the client side by intercepting communication between the attacker's web browser and the online application.
- Kali Linux
- **Critical security risks and vulnerabilities:**
- Broken access control
- Cryptographic failures
- Injection
- Insecure design
- Security misconfiguration
- Vulnerable and outdated components
- Identification and authentication failures
- Software and data integrity failures
- Security logging and monitoring failures
- Server-side request forgery
- As a result, I learned the principles of web applications and the way that HTTP functions between a web browser and a web application during this chapter. Additionally, the following exercises possessed me with the ability to model different kinds of online application cyberattacks on susceptible apps in order to identify and take advantage of security holes in a target.

- Getting started with FoxyProxy and Burp Suite
- To initiate this lab, I went through the entire process to set up Foxy Proxy. Setting up and verifying all configurations are checked off is crucial in this lab, as a mistake will result in issues.

Add Proxy

Title or Description (optional)
Burp Suite Proxy

Proxy Type
HTTP

Color
#66cc66

Proxy IP address or DNS name ★
127.0.0.1

Port ★
8080

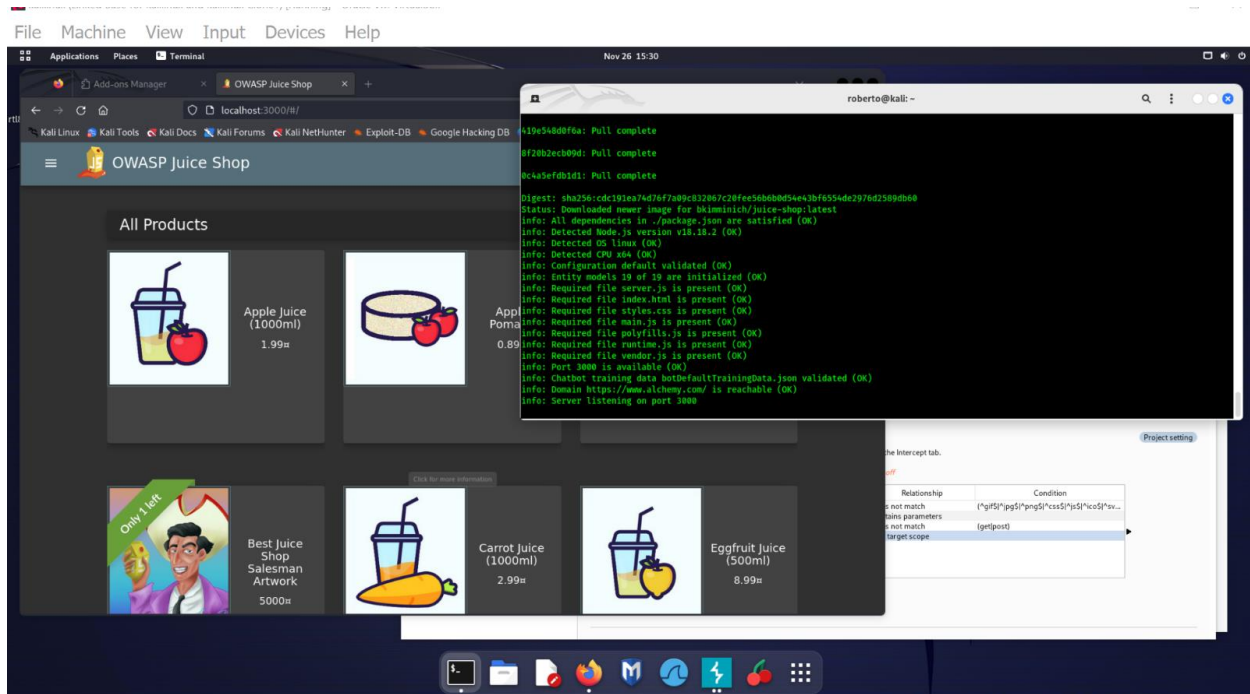
Username (optional)
username

Password (optional)

