

Ethical Hacking and Penetration Testing

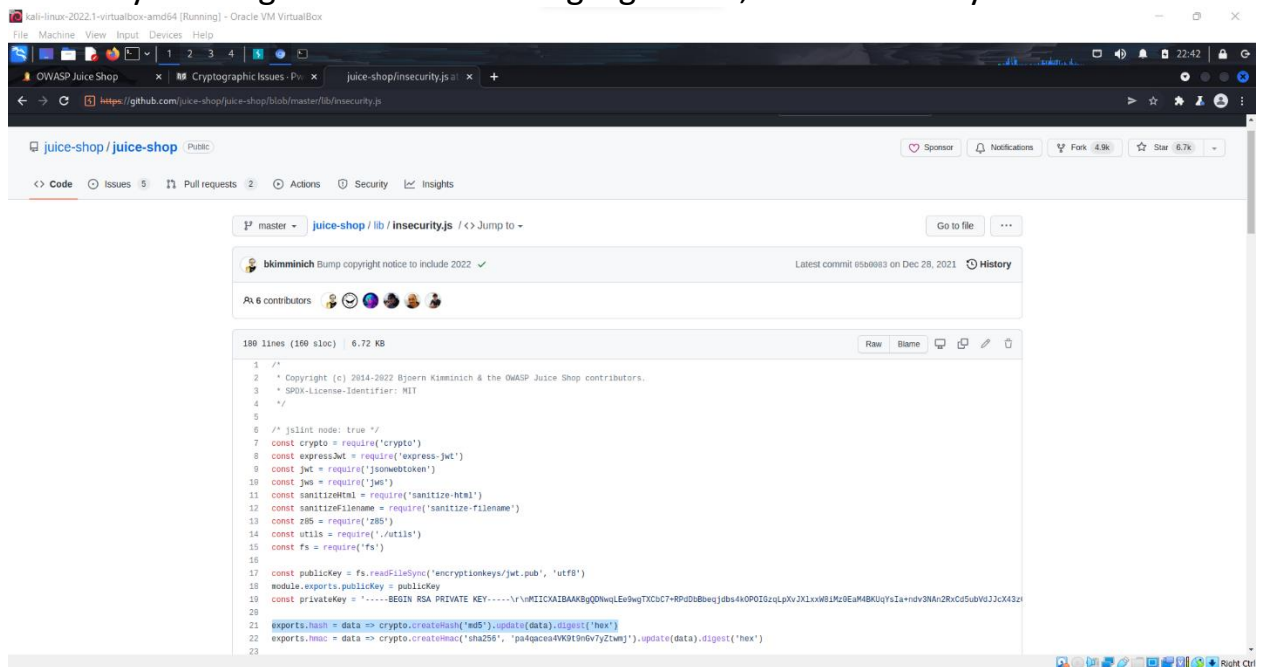
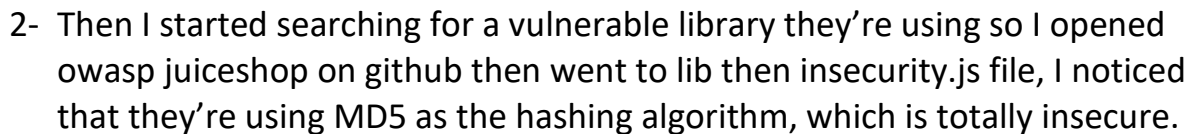
Assignment 5

Web Exploitation (Juice-Shop)

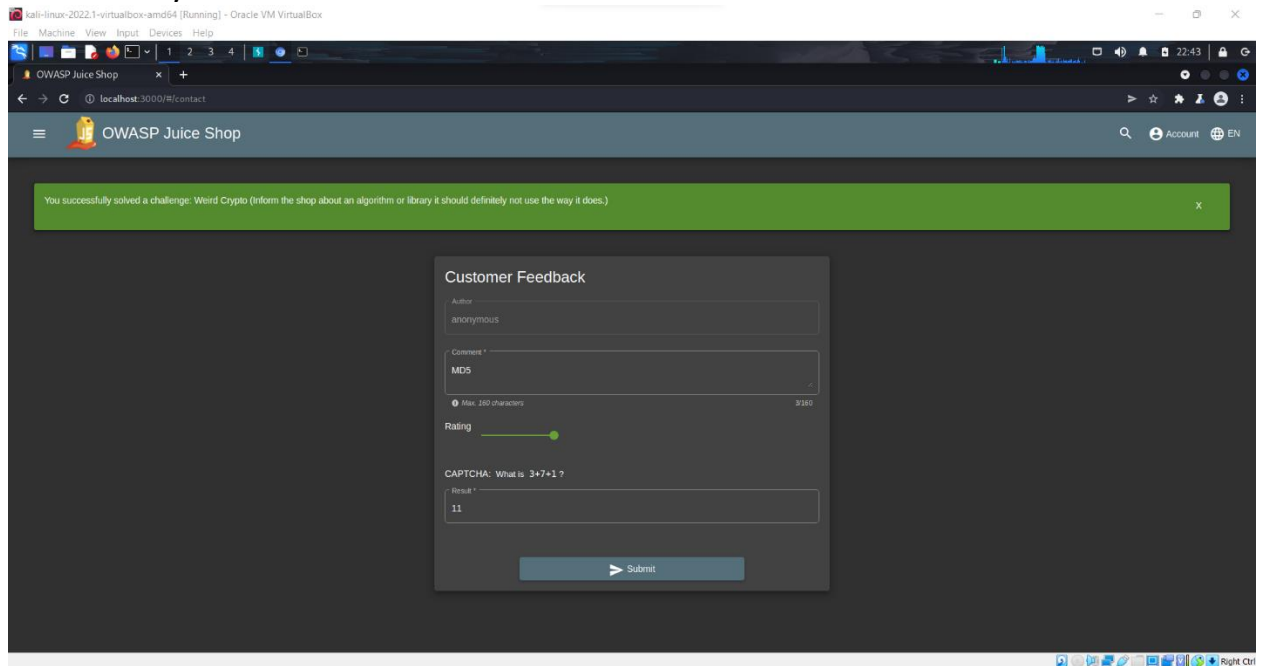
Name: John Ehab

ID: 100-2096

- 1- First, I read about the challenge and noticed that I have to submit the name of the library as feedback on the website.

