

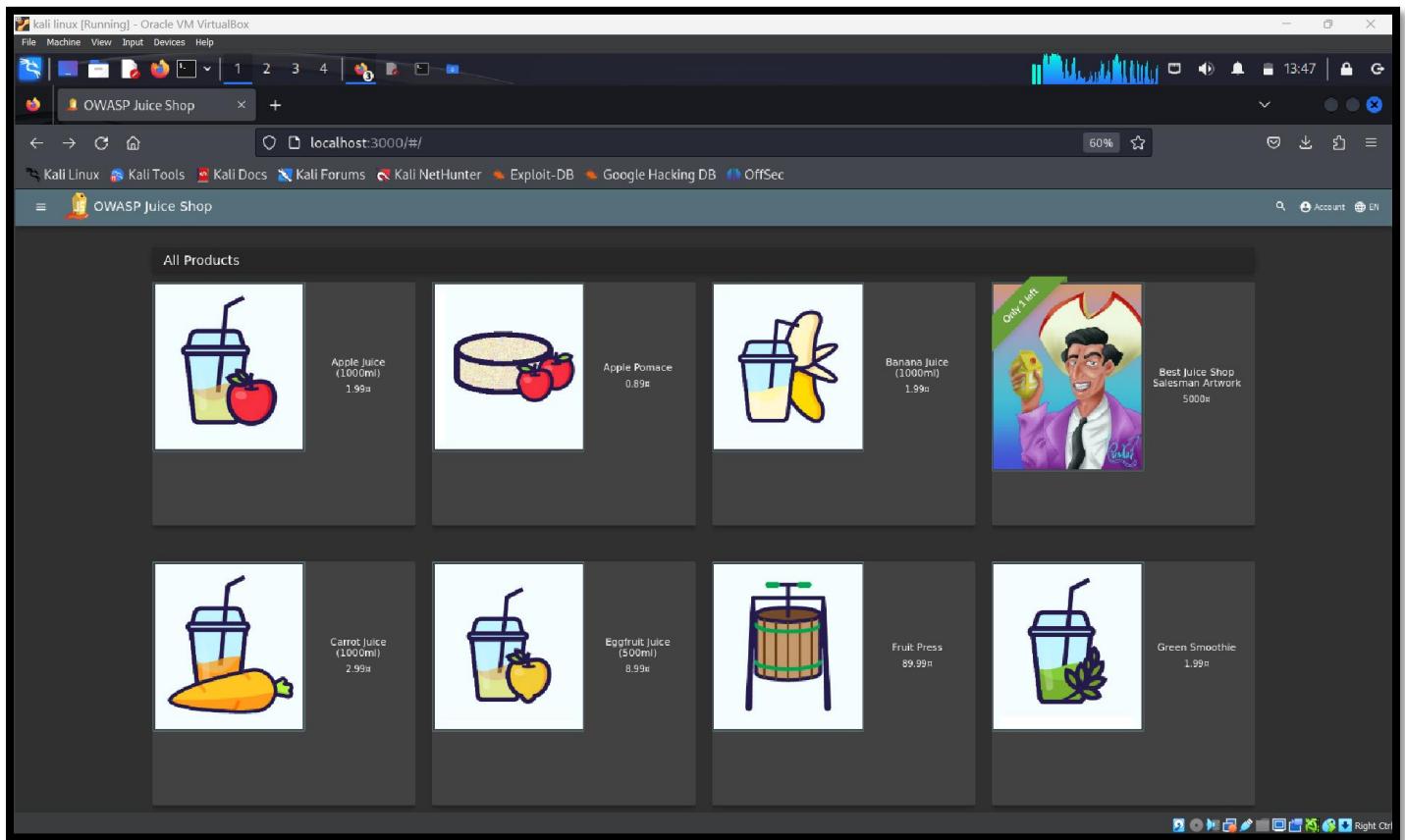
# **Subject: Security Testing**

## **SENG 8061**

### **Project Part - 03**

Name	Student ID
Shivani Varu	8941914
Mohammed Rafique	8954785

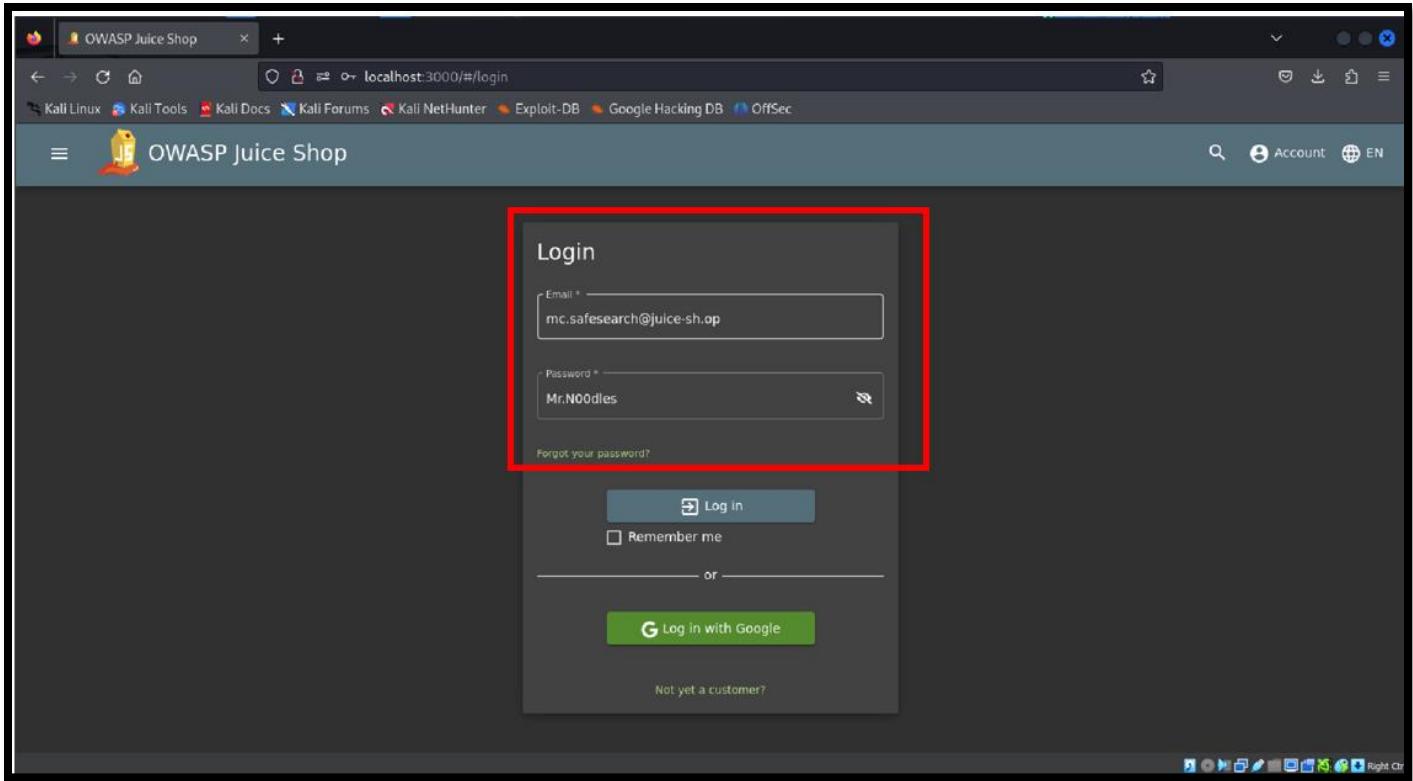
Screenshot 1: Go to <http://localhost:3000/>



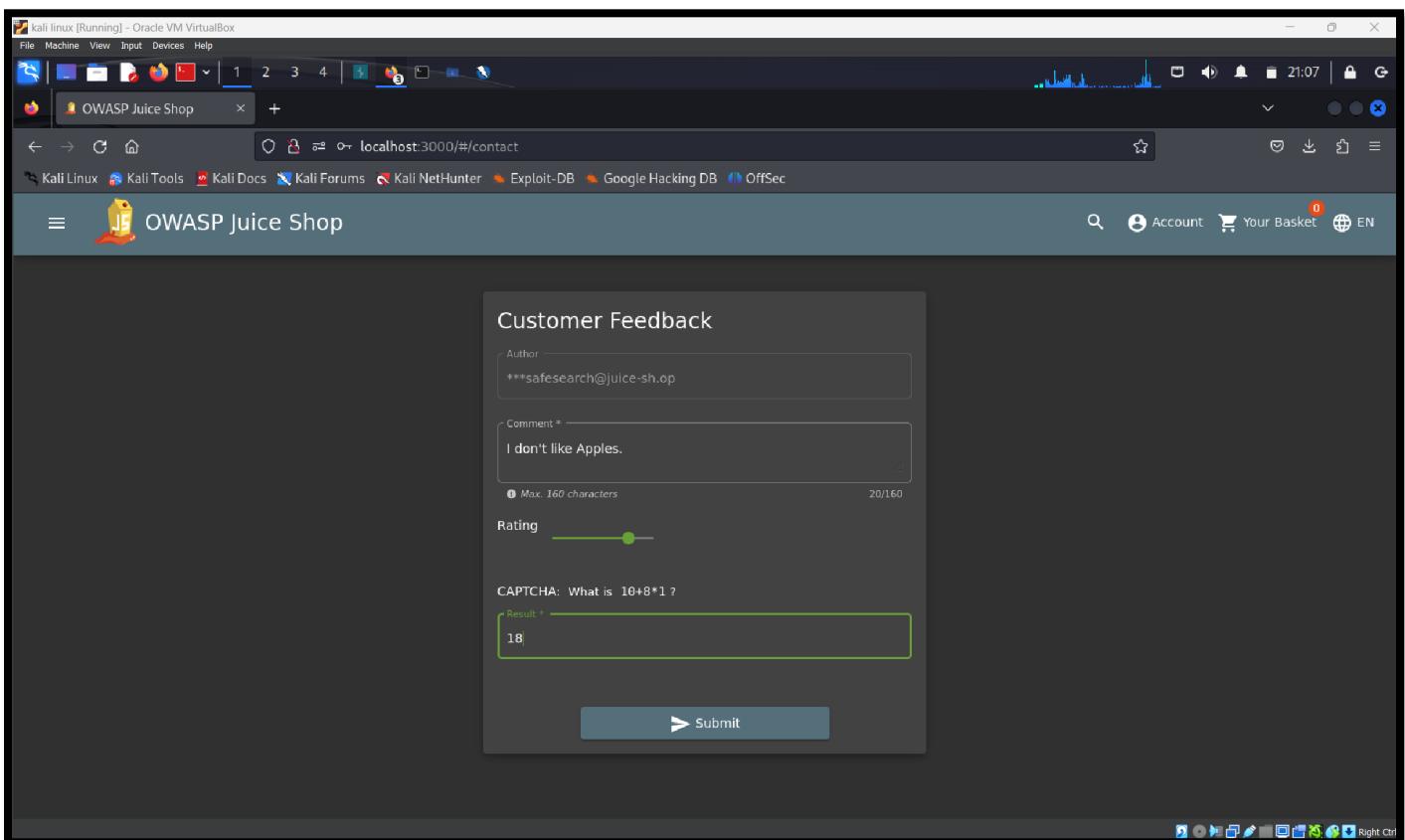
## TASK 1

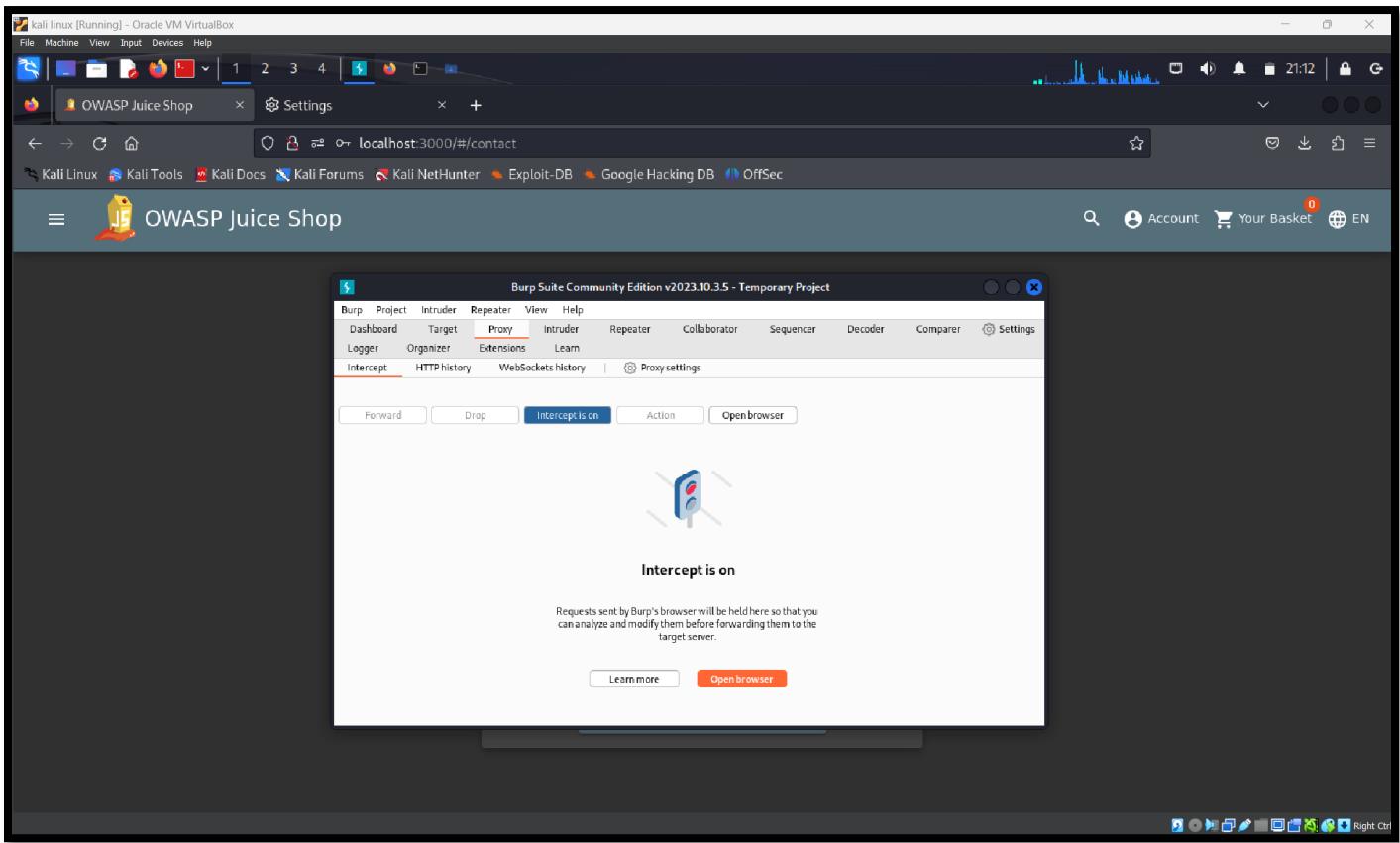
Post some feedback in another user's name

In this challenge, we recalled the email and password of mc. safesearch via project part 2's Task2. We entered his username and password to log in to his account and provide Customer Feedback, which resided in the hamburger menu.



While inspecting the Juice Shop application, we found an option called customer feedback and attempted to enter some feedback. With the help of the Burp tool in Kali, we intercepted the customer feedback request form.