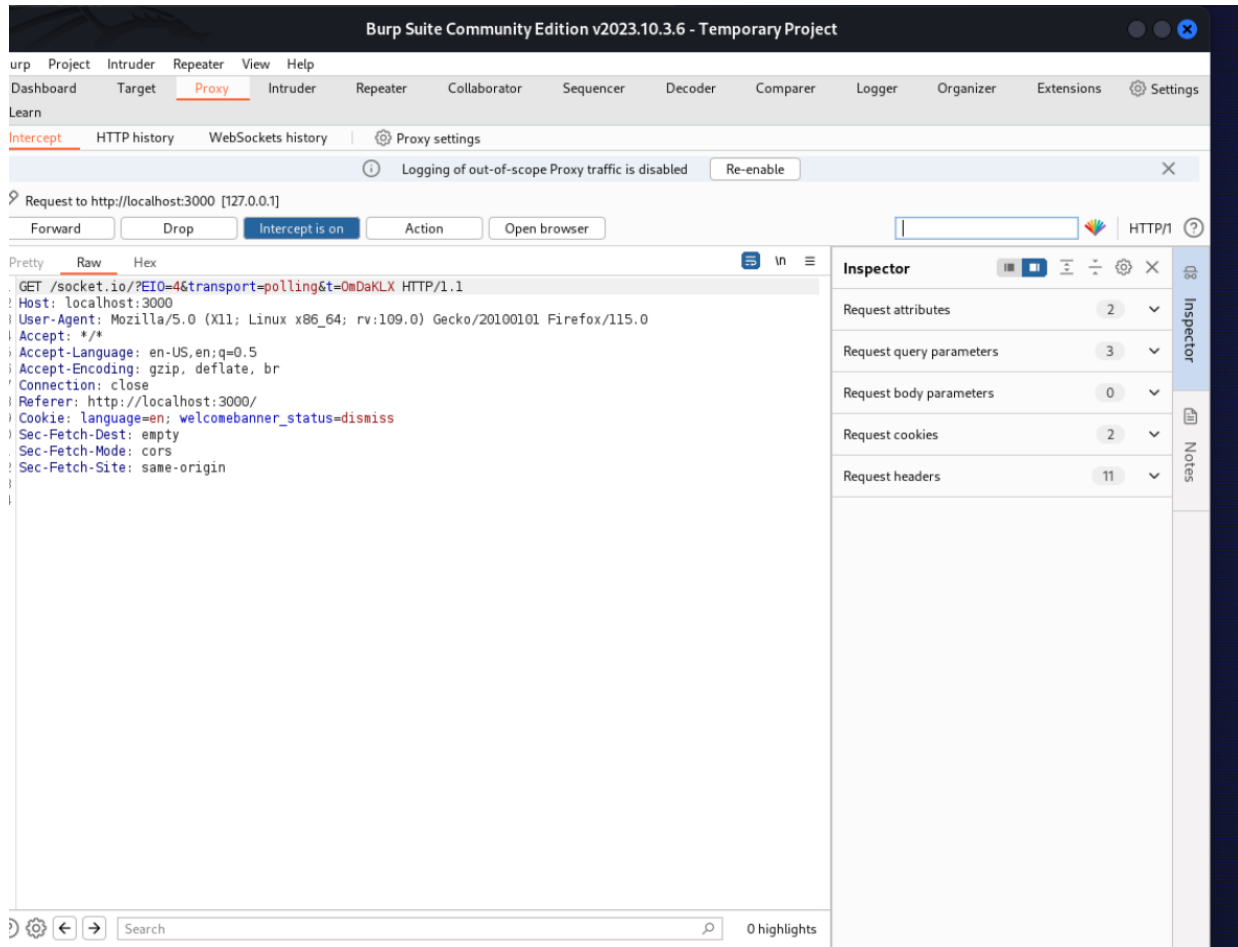
Pattern Shortcuts

- Enabled
- Add whitelist pattern to match all URLs
- Do not use for localhost and intranet/private IP addresses

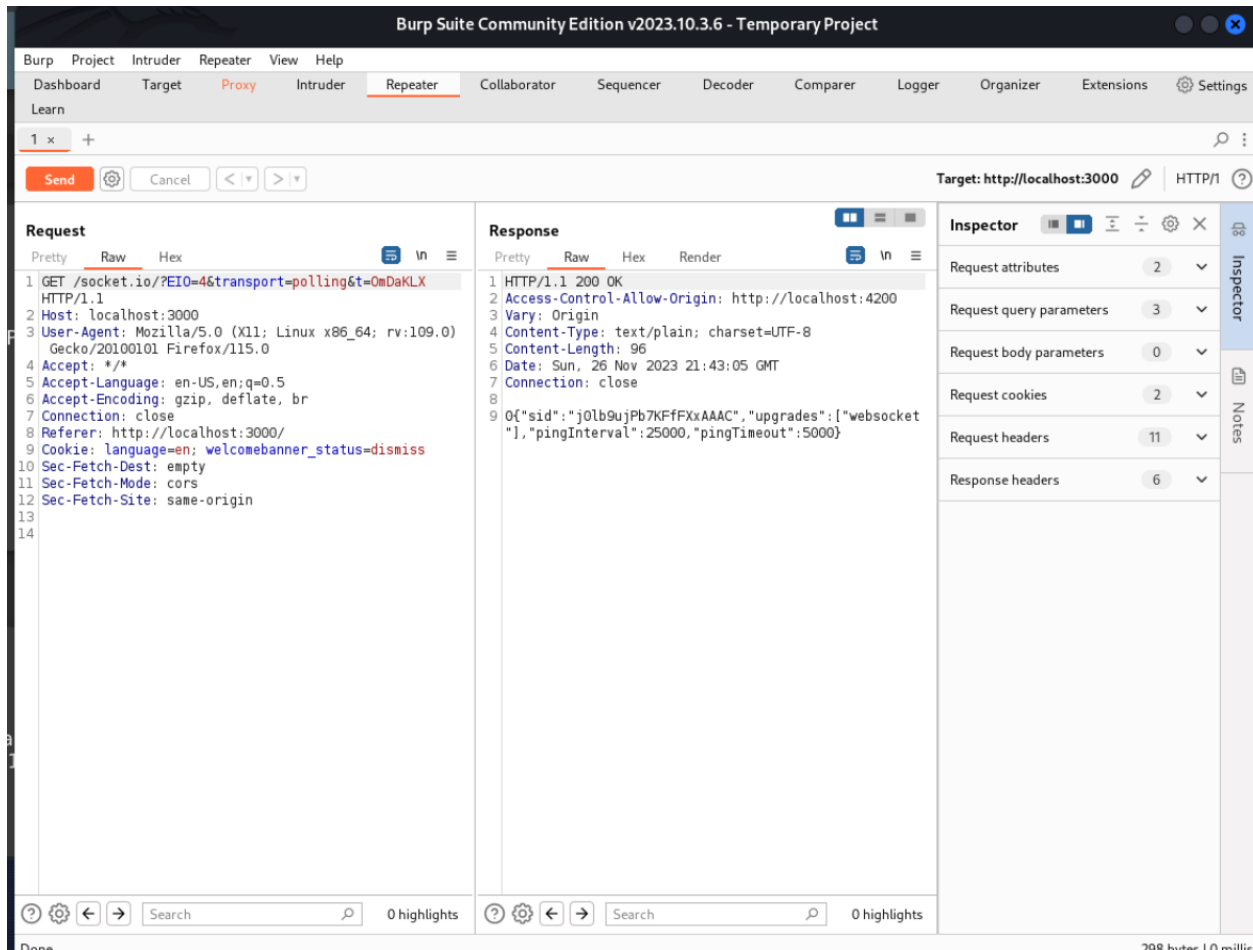
- Next, I had to initiate the Burp Suite application. The Kali Linux one was outdated so the features were limited. To fix the issue, I went through and updated the application successfully.
- After going through the steps from the book, I successfully set up the Burp Suite application and booted up the OWASP Juice Box application with no problem.



- This section helped me get familiar with the features within the Burp Suite application.
- By turning on the intercept, Burp Suite will be able to intercept the web request as it leaves the browser and heads straight to the web application.
- By completing this, Burp Suite recorded a message containing an HTTP GET request.



- Next, I forwarded the HTTP GET message to the repeater, which is another feature in the Burp Suite application.
- Burp Suite's Repeater functionality lets penetration testers edit a web request message before delivering it to the target web application.
- After sending it to the repeater, I click send, which will forward it to the response section next to the request. By doing this, I am able to modify the parameters in the request message and even hijack the web application.



- After completing those tasks, I referred to the OWASP score board and was able to see that I had successfully completed the challenge.
- After completing this section, I now understand the fundamentals of using Burp Suite and how to set up Foxy Proxy as a web browser's proxy switcher.

Score Board 1% Coding Score 0%

1/14 0/14 0/23 0/25 0/18 0/12

Show all Show solved Show tutorials only Show unavailable

Broken Access Control Broken Anti Automation Broken Authentication Cryptographic Issues Improper Input Validation Injection Insecure Deserialization

Miscellaneous Security Misconfiguration Security through Obscurity Sensitive Data Exposure Unvalidated Redirects Vulnerable Components XSS XXE

Hide all

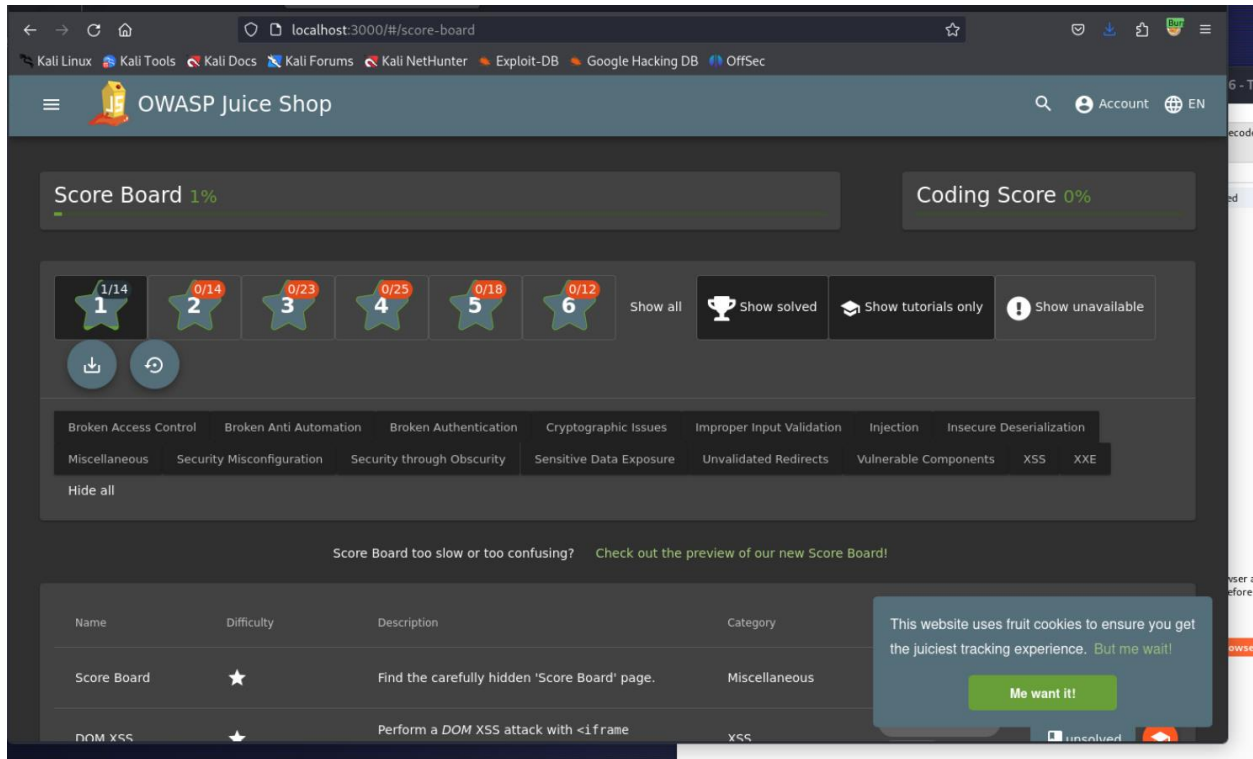
Score Board too slow or too confusing? Check out the preview of our new Score Board!

Name	Difficulty	Description	Category
Bonus Payload	★	Use the bonus payload <iframe width=100% height=166 scrolling=no frameborder=no allow=autoplay src=https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&...	XSS

This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!

Me want it!

- Performing a SQL injection attack
- In this section, using Burp Suite on Kali Linux, I discovered how to leverage SQL injection to obtain administrator access to a web application, as in OWASP Juice Shop.
- I ensured that my Kali Linux is powered on and verified that the OWASP Juice Shop Docker instance is running.



- Prior to commencing with this section, I made sure FoxyProxy was configured to use the Burp Proxy configurations, launch Burp Suite, and confirm that intercept was enabled.
- I went to the log-in section of the OWASP Juice Shop web application and entered a random email and password.
- In the screenshot below, I refer back to the HTTP POST message and see the random email address and password I entered.

The screenshot displays the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' section shows a request to `http://localhost:3000` [127.0.0.1] with 'Intercept is on' enabled. The request is shown in 'Pretty' format, displaying the following details:

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 48
4 sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/119.0.6045.159 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss
18 Connection: close
19
20 {"email":"rflor033@gmail.com","password":"1234"}

```

The 'Inspector' panel on the right provides a structured view of the request:

- Request attributes:** 2 attributes, Protocol: HTTP/1.
- Method:** POST
- Path:** /rest/user/login
- Request query parameters:** 0 parameters (It's empty in here).
- Request cookies:** 2 cookies:

Name	Value
language	en
welcomebanner_status	dismiss
- Request headers:** 17 headers.

- Additionally, in the response section, I can see the unauthorized status and a message indicating the email or password is invalid.
- The next part will involve me committing a SQL injection attack. I injected the email as admin@email.com into the request tab, which will prompt this login information to the web application.

The screenshot displays the Burp Suite interface with the 'Repeater' tab selected. The target is set to 'http://localhost:3000'. The request is a POST to '/rest/user/login' with the following body:

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 48
4 sec-ch-ua: "Chromium";v="119",
  "Not?A Brand";v="24"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
  x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/119.0.6045.159 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Origin: http://localhost:3000
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: language=en; welcomebanner_status=dismiss
18 Connection: close
19
20 {
  "email": "rflor033@gmail.com",
  "password": "1234"
}

```

The response is an HTTP 401 Unauthorized status with the following headers:

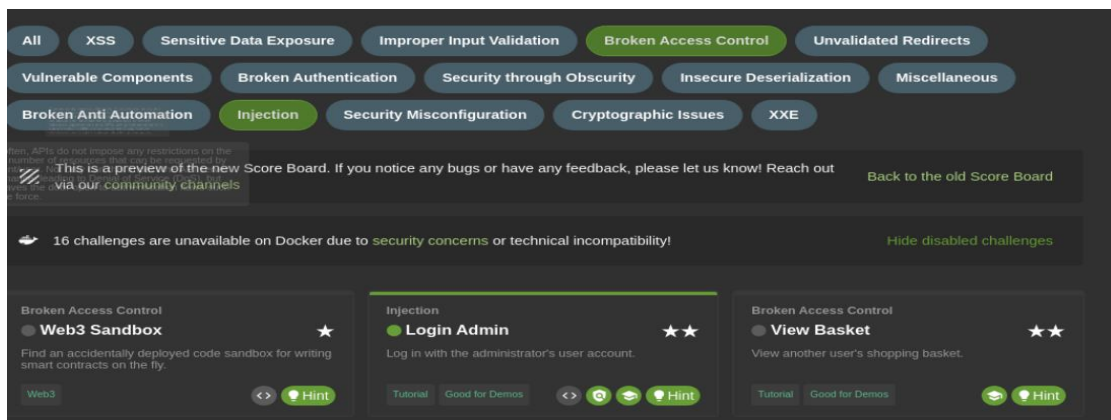
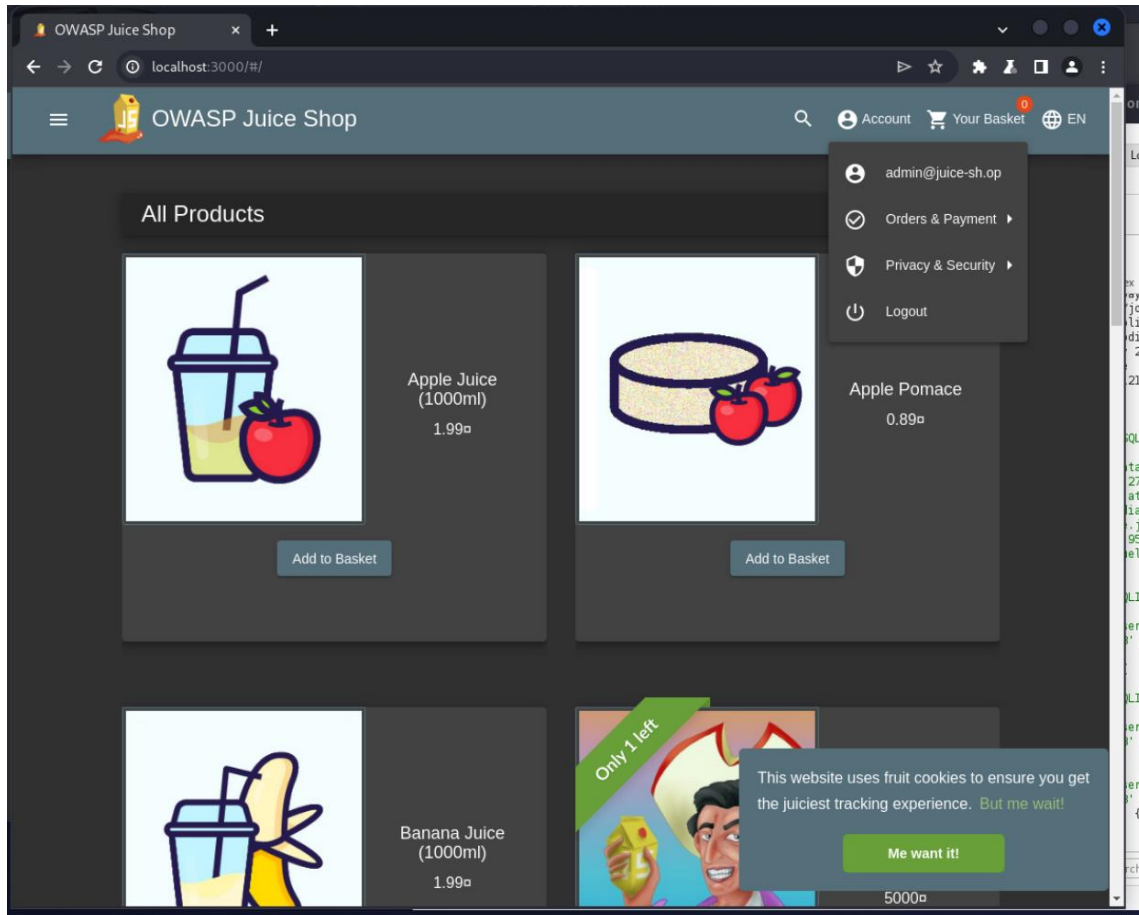
```

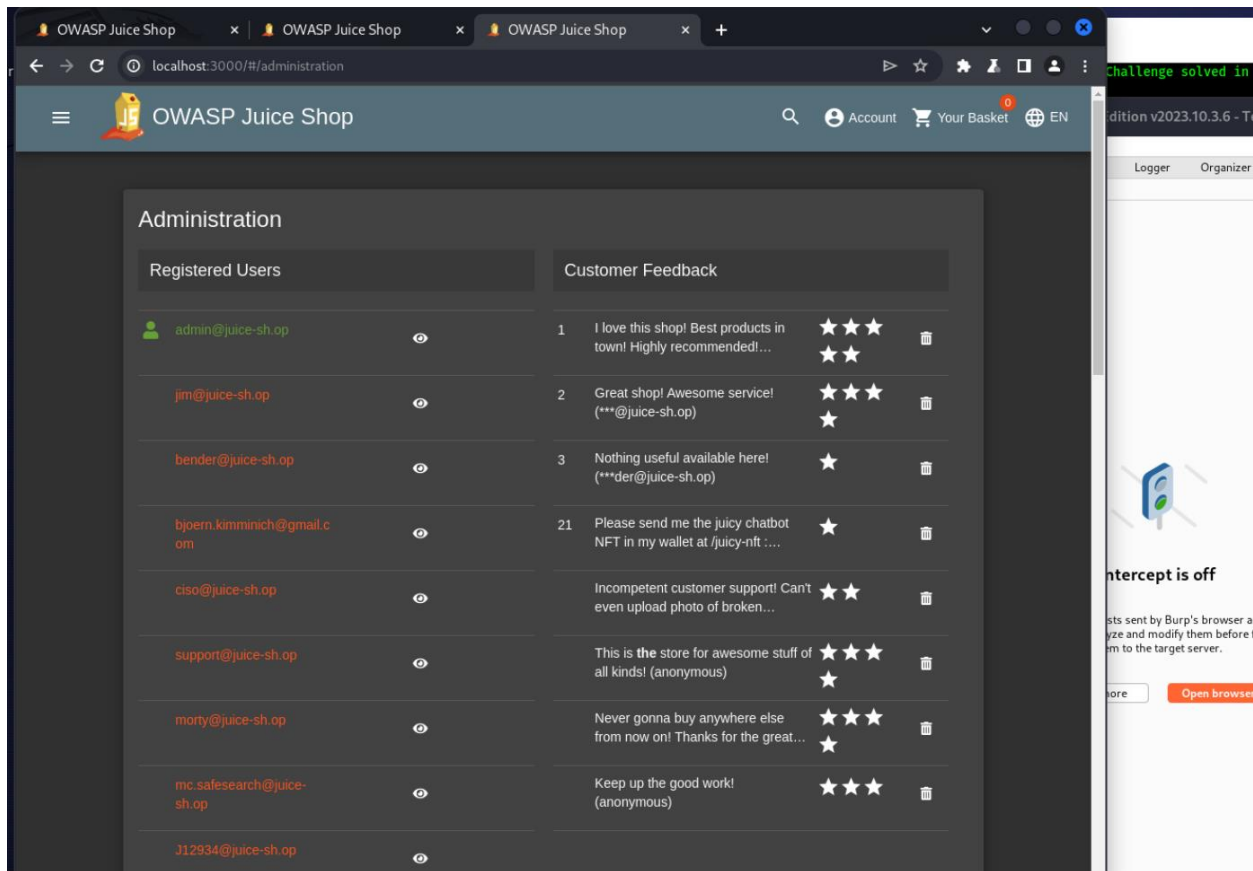
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Content-Length: 26
9 ETag: W/"1a-AR3vVK+smzAF3QQve2mDSG+3Eus"
10 Vary: Accept-Encoding
11 Date: Sun, 26 Nov 2023 22:30:39 GMT
12 Connection: close
13
14 Invalid email or password.

```

The right-hand side of the interface shows the 'Inspector' panel with a tree view containing 'Request attributes', 'Request query parameters', 'Request cookies', 'Request headers', and 'Response headers'. The 'Request headers' section is currently expanded, showing 17 items.

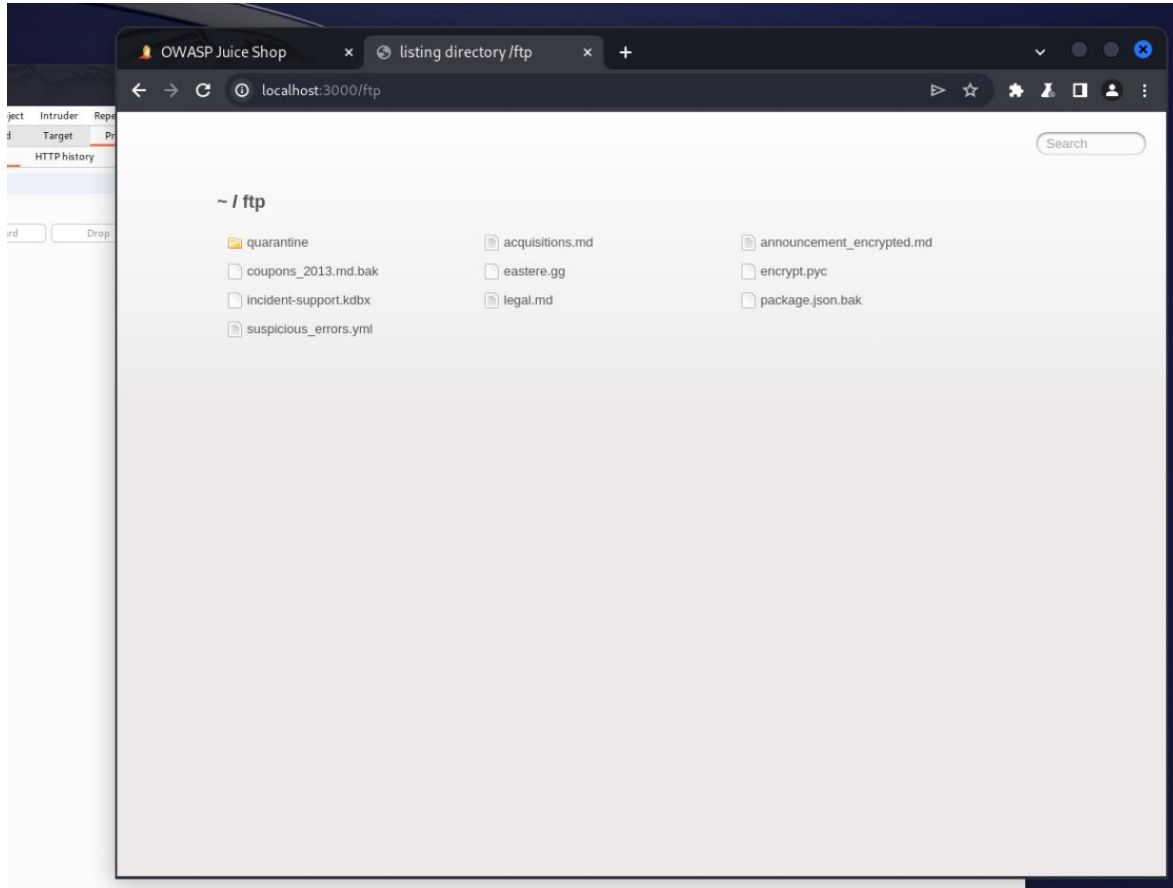
- The web application is vulnerable to a SQL injection attack
- Upon successfully completing the tasks, I know that the web application is vulnerable to a SQL injection attack.
- Additionally, I am able to successfully gain access to the administrator's user account on the vulnerable web application.





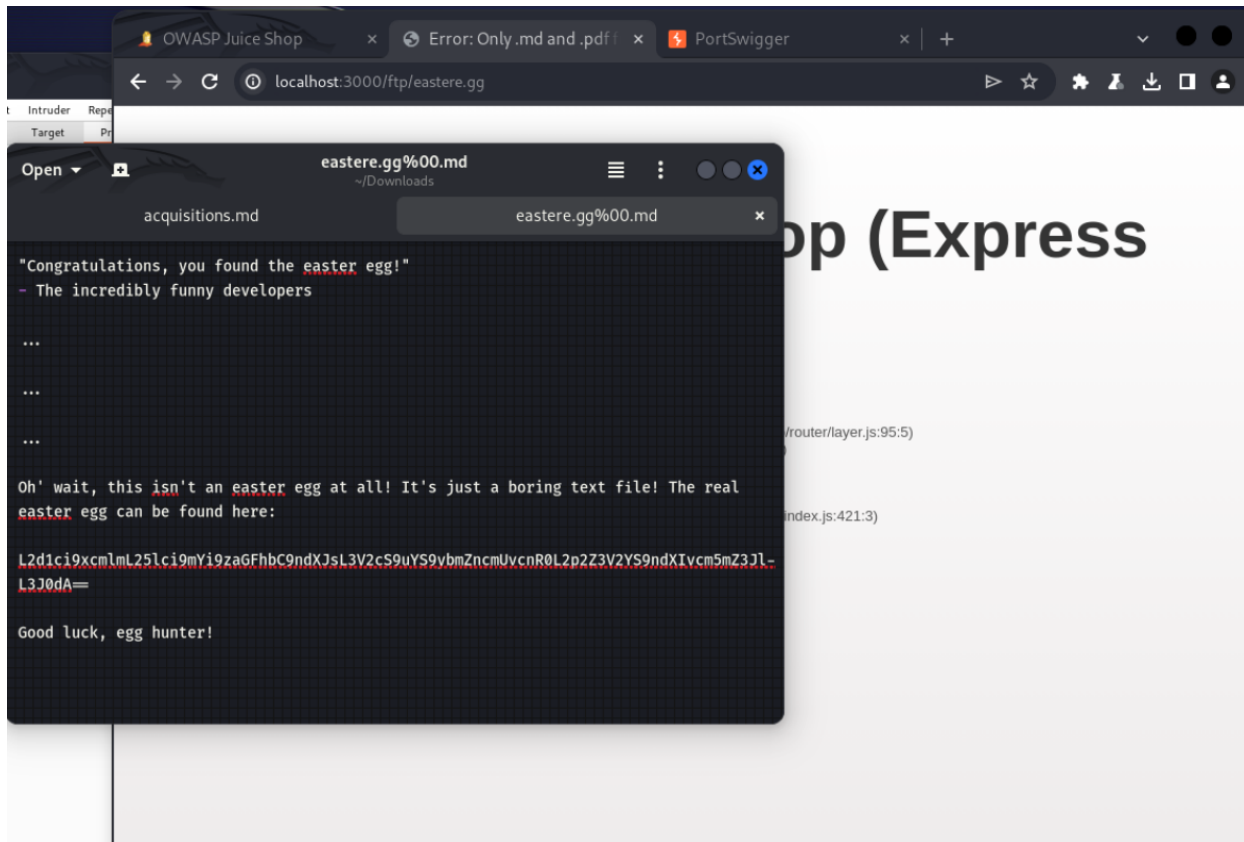
- Exploiting cryptographic failures
- This section of the lab involved me completing the Nested Easter Egg Challenge in order to illustrate the security vulnerabilities and potential exploits.

- The set-up process is the same as the rest of the previous challenges. I am using Kali Linux and started at the OWASP Juice Shop Score Board section.
- The first step is to input the following link and browse the directory to view the contents, I will be using eature.gg file.



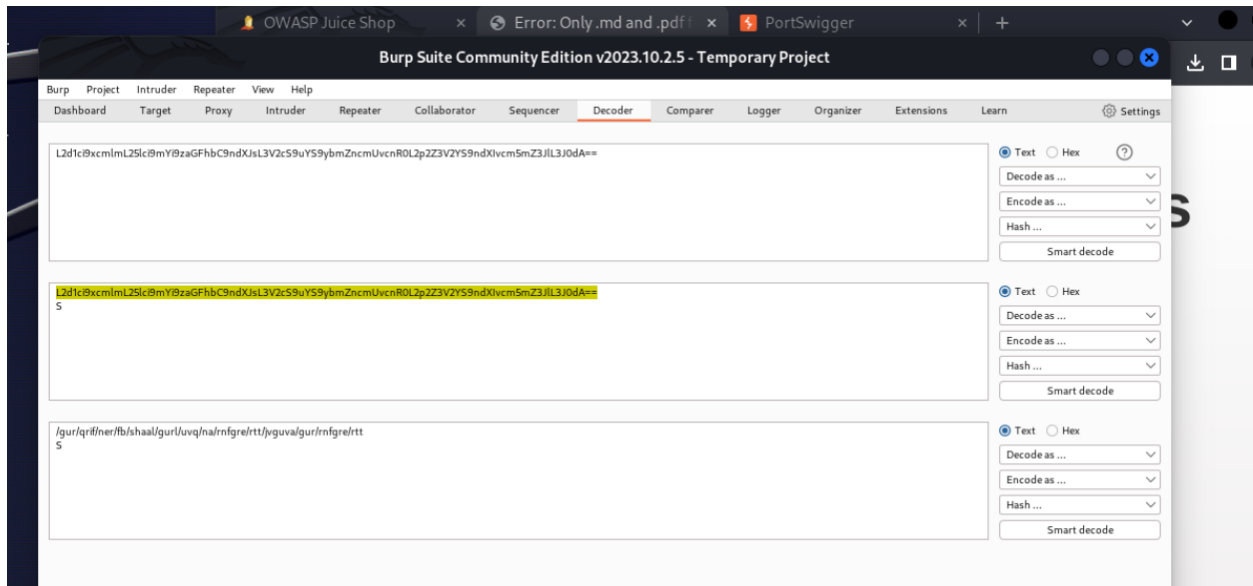
- Upon opening the file after converting, the contents are shown within the mousepad editor.

- The contents of the file display a hash value. I will use an identifier to decrypt the contents of the value.



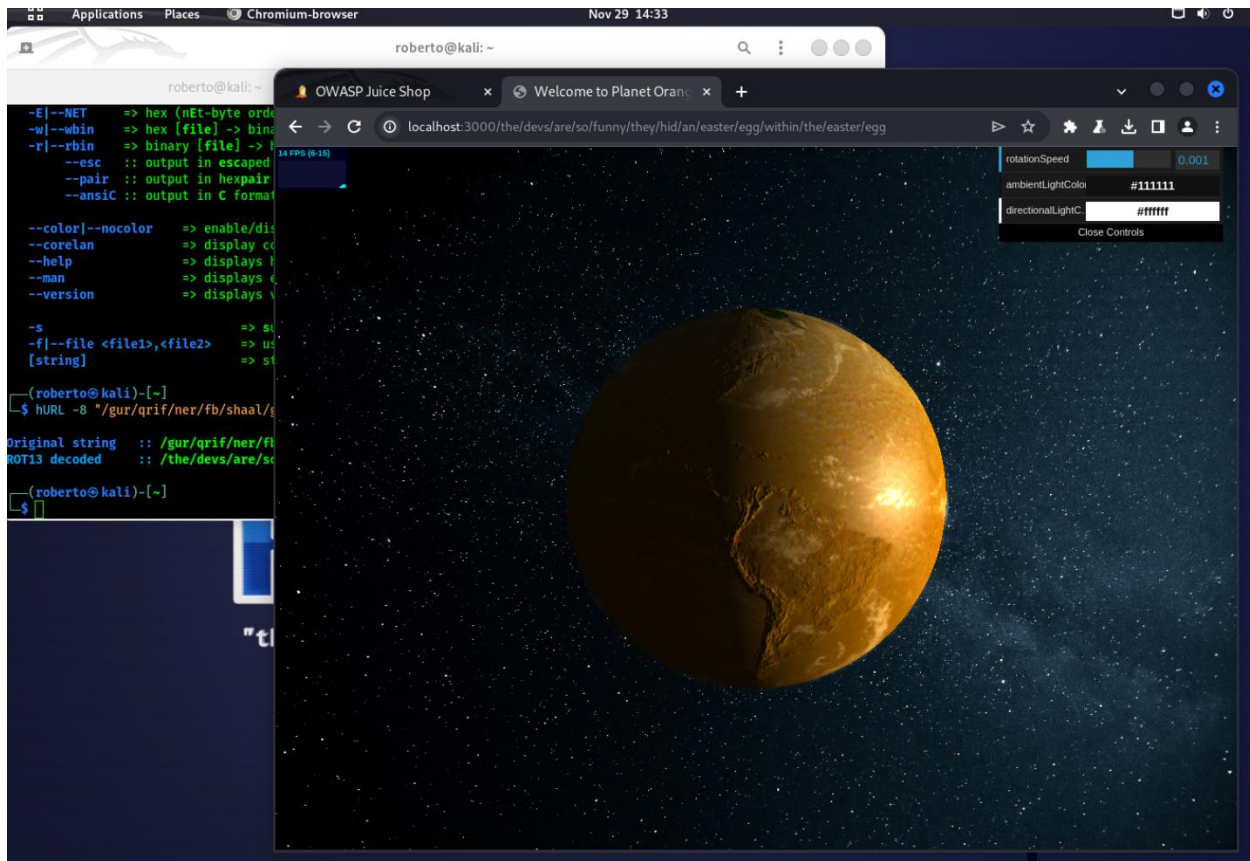
- The decoder tool I used is from the Burp Suite application.

- The result demonstrated a cryptographic hash for something that looks like the path of a web address.



- The next step was to install hURL, which is a tool used to encode and decode various types of character-offset encryption ciphers.

- As a result, I was promoted with the plaintext message that it had been successfully decrypted. Using the message, I pasted it into my web browser and got the following result.



- Exploiting security misconfigurations

- The screenshot displays the Burp Suite Community Edition v2023.10.3.6 interface. The top menu bar includes options like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The main workspace is divided into three panes:

 - Request Pane:** Shows a GET request to `/rest/fakepath/socket.io/?EIO=4&transport=polling&t=OmE3_Sy&sid=Ahcrc-MkvShOtFBAAAY`. The request body contains various headers and cookies.
 - Response Pane:** Displays the server's response, which is an HTML document. It starts with a meta charset declaration and includes a title "Error: Unexpected path:". Below the title, there is a large CSS style rule for a gradient background.
 - Inspector Pane:** Provides detailed information about the selected element in the response, currently showing the root HTML document structure.

The status bar at the bottom indicates the target is `http://localhost:3000` and shows the number of highlights for the selected element.

- To complete this section, I uploaded an unsupported file to the OWASP Juice Shop web application, such as the one below (an xml file). When hitting the submit file button, I understood now that the web application accepts the unsupported file type, which is another indication of a security misconfiguration that can be exploited.