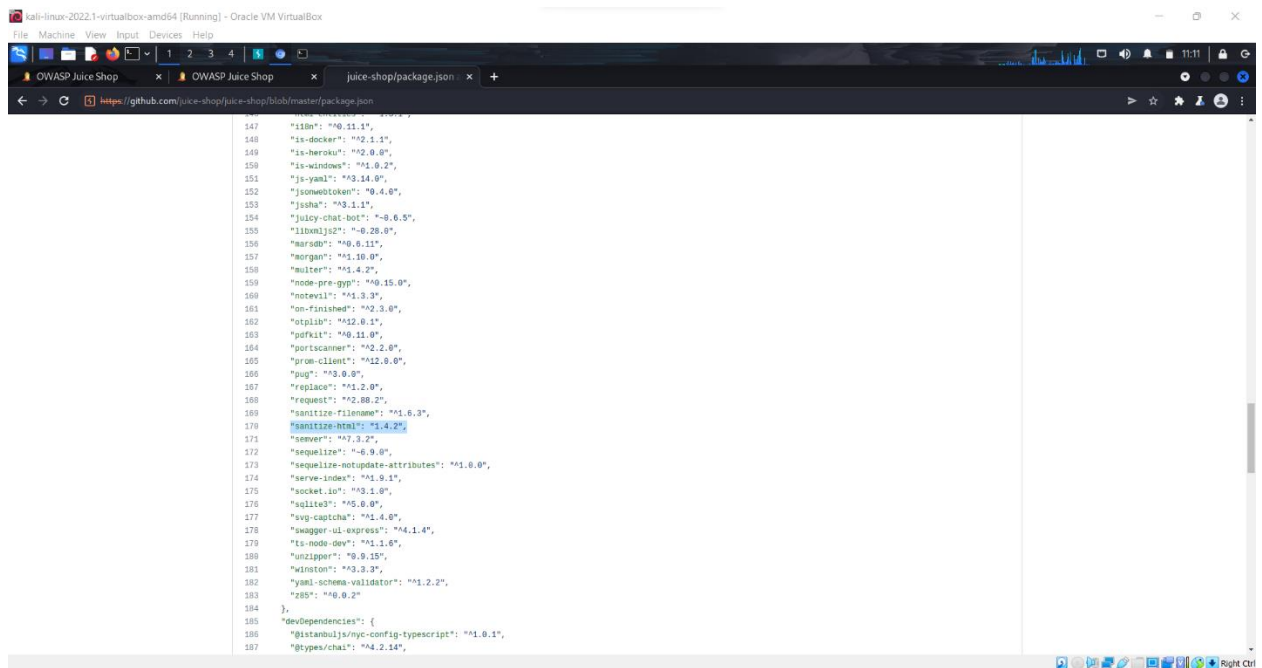


3- I went for the website to submit my feedback completing the challenge successfully.



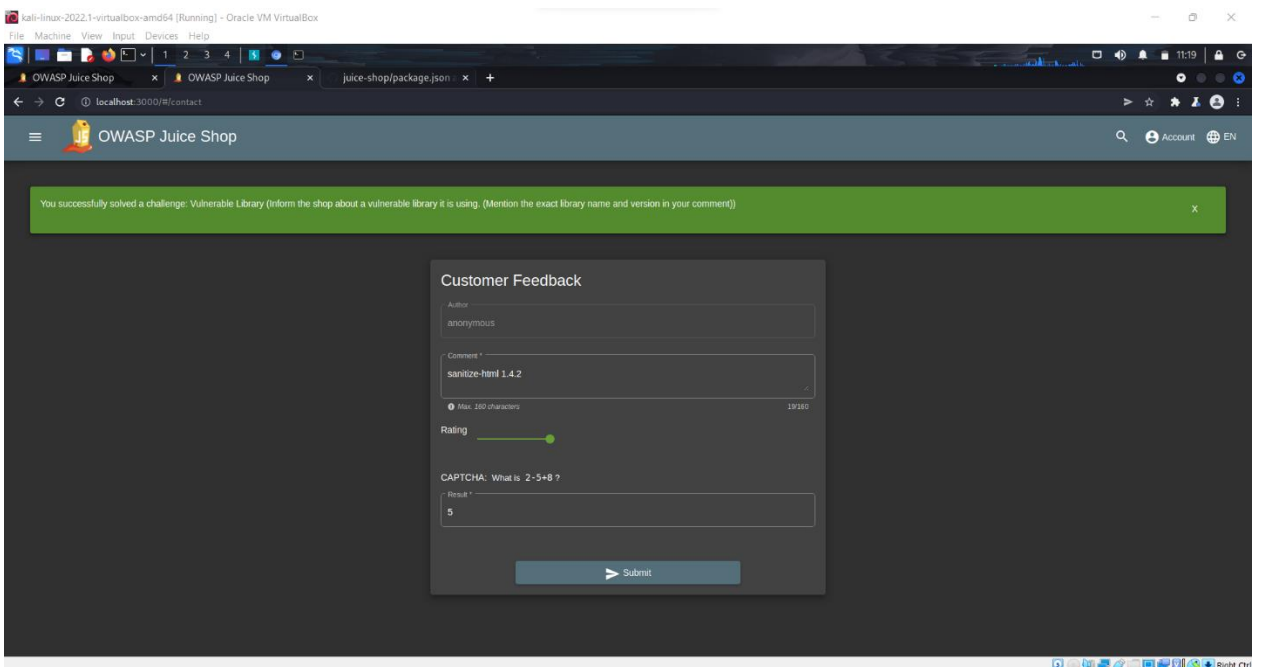
Attack number 2

- 1- Again it's shown in the challenge description that I have to submit the name of the library and its version as feedback on the website. Then also I went for owasp juiceshop on github then went to package.json file to have a look on the dependencies, after doing my research I discovered that they're using sanitize html which is vulnerable to Cross-site Scripting (XSS), Arbitrary Code Execution, Validation Bypass, and Access Restriction Bypass attacks.



```
147 "isbot": "0.11.1",
148 "is-docker": "2.1.1",
149 "is-heroku": "2.0.0",
150 "is-windows": "1.0.2",
151 "js-yaml": "4.1.0",
152 "jsonwebtoken": "9.4.0",
153 "jshash": "1.1.1",
154 "juicy-chat-bot": "0.6.5",
155 "libxmljs2": "0.28.0",
156 "maradb": "0.6.11",
157 "morgan": "1.10.0",
158 "multer": "1.4.2",
159 "node-pre-gyp": "10.15.0",
160 "notewise": "1.3.3",
161 "on-finished": "2.3.0",
162 "otplib": "12.0.1",
163 "pdfkit": "0.11.0",
164 "portscanner": "2.2.0",
165 "prom-client": "12.0.0",
166 "pug": "3.0.0",
167 "replace": "1.2.0",
168 "request": "2.88.2",
169 "sanitize-filename": "1.6.3",
170 "sanitize-html": "1.4.2",
171 "sewer": "7.3.2",
172 "sequelize": "6.9.0",
173 "sequelize-notupdate-attributes": "1.0.0",
174 "serve-index": "1.9.1",
175 "socket.io": "3.1.0",
176 "sweetalert": "5.0.0",
177 "svg-captcha": "1.4.0",
178 "swagger-ui-express": "4.1.4",
179 "ts-node-dev": "1.1.0",
180 "unzipper": "0.9.10",
181 "vinston": "3.3.3",
182 "yaml-schema-validator": "1.2.2",
183 "z85": "0.0.2",
184 },
185 "devDependencies": {
186   "@istanbuljs/nyc-config-typescript": "1.0.1",
187   "@types/": "4.2.14",
```

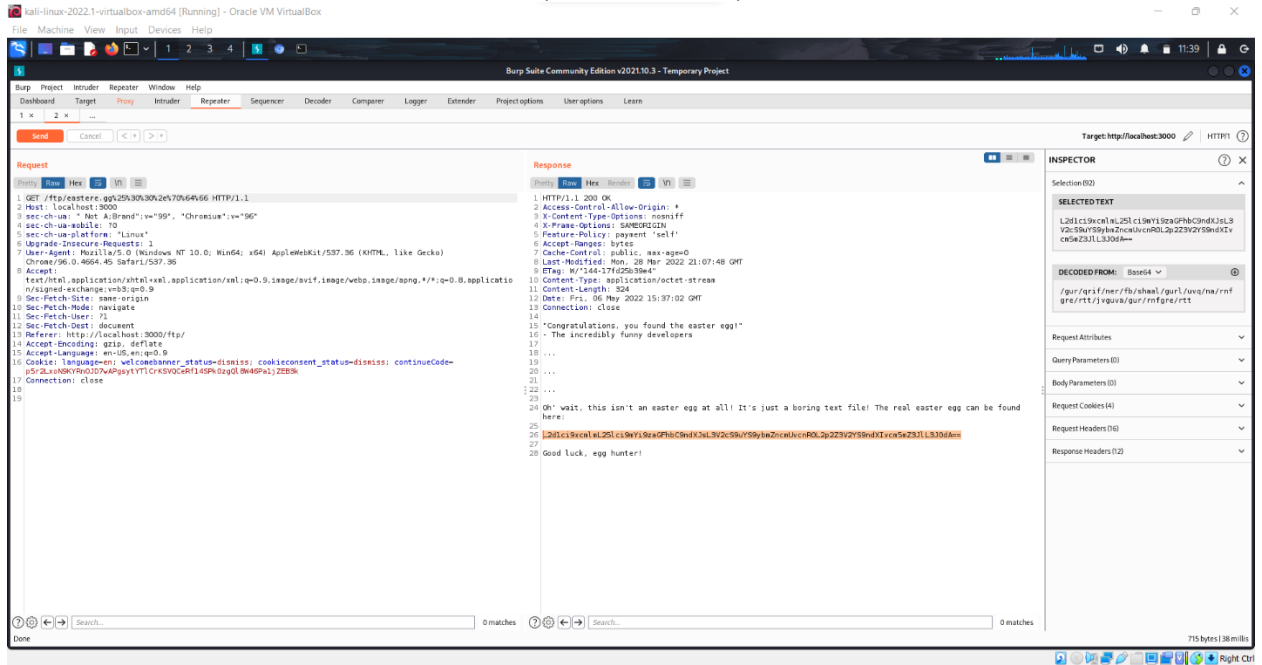
- 2- Then I went back for the website to submit the feedback with the library name and version completing the challenge successfully.



Attack number 3

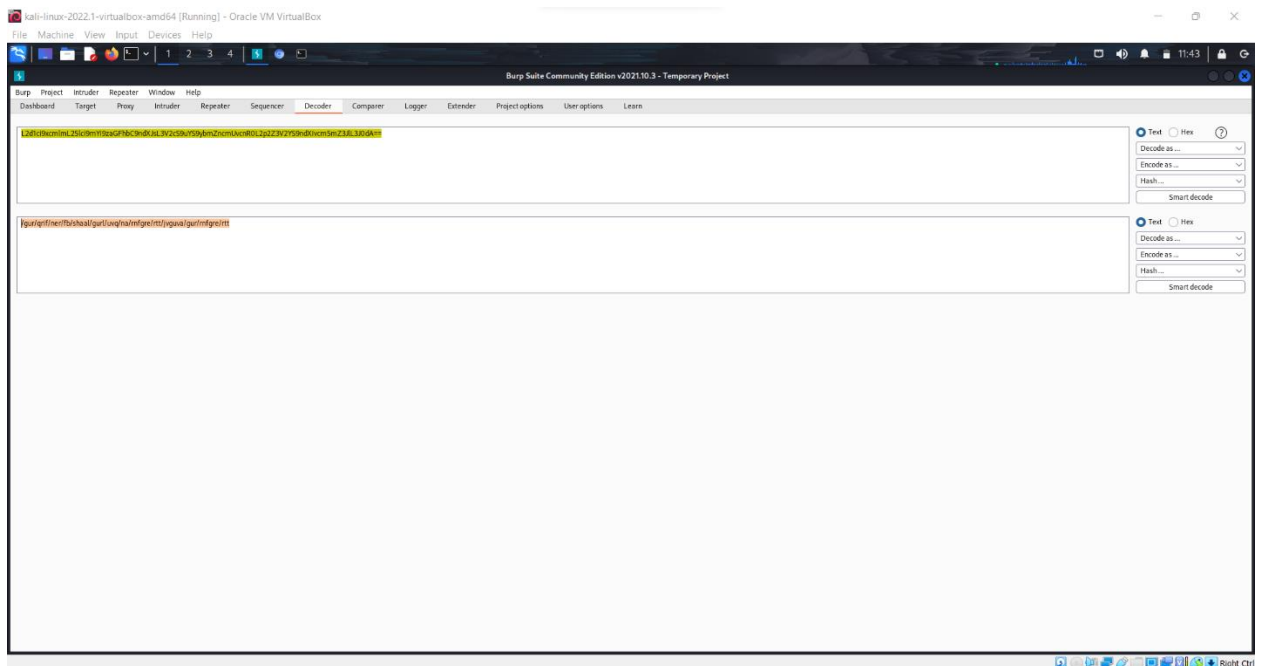
- 1- Following attack number 6 in Assignment 4, when I downloaded the eastere.gg file it said that this isn't the easter egg, and it gave me a hint that easter egg can be found here:

L2d1ci9xcmlmL251ci9mYi9zaGFhbc9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXlvcn5mZ3JlL3J0dA==



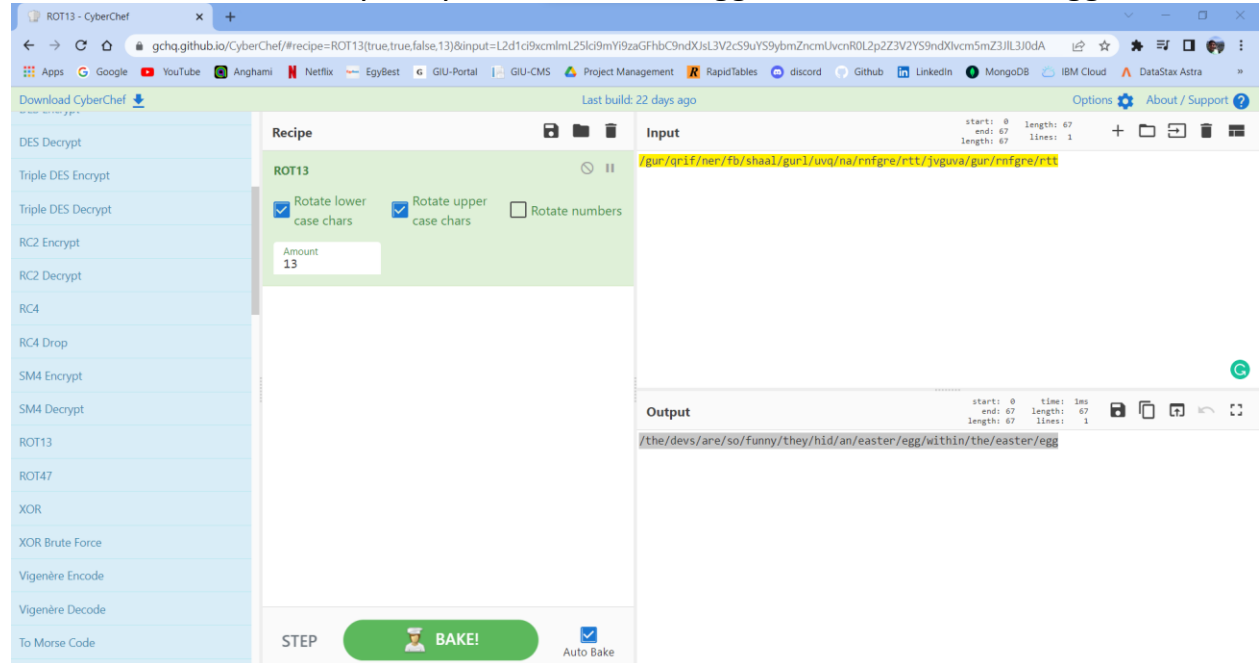
- 2- I'll do my cryptanalysis on that string. First that == at the end gave me a hint that it might be encoded as base64, so I'll use burpsuite decoder to decode it, I'll get:

/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt

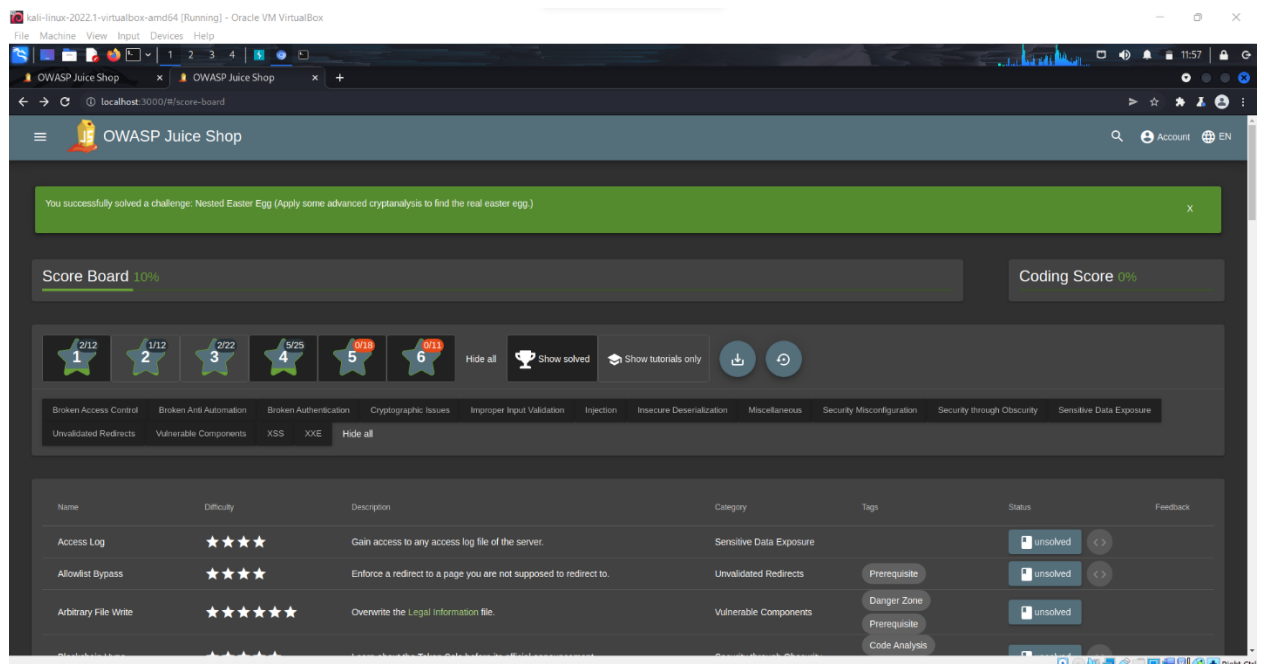


- 3- It seems that the text is an extension but it's still not meaningful, it seems that it's encrypted, so I'll try to do another part of analysis on any online encryption/decryption tool by trying different decryption algorithms till I get meaningful words for that extension. I'll use CyberChef, and I kept trying till I found out that ROT13 is the right decryption algorithm to be used to give meaningful extension. It will give:


/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg



- 4- I'll add this extension on juiceshop to access the real easter egg and complete the challenge successfully.



- 1- After opening juiceshop website then going to the developers tools then opening the main.js and pressing on pretty print to format it, I searched for "redirect" keyword and started navigating the code till I found a url extension that juiceshop allows to redirect to, which is:



The screenshot shows a Kali Linux terminal window titled "kali-linux-2022.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal has a dark theme. At the top, there's a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". Below the menu is a toolbar with icons for file operations and system settings. The main area of the terminal shows a web browser window with a dark theme. The browser's address bar displays "localhost:3000/redirect?to=https://google.com". The browser's content area shows a red error message: "Error: Unrecognized target". The browser's tab bar shows two tabs: "Error: Unrecognized target" and "OWASP Juice Shop". The browser's status bar at the bottom shows various icons for navigation and settings.

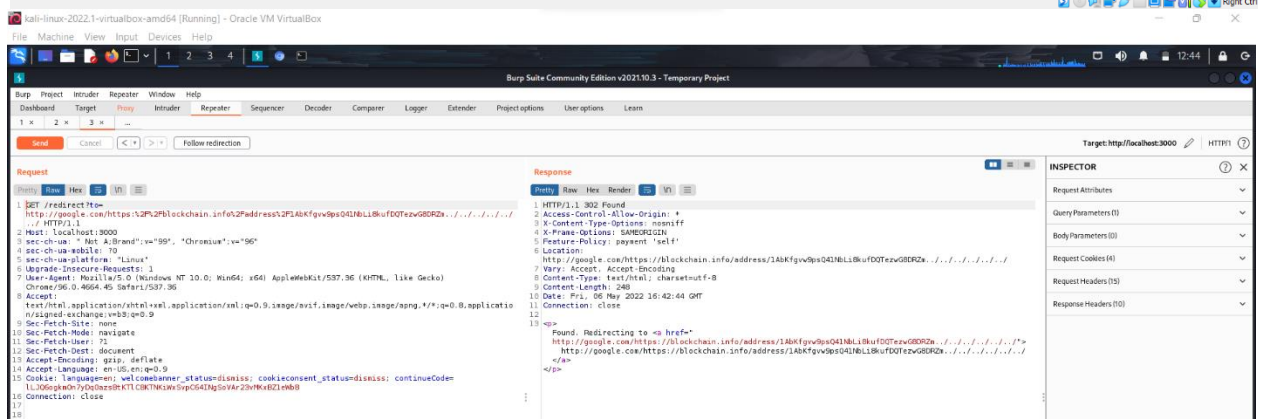
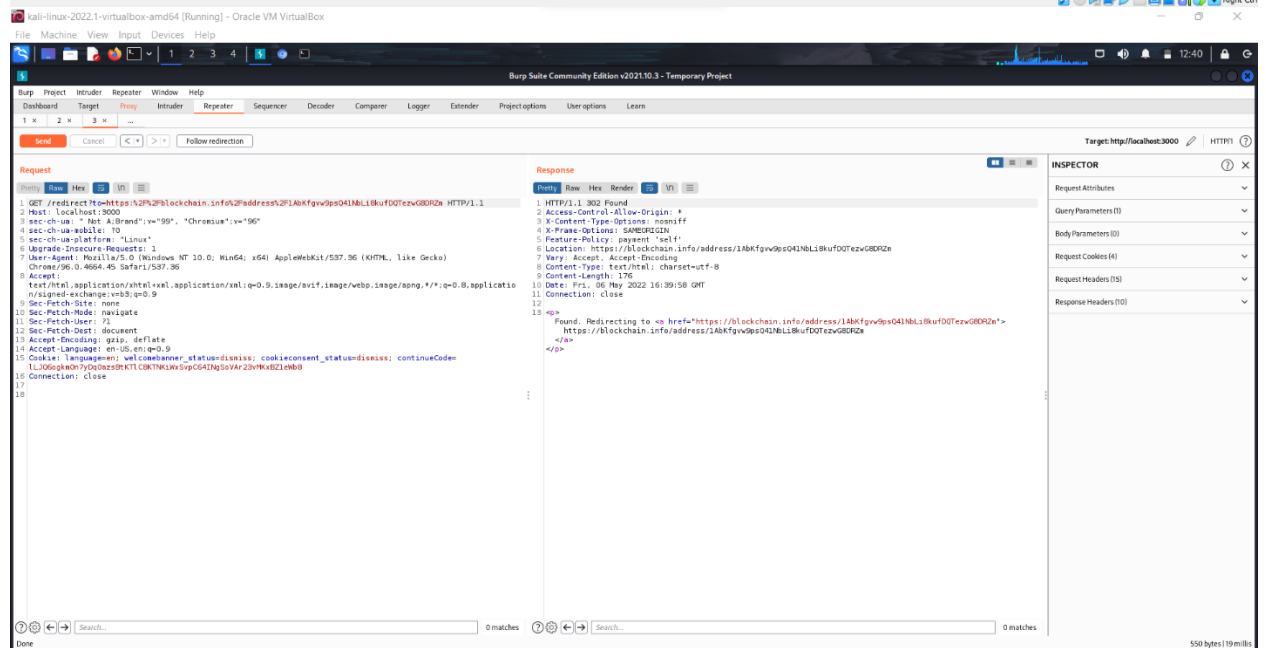
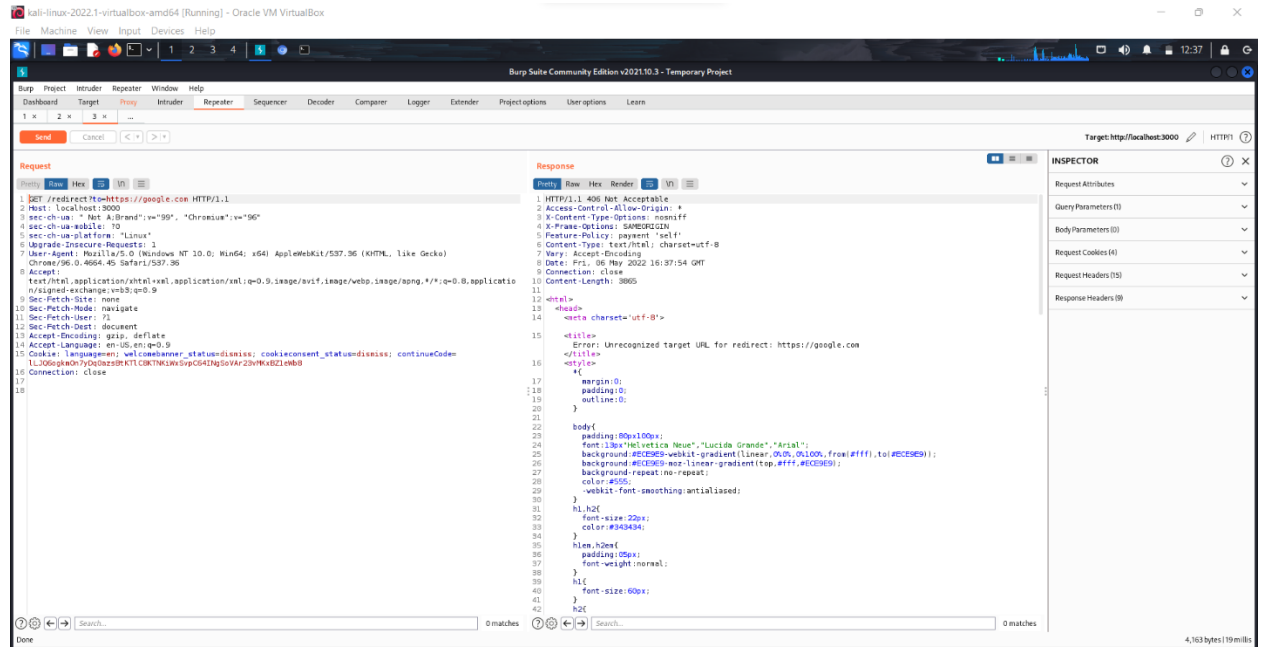
406 Error: Unrecognized target URL for redirect: https://google.com

```

at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:20:18
at Layer.handle [as handle_request] (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:5)
at process.nextTick (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:137:10)
at Route.dispatch (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:112:3)
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:95:5)
at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:281:22
at Function.process_params (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:341:12)
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:275:10)
at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:133:5
at Layer.handle [as handle_request] (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:5)
at process.nextTick (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:137:10)
at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:284:7
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:95:5)
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:275:10)
at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:89:5
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:95:5)
at trim_prefix (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:323:13)
at home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:284:7
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:341:12)
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:275:10)
at tologer (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:144:5)
at next (home.kaleido.DownloadsGithub.conjonejs-shupbldmtools-hredist.js:95:5)

```

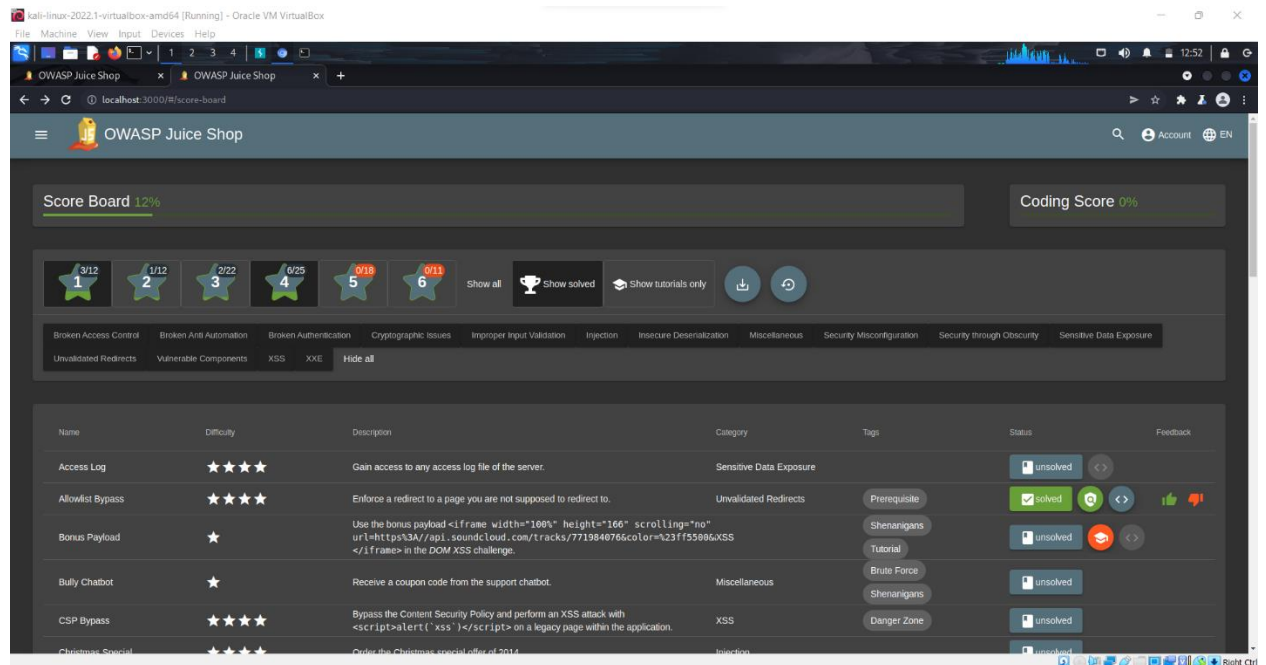
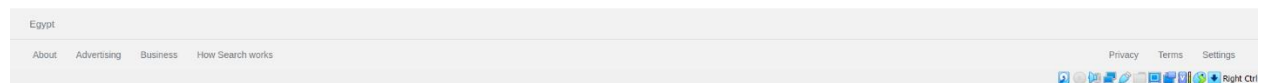
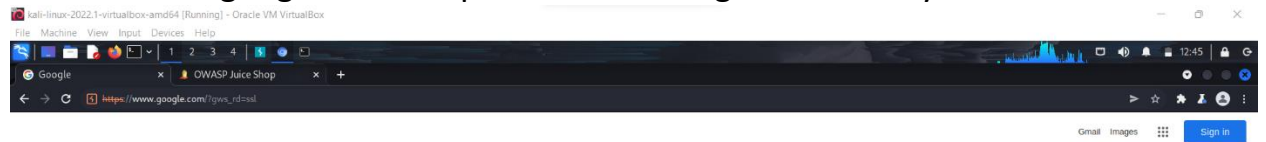

3- I'll intercept this get req with burpsuite and send it to the repeater, I'll fool the filtering by adding the url of google followed by the url of the blockchain to bypass the allowlist followed by some ../ to get back to google.



4- Then I'll write

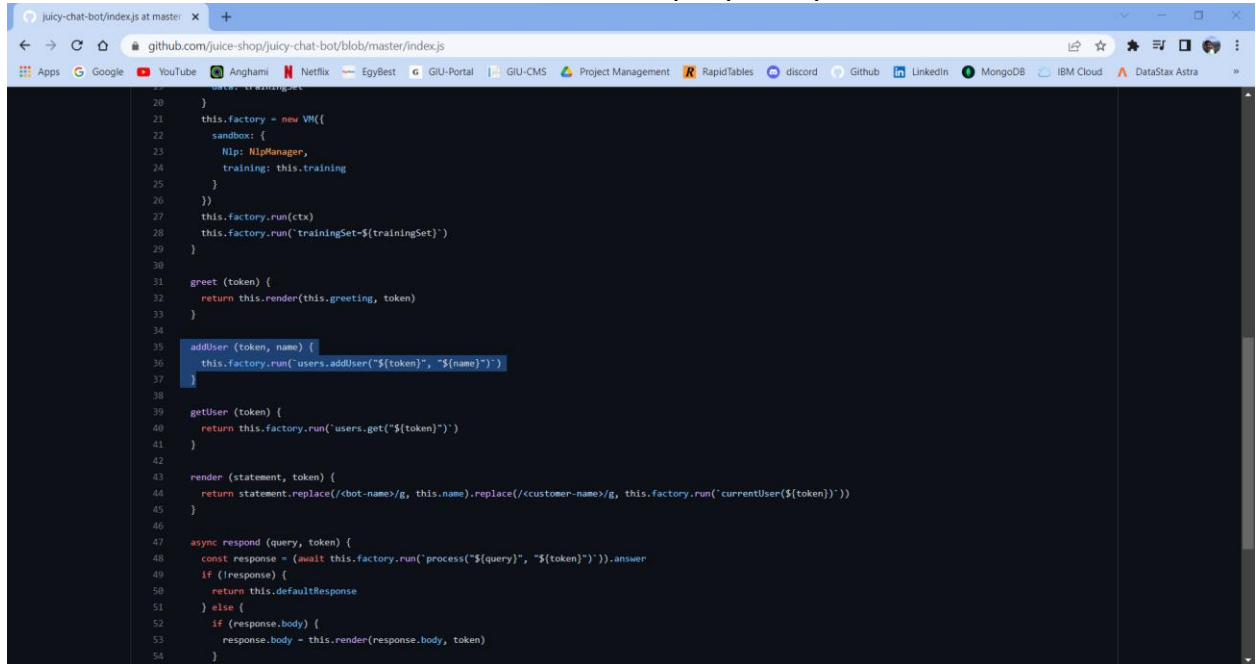
<http://localhost:3000/redirect?to=http://google.com/https:%2F%2Fblockchain.info%2Faddress%2F1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm.../.../.../.../>

to redirect to google and complete the challenge successfully.



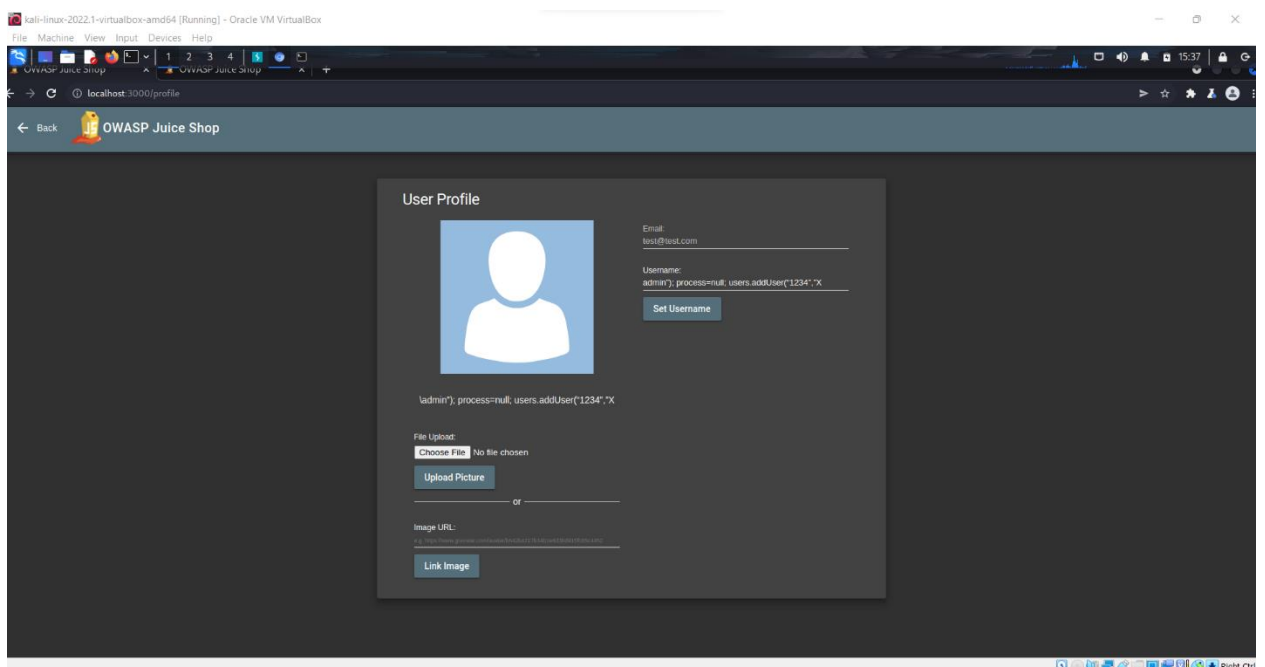
Attack number 5

- 1- By navigating through the juiceshop src code on github then going to juicy-chat-bot then index.js file, I found that there's a vulnerable method which is addUser that adds the username without proper input validation.

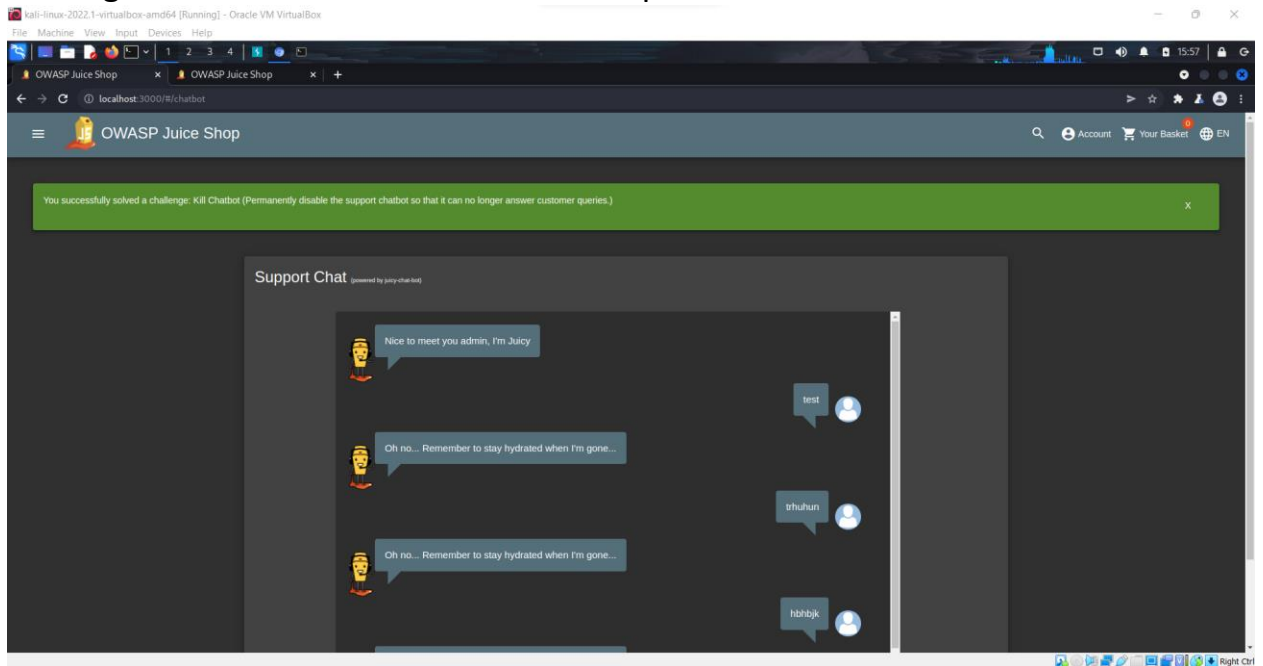


```
20 }
21 this.factory = new VM({
22   sandbox: {
23     Nlp: NlpManager,
24     training: this.training
25   }
26 })
27 this.factory.run(ctx)
28 this.factory.run('trainingSet-${trainingSet}')
29 }
30
31 greet(token) {
32   return this.render(this.greeting, token)
33 }
34
35 addUser(token, name) {
36   this.factory.run('users.addUser(`${token}`, `${name}`)')
37 }
38
39 getUser(token) {
40   return this.factory.run('users.get(`${token}`)')
41 }
42
43 render(statement, token) {
44   return statement.replace(/<bot-name>/g, this.name).replace(/<customer-name>/g, this.factory.run('currentUser(`${token}`)'))
45 }
46
47 async respond(query, token) {
48   const response = (await this.factory.run('process(`${query}`, `${token}`)')).answer
49   if (!response) {
50     return this.defaultResponse
51   } else {
52     if (response.body) {
53       response.body = this.render(response.body, token)
54     }
55   }
56 }
```

- 2- So I'll exploit this vulnerability to kill the chatbot. I'll login with any registered account, then go to the user profile and enter in the username: admin"); process = null; users.addUser("1337","X to kill the process of the chatbot.



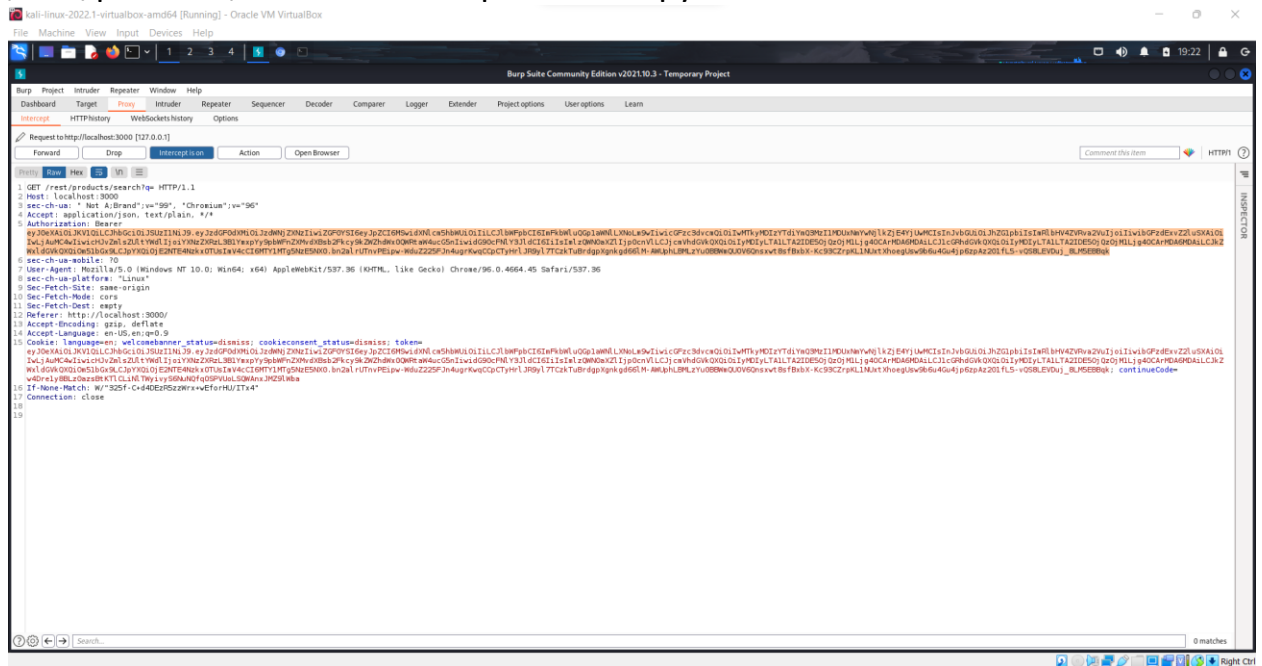
- 3- Then go to the support chat webpage on juice shop and try to type any test message to make sure that the chat bot's process is killed.



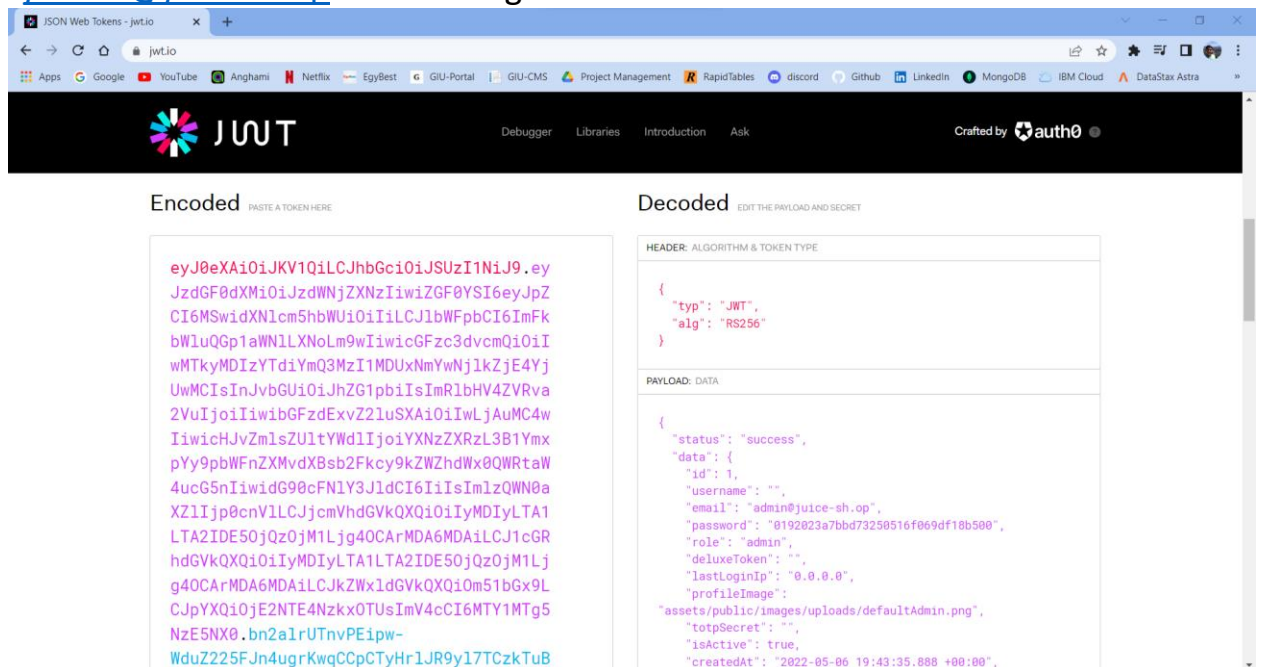
Attack number 6 is not implemented yet in the juice app itself, so It'll be skipped.

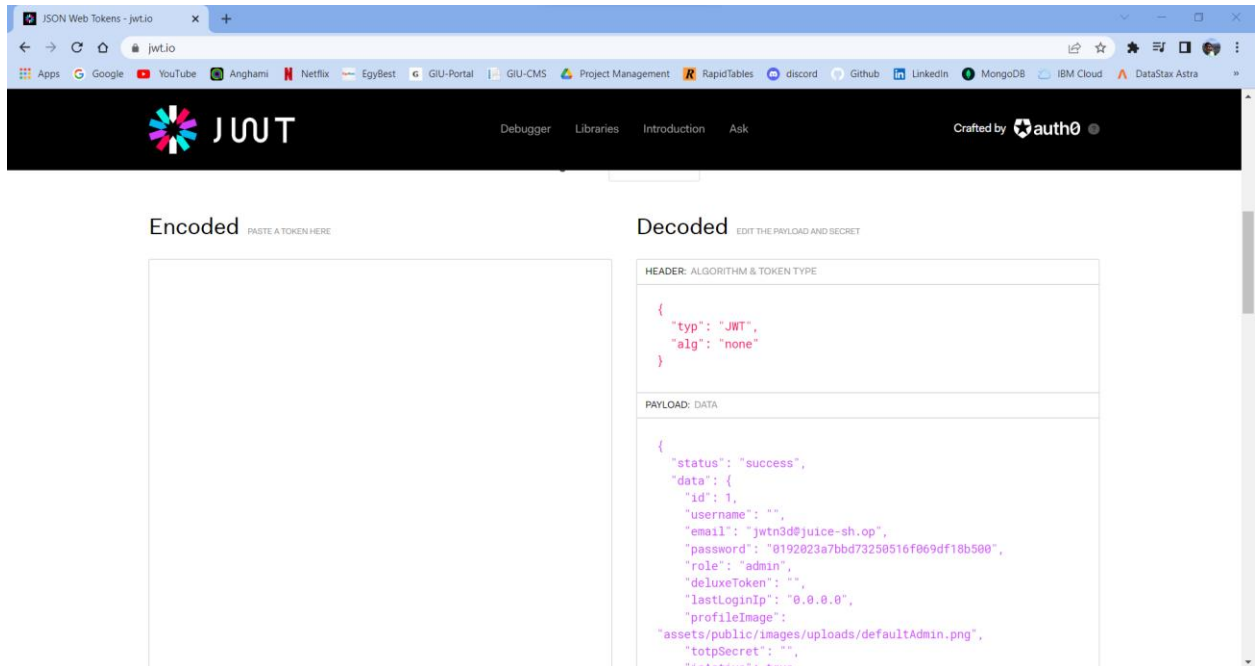
Attack number 7

- 1- First I'll login with any registered account, I'll intercept the req of the /rest/products/search with burpsuite to copy the authorization bearer.



- 2- Then I'll use jwt.io to decode and manipulate it. I'll change the email to "jwt3d@juice-sh.op" and the alg to "none".





- 3- Then I'll open any encoding online tool. In my case I'll will use base64url.com. And I'll take the header and the payload and encode them to base64URL then concatenate them to each other and copy paste them to the authorization bearer in the proxy and send it to complete the challenge successfully.