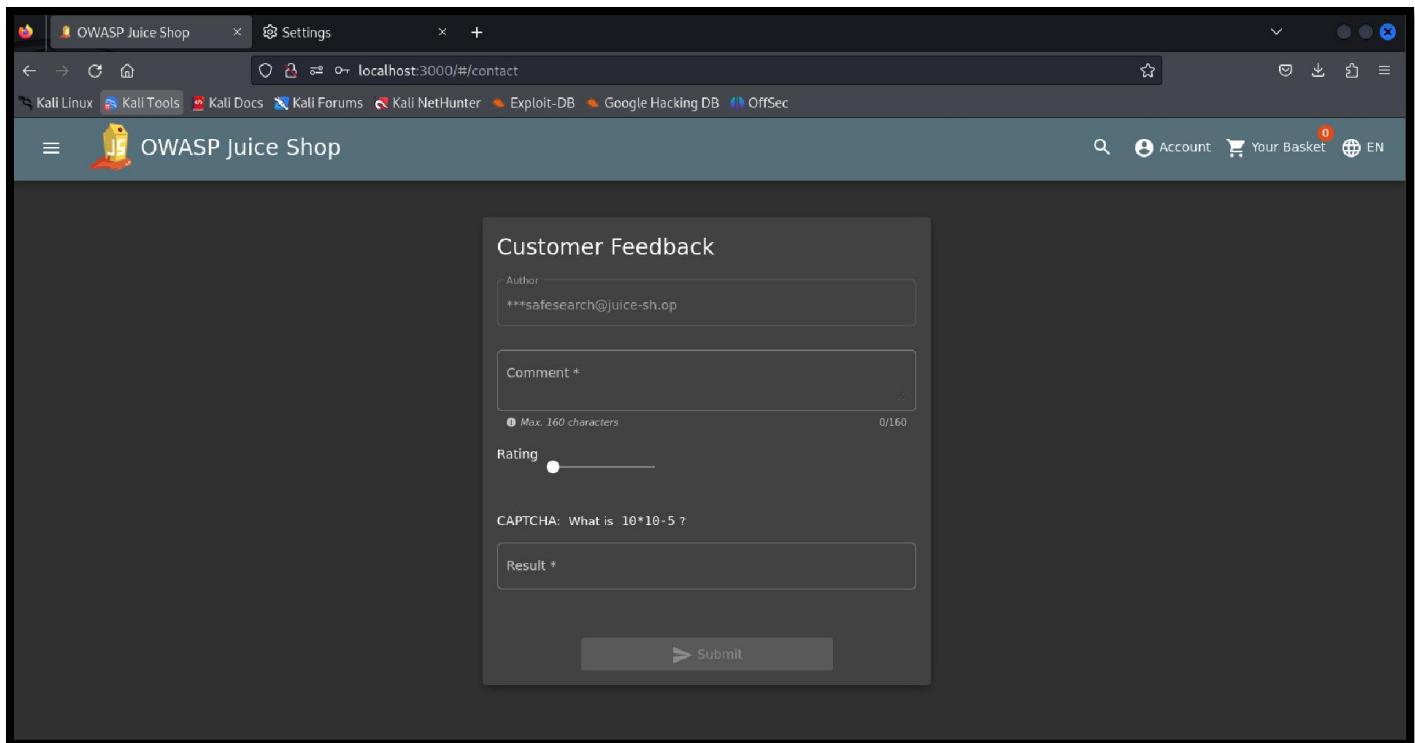


Upon inspecting the request, we discovered a comment id, captcha id, and rating, but nothing was mentioned about the user id.

```

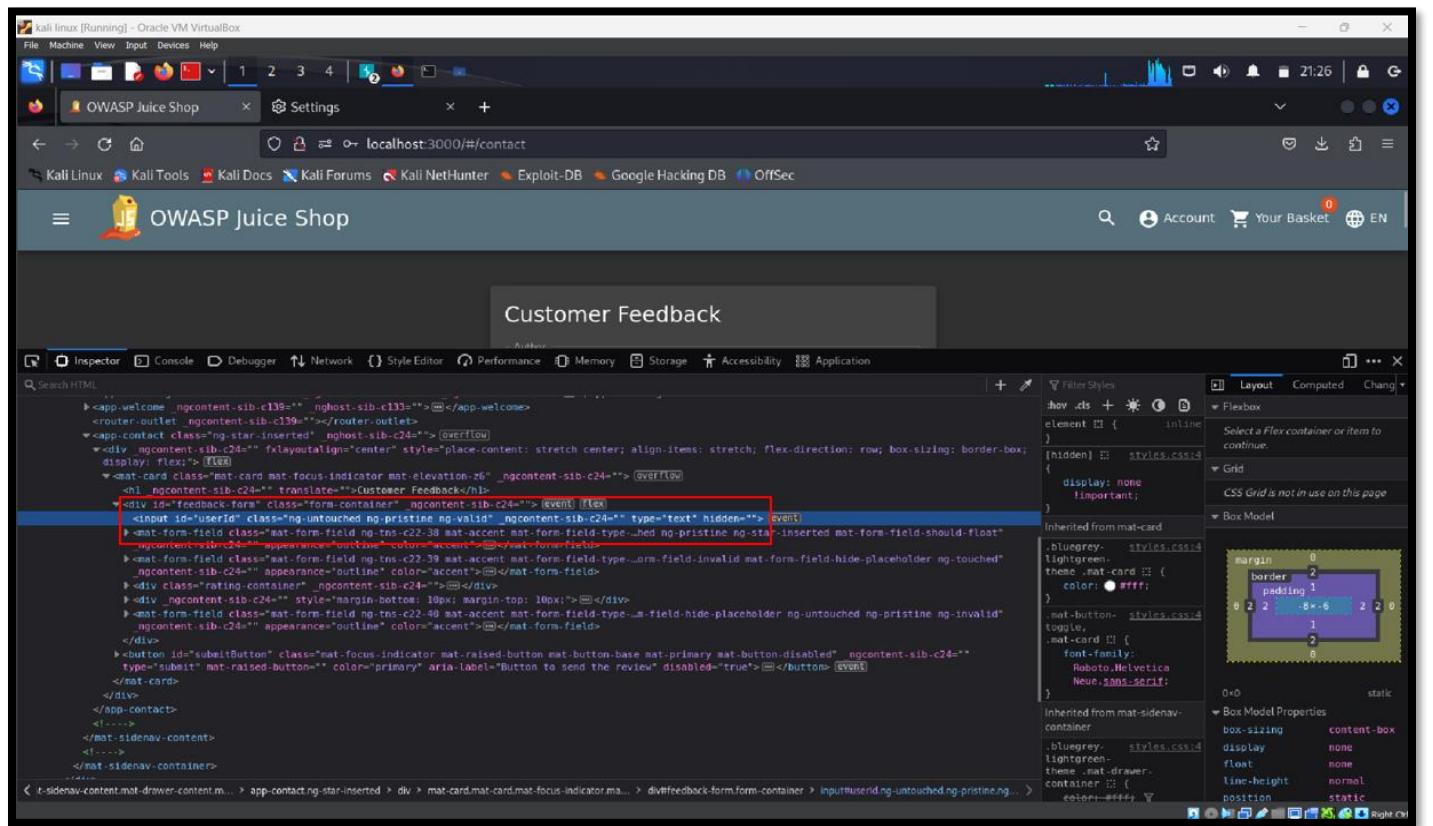
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36
4 Accept: application/json, text/plain, */*
5 Accept-Encoding: gzip, deflate, br
6 Accept-Language: en-US,en;q=0.9
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkIjpbG0dKH0sJzdwWjZxNcTiwsZP0YStKeyJpZCf600wLdXNlcm5hWj1sJzLcJ3bMfpbCD6tEl1JmNhZewVzZWPyY2hAenVpY2ltc2gab5s1Lc3wYXnsd29yZC161tTwD2Y
8f9BfYThMDU4ZewYMMYzbyYzR0TfUzrH41vcse9zS6t6mNL50rbmKV1ivz20sVkh1V59rZh40s1llLc3sYXNOTGwmn53cC6t3EfyhAvLsAmNSIsInBy62zbGVJbmFnZS16t#Fcc2v0cy9wdHjswhwrl1JZ2vL3V
9wG9bzHmZ0vYXNs952dyC52dyCLC30b39wQ2V)c+VO5) s1Jivs1aKNBv3pdaULOrnLiCtly2fHfOZyRbC16t)2wH) tNDRH HTLgMDAGHtHbWc0utA01CsuvDbwvCEsInVwZPfDwFBtC16t)1wH) tNDRH HTLgMDAGHbU
uHfY41csuMew1s1am1bbGvZmHdCtDbevshHos1elhdCtDfHxHDCb1c1M40.95EHLsQdGAM8dHmVz1nWfDy0s12dc3LLaPAT5G2Ra4VrcAro_aDw4KnrBpzr23A5s1nL4wVtsgIS1Kax11pgjgkN-zEtPfCp24C)EKD
eA5hA1cgb9vPhZ0A36spfGOpIZxJ+nbushZP0Lvg0t6JwJch35gWw5e09HtA8
10 Content-Type: application/json
11 Content-Length: 113
12 Origin: http://localhost:3000
13 Connection: close
14 Referer: http://localhost:3000/
15 Cookies: language=en; welcome_status=dississ; cookieconsent_status=dississ; continueCode=eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkIjpbG0dKH0sJzdwWjZxNcTiwsZP0YStKeyJpZCf600wLdXNlcm5hWj1sJzLcJ3bMfpbCD6tEl1JmNhZewVzZWPyY2hAenVpY2ltc2gab5s1Lc3wYXnsd29yZC161tTwD2Y
16 eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkIjpbG0dKH0sJzdwWjZxNcTiwsZP0YStKeyJpZCf600wLdXNlcm5hWj1sJzLcJ3bMfpbCD6tEl1JmNhZewVzZWPyY2hAenVpY2ltc2gab5s1Lc3wYXnsd29yZC161tTwD2Y
17 f9BfYThMDU4ZewYMMYzbyYzR0TfUzrH41vcse9zS6t6mNL50rbmKV1ivz20sVkh1V59rZh40s1llLc3sYXNOTGwmn53cC6t3EfyhAvLsAmNSIsInBy62zbGVJbmFnZS16t#Fcc2v0cy9wdHjswhwrl1JZ2vL3V
18 wG9bzHmZ0vYXNs952dyC52dyCLC30b39wQ2V)c+VO5) s1Jivs1aKNBv3pdaULOrnLiCtly2fHfOZyRbC16t)2wH) tNDRH HTLgMDAGHtHbWc0utA01CsuvDbwvCEsInVwZPfDwFBtC16t)1wH) tNDRH HTLgMDAGHbU
uHfY41csuMew1s1am1bbGvZmHdCtDbevshHos1elhdCtDfHxHDCb1c1M40.95EHLsQdGAM8dHmVz1nWfDy0s12dc3LLaPAT5G2Ra4VrcAro_aDw4KnrBpzr23A5s1nL4wVtsgIS1Kax11pgjgkN-zEtPfCp24C)EKD
19
20 {
21   "UserId": 1,
22   "captcha_id": 1,
23   "captcha": "38",
24   "comment": "I don't like Apples. (***safeSearch@juice-shop*)",
25   "rating": 4
26 }

```

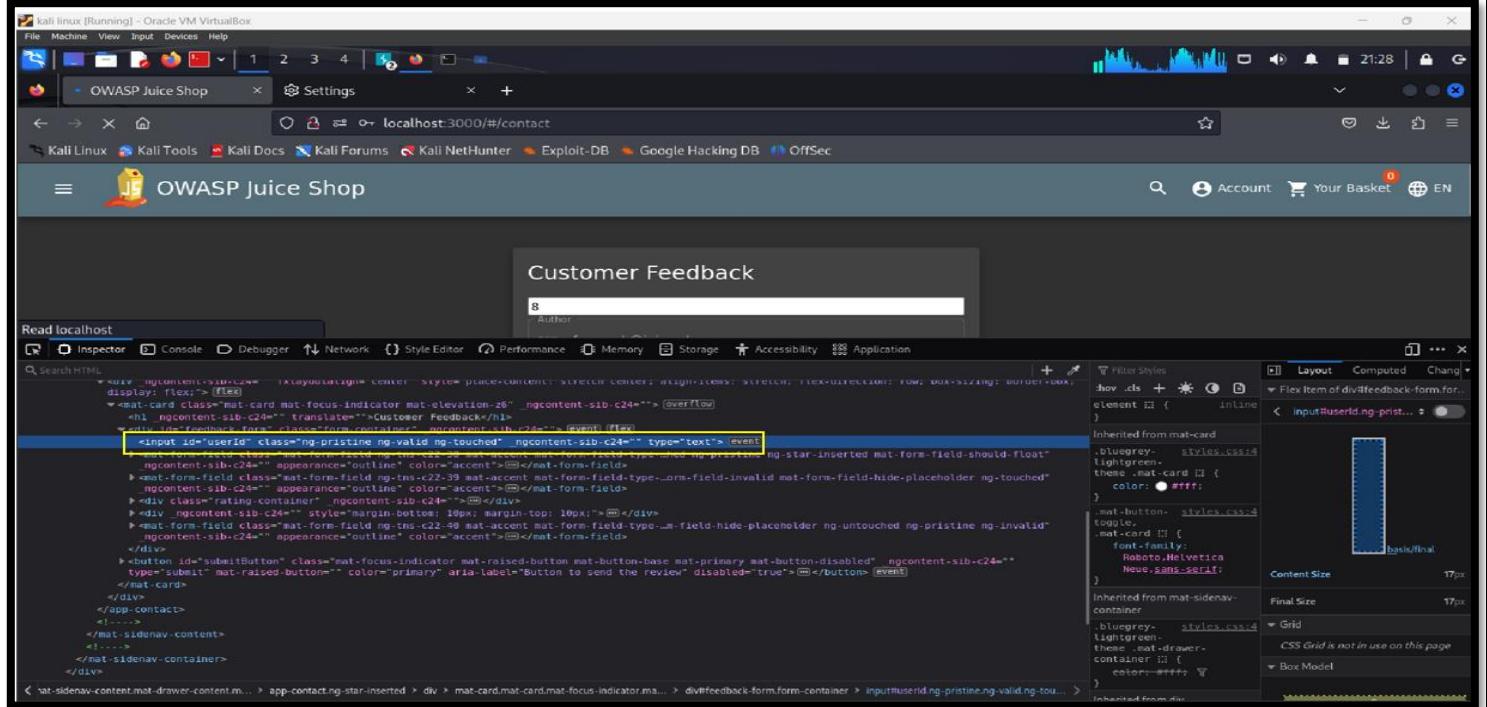


Afterward, we turned off the intercept and submitted the user feedback. We then reopened Burp Suite and intercepted to see the response in the HTTP history. We observed that we received a response status of success.

After that, we sent these to repeater. To investigate further, we opened the browser and inspected the feedback form using the developer tools to find something related to the user id. Upon inspection, we noticed that the user ID was included as a parameter in the form submission, and we found a hidden tag, which essentially meant that the input box couldn't be seen in the UI.

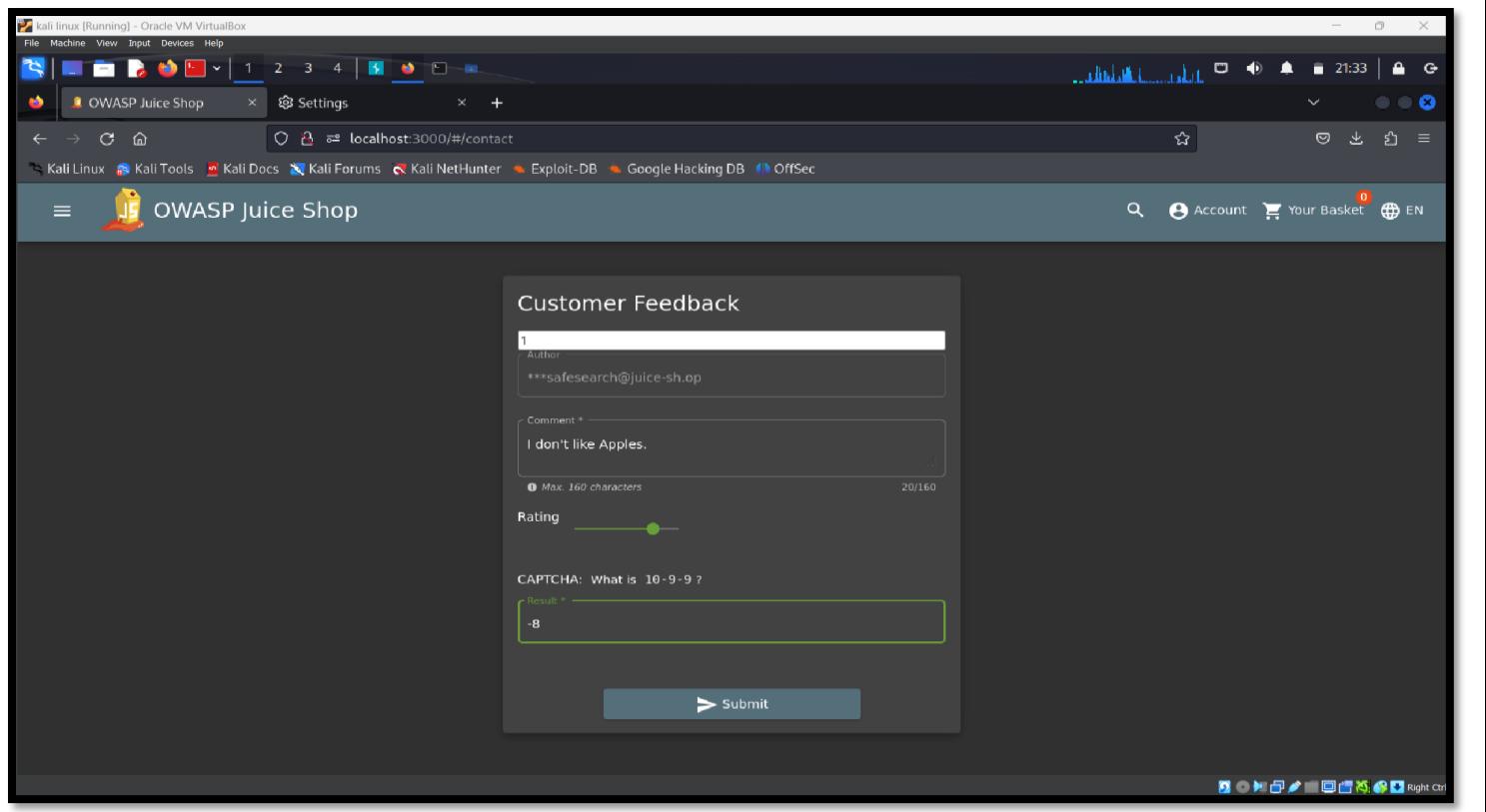


Therefore, we deleted that hidden tag and hit enter to observe the outcome. We observed that an input box popped up, displaying the user ID 8, which we retrieved from BurpSuite. Using the developer tools, we modified the HTML code of the form to replace our own user ID with the ID of another user.



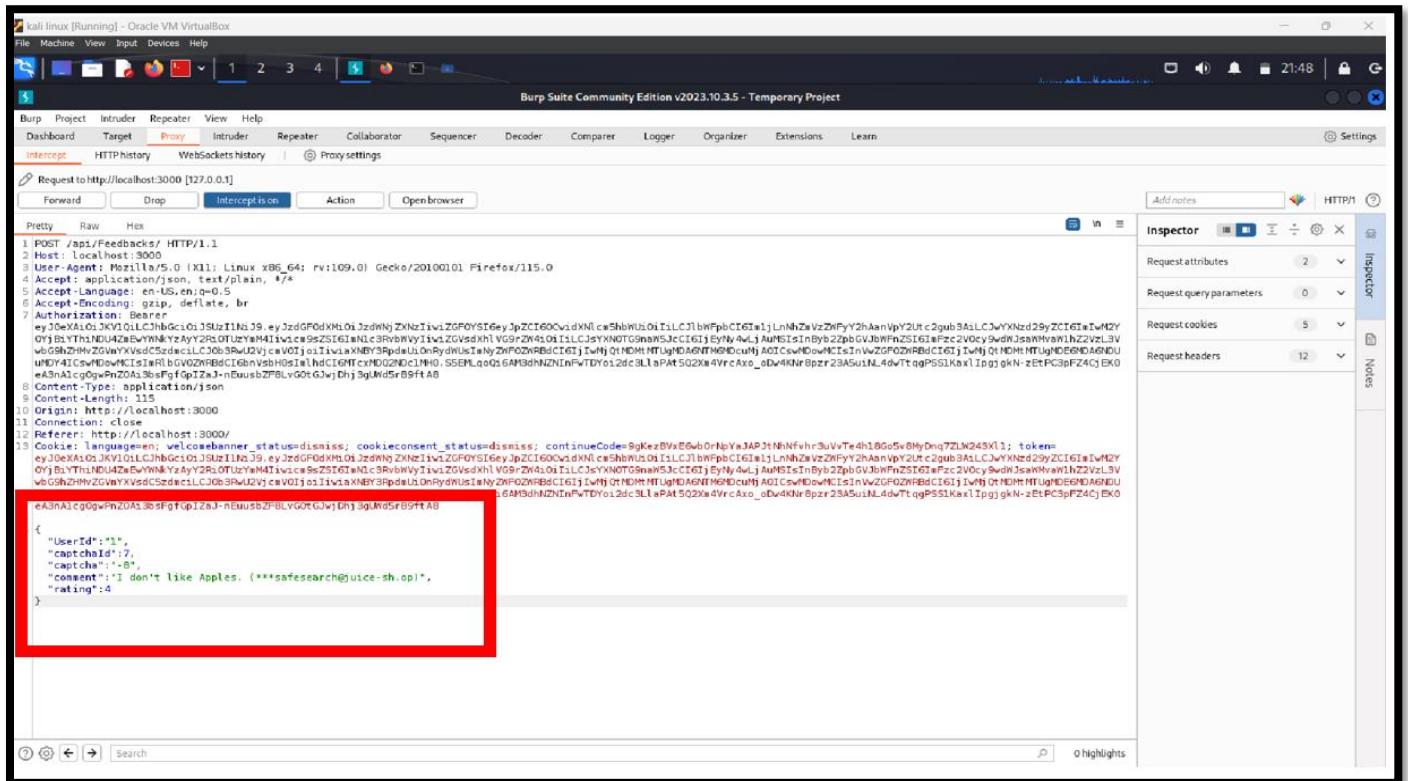
The screenshot shows a browser window for the OWASP Juice Shop. The URL is `localhost:3000/#/contact`. The page displays a 'Customer Feedback' form. In the developer tools, the `Elements` tab is active, showing the HTML structure of the form. The `<input id="userId" ...>` line is highlighted. The right panel of the developer tools is the `Style Editor`, showing the computed styles for the selected element. The value '8' is visible in the input field.

To complete our task, we changed the userid from 8 to 1 to see the result. Thus, we turned off the intercept in BurpSuite, then entered userid as 1, filled the feedback form, and intercepted before submitting the feedback.

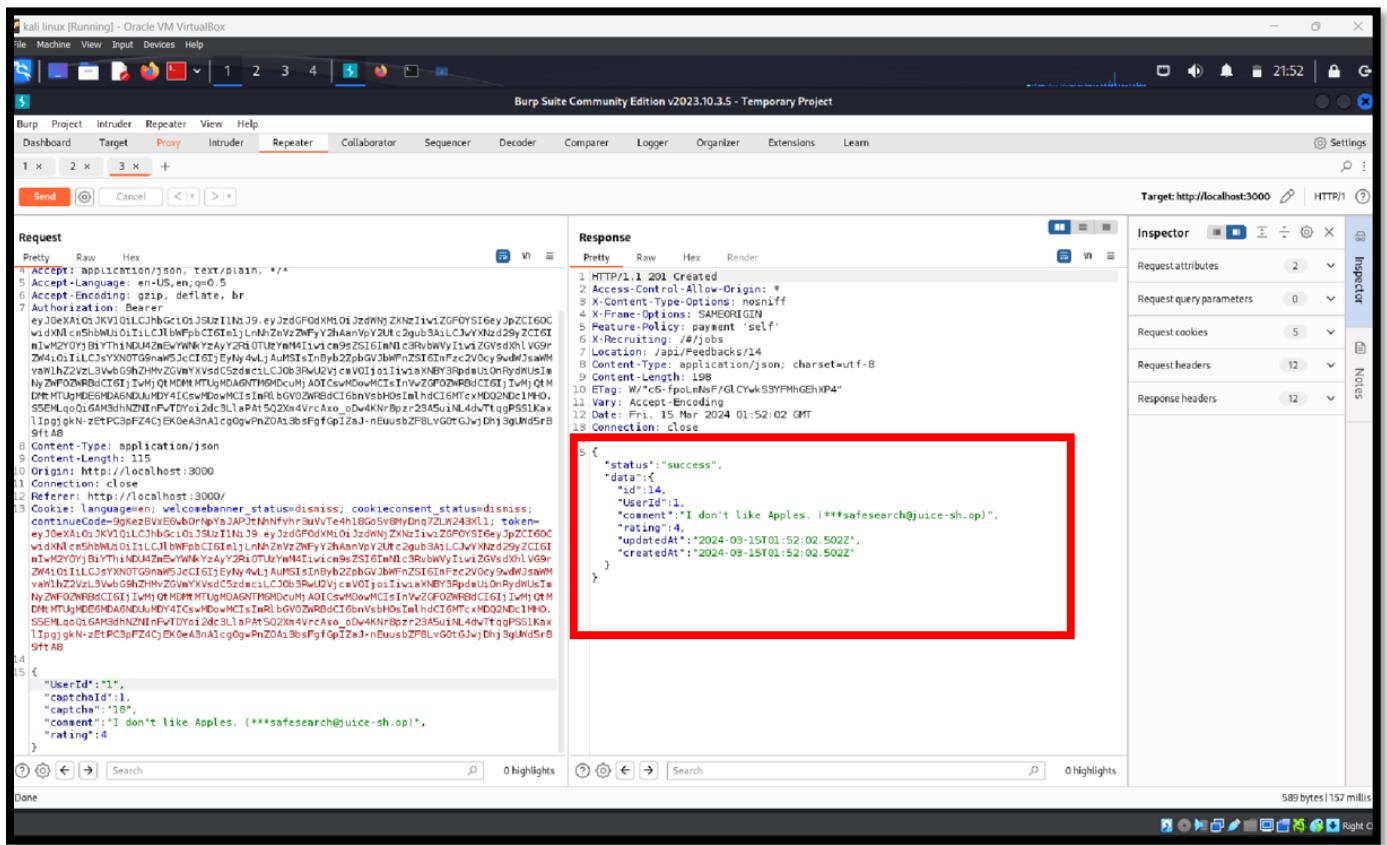


The screenshot shows the same browser window after modifying the `userId` value. The input field now contains '1'. The rest of the form fields are visible: 'Author' (with value `***safesearch@juice-sh.op`), 'Comment' (with value `I don't like Apples.`), 'Rating' (set to 1), and 'CAPTCHA' (with question `What is 10 - 9 - 9 ?` and answer `-8`). A large blue button at the bottom right is labeled `> Submit`.

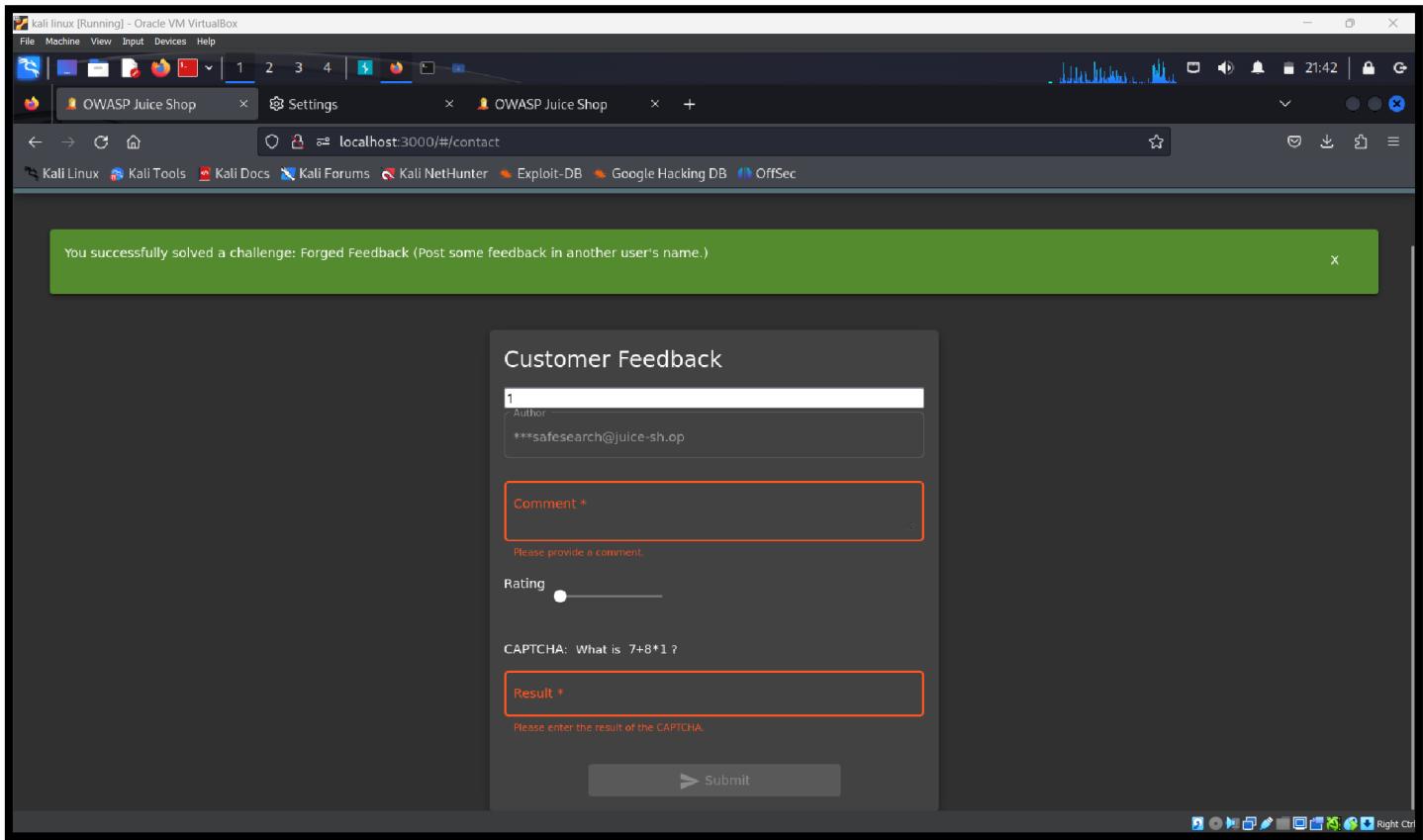
Before submission, we turned on the intercept in BurpSuite, and upon clicking the submit button, we received a response on BurpSuite indicating that the user id was changed. We forwarded that to repeater.



Then, we clicked the send button to see what response status we would get.



We received a response status of success, indicating that we had successfully created new feedback. Finally, we turned off the intercept to check what we would get on the feedback page, confirming that we had successfully solved this challenge by posting feedback under another user's name.



### Vulnerabilities Exploited by Task 1:

- Insecure Direct Object Reference (IDOR)
- Lack of Access Control
- Insufficient Input Validation

### How Vulnerabilities Can Be Fixed or Addressed:

- Implement Proper Access Controls
- Validate User Input
- Use CSRF Tokens
- Secure API Endpoints

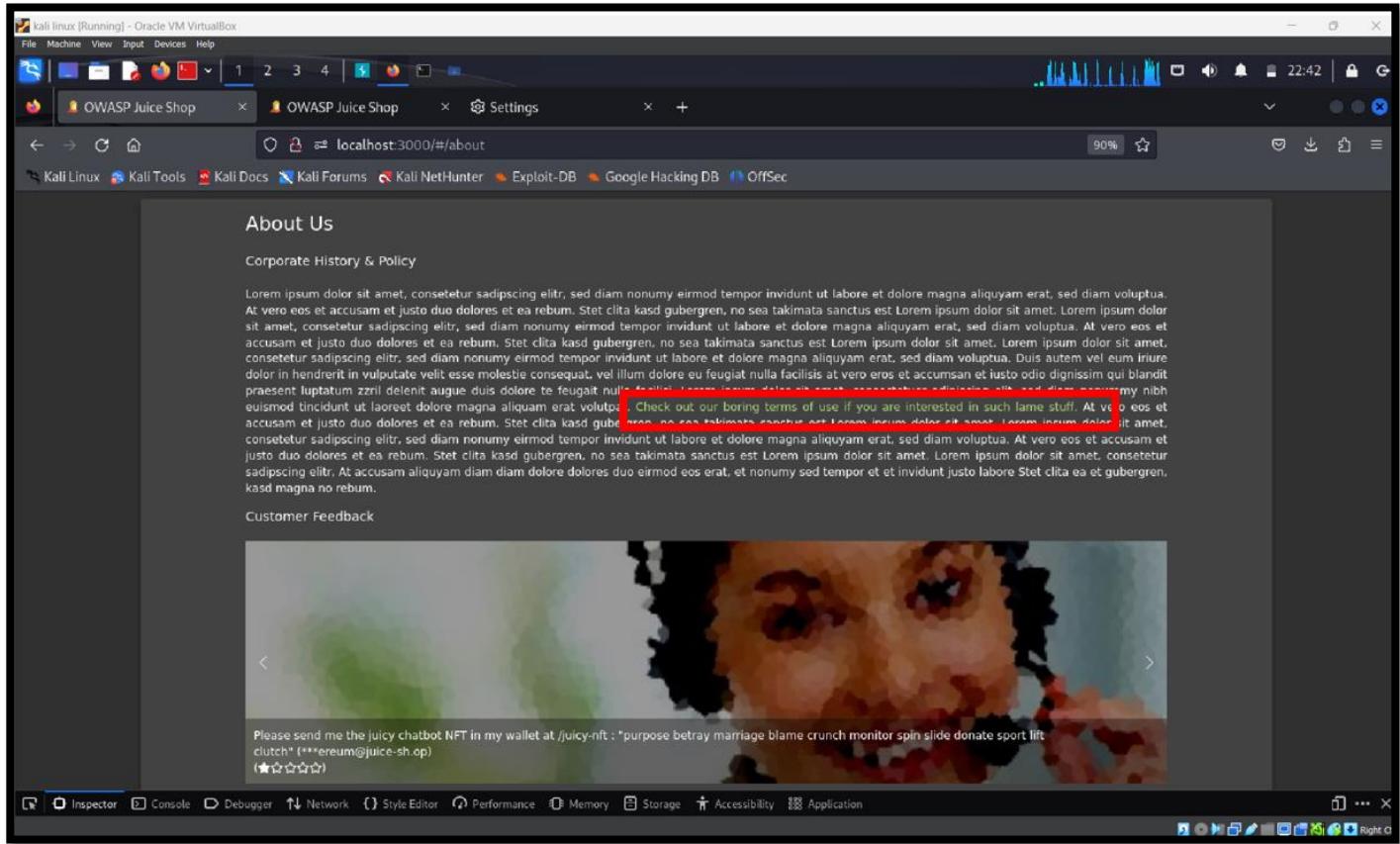
### Real-World Scenarios:

- Exploiting vulnerabilities like IDOR and inadequate access controls can lead to severe consequences.
- Attackers may impersonate users, post malicious content, or manipulate sensitive data.
- This can result in reputation damage, legal consequences, and financial losses for organizations.
- The attacker can steal information from other users and conduct passive reconnaissance.

## TASK 2.1

### Access a confidential document

In this challenge, we tried to expose a confidential document in the Juice Shop application. We attempted to search for something useful within the Juice Shop application. An interesting item appeared on the about page of the Juice Shop.



We opened the Burp tool to inspect this link. While inspecting this request, we found a directory named FTP containing a file named legal.md.

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Request to http://localhost:3000 [127.0.0.1]

Pretty Raw Hex

```

1 GET /ftp/legal.md HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=9gKezBVxE6wbOrNpYaJAPjtNhNfvr3uvvTe4h18G05v8MyDnq7ZLW243Xl1; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwvZGFOySI6eyJpZC16OwcidXNlc5hbWliOiIiLCJlWFBpbCI6Im1jLnNhZmVzZWYyY2hAanVpY2Ut2gb3AiLC2wYXNzd29yZC16Im1wM2Y0YjBtYThhNDU4ZnEwYNNkYzAyY2RiOTUzYmM4IiwiZG5GImNlc3RhbmVvIiwiZG9rZm1iLcJ3YmNOTGsmNSJcCIGIjEyn4vljAuMSIsInByb2ZpbGVjbFnZSI6InFzc2V0cy9vdWJsaWMaW1hZ2VzL3WbG9hZMvZGVxYXVsdc5zdmciLCJ0b3RwU2VjcmVOIjoiIiiviaXNBY3RpdmUiOnPydwUsInIyZWF0ZWRbdCI6IjIwMjQtMDMtMTUgMDAGNTMSMdCmJjAOICswMDowMCIsInVwZGF0ZWRbdCI6IjIwMjQtMDMtMTUgMDAGNDUuMDY4ICswHdoMCIsInRlbgV0ZWRbdCI6bnVsBH0sInlhdc16HTcxMD02Ndc1MH0.S5EHLqo16AM3dhNZNINfWtDYo12dc3llaPAT502X4VrcAxo_oDw4KnbPzr29ASu1NL4dwTtqgPSS1KaXlIpqjgkN-zEtPC3pFZ4cjEk0eA3nA1cg0gwPnZ0A3bsFgfGpIZaJ-nEuusbZP8LvG0tgJwjDhj3gUm5rB9ftA8
10 Upgrade-Insecure-Requests: 1
11
12

```

After forwarding the request to the server, we found this file. But we needed to access a confidential file, so we inspected the directory named FTP. We opened the Burp tool to inspect this link.

Request raw Hex

```

1 GET /ftp/legal.md HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=9gKezBVxE6wbOrNpYaJAPjtNhNfvr3uvvTe4h18G05v8MyDnq7ZLW243Xl1; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwvZGFOySI6eyJpZC16OwcidXNlc5hbWliOiIiLCJlWFBpbCI6Im1jLnNhZmVzZWYyY2hAanVpY2Ut2gb3AiLC2wYXNzd29yZC16Im1wM2Y0YjBtYThhNDU4ZnEwYNNkYzAyY2RiOTUzYmM4IiwiZG5GImNlc3RhbmVvIiwiZG9rZm1iLcJ3YmNOTGsmNSJcCIGIjEyn4vljAuMSIsInByb2ZpbGVjbFnZSI6InFzc2V0cy9vdWJsaWMaW1hZ2VzL3WbG9hZMvZGVxYXVsdc5zdmciLCJ0b3RwU2VjcmVOIjoiIiiviaXNBY3RpdmUiOnPydwUsInIyZWF0ZWRbdCI6IjIwMjQtMDMtMTUgMDAGNTMSMdCmJjAOICswMDowMCIsInVwZGF0ZWRbdCI6IjIwMjQtMDMtMTUgMDAGNDUuMDY4ICswHdoMCIsInRlbgV0ZWRbdCI6bnVsBH0sInlhdc16HTcxMD02Ndc1MH0.S5EHLqo16AM3dhNZNINfWtDYo12dc3llaPAT502X4VrcAxo_oDw4KnbPzr29ASu1NL4dwTtqgPSS1KaXlIpqjgkN-zEtPC3pFZ4cjEk0eA3nA1cg0gwPnZ0A3bsFgfGpIZaJ-nEuusbZP8LvG0tgJwjDhj3gUm5rB9ftA8
10 Upgrade-Insecure-Requests: 1
11
12

```

On the HTTP history, we got the URL path, so we found something important for this task. Therefore, we sent this URL path to repeater.

kali linux [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Dashboard Target Proxy Intruder Repeater View Help

Repeater

Send Cancel < >

Target: http://localhost:3000 | HTTP/1.1

Request

Pretty Raw Hex

1 GET /ftp/legal\_md HTTP/1.1

2 Host: localhost:3000

3 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/115.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate, br

7 Connection: close

8 Referer: http://localhost:3000/

9 Cookie: language=en; welcomebanner\_status=dismiss; cookieconsent\_status=dismiss; continueCode=q9KerzBVzEfwhOrNpYiAPJtMhNfvhr3uVtEhBgc5w8MyDng72LWZ49Xl; token=eyJxExAL01JKY101LCJhbGciOiJSUzI1NiJ9.eyJzdjF0dXNlOiJzdnNZNXNzIiViZGPOyS16eyJpZC16OCwdhNMcm5hDWm1iL1iC1iWfPbC16i1iLnNhVzZWfFy2hAnVpY2Utc2guBAiLCjWvNzId29yZC16iNzWh2Y0YiBYTh1NDU42nEwYMKy2aYr2p4OTUzYmM1iwiG9sZSI6iNlc3RvbWViiZGVygdh1VG9rWZ4i0i1LCj9XN0TG9naW5jC16iEyNy4LjAhMSLInBybz2pbgvJbfwnZSLi6fFc2V0cy9wdMsWmvarlh2ZLw3WvbGhZMzyZCvnXVsdCSzdrclLCJ0B3fWu2VjcmV0i1i1i1i1aXNB7SPdmuL0RydhUs1mNyZWF02WRBcG1jIwH0QDMHUGMDA9TMSHdC0uAi01CsMdwoMCIsInWzGF0ZWRBdC16iJwH0QH0MHTUgjD4EM9A6NDUuMDY4LcsWb0WkC1mRlbgvQ2WfBcG16bNVbHbOsInh4c60TcXxDQ2N2c1MM0.SSEc0L0.6AMhdhNzNnf1DYo1zdc3LLeAt502k04vAxo\_oB4KNr8pzs2945uiNLdgg1TtgPS51kex1a3jgAKK-zEtCP5pZ4CjEK9eA9nAlcg0g9PnZ0A13b5F9fOpj2aJ-nEuusb2P8LvG0tGjvDhj3gJnd5FB9ft48

10 Upgrade-Insecure-Requests: 1

11

12

Response

Pretty Raw Hex Render

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 5

Request headers 9

Notes

After that, we deleted the legal.md file path from the request and sent that request again. On the right-hand side in response, we got some information. We scrolled through it last. After inspecting the response, we found a file named acquisitions.md.

Request

```

1 GET /ftp HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
10 continueCode=9gkezBVxE6vbOrNyAJP0tNnhFvhru3uVvTe4h1B8go5ByhDng7ZLW24Xkl; token=eyJ0eXA1OjJKV101LCJhbGciOiJSUzI1NiJ9.eyJzdGFpdkIjMLOiJzdnNjZXNzIiVlZP0YSl6eyJpZC160cwidXNmcShmJ0i1LCJtUWFpbCI6Ij1LnNhZnVzDWFyY2hAanVpV2Utc2ub3a1CjVvNzd29yZC161IwM2Y0Yj81YTh1NUu2hEWyNkYzAyY2RiUTUyM41iLvcn9eZS16In1c3RbWvYi1vZGysdXhLVG9rZW4i0iLcJyXN0Tgnw53c1C1GjEyNy4wLjAuMSisInby22pbGVbFnZS16Infc2V0cy9wdJ3swMvanhKZvzL5vbGsh2MMZ0vMyVsdczdc1Lc0b8wU2V) cVoijilivivaNkY8PpdwU0nydflus1mNy2z02NRbd(CIGj1wM) OTHMTuHDA0NTM9DCuHjA01CsHdwMCisInwZfQ2NRbd(CIGj1wM) OTHMTuHDE69DABNDUMDY1GsWDbwKC1i80bGv02wBc1C6hnsHoi1nldC1MHTxMDQ2D1M9OSShMq0i6AM83hNZN1nfV1D6i2d3LlaPA1502X4VrAcko_oBwAKN8bzr23ASuiNL4dvTqgPSl1KxLipigikN-2ETPC3pFZ4C) EK0eA3nA1cgoqwhnZ0A18sFgfpIzaJ-nBuusb2PwlvG0tGjwDh) 3gJhdsrB9ft48
11 Upgrade-Insecure-Requests: 1
12

```

Response

```

358
359

```

Inspector

We copied that file path and pasted it on the request.

Request

```

1 GET /acquisitions.md HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
10 continueCode=9gkezBVxE6vbOrNyAJP0tNnhFvhru3uVvTe4h1B8go5ByhDng7ZLW24Xkl; token=eyJ0eXA1OjJKV101LCJhbGciOiJSUzI1NiJ9.eyJzdGFpdkIjMLOiJzdnNjZXNzIiVlZP0YSl6eyJpZC160cwidXNmcShmJ0i1LCJtUWFpbCI6Ij1LnNhZnVzDWFyY2hAanVpV2Utc2ub3a1CjVvNzd29yZC161IwM2Y0Yj81YTh1NUu2hEWyNkYzAyY2RiUTUyM41iLvcn9eZS16In1c3RbWvYi1vZGysdXhLVG9rZW4i0iLcJyXN0Tgnw53c1C1GjEyNy4wLjAuMSisInby22pbGVbFnZS16Infc2V0cy9wdJ3swMvanhKZvzL5vbGsh2MMZ0vMyVsdczdc1Lc0b8wU2V) cVoijilivivaNkY8PpdwU0nydflus1mNy2z02NRbd(CIGj1wM) OTHMTuHDA0NTM9DCuHjA01CsHdwMCisInwZfQ2NRbd(CIGj1wM) OTHMTuHDE69DABNDUMDY1GsWDbwKC1i80bGv02wBc1C6hnsHoi1nldC1MHTxMDQ2D1M9OSShMq0i6AM83hNZN1nfV1D6i2d3LlaPA1502X4VrAcko_oBwAKN8bzr23ASuiNL4dvTqgPSl1KxLipigikN-2ETPC3pFZ4C) EK0eA3nA1cgoqwhnZ0A18sFgfpIzaJ-nBuusb2PwlvG0tGjwDh) 3gJhdsrB9ft48
11 Upgrade-Insecure-Requests: 1
12

```

Response

```

358
359

```

Inspector

Afterward, we sent this request again. We went back to the Burp repeater and edited the file to acquisitions.md. After editing, we sent the request again. After checking the response, we found that this document was confidential.

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Request

```

1 GET /ftp/acquisitions.ms HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:3000/
9 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
continueCode=9gKzBVxE5wbdNpYyJAPtNNFvh3uVtTe4h18Gg5vBMyDng7ZLw243Wl1; token=eyJpXAxI01JKV01LcJhGc101J9u11N1J9.yJzGCP0dXMs01JzdwNjZXn21v1ZGPOYS1eeyjpcZC60CwidXNcnShbMUs01isLCJ1bWfpbC161m1uNmNzv2WFy2hAnvY2Urtc2gbub3a1LCjwYMNzd29yZC161nIVw2Y09sBlYTthND4m5yWWYzAy2i0TUzrM41ivicm9sZS161m1c19tbbnvylivizovsdxh1vg9rZW4i011C3jsYXNTD9gnw5Jc161jEyNy4wLjAU5Ms1Inby22pbGVJbWFnZS161nFz2W0cy9vdWjsawMvan1hZZv2L3wv0Gn2HMyZGwvYxsdSzdmcilC10b3HwQV1cwVOijoi11viaXNBY8PpdwUOnRydWUs1mNy2WF02WFBc1G1; iWj0tMDM HTlgMDA9NTM9MDcUmj1AD0Cswh0wMCis1nVwZC0f2WFBc1G1; iWj0tMDM HTgj0tMD0BMDA6NDUuMDY41C5wMDwMCis1mRbGv02WFBdC16mhdC16MtxMDQ2NDc1MHOSS6M1qq0i6AM5shhNZIIifwT0y12dc3L1pAt502X4VrcAko_0w4KNr8prz23Asu1L4dWtqgPSS1kx1IppgkN-zEtPc3pFZ4C)BK0eA8nA1cg0gwPn2oAi3bsFgFqjzaJ-nEusbzP8lwGtQjwDhj3gWJ5rb9f4AB
10 Upgrade-Insecure-Requests: 1
11
12

```

Response

```

1 HTTP/1.1 200 OK
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 Accept-Ranges: bytes
8 Cache-Control: public, max-age=0
9 Last-Modified: Tue, 19 Dec 2023 13:12:10 GMT
10 ETag: W/"98d-18c62348810"
11 Content-Type: text/markdown; charset=UTF-8
12 Content-Length: 909
13 Vary: Accept-Encoding
14 Date: Fri, 15 Mar 2024 09:53:48 GMT
15 Connection: close
16
17 # Planned Acquisitions
18
19 > This document is confidential! Do not distribute!
20
21 Our company plans to acquire several competitors within the next year.
22 This will have a significant stock market impact as we will elaborate in
23 detail in the following paragraph:
24
25 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
26 eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
27 voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
28 clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
29 amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
30 nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
31 sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
32 rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
33 ipsum dolor sit amet.
34
35 Our shareholders will be excited. It's true. No fake news.
36
37

```

Inspector

Request attributes: 2

Request query parameters: 0

Request body parameters: 0

Request cookies: 5

Request headers: 9

Response headers: 14

After turning off the interceptor in Burp, we could see that we successfully solved this challenge and accessed the confidential document.

kali linux [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

OWASP Juice Shop Settings

localhost:3000/#/about

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

OWASP Juice Shop

You successfully solved a challenge: Confidential Document (Access a confidential document.)

About Us

Corporate History & Policy

Customer Feedback

### **Vulnerabilities Exploited by Task 2.1:**

- Listing lax file directory access rules led to the exploitation of a traversal vulnerability.
- Inadequate authorization checks resulted in illegal access to confidential documents.

### **How Vulnerabilities Can Be Fixed or Addressed:**

- Implement Access Controls
- Validate User Input
- Encrypt Confidential Documents

### **Real-World Scenarios:**

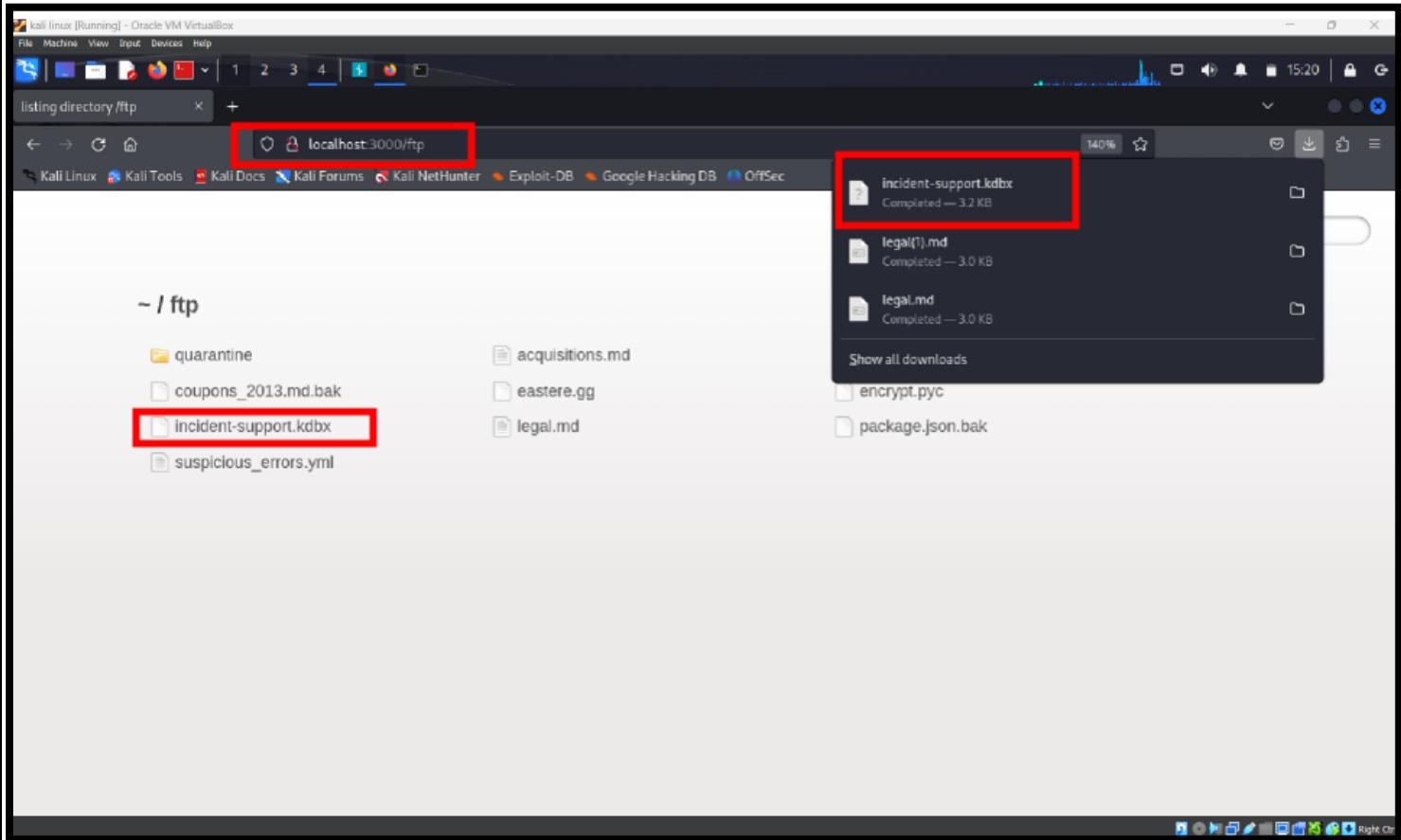
- Exploiting directory traversal vulnerabilities can lead to unauthorized disclosure of sensitive information.
- This may include bank records, consumer information, or confidential documents.
- Consequences for the affected organization can include financial losses, loss of trust from customers, and legal penalties.

## TASK 2.2

### Gain access to any access log file of the server

In the previous challenge, we had already accessed the directory called FTP.

In this directory, after inspecting all the files, we found a file called incidentsupport.kdbx. We attempted to



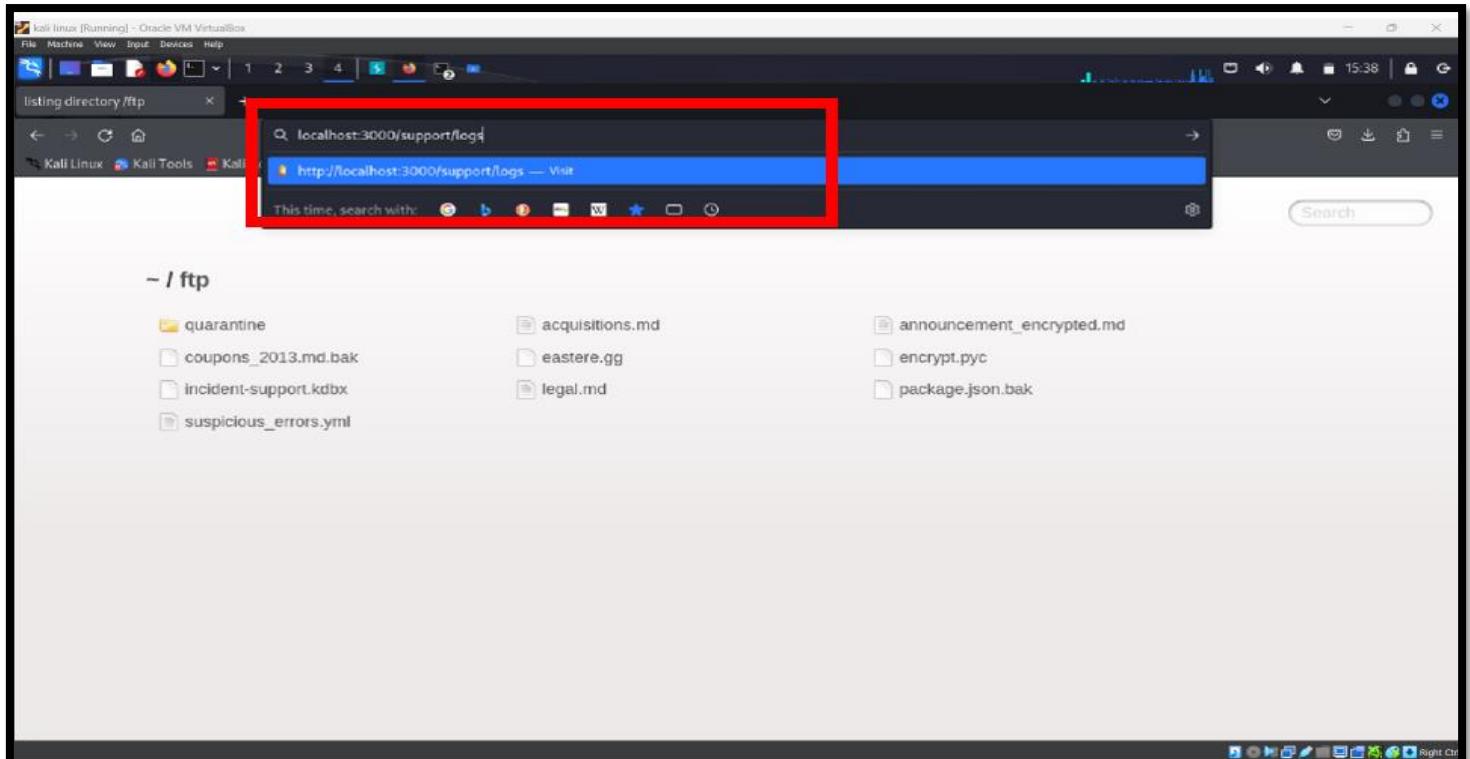
download the file and inspect it. However, no application worked for this type of file.

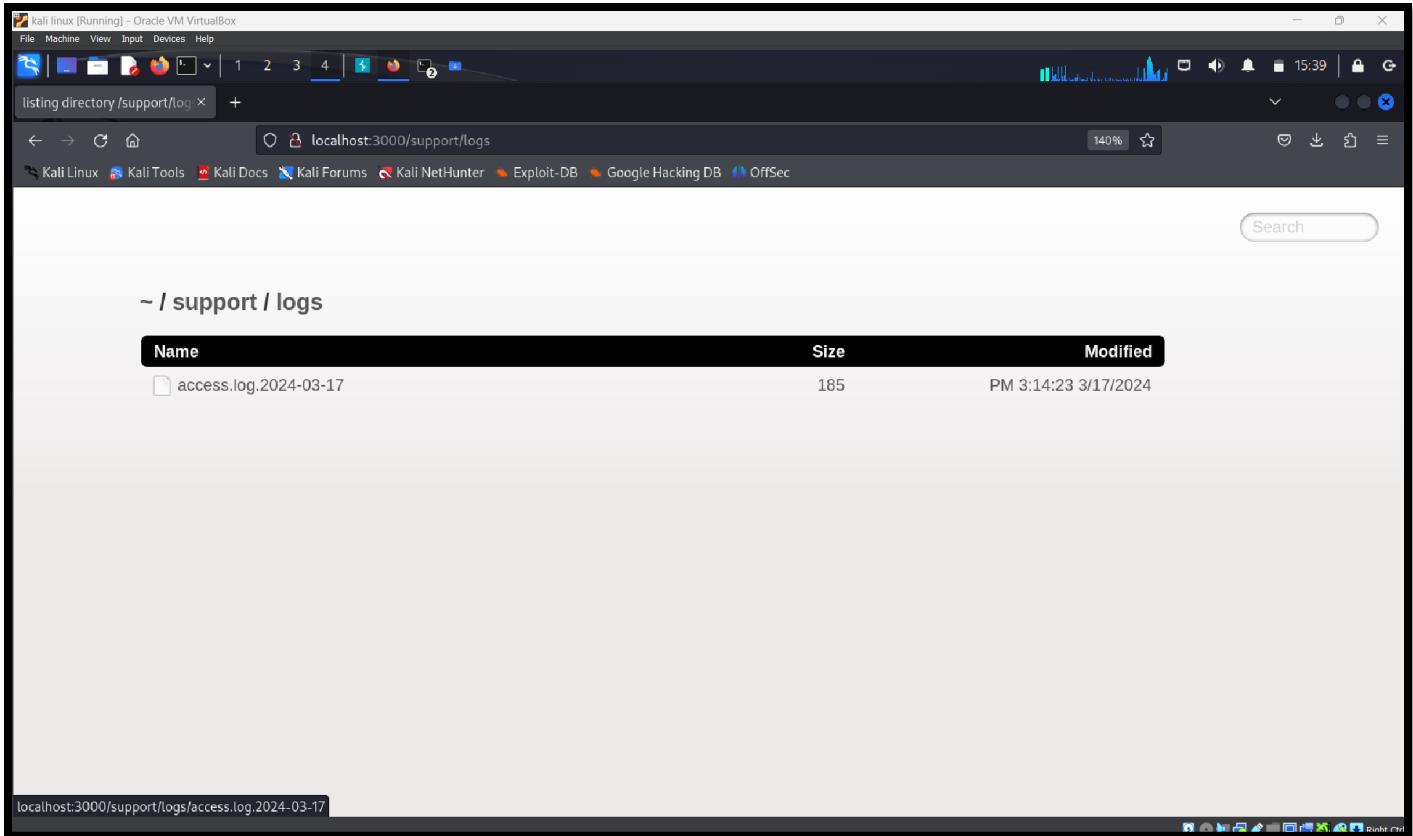
```
shivanivar@shivanivar: ~/Downloads
File Actions Edit View Help favorite-hiking-place.png
(shivanivar@shivanivar)-[~]
$ cd Downloads
(shivanivar@shivanivar)-[~/Downloads]
$ ls
IMG_4253.png
SENG8060_project_part01(1).doc
SENG8060_project_part01.doc
ZAP_2_14_0_unix.sh
best1050.txt
burpsuite_community_linux_v2024_1_1_6.sh
cacert.der
favorite-hiking-place.png
incident-support.kdbx
juice-shop-16.0.0-node18_linux_x64.tgz
juice-shop_16.0.0
legal(1).md
legal.md

(shivanivar@shivanivar)-[~/Downloads]
$ cat incident-support.kdbx
```

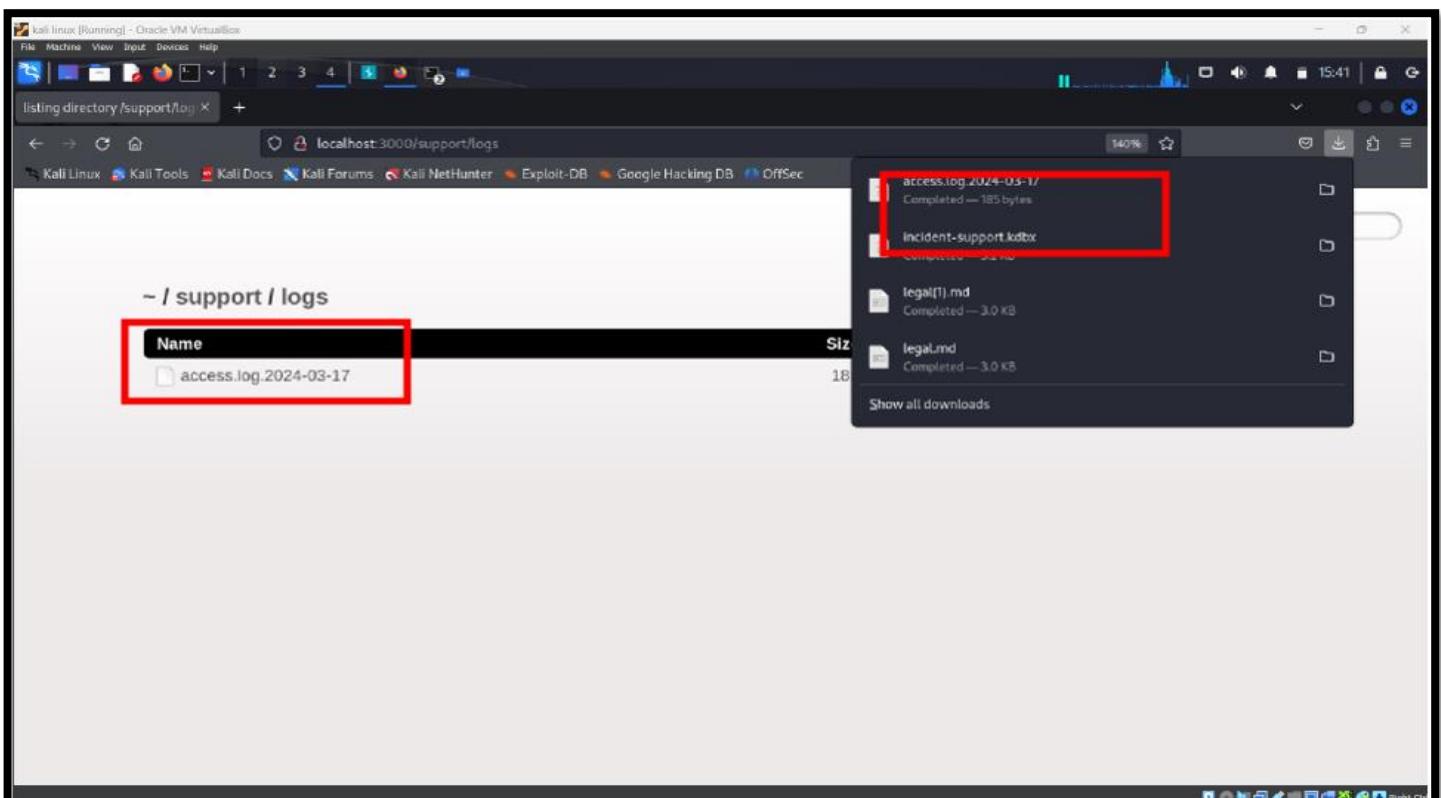
We used the cat command to open this file in the terminal and found its contents.

We then tried to open the support directory and found 1 access log files.





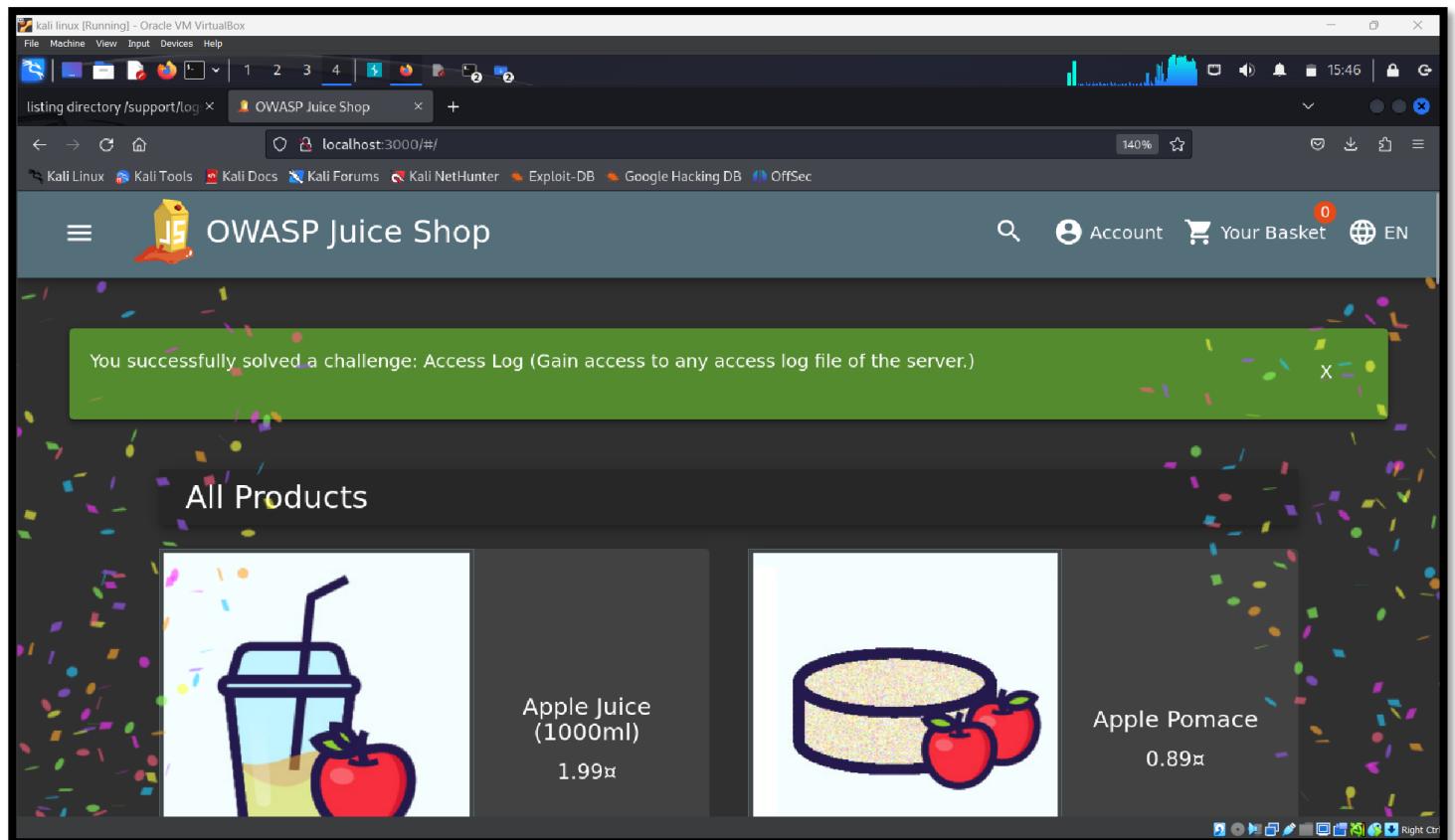
We were able to access and download these files.



Upon downloading the access log files, we found this type of data in them.

```
~/Downloads/access.log.2024-03-17 - Mousepad
File Edit Search View Document Help
New Open Save Close Find Replace Copy Cut Paste Undo Redo Select All Find Next Find Previous Select All
1 ::ffff:127.0.0.1 - - [17/Mar/2024:19:14:23 +0000] "GET /favicon.ico HTTP/1.1" 200 - "http://localhost:3000/ftp" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
2
```

We then went back to the OWASP juice shop home page and confirmed that we had successfully solved this challenge in the Juice Shop.



### **Vulnerabilities Exploited by Task 2.2:**

- **Inappropriate file permissions:** Allows illegal access to log files and sensitive data.
- **The lack of file type validation:** Enables potentially sensitive files accessible.

### **How Vulnerabilities Can Be Fixed or Addressed:**

- Strict Access Controls for Sensitive Documents
- Implement strict access controls.
- Use file type validation for approved file viewing/download.
- Protect sensitive documents/log files in secure, encrypted locations.

### **Real-World Scenarios:**

- Unauthorized access to confidential documents and log files.
- Potential consequences include data breaches, regulatory fines, and reputational harm.
- Cloud storage service providers may be targets.
- Malicious attackers can bypass proper authorization and access directories.

## TASK 3

### Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to

We were logged in to the OWASP Juice Shop with the Administrator's user credentials which were caused by the Sensitive Data Exposure. We entered the administrator's credentials in the "Log in Box" by tracking the email address and password.

The screenshot shows the OWASP Juice Shop administration interface. In the 'Registered Users' section, there are seven entries:

- admin@juice-sh.op
- jim@juice-sh.op
- bender@juice-sh.op
- bjoern.kimmich@gmail.com
- ciso@juice-sh.op
- support@juice-sh.op
- morty@juice-sh.op
- mc.safesearch@juice-sh.op

In the 'Customer Feedback' section, there are several reviews:

- I love this shop! Best products in town! Highly recommended! (\*\*\*in@juice-sh.op) ★★★★★
- Great shop! Awesome service! (\*\*@juice-sh.op) ★★★★
- Nothing useful available here! (\*\*\*der@juice-sh.op) ★
- Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray mariage..." (\*\*\*nft@juice-sh.op) ★
- Incompetent customer support! Can't even upload photo of broken purchase!... (\*\*\*customer@juice-sh.op) ★★
- This is the store for awesome stuff of all kinds! (anonymous) ★★★★★
- Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★★
- Keep up the good work! (anonymous) ★★★

To solve this challenge firstly we searched on Google where we saw a bunch of links related to that but that's not what we were looking for.

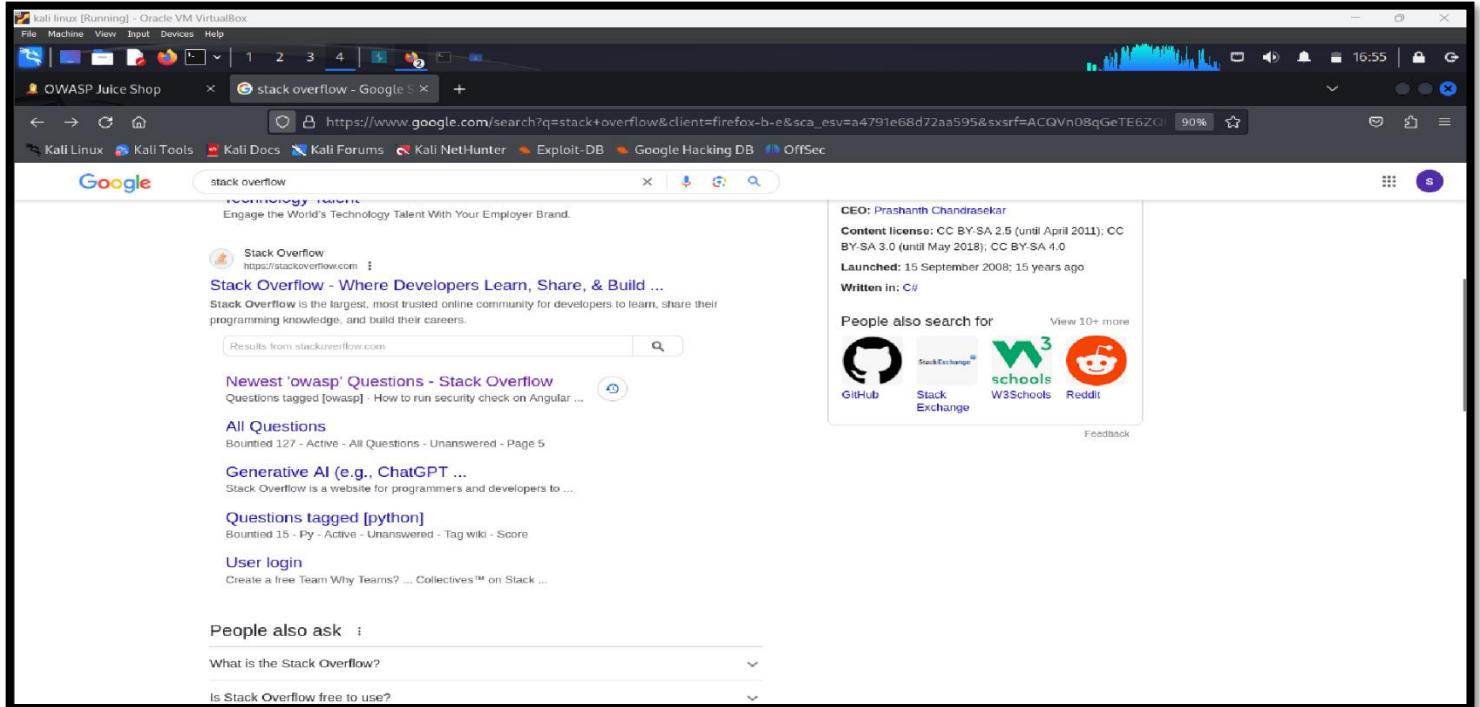
The screenshot shows a Google search results page for the query "owasp juice shop leaked access logs stackoverflow". The top result is a Stack Overflow post titled "Hacking OWASP's Juice Shop Pt. 61: Leaked Access Logs". The post discusses how to use a normal query to zap the database and provides a link to the "Leaked Access Logs" page on the OWASP Juice Shop.

Other search results include:

- A GitHub repository for "owasp-juice-shop/README.md" with a note about leaked access logs.
- A question on Stack Overflow about how to use a normal query to zap the database.
- A post on hackerbartender.com about leaked access logs.
- A GitHub repository for "owasp-juice-shop/README.md" with a note about leaked access logs.
- A GitHub repository for "owasp-juice-shop/README.md" with a note about leaked access logs.

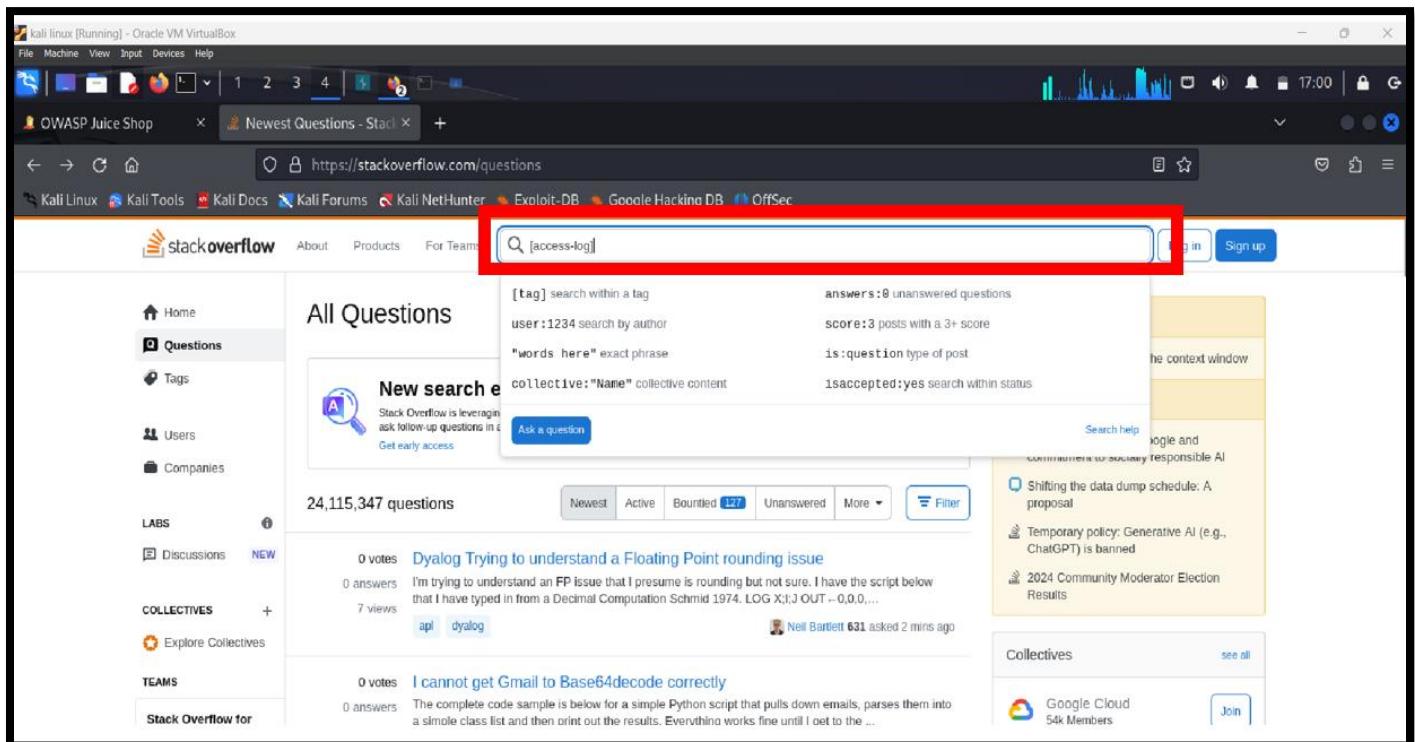
so we did some research on it. We found out that we had to go to a Stack Overflow solution in the Firefox browser.

Then we went to the question, then we searched with the square brackets which happened to be a tag in



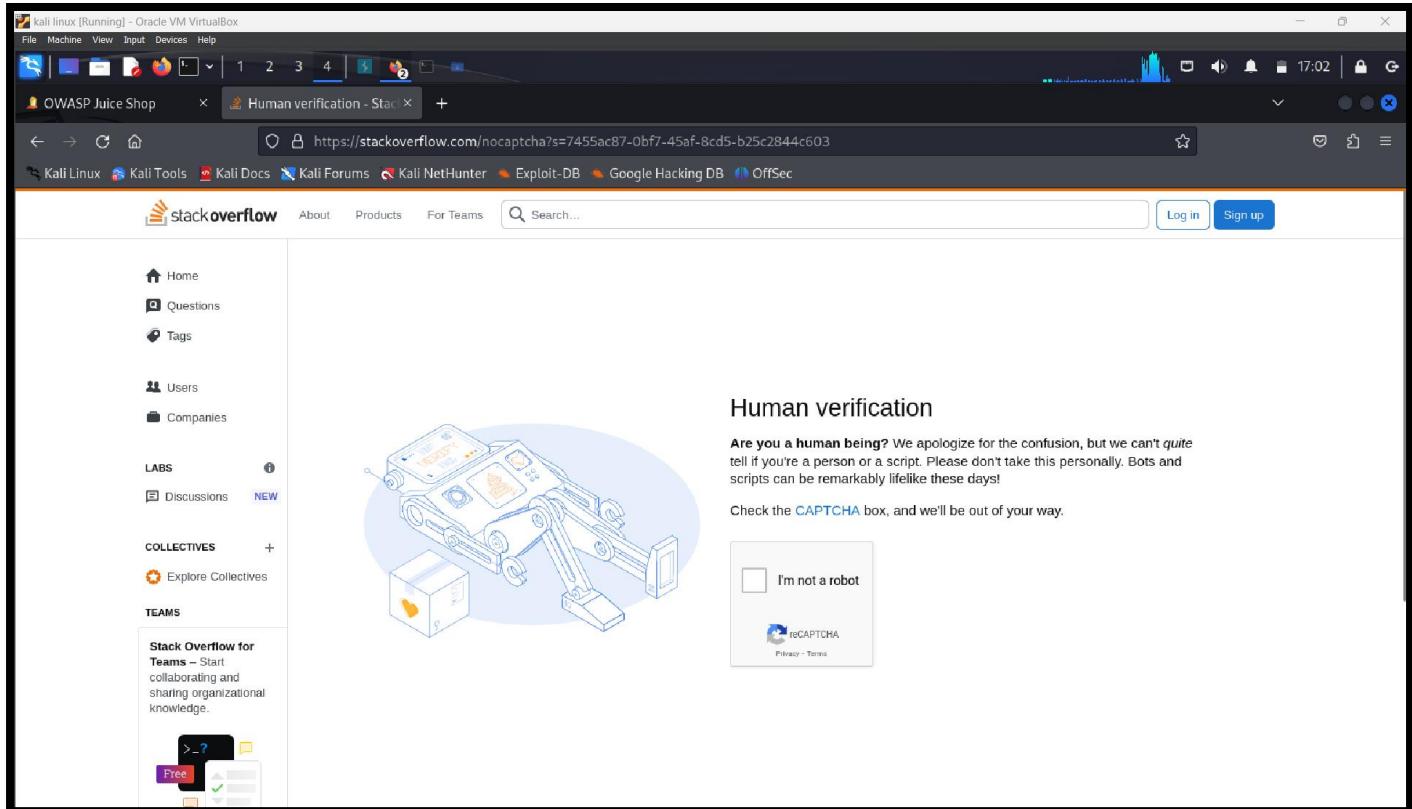
Google search results for "stack overflow". The top result is the official Stack Overflow website. The sidebar on the right provides details about the site, such as its CEO, license, and search links to related platforms like GitHub, Stack Exchange, and W3Schools.

Stack Overflow and we typed "access-log" and then we closed the square bracket. We looked for it.

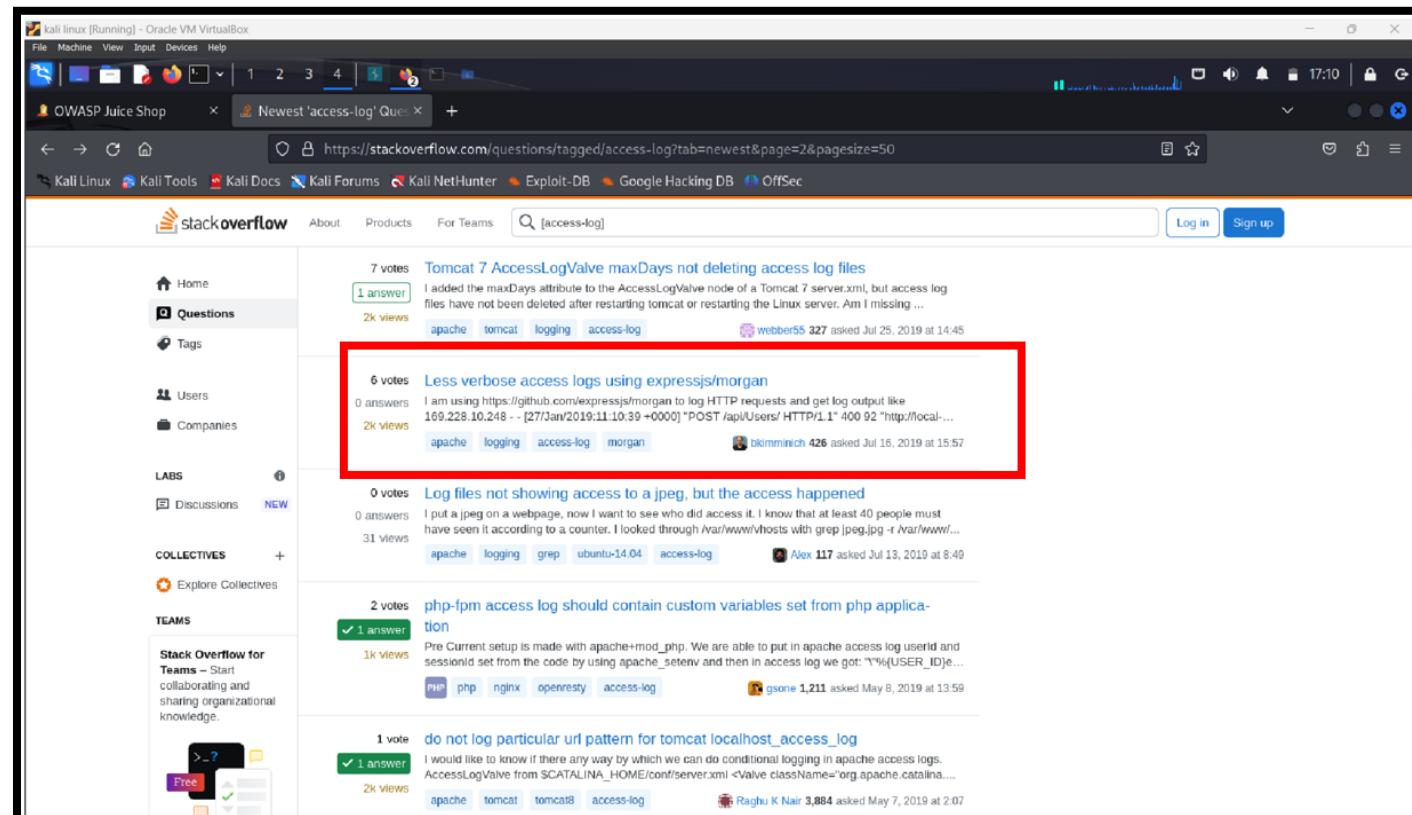


The screenshot shows the Stack Overflow homepage with a search query "[access-log]" entered into the search bar. The search results page displays several questions related to "access-log", with one prominent result by user "dyalog" asking about floating-point rounding issues.

It asked about human verification. We solved that captcha



then scrolled through multiple solutions, where we finally found something that was relevant to our task.



We saw there was localhost:3000 related information so we opened that solution. Then we found a link to Pastebin.

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open. The URL in the address bar is <https://stackoverflow.com/questions/57061271/less-verbose-access-logs-using-expressjs-morgan>. The page content discusses log verbosity and includes a link to a pastebin at <https://pastebin.com/4U1v1UjU>. A red box highlights this link.

When we went over that link, we found different events and timestamps, each event occurred, so what we were basically looking for in this was log dump, is a keyword is password, so here we found out the password. So, it's invoking a get request for the current password to be this and then the new password would be changed over to this

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open. The URL in the address bar is <https://pastebin.com/4U1v1UjU>. The page content is a log dump from a server. A red box highlights the password "kjwmmw39VFYEMK2AT799-1Fg1L6t66fB&new=sjss22%ME2%82%AC55jaJasj!.k&repeat=sjss22%ME2%82%AC55jaJasj!.k8".

A screenshot of a Firefox browser window running on Kali Linux. The title bar says "kali linux [Running] - Oracle VM VirtualBox". The address bar shows the URL "https://www.google.com/search?client=firefox-b-e&q=url+encoder". The search term "url encoder" is entered in the search bar. Below the search bar, there are filters for "All", "Videos", "Images", "Shopping", "News", "More", and "Tools". There are also buttons for "Decode", "Java", "JavaScript", "Python", "For SVG", "C#", "PHP", "HTML", and "Org". The search results page displays approximately 27,400,000 results found in 0.29 seconds. The first result is a link to "URL Encode" at <https://www.urlencoder.org>. The second result is "URL Encode and Decode - Online" from meyerweb.com. The third result is "URL Decoder/Encoder" from W3Schools.

A screenshot of a Firefox browser window running on Kali Linux. The title bar says "kali linux [Running] - Oracle VM VirtualBox". The address bar shows the URL "https://meyerweb.com/eric/tools/dencoder/". The page title is "URL Decoder/Encoder". A large text input field contains the string "0Y8rMnw\$\*9VFYE%C2%A759-!Fg1L6t&6lB&". Below the input field are two buttons: "Decode" and "Encode". A list of instructions follows:

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

so we copied that password and went to URL encoder/decoder and pasted that. After we clicked on the decode button so finally we got the decoded password.

0Y8rMnw\$\*9VFYEg59-!Fg1L6tG6lBc

Decode | Encode

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

We tried to log in as an administrator by using the SQL injection. We clicked on the account and logged out. We logged in again, on the login page to input the email and password.

We searched the administration on localhost.

You successfully solved a challenge: Admin Section (Access the administration section of the store.)

## Administration

Registered Users	Customer Feedback
admin@juice-sh.op	1 I love this shop! Best products in town! Highly recommended! (**@juice-sh.op) ★★★★☆
jim@juice-sh.op	2 Great shop! Awesome service! (**@juice-sh.op) ★★★★☆
bender@juice-sh.op	3 Nothing useful available here! (**der@juice-sh.op) ★
bjoern.kimminich@gmail.com	21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray mar... ★
ciso@juice-sh.op	Incompetent customer support! Can't even upload photo of broken purchase! ★★

Here we found the list of all registered administrators. All we needed to do was match which user's hashed password was this.

primejuice@juice-sh.op  
bender@juice-sh.op  
bjoern.kimminich@gmail.com  
ciso@juice-sh.op  
support@juice-sh.op  
morty@juice-sh.op  
mc.safesearch@juice-sh.op  
j12934@juice-sh.op  
wurstbrot@juice-sh.op

2 Great shop! Awesome service! (\*\*@juice-sh.op) ★★★★☆  
3 Nothing useful available here! (\*\*der@juice-sh.op) ★  
21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray mar... ★  
Incompetent customer support! Can't even upload photo of broken purchase! ★★  
This is the store for awesome stuff of all kinds! (anonymous) ★★★★☆  
Never gonna buy anywhere else from now on! Thanks for the great service! (anonymo... ★★★★☆  
Keep up the good work! (anonymous) ★★★★☆

A screenshot of a web browser window titled 'OWASP Juice Shop' on the address bar. The URL is 'localhost:3000/#/administration'. The page displays a list of reviews. One review, from 'j12934@juice-sh.op', is highlighted with a red rectangular box.

User	Review	Rating
admin@juice-sh.op	I love this shop! Best products in town! Highly recommended! (**@juice-sh.op)	★★★★★
jim@juice-sh.op	Great shop! Awesome service! (**@juice-sh.op)	★★★★★
bender@juice-sh.op	Nothing useful available here! (****der@juice-sh.op)	★
bjoern.kimminich@gmail.com	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriag..."	★
cisa@juice-sh.op	Incompetent customer support! Can't even upload photo of broken purchase!...	★★
support@juice-sh.op	This is the store for awesome stuff of all kinds! (anonymous)	★★★★★
morty@juice-sh.op	Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)	★★★★★
mc.safesearch@juice-sh.op	Keep up the good work! (anonymous)	★★★
j12934@juice-sh.op	(highlighted)	
wurstbrot@juice-sh.op		

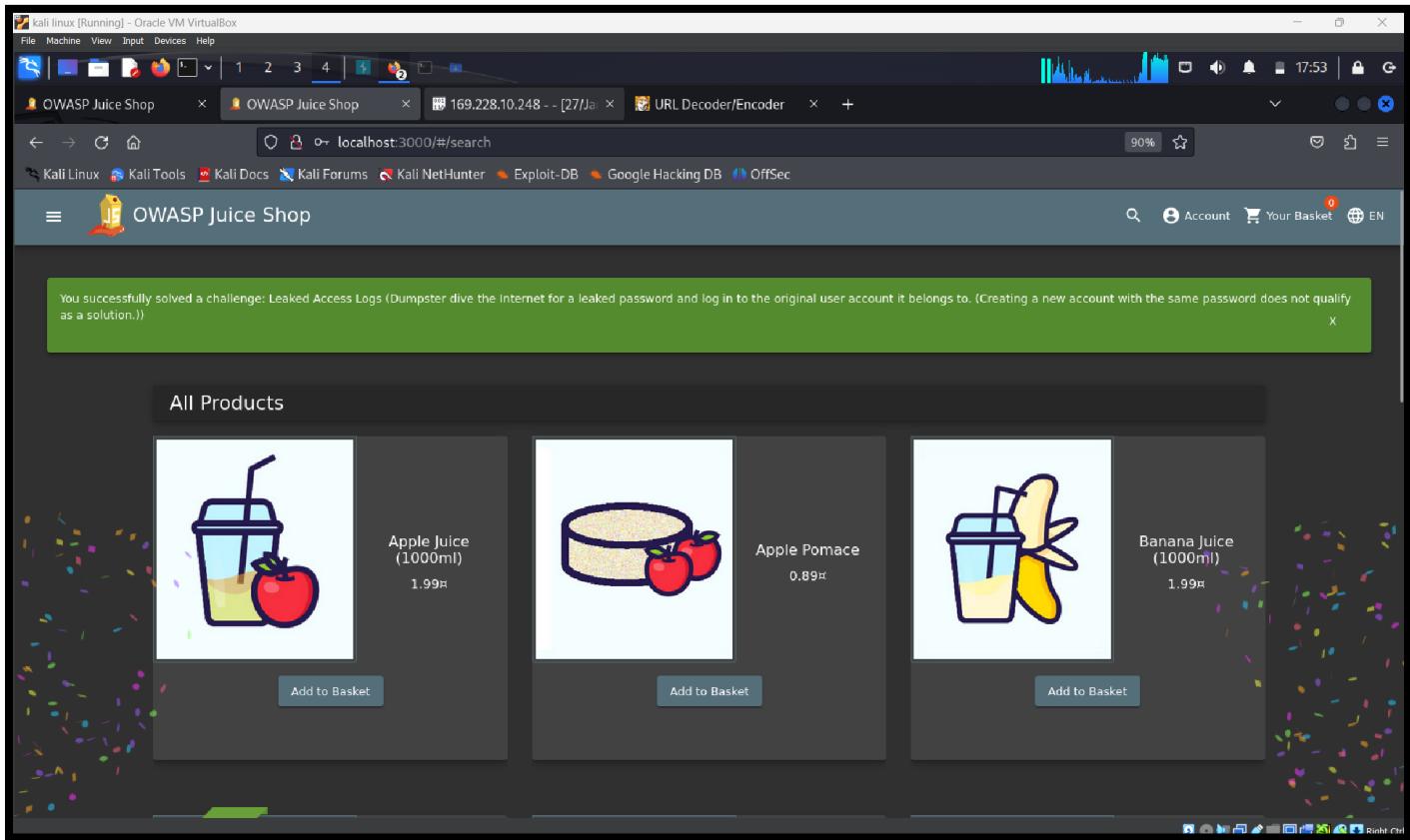
We manually matched one by one all the emails of administrators with that password and the email j12934@juice-sh.op matched with that.

A screenshot of a web browser window titled 'OWASP Juice Shop' on the address bar. The URL is 'localhost:3000/#/login'. The page shows a 'Login' form. The 'Email' field contains 'j12934@juice-sh.op' and the 'Password' field contains '0Y8rMnww\$\*9VFE\$59-fFg1L6t&6IB'.

Login Form Fields:

- Email: j12934@juice-sh.op
- Password: 0Y8rMnww\$\*9VFE\$59-fFg1L6t&6IB
- Forgot your password?
- Log in button
- Remember me checkbox
- or
- Log in with Google button
- Not yet a customer?

With these credentials we could solve this challenge successfully.



### Vulnerabilities' Exploited by Task 3:

- Password exposure due to data breaches/leaks.
- Unauthorized access to user accounts.
- Risk of security breaches for users and companies.
- Hacked passwords made public online.

### How Vulnerabilities Can Be Fixed or Addressed:

- Implement strong password standards requiring multi-factor authentication, unique passwords, and frequent password changes.
- Use hashing and encryption methods for secure password storage in databases.
- Regularly monitor user account activity for unauthorized access or questionable behavior.
- Educate users on password security importance and provide instructions for secure password management.
- Use threat intelligence tools to monitor and mitigate user-related password leaks.

### Real-World Scenarios:

- Database breaches can lead to leaked user credentials.
- These credentials can be shared online, allowing attackers to access user accounts.
- Attackers can exploit sensitive data, compromise user privacy, and initiate financial fraud, identity theft, or unauthorized transactions.
- Organizations can suffer financial losses, reputational damage, and legal penalties.

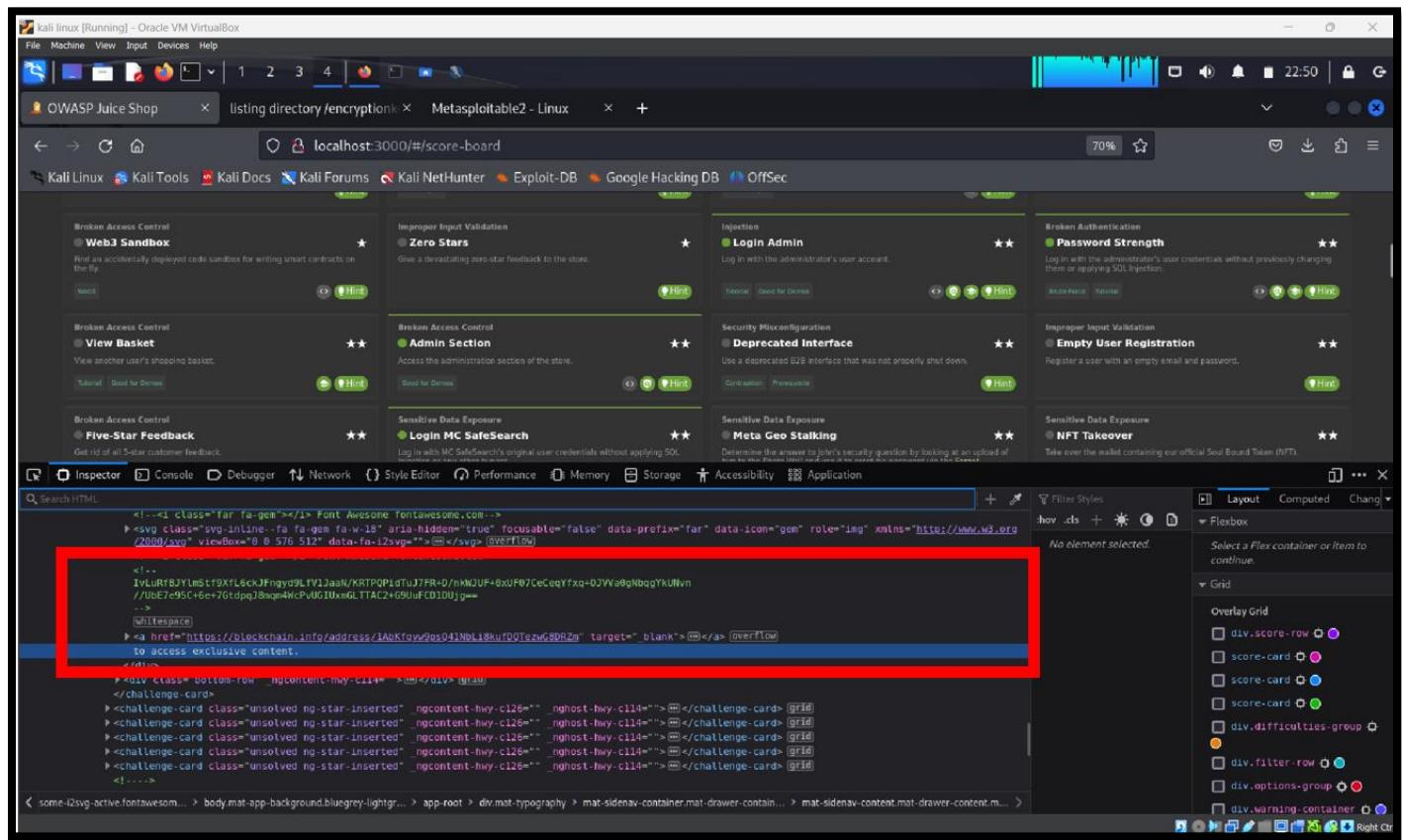
## TASK 4

Unlock Premium Challenge to access exclusive content

To solve that challenge, we first Inspected the HTML source of the corresponding row in the Score Board table, revealing a HTML comment that was obviously encrypted, which showed like this and in Screenshot:

IvLuRfBJYImStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeQYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8m  
qm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg==-->

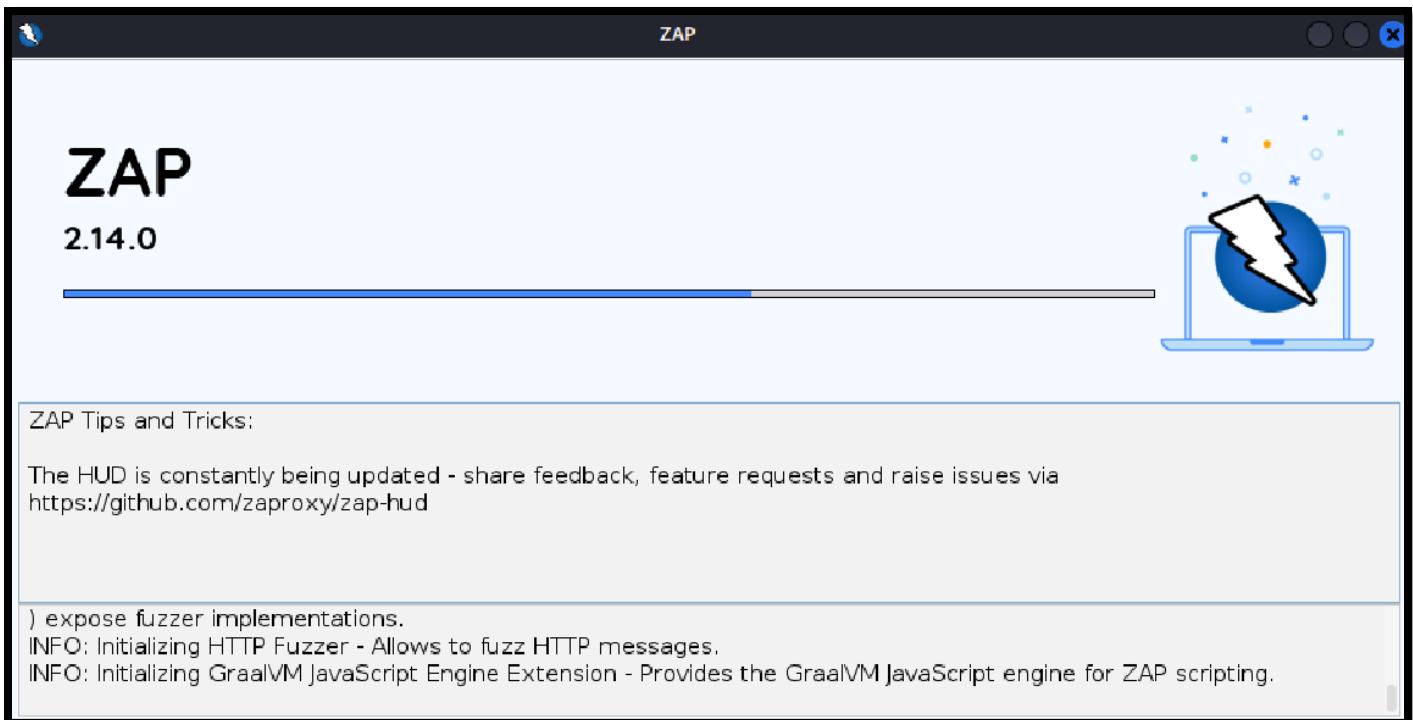
This was a cipher text that came out of an AES-encryption using AES256 in CBC mode.



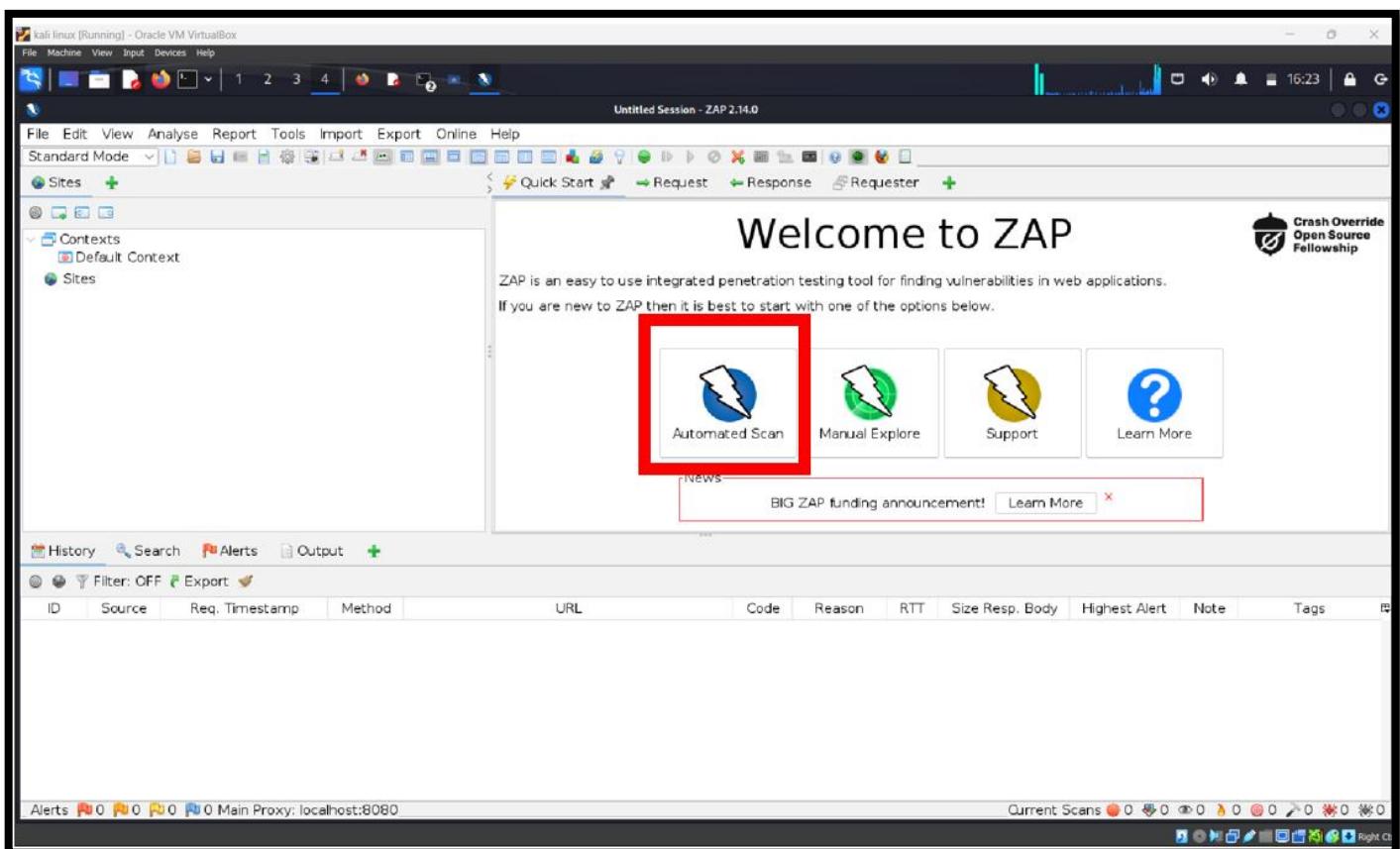
The screenshot shows a Kali Linux desktop environment with several windows open. In the foreground, a Firefox browser window displays the OWASP Juice Shop application at localhost:3000/#/score-board. The page lists various security challenges with their respective scores and descriptions. A developer tools window is open over the browser, showing the HTML source code. A specific comment block is highlighted with a red box, containing the following encrypted text:

```
<!--> IvLuRfBJYImStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeQYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8m  
qm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg==-->
```

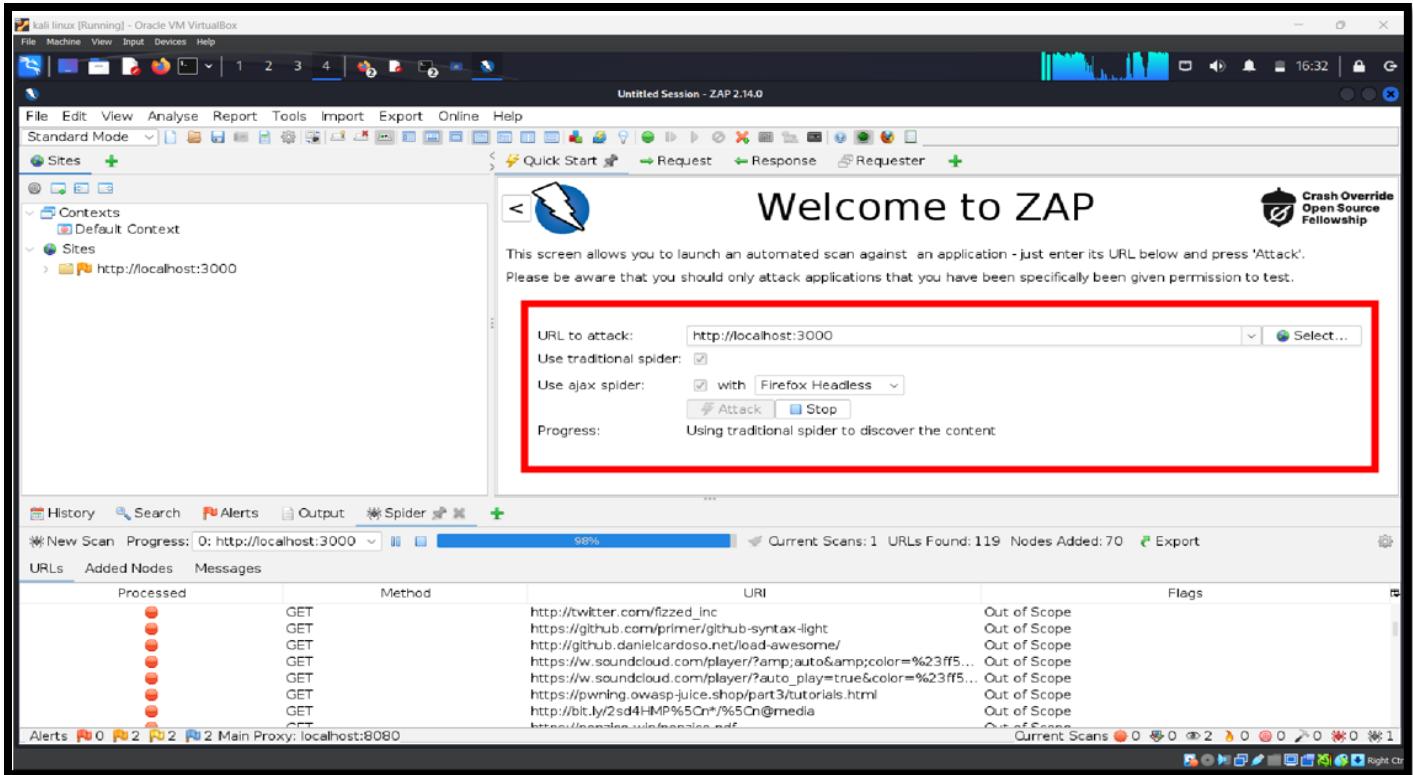
To get the key and the IV, we ran a Forced Directory Browsing attack against the application. We used OWASP ZAP for this purpose.



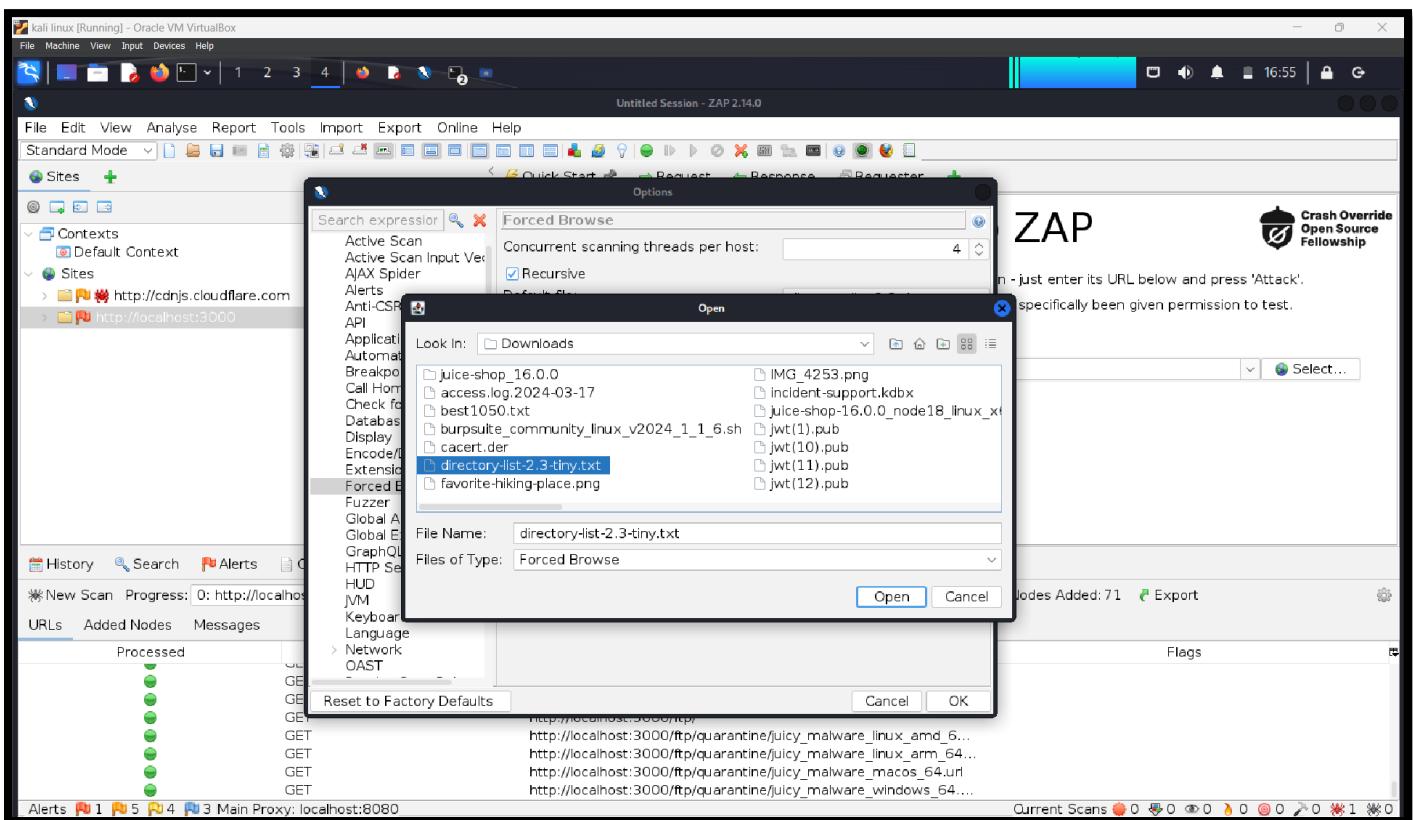
We first launched ZAP, after which we clicked on automated scan.



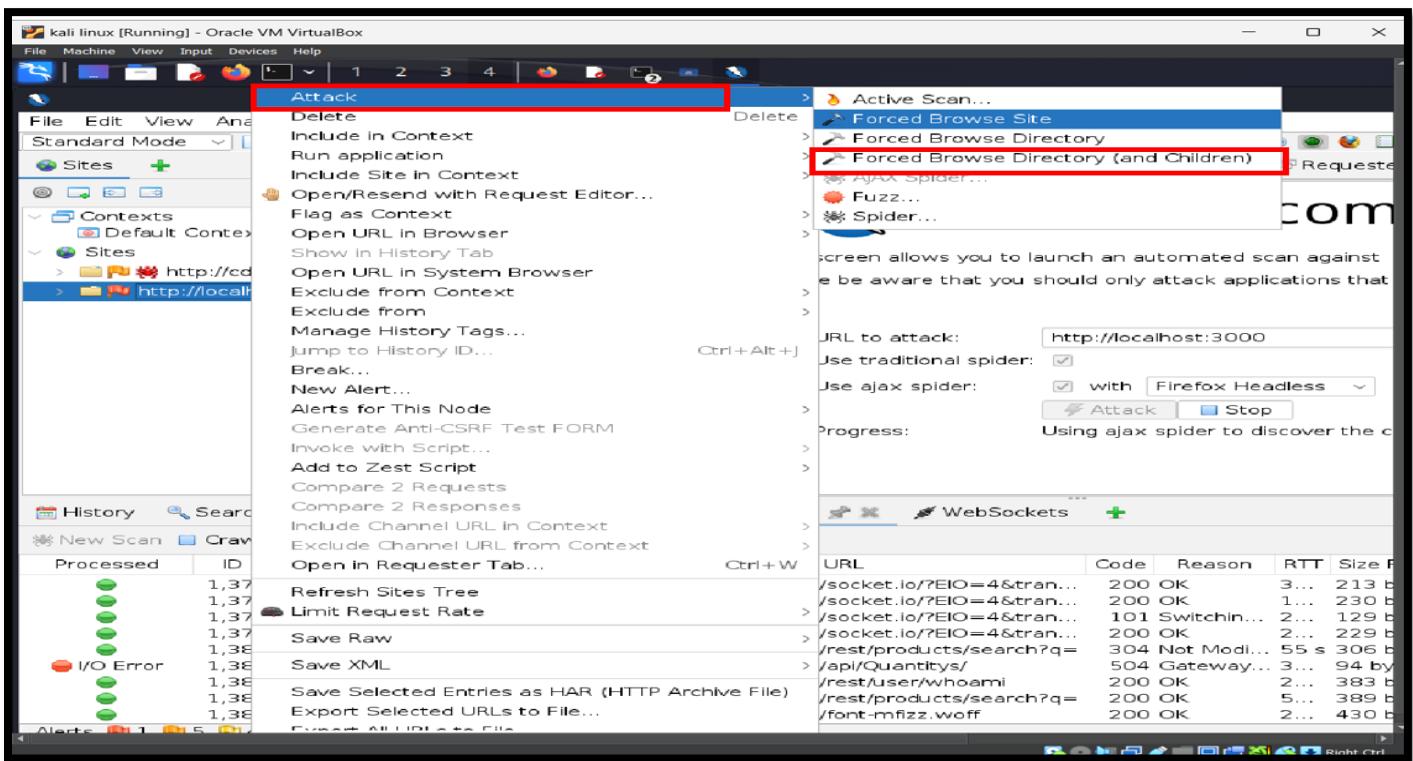
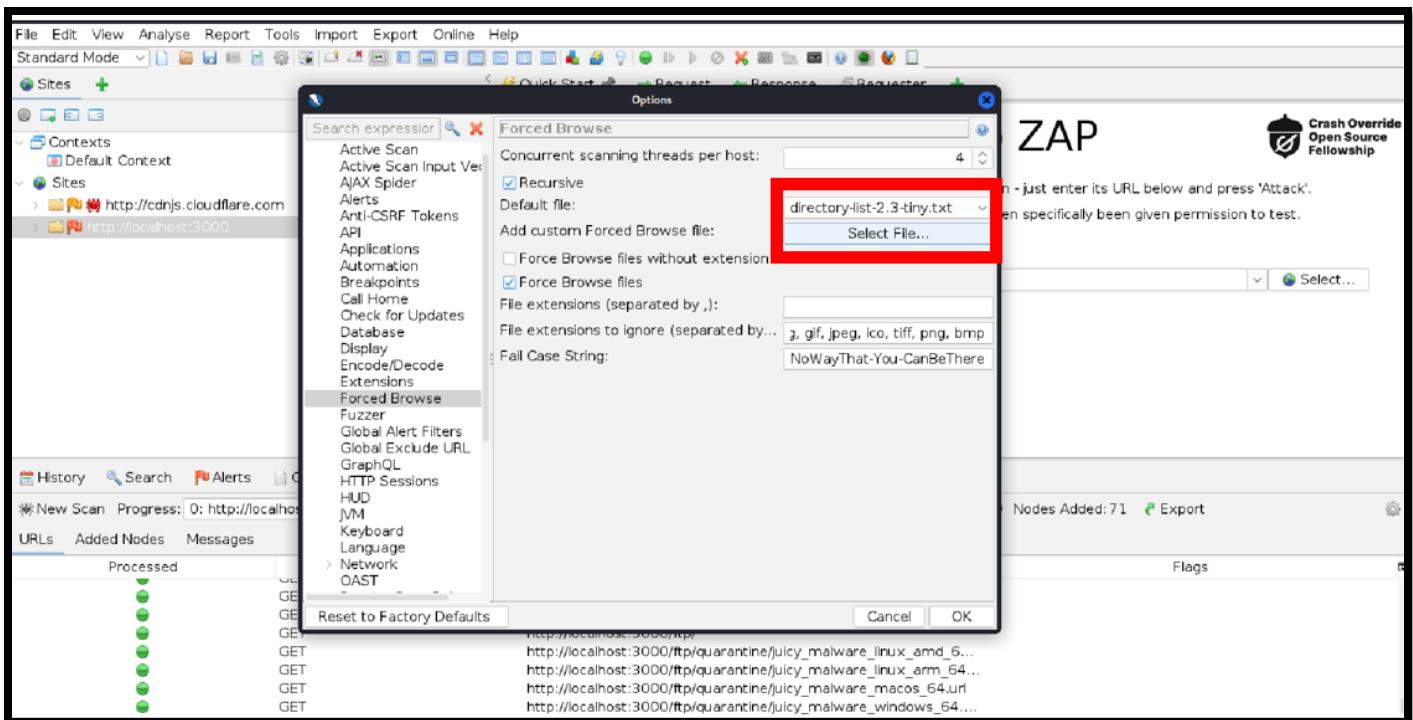
Here, in the URL, we put **localhost:3000** as our target. After that, we selected use traditional spider and used ajax spider and clicked on the attack button.



The attack started, and after successfully attacking localhost, we got directory trees on the left side under sites.



We went to the option menu, which resided under the tools menu, and clicked on forced browse configuration, where we found the option to add a custom wordlist. We downloaded the directory-list-2.3-tiny file from econestoga and added it to this custom wordlist under forced browse configuration.



After that, we clicked on forced browser and directory under the attack.

The screenshot shows the ZAP interface. In the top right, it says "Welcome to ZAP". Below that, there's a note: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'. Please be aware that you should only attack applications that you have been specifically given permission to test." A URL input field contains "http://localhost:3000". Underneath, there are checkboxes for "Use traditional spider:" (checked) and "Use ajax spider:" (checked, with "Firefox Headless" selected). A progress bar indicates "Using ajax spider to discover the content". At the bottom, a table lists network traffic with columns: Req. Timestamp, Resp. Timestamp, Method, URL, Code, Reason, Size, Resp. Header, and Size Resp. Body. One row for "http://localhost:3000/encryptionkeys/" is highlighted with a red box. The status bar at the bottom shows "Alerts 0 5 4 3 Main Proxy: localhost:8080" and "Current Scans 0 0 0 0 0 0 0 0 0 1 1 0".

Then, we found <http://localhost:3000/encryptionkeys> as a browsable directory.

The screenshot shows a browser window on a Kali Linux desktop. The address bar shows "localhost:3000/encryptionkeys". The page content displays a directory listing for "/encryptionkeys" with the following table:

Name	Size	Modified
jwt.pub	248	AM 8:12:10 12/19/2023
premium.key	50	AM 8:12:10 12/19/2023

Next, we opened the browser for the encryption keys on localhost and downloaded the premium.key file.

On the desktop, we created a .txt file named sample.txt. Here, first, we checked its current directory using the pwd command to check our current directory. After that, we created a file using the echo command in the terminal and copied the content from the premium.key to the sample.txt file.

```
shivanivaru@shivanivaru: ~
File Actions Edit View Help Difficulty Status
(shivanivaru@shivanivaru)-[~]
$ pwd
/home/shivanivaru
(shivanivaru@shivanivaru)-[~]ographic Issues XXE
$ echo "IvLuRFBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07
CeCeqYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUmGLTTAC2+G9UuFC
D1DUjg==" > sample.txt
ease let us know! Reach out via our community channels
```

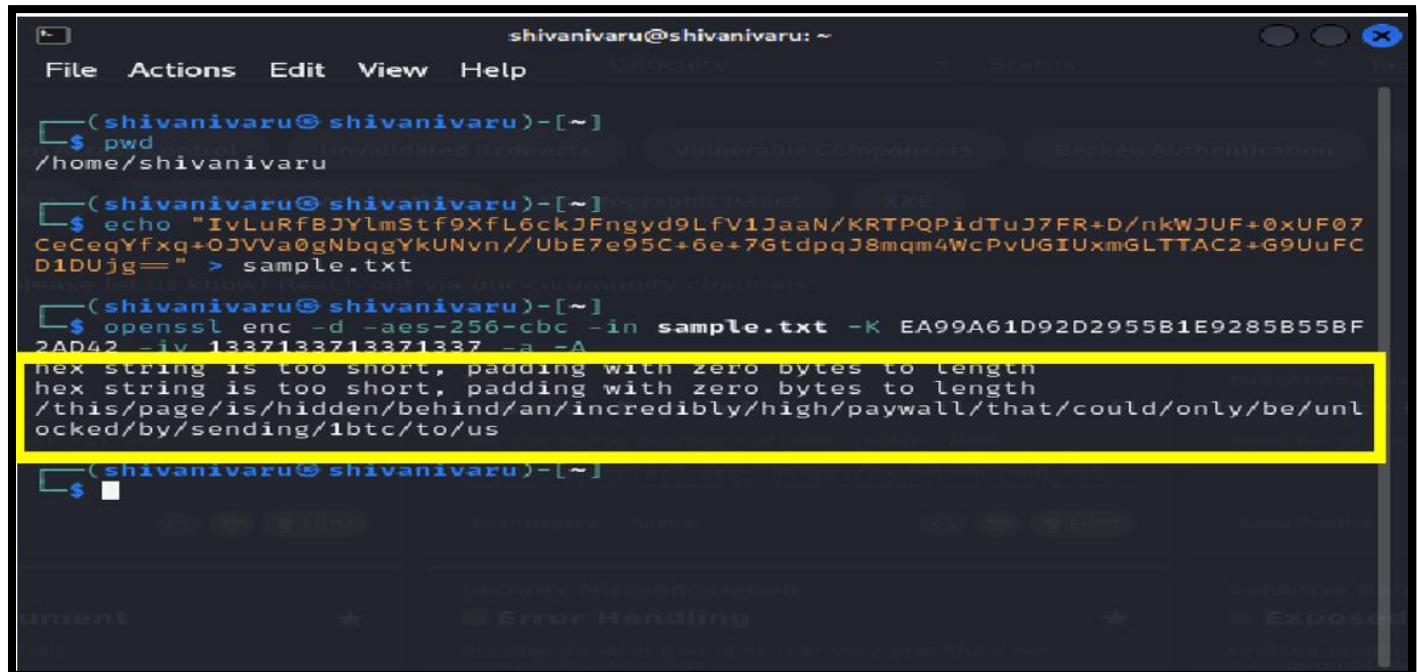
The terminal window shows the user's session on the host shivanivaru. The user runs 'pwd' to check the current directory, which is /home/shivanivaru. Then, they run an 'echo' command to create a file named 'sample.txt'. The content of the file is a base64 encoded string: IvLuRFBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07 CeCeqYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUmGLTTAC2+G9UuFC D1DUjg==. A yellow box highlights this command and its output.

Then, we opened the terminal and used the command openssl with the necessary command to decipher the text as shown in the screenshot.

```
shivanivaru@shivanivaru: ~
File Actions Edit View Help Difficulty Status
(shivanivaru@shivanivaru)-[~]
$ pwd
/home/shivanivaru
(shivanivaru@shivanivaru)-[~]ographic Issues XXE
$ echo "IvLuRFBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07
CeCeqYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUmGLTTAC2+G9UuFC
D1DUjg==" > sample.txt
(shivanivaru@shivanivaru)-[~]
$ openssl enc -d -aes-256-cbc -in sample.txt -K EA99A61D92D2955B1E9285B55BF
2AD42 -iv 1337133713371337 -a -A
xss Miscellaneous
```

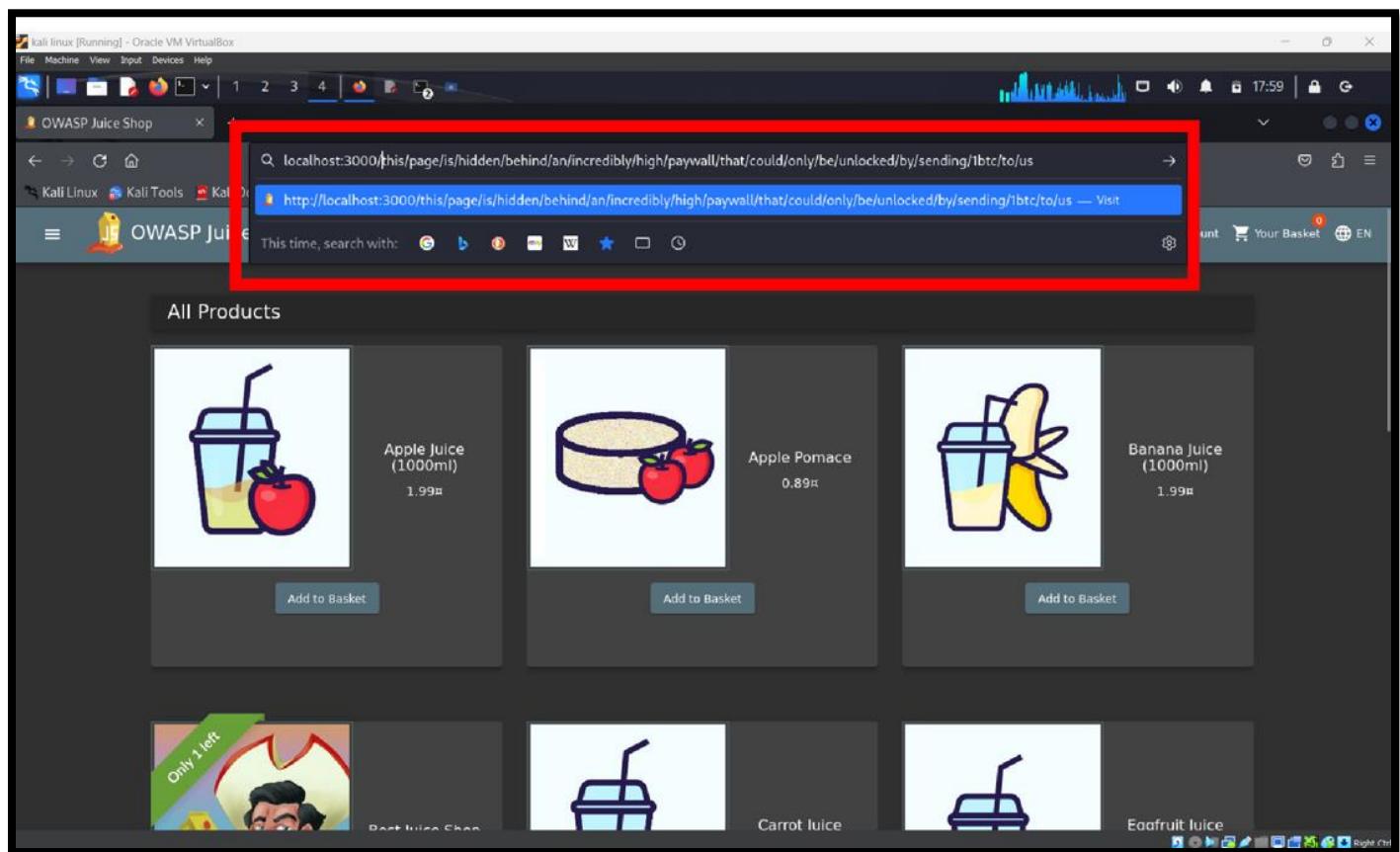
The terminal window shows the user's session on the host shivanivaru. The user runs 'pwd' to check the current directory, which is /home/shivanivaru. Then, they run an 'echo' command to create a file named 'sample.txt'. The content of the file is a base64 encoded string: IvLuRFBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07 CeCeqYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUmGLTTAC2+G9UuFC D1DUjg==. In the next step, the user runs an 'openssl' command to decrypt the file. The command is: openssl enc -d -aes-256-cbc -in sample.txt -K EA99A61D92D2955B1E9285B55BF2AD42 -iv 1337133713371337 -a -A. A yellow box highlights this command and its output.

We got the decrypted path where we copied that path.

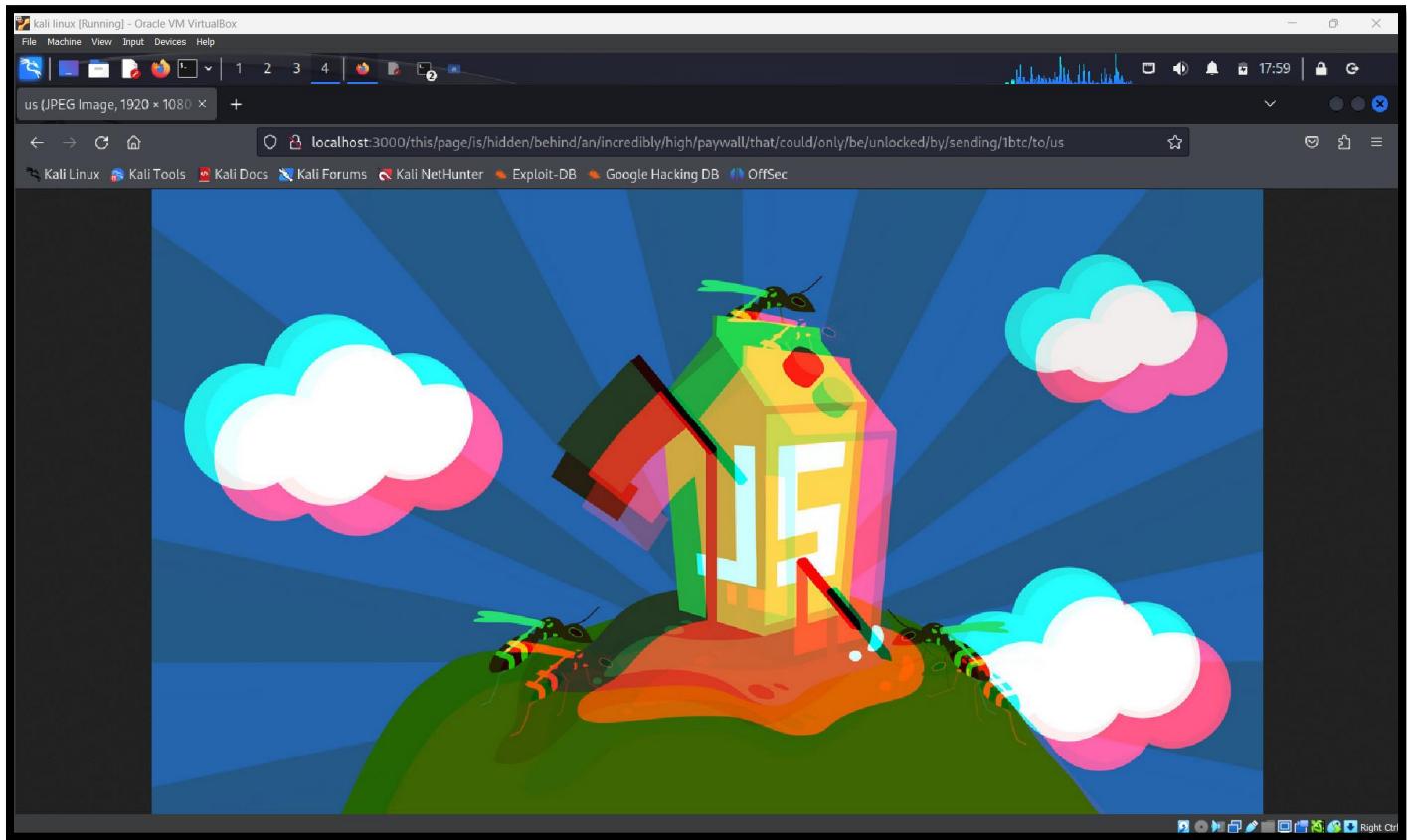


```
shivanivar@shivanivar: ~
File Actions Edit View Help
(shivanivar@shivanivar)-[~]
$ pwd
/home/shivanivar
(shivanivar@shivanivar)-[~]
$ echo "IvLuRFBJYlmStf9XF6ckJFngyd9LFV1JaaN/K RTPQPidTuJ7FR+D/nkWJUF+0xUF07
CeCeqYfxq+OJVVa0gNbqgYkUnvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUmGLTTAC2+G9UuFC
D1DUjg==" > sample.txt
(shivanivar@shivanivar)-[~]
$ openssl enc -d -aes-256-cbc -in sample.txt -K EA99A61D92D2955B1E9285B55BF
2AD42 -iv 1337133713371337 -a -A
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
/t his/page/is/hidden/behind/an/incredibly/high/paywall/that/could/only/be/unlocked/by/sending/1btc/to/us
(shivanivar@shivanivar)-[~]
```

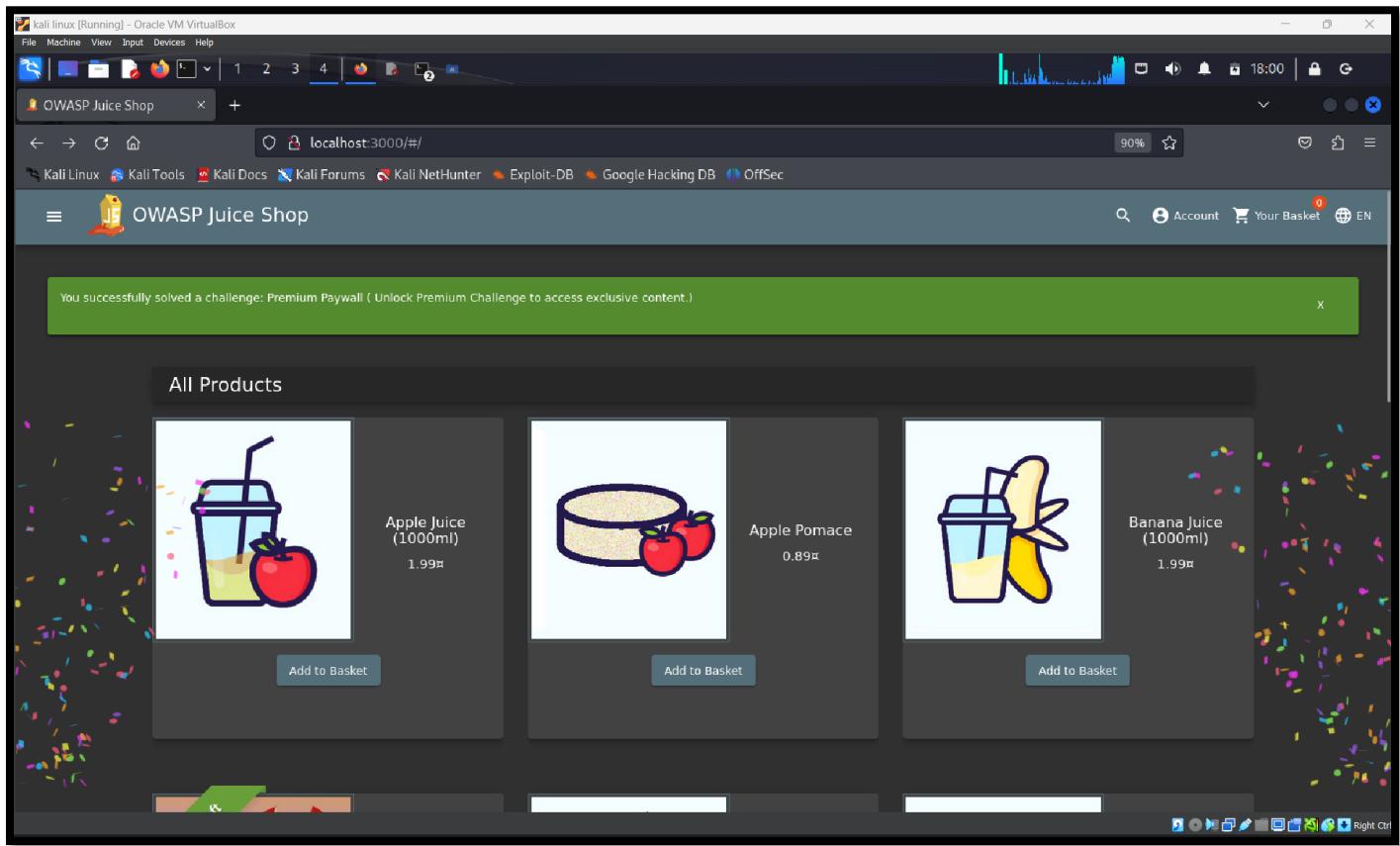
and then we put it on the browser with localhost:3000, and it opened the exclusive content.



We saw some "Marvel at the premium VR wallpaper."



Finally, we solved the challenge.



### **Vulnerabilities' Exploited by Task 4:**

- Inadequate encryption key protection leads to unauthorized access.
- The comment section of an HTML page contains a publicly encrypted premium key, which exposes sensitive data and poses a significant threat.
- Malicious attackers exploit this vulnerability to decrypt sensitive data.
- Forced Browsing Attacks exploit vulnerable local host browsers, allowing open-SSL connections and premium access.key files, enabling decoding of encrypted text like AES encryption.

### **How Vulnerabilities Can Be Fixed or Addressed:**

- Sensitive information should not be exposed, even if encrypted.
- Encryption keys should not be accessible to open SSL connections, as this would compromise security.
- Use industry-standard encryption techniques and secure storage solutions.
- Implement strong access controls, including RBAC and least privilege principles.
- Regularly rotate encryption keys to reduce breach risks.
- Use Multi-Factor Authentication for accessing encryption keys and sensitive cryptographic activities.

### **Real-World Scenarios:**

- Encryption key vulnerabilities lead to unauthorized access to sensitive data.
- Attackers can exploit compromised encryption keys to gain confidential information.
- Consequences include data breaches, financial losses, and damage to reputation.
- Compromised keys enable decryption of encrypted messages/files, compromising privacy.

## **References**

1. URL Decoder/Encoder. (n.d.). <https://meyerweb.com/eric/tools/dencoder/>
2. <https://conestoga.desire2learn.com/d2l/le/content/1002737/viewContent/21555039/View>
3. A01 Broken Access Control - OWASP Top 10:2021. (n.d.).  
[https://owasp.org/Top10/A01\\_2021-Broken\\_Access\\_Control/](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)
4. OWASP Top Ten 2017 | A3:2017-Sensitive Data Exposure | OWASP Foundation. (n.d.). [https://owasp.org/www-project-top-ten/2017/A3\\_2017-Sensitive\\_Data\\_Exposure](https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure)

## **Video Recording Link:**