The American University in Cairo
School of Sciences and Engineering

AUC

CyberRange Vulnerabilities

By:

Michael Reda ID: 900203291

Freddy Amgad ID:900203088

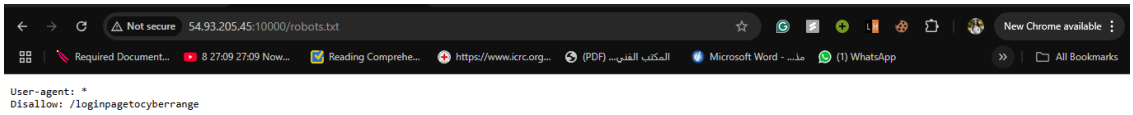
Shahd Elmahallawy ID: 900194441

CSCE493005 –Web Secuirty

Fall 2024

Authentication and authorization

1. Robots.txt



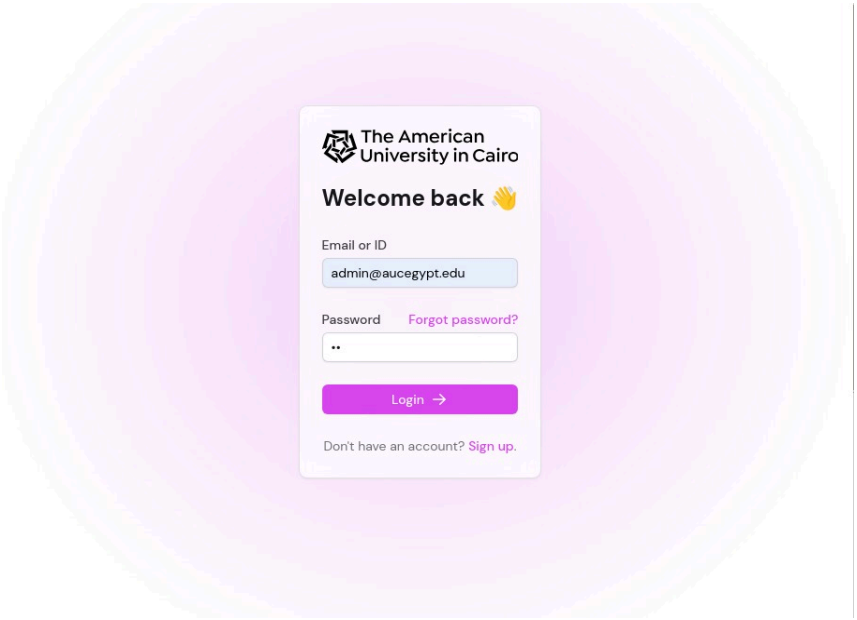
Description:

The first vulnerability is robots.txt which has the directory to login into the website

Mitigation:

Avoid placing sensitive directories or files in robots.txt that could disclose information to unauthorized users.

2. Login with any Password



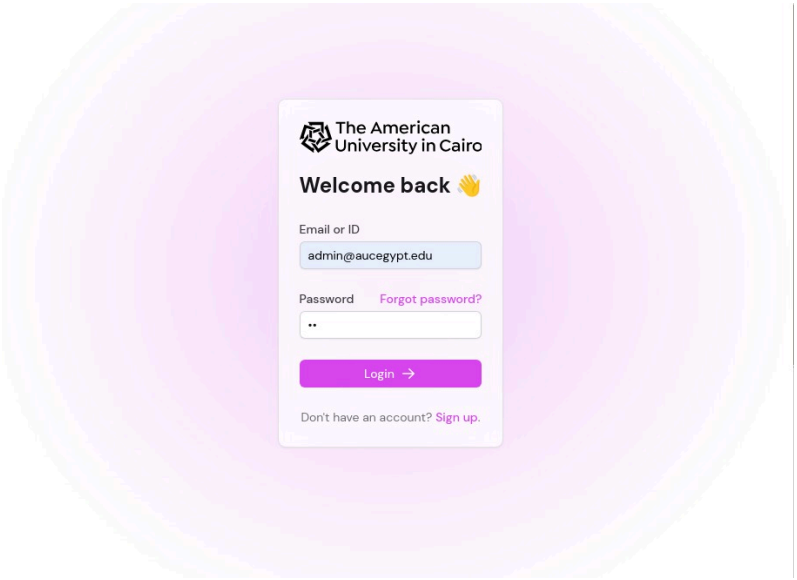
Description:

The user can enter any password no matter its length to log in. So, the website only checks the username.

Mitigation:

Add a condition to check the password in the backend by validating it in with the one stored in the database.

3. Access By Admin Email



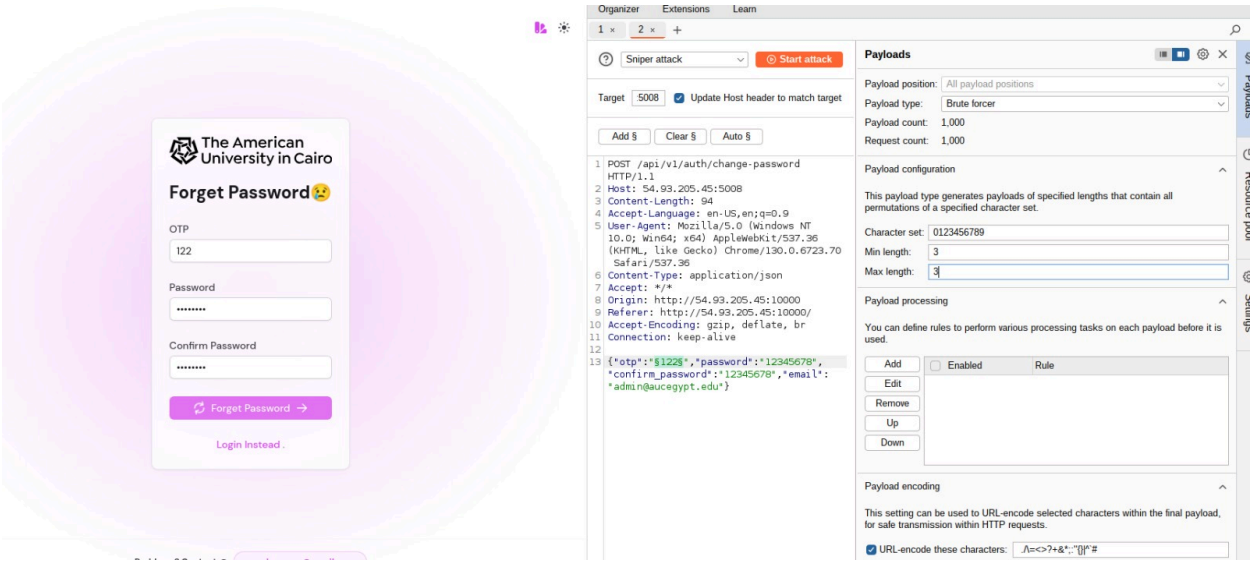
Description:

Since the website only checks the username while logging in, you can easily guess the admin username, and get access to the admin portal.

Mitigation:

Change the admin username to something less guessbale or less known.

4. BruteForce OTP



Description:

OTP is only 3 digits (which is obvious from the frontend) we can easily brute force it.

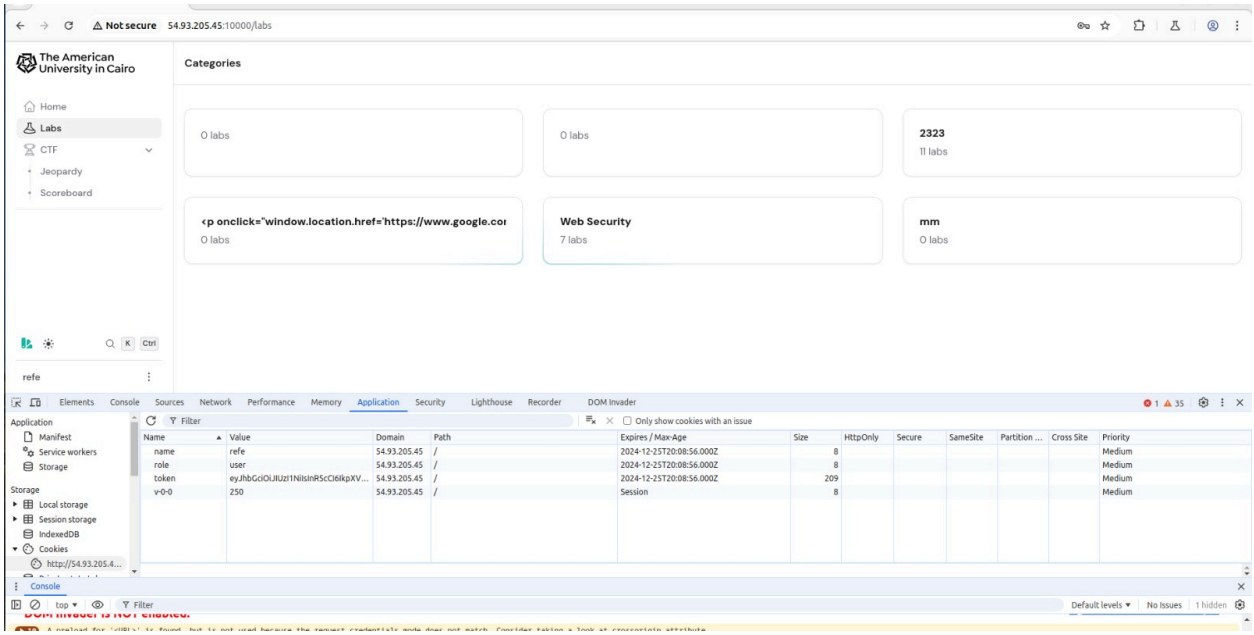
Besides, OTP doesn't have an expiry date, so if the user didn't request a new OTP, it will be the same.

Mitigation:

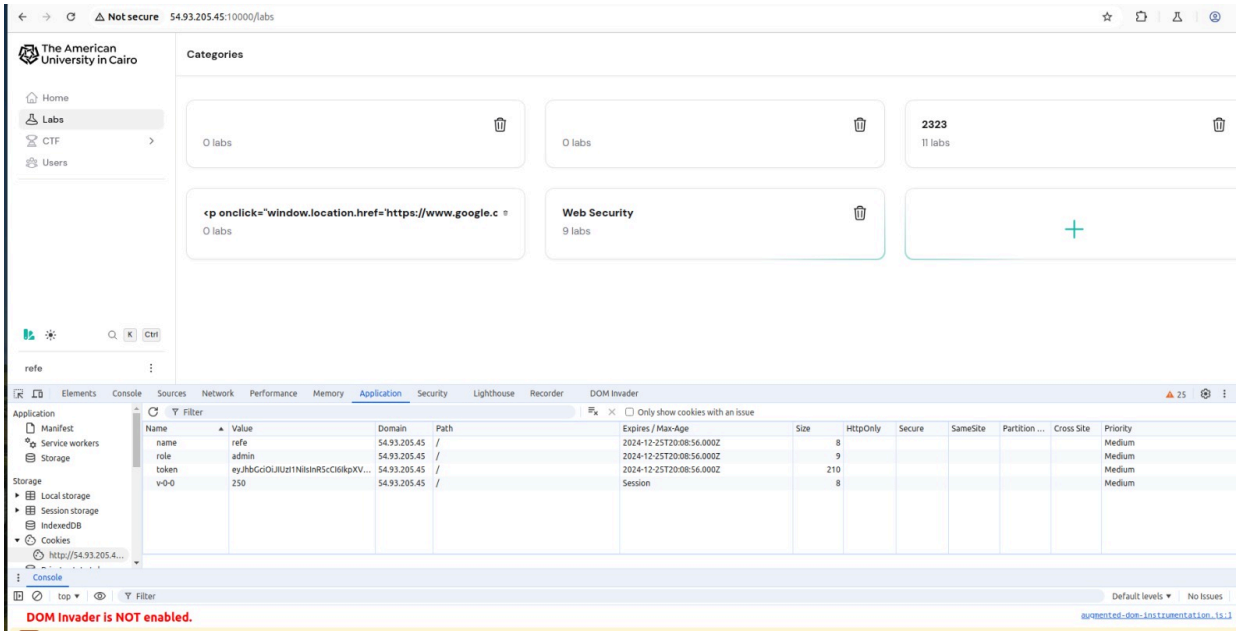
1. Remove the suggest OTP form the frontend, so the user cannot know its length.
2. Add expiry time to the OTP

5. Escalate a user to admin

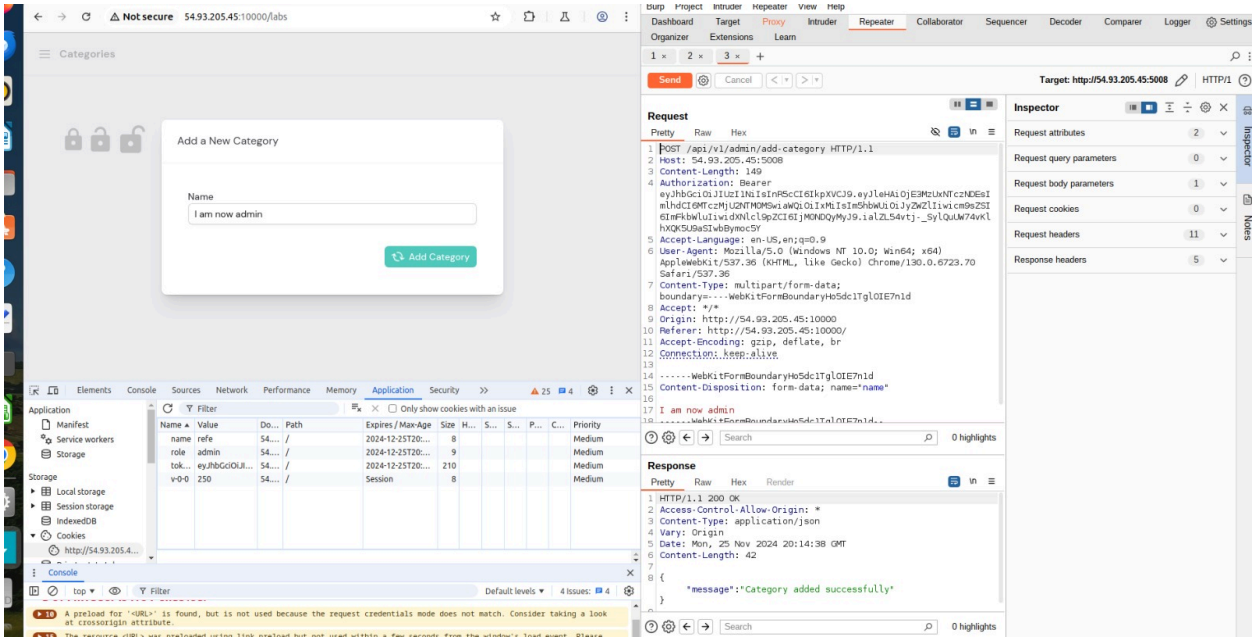
1. Normal user before making him admin and changing the token



2. The Normal user after becoming an admin.



3. The normal user adding a category himself after changing the token.



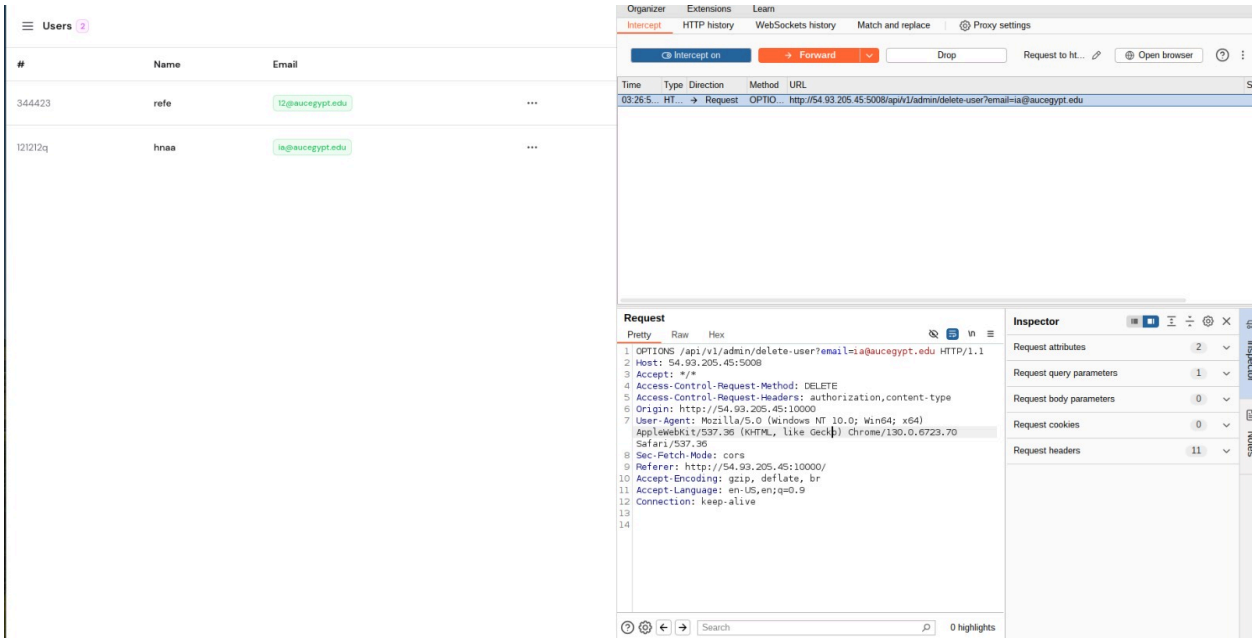
Description:

The user can be escalated to admin by changing the role in the cookie to equal admin in the cookie and changing it in the token as well as adding the secret using jwt.io to reflect in the backend as well.

Mitigation:

1. Ensure that JWTs are signed and validate the signature on the server side for every request to confirm that the token has not been tampered with.
2. Implement and enforce expiration dates for tokens to reduce the window of opportunity for an attacker to use a stolen token.
3. Instead of relying solely on client-side storage for critical data (like user role information), manage session states and roles on the server. When a session is established, the server should keep a record of the user's role and permissions and verify them for each request.

6. Delete Admin



Description:

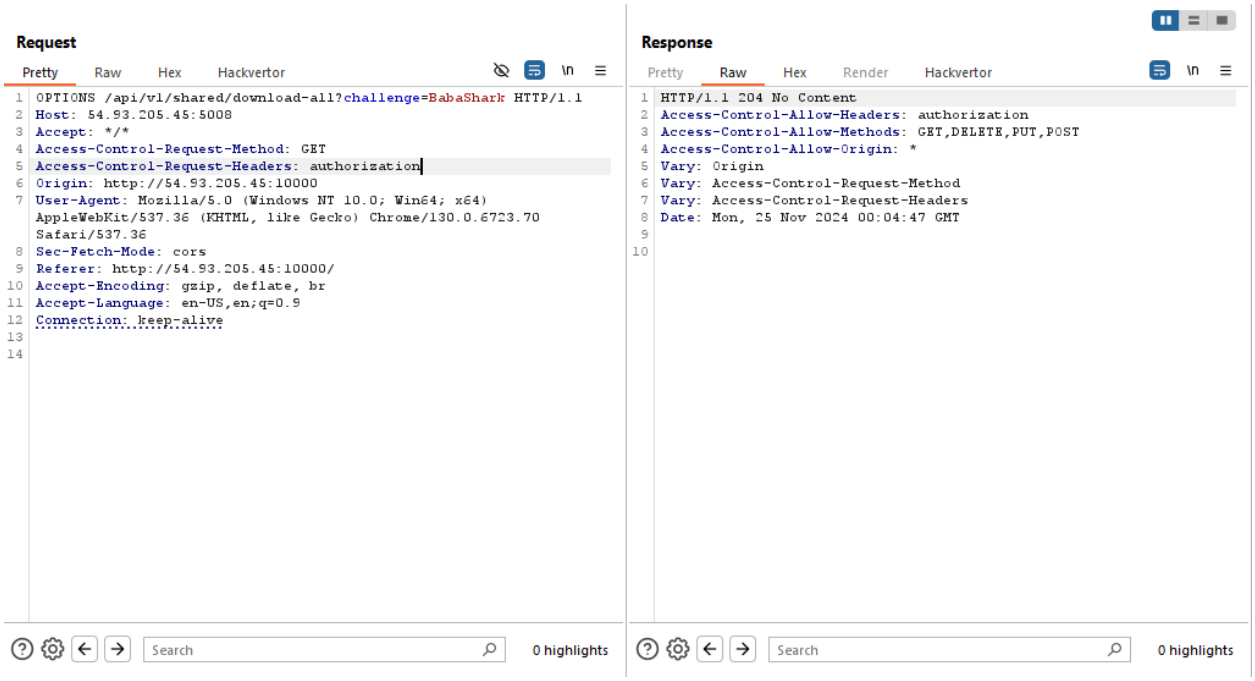
The Admin can delete other admins by changing the request when deleting a normal user, by intercepting the request, and changing the email in the request.

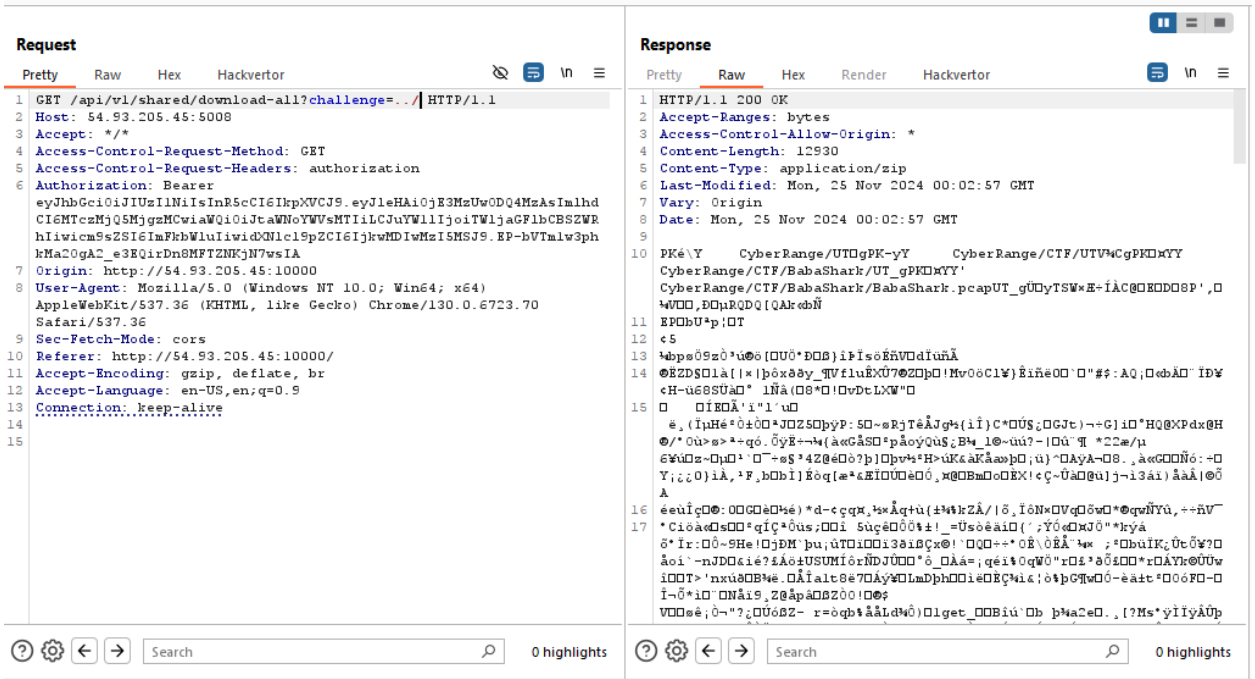
Mitigation:

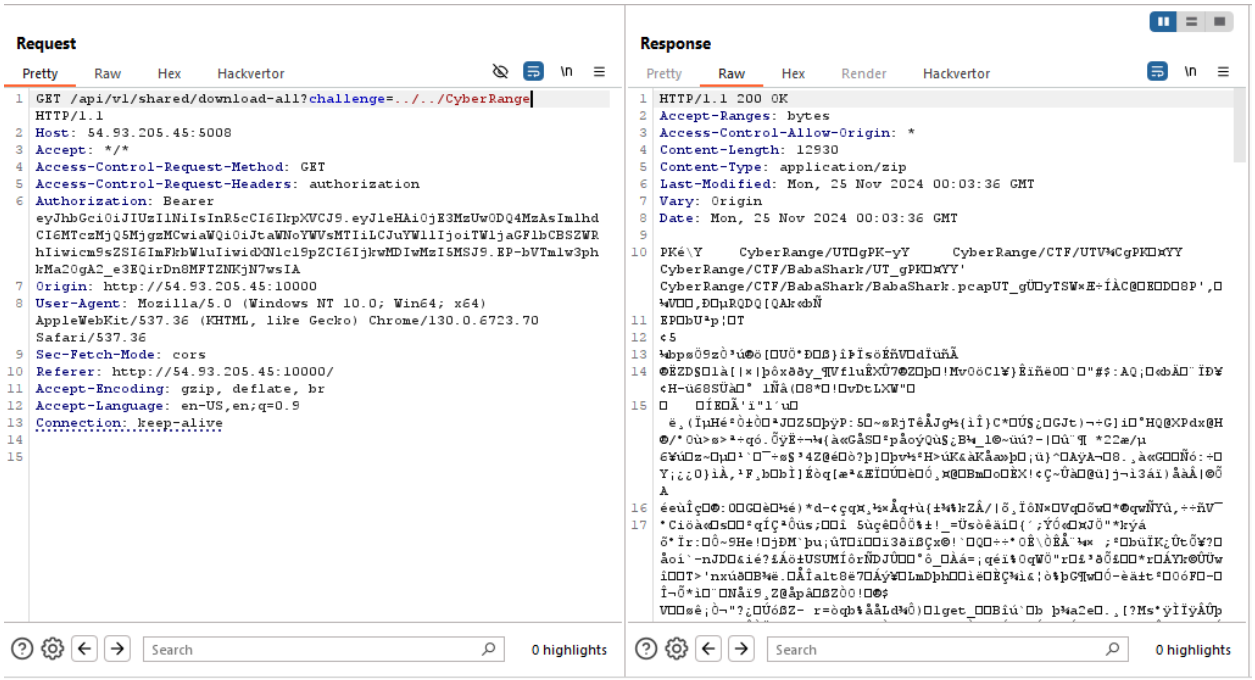
Always validate inputs on the server-side to prevent unauthorized actions due to manipulated requests. Ensure the email address is verified and that it is the user’s input before proceeding with any action.

Directory traversal:

1. Challenge=../../CyberRange







Description:

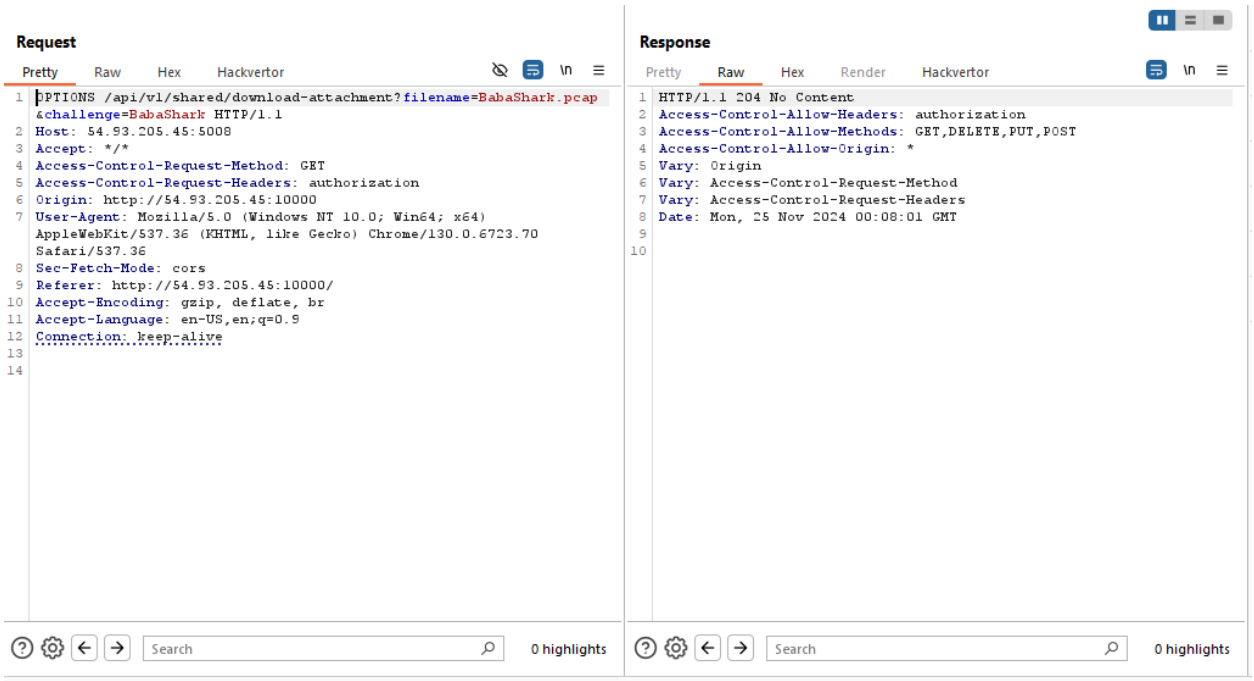
When the admin clicks on any of the challenges in the jeopardy. You can intercept the request:
Options /api/v1/shared/download-all?challenge=BabaShark HTTP/1.1
Change it to GET request and add the Authorization from any other GET request and then you can traverse to ../ or ../../CyberRnage

- GET /api/v1/shared/download-all?challenge=../ HTTP/1.1
- GET /api/v1/shared/download-all?challenge=../../CyberRange HTTP/1.1

Mitigation:

Use secure coding practices like rejecting path traversal patterns (e.g., "../"). Implement strict access controls and audit logs. Canonicalize the URL.

2. challenge=../ctf/



Request

Pretty Raw Hex Hackvertor

```
OPTIONS /api/v1/shared/download-attachment?filename=BabaShark.pcap
&challenge=BabaShark HTTP/1.1
Host: 54.93.205.45:5008
Accept: */*
Access-Control-Request-Method: GET
Access-Control-Request-Headers: authorization
Origin: http://54.93.205.45:10000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70
Safari/537.36
Sec-Fetch-Mode: cors
Referer: http://54.93.205.45:10000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
```

Search 0 highlights

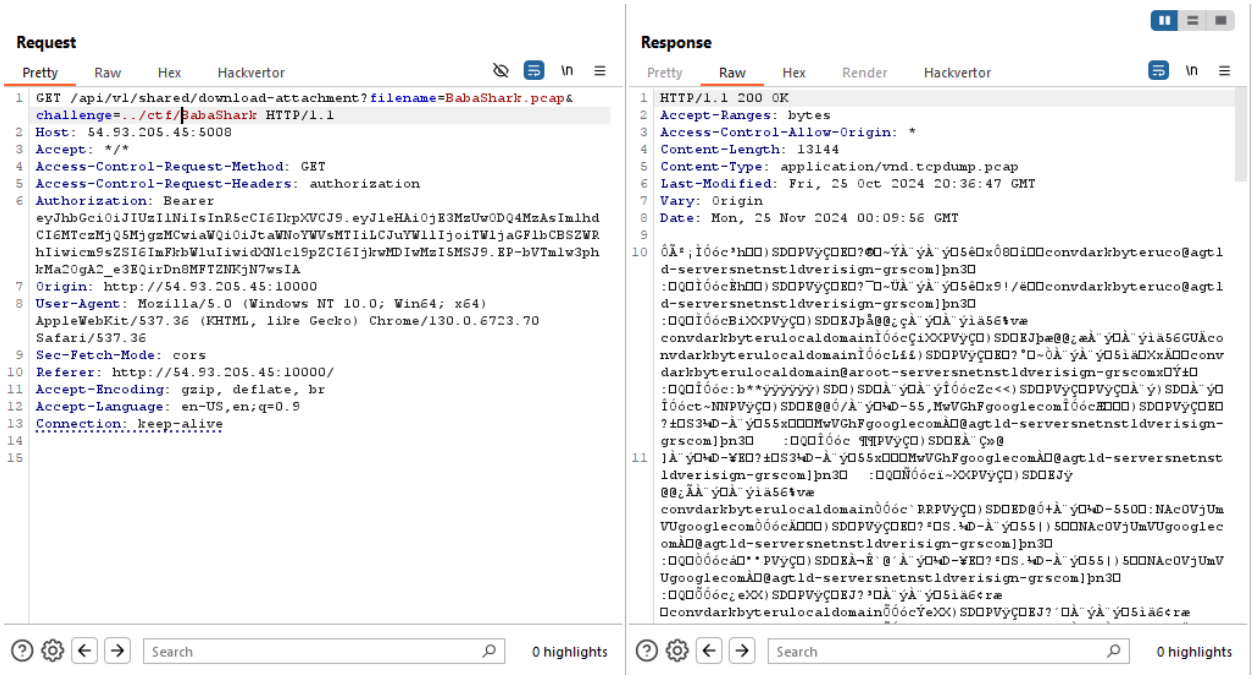
Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 204 No Content
Access-Control-Allow-Headers: authorization
Access-Control-Allow-Methods: GET,DELETE,PUT,POST
Access-Control-Allow-Origin: *
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Date: Mon, 25 Nov 2024 00:08:01 GMT
```

Search 0 highlights

Done



Request

Pretty Raw Hex Hackvertor

```
GET /api/v1/shared/download-attachment?filename=BabaShark.pcap&
challenge=../ctf/BabaShark HTTP/1.1
Host: 54.93.205.45:5008
Accept: */*
Access-Control-Request-Method: GET
Access-Control-Request-Headers: authorization
Authorization: Bearer
eyJhbGci0iJIUzIlNiIsInR5cCI6IkpXVCJ9.eyJleHAi0jE3MzUw0DQ4MzAsImlhd
CI6MTczMjQ5MjgzMCwiaWQi0iJtaWNoYWVsMTIiLCJuYWllIjoiTWljaGFlbCBSZWR
hIiwicm9sZSI6ImFkbWluIiwidXNlcl9pZCI6IjkwMDIwMzISMSJ9.EP-bVTmlw3ph
kMa20gA2_e3EQirDn8MFTZNKjN7wsIA
Origin: http://54.93.205.45:10000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70
Safari/537.36
Sec-Fetch-Mode: cors
Referer: http://54.93.205.45:10000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
```

Search 0 highlights

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Content-Length: 13144
Content-Type: application/vnd.tcpdump.pcap
Last-Modified: Fri, 25 Oct 2024 20:36:47 GMT
Vary: Origin
Date: Mon, 25 Nov 2024 00:09:56 GMT

ÔÃ²;ÌÔóc³h□□)SD□PVýÇ□E□?€□~ÝÀ¨ýÀ¨ý□5é□x□8□i□□convdarkbyteruco@agtl
d-serversnetnstldverisign-grscom]þn3□
:□Q□ÌÔócEh□□)SD□PVýÇ□E□?¯□~ÜÀ¨ýÀ¨ý□5é□x9!/é□□convdarkbyteruco@agtl
d-serversnetnstldverisign-grscom]þn3□
:□Q□ÌÔócBiXXPVýÇ□)SD□EJpå@@¿çÀ¨ý□À¨ýiä5é%væ
convdarkbyterulocaldomainÌÔóçiXXPVýÇ□)SD□EJpæ@@¿æÀ¨ý□À¨ýiä5éGUÀco
nvdarkbyterulocaldomainÌÔócLåí)SD□PVýÇ□E□?°□~ûÀ¨ýÀ¨ý□5iä□XxÀ□□conv
darkbyterulocaldomain@aroot-serversnetnstldverisign-grscom□Ý±□
:□Q□Ì□óc:b**ÿÿÿÿÿÿ)SD□)SD□À¨ý□À¨ýÌÔócZc<<)SD□PVýÇ□PVýÇ□À¨ý)SD□À¨ý□
Ì□óct~NNPVýÇ□)SD□E@@□/À¨ý□¼D-55,MwVGhFgooglecomÌ□óc□□□)SD□PVýÇ□E□
?±□S3¼D-À¨ý□55x□□□MwVGhFgooglecomÀ□@agtld-serversnetnstldverisign-
grscom]þn3□  :□Q□Ì□óc ¶¶PVýÇ□)SD□EÀ¨Ç»@
]À¨ý□¼D-¥E□?±□S3¼D-À¨ý□55x□□□MwVGhFgooglecomÀ□@agtld-serversnetnst
ldverisign-grscom]þn3□  :□Q□Ñ□óci~XXPVýÇ□)SD□EJý
@@¿ÅÀ¨ý□À¨ýiä5é%væ
convdarkbyterulocaldomainÒ□óc`RRPVýÇ□)SD□ED@□+À¨ý□¼D-550□:NAc0VjUm
VUgooglecom□□ócÀ□□□)SD□PVýÇ□E□?ªS.¼D-À¨ý□55|)5□□NAc0VjUmVUgooglec
omÀ□@agtld-serversnetnstldverisign-grscom]þn3□
:□Q□□□ócÁ□**PVýÇ□)SD□EÀ¬Ê¨@¨À¨ý□¼D-¥E□?ªS.¼D-À¨ý□55|)5□□NAc0VjUmV
UgooglecomÀ□@agtld-serversnetnstldverisign-grscom]þn3□
:□Q□□□óc¿eXX)SD□PVýÇ□EJ?³□À¨ýÀ¨ý□5iä6¢ræ
□convdarkbyterulocaldomain□□ócÝeXX)SD□PVýÇ□EJ?´□À¨ýÀ¨ý□5iä6¢ræ
```

Search 0 highlights

Description:

When the admin clicks on any of the challenges in the jeopardy, and clicks on any of the files that are attached to the challenge. You can intercept the request:

Options /api/v1/shared/download-attachment?challenge=../ctf/ file name=BabaShark.pcap HTTP/1.1

Change it to GET request and add the Authorization from any other GET request and then you can traverse to ../ctf/BabaShark *(or any challenge name)*

GET /api/v1/shared/download-attachment?challenge=../ctf/challenge HTTP/1.1

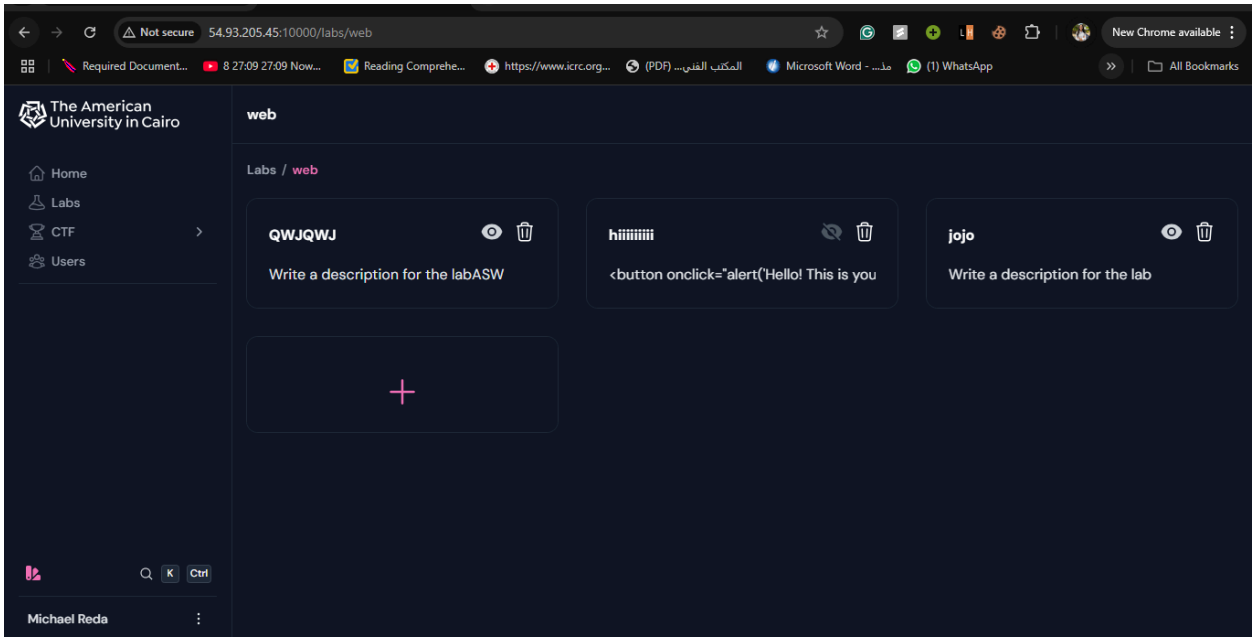
Mitigation:

Use secure coding practices like rejecting path traversal patterns (e.g., "../"). Implement strict access controls and audit logs. Canonicalize the URL.

3. /labs/(anything)

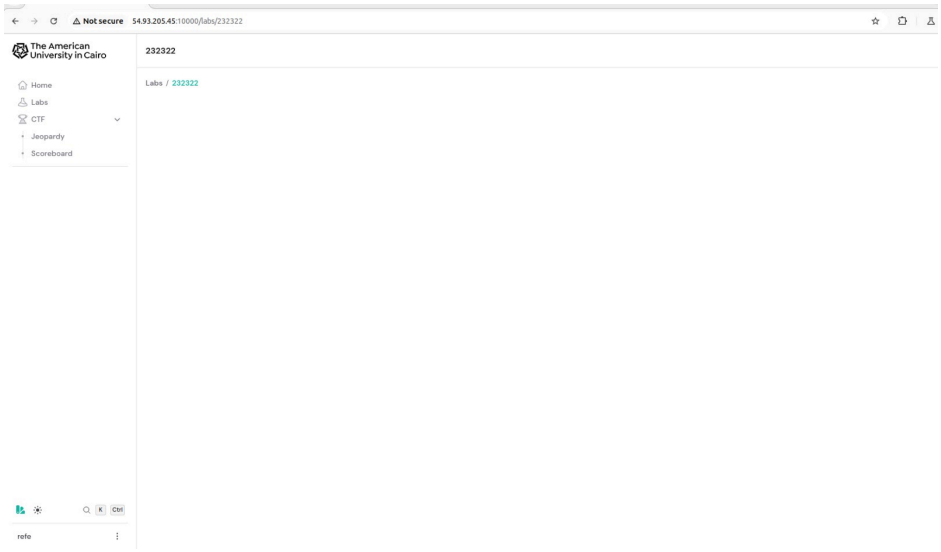
Description:

Changing the url from above after /labs to access any tabs there



By traversing to the following: http://54.93.205.45:10000/labs/web, the user see the hidden lab.

Another shape for it even if there is no labs inside

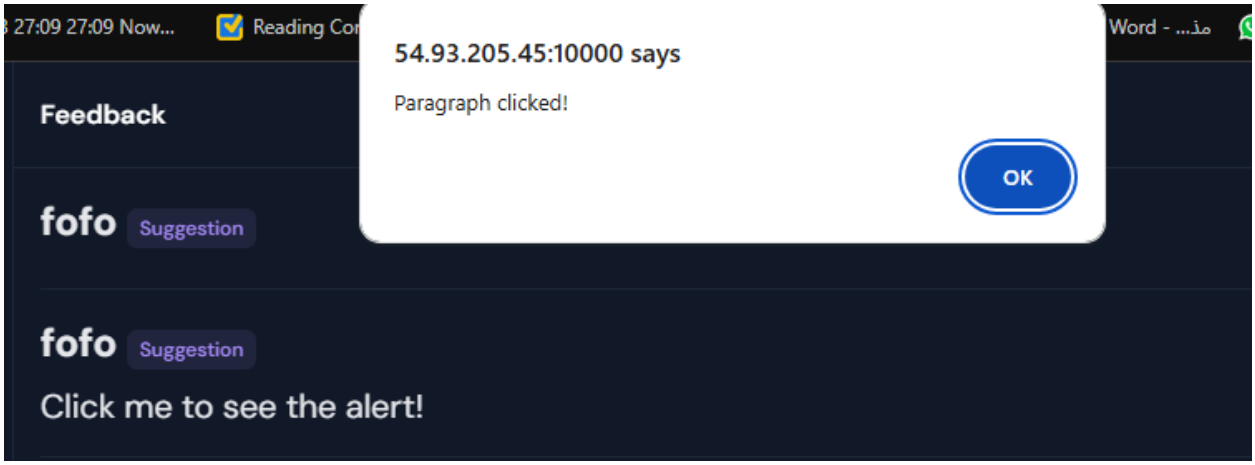


Mitigation:

Use secure coding practices like rejecting path traversal patterns (e.g., "../"). Implement strict access controls and audit logs. Canonicalize the URL.

XSS:

1. Stored XSS in the feedback message



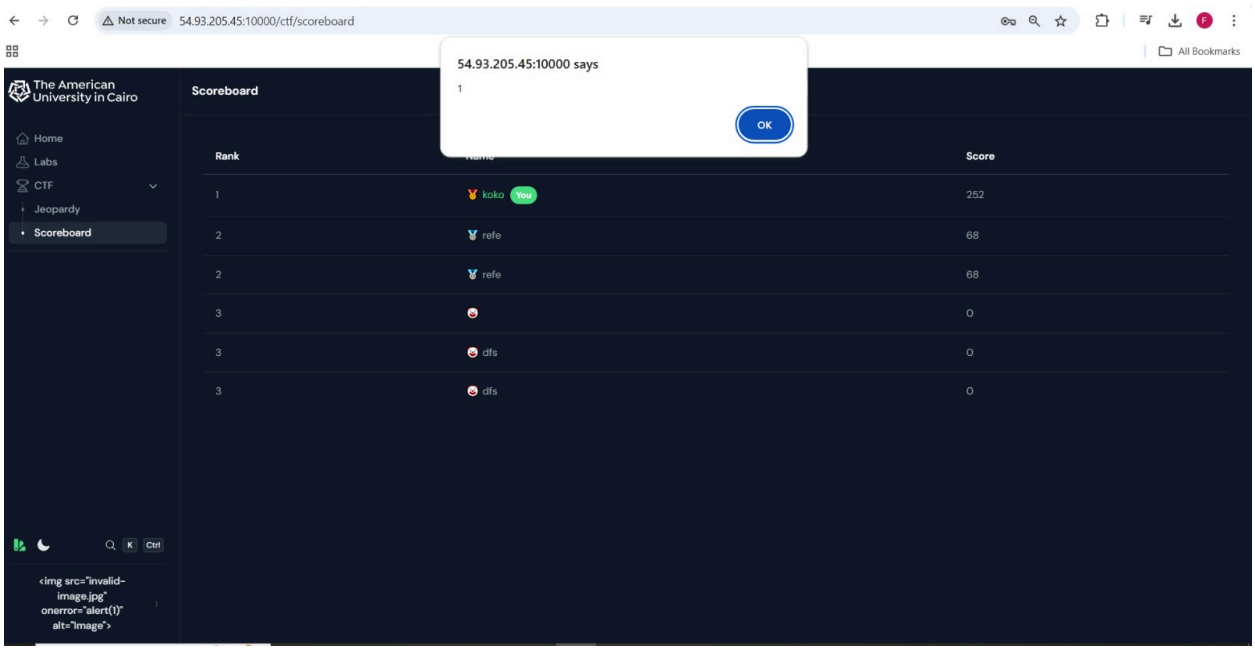
Description:

The user can inject an HTML code into the feedback message.

Mitigation:

Sanitize and validate user inputs before storing them and make sure that the outputs is not outputting unexpected outputs.

2. Stored XSS in user name.



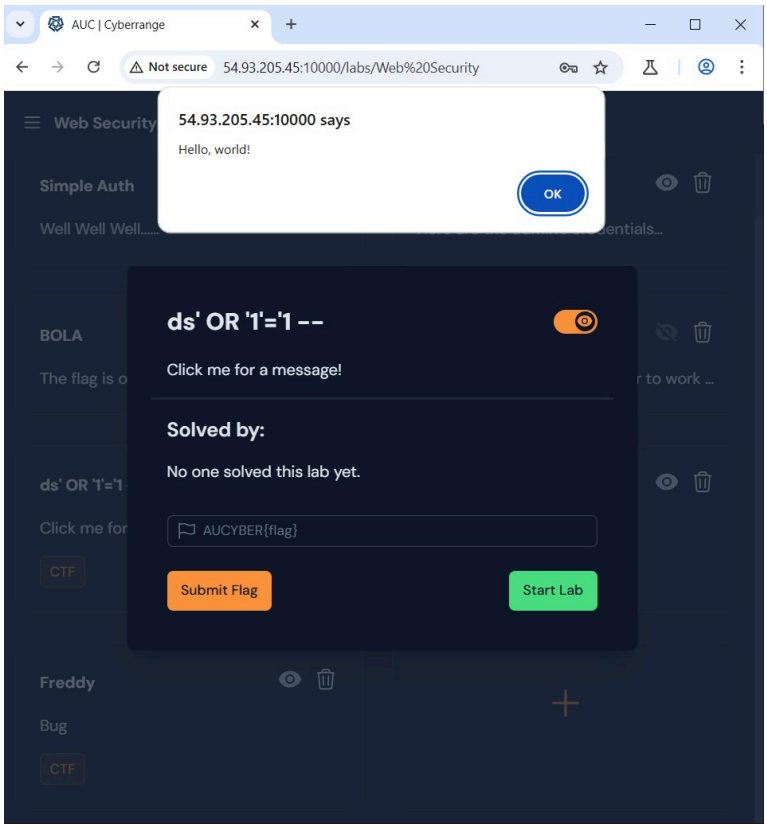
Description:

The user can inject an HTML code into the change name and save changes.

Mitigation:

Sanitize and validate user inputs before storing them and make sure that the outputs is not outputting unexpected outputs.

3. Stored XSS Lab Description



Description:

Can Store HTML in the lab description

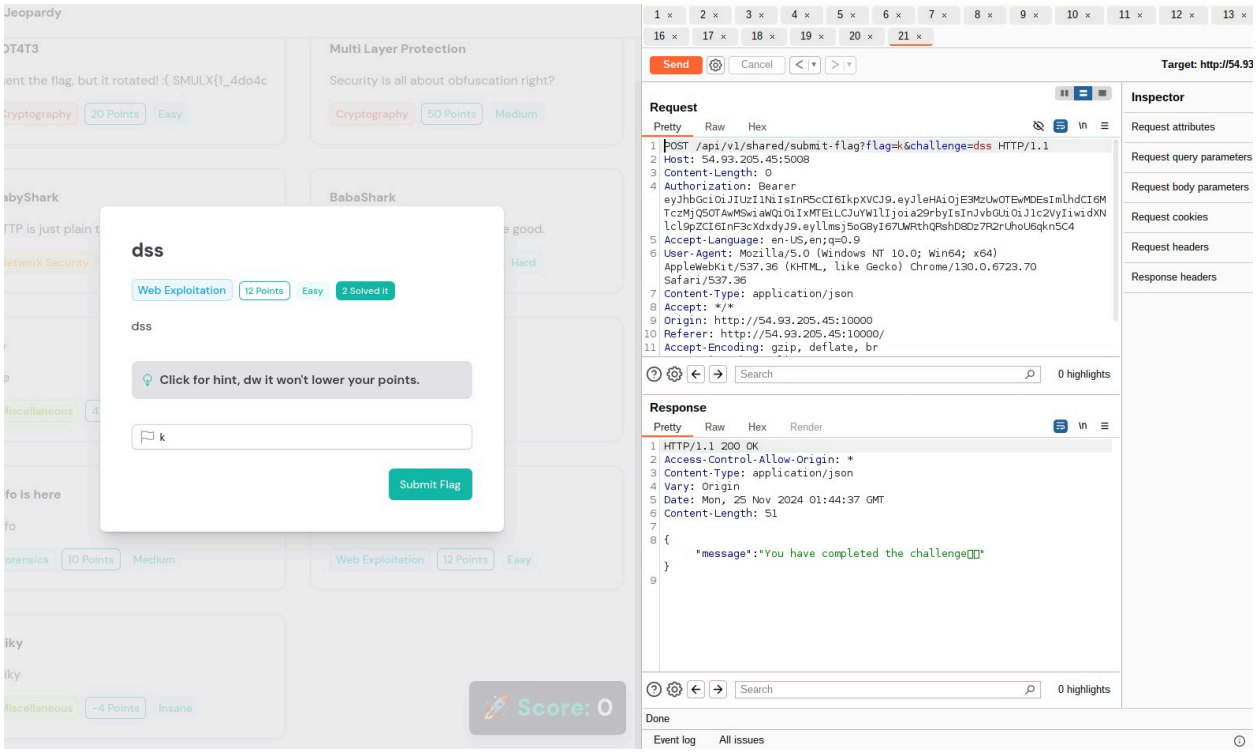
Mitigation:

Sanitize and validate user inputs before storing them and make sure that the outputs is not outputting unexpected outputs.

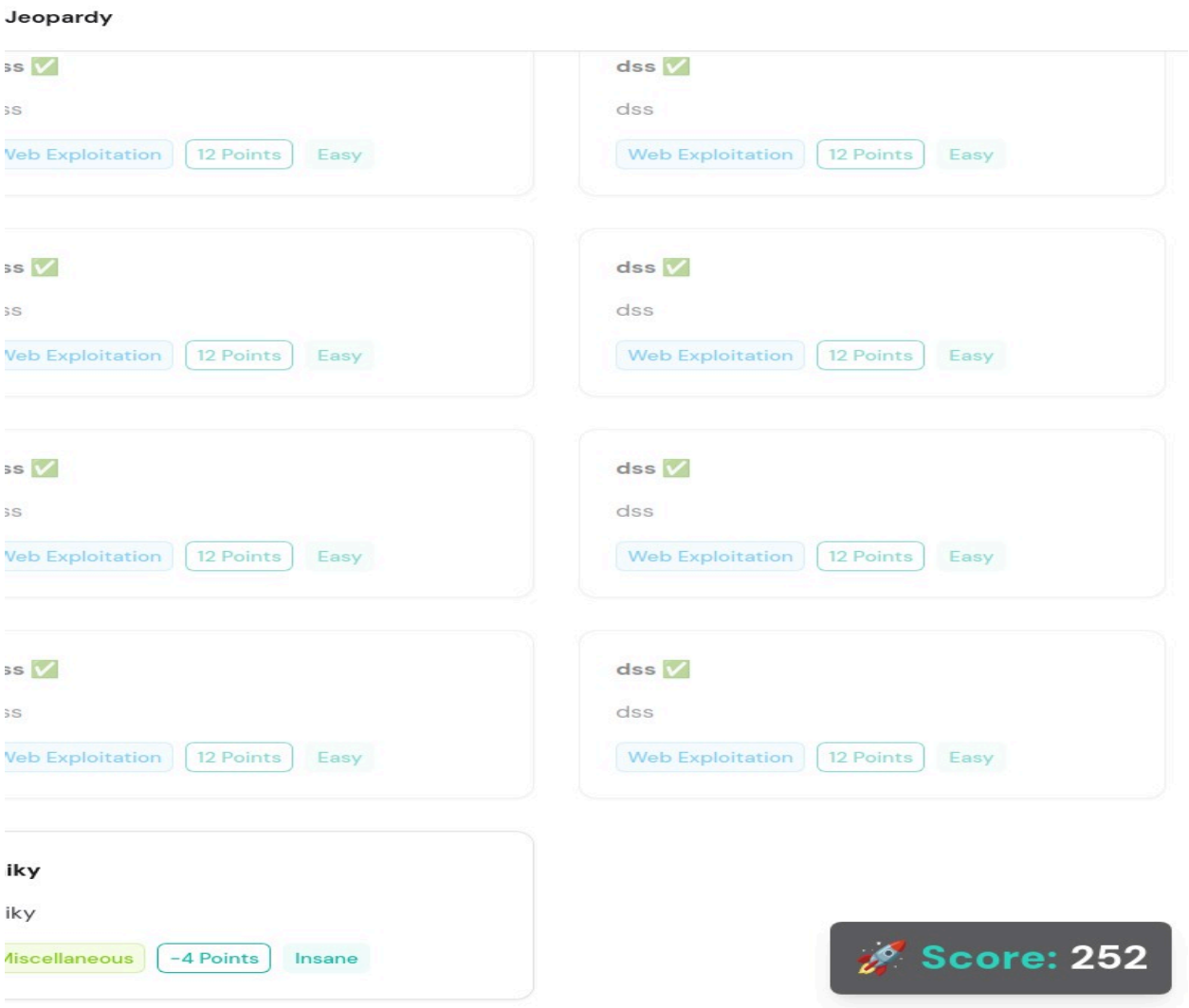
Business Logic:

1. Increase Score

1. Before submitting the flag:



2. After submitting the lab many times:



Description:

The user can take the request and submit the challenge multiple times to increase his score.

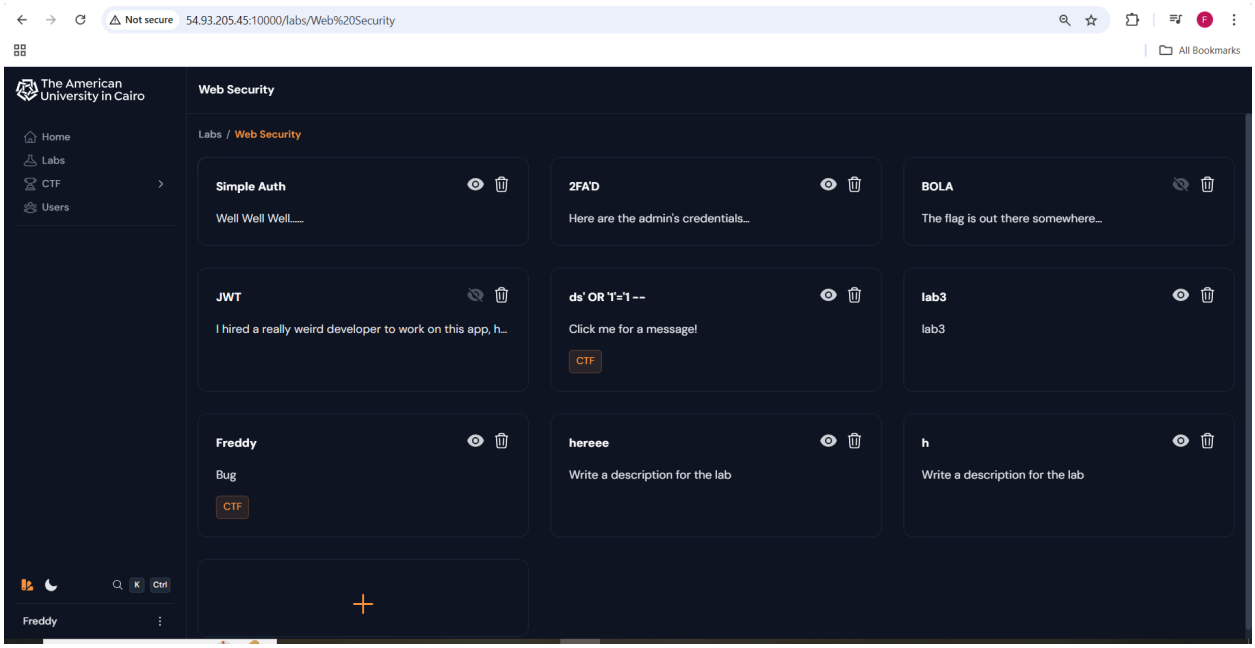
Mitigation:

Implement a server-side check to ensure that a challenge can only be submitted once per user by maintaining a submission record. Reject duplicate submissions and notify the user.

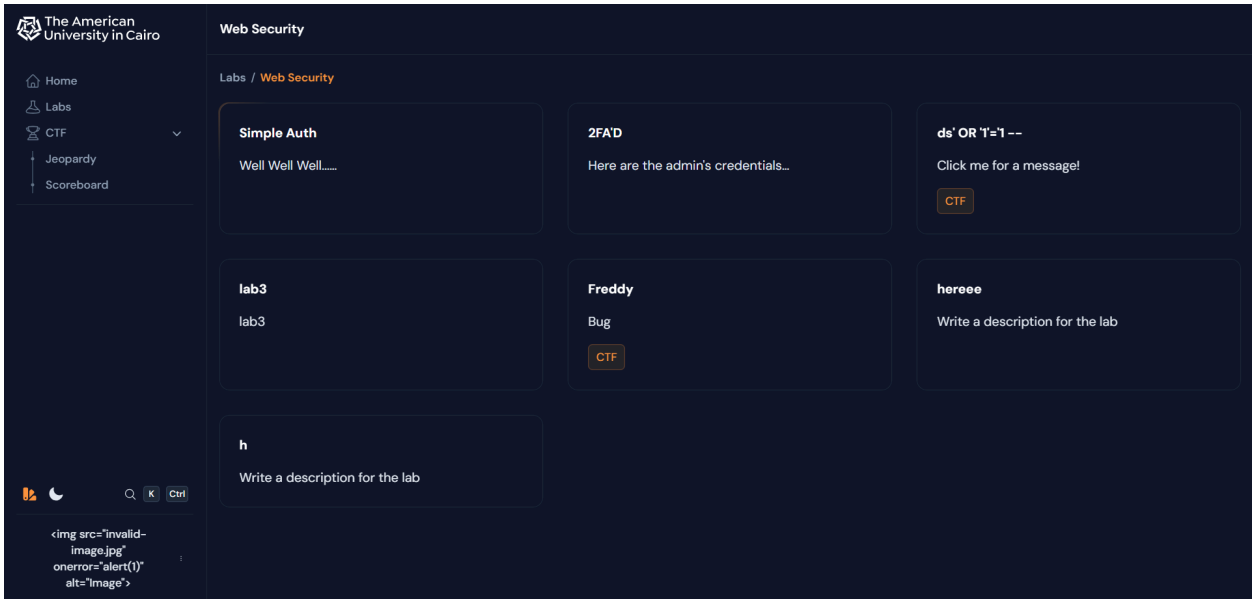
2. The user can access hidden labs by just typing the name of the lab.

Description:

1. The hidden lab as shown is BOLA

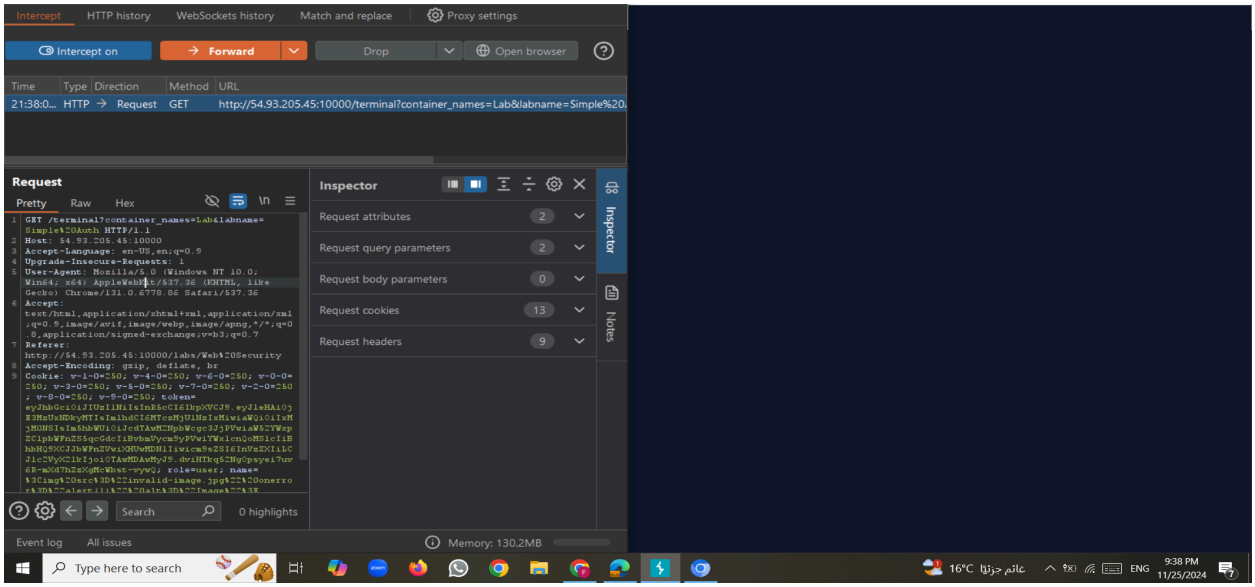


2. Then from the user's perspective

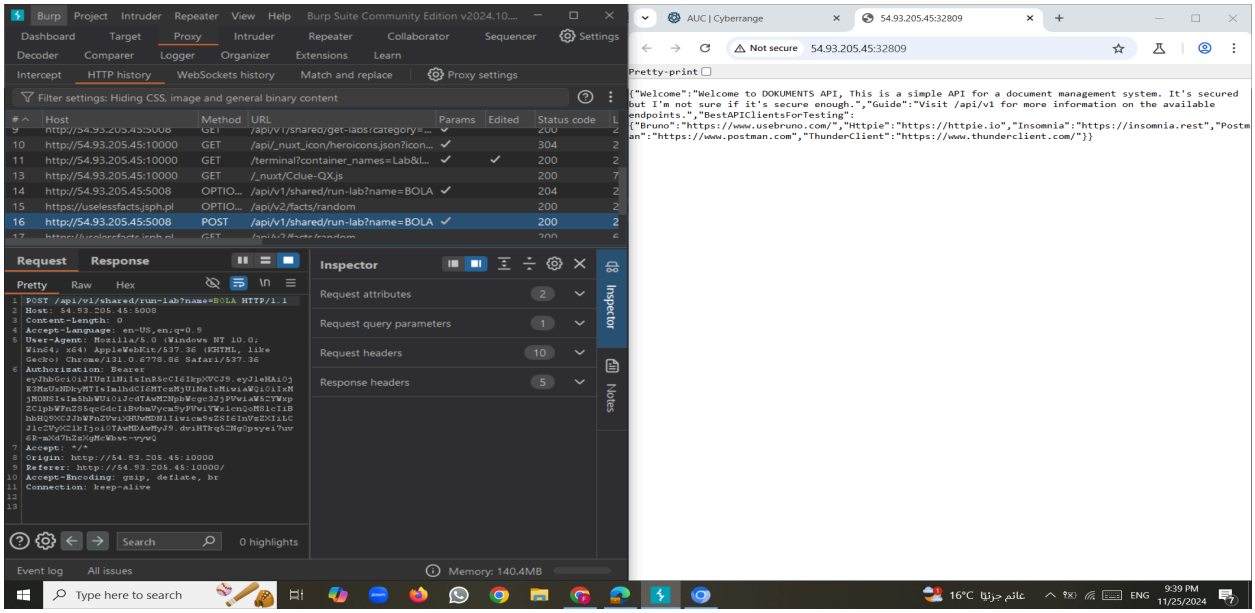


3. The original request of the user trying to access an existing lab

My



4. Then the result after changing the name in the request to the hidden lab



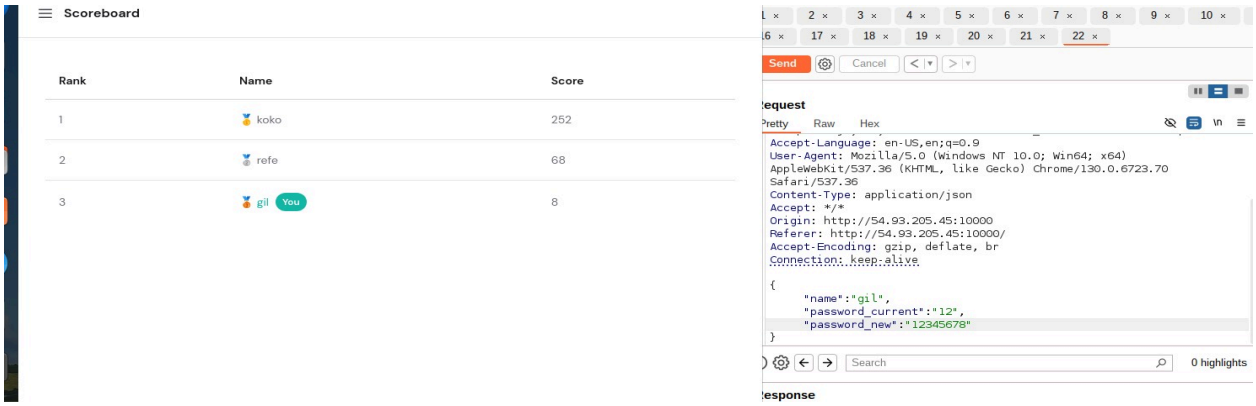
Mitigation:

Validate the requested lab against a database of labs accessible to the user based on their permissions. Reject requests for labs not assigned to the user.

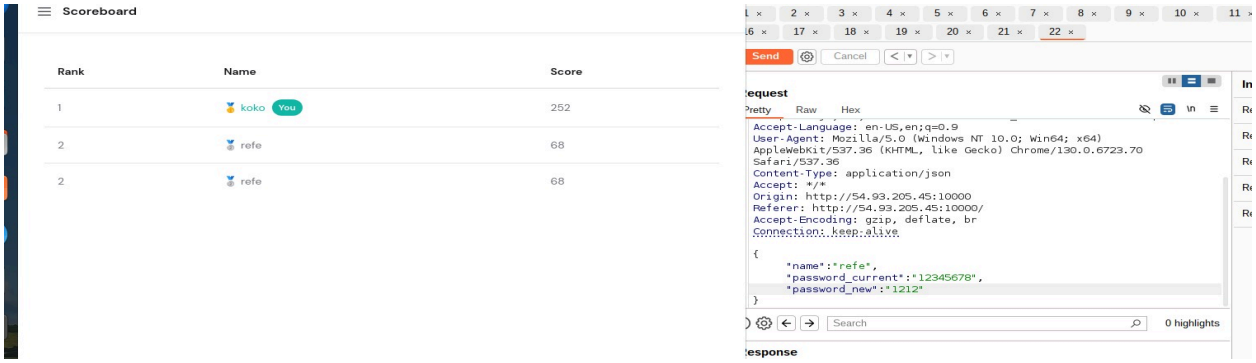
3. Change the name

Description:

1. Before changing the user name to another name that already exists



2. After changing the user name to a user name that already exists this copies the already existing user score to the user

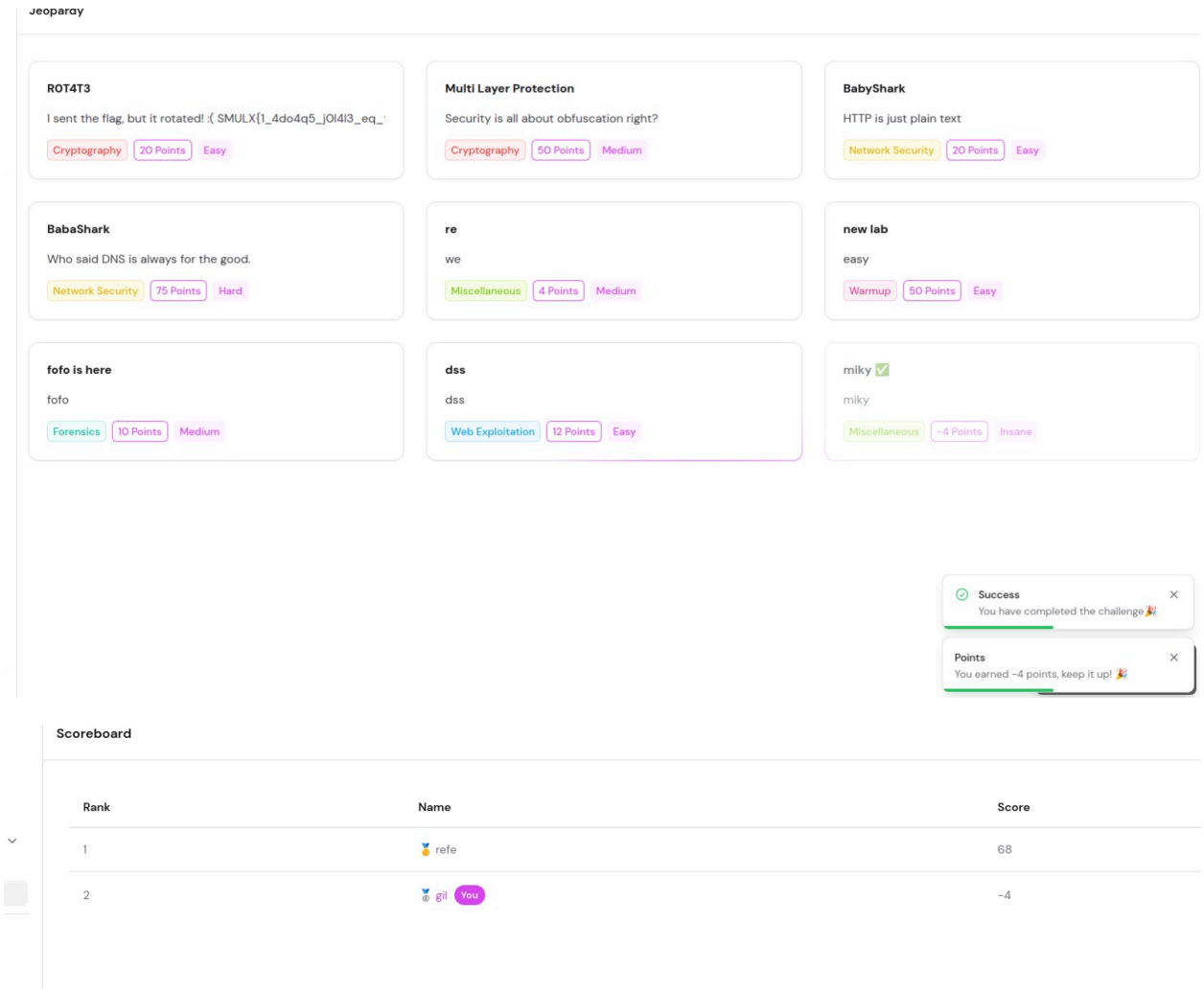


3. By doing so the rank of the user will change to maybe a user with a higher rank

Mitigation:

Use email as the identifier and tie points to an immutable user ID, not the username, to prevent tampering.

4. Change the score to negative



Description:

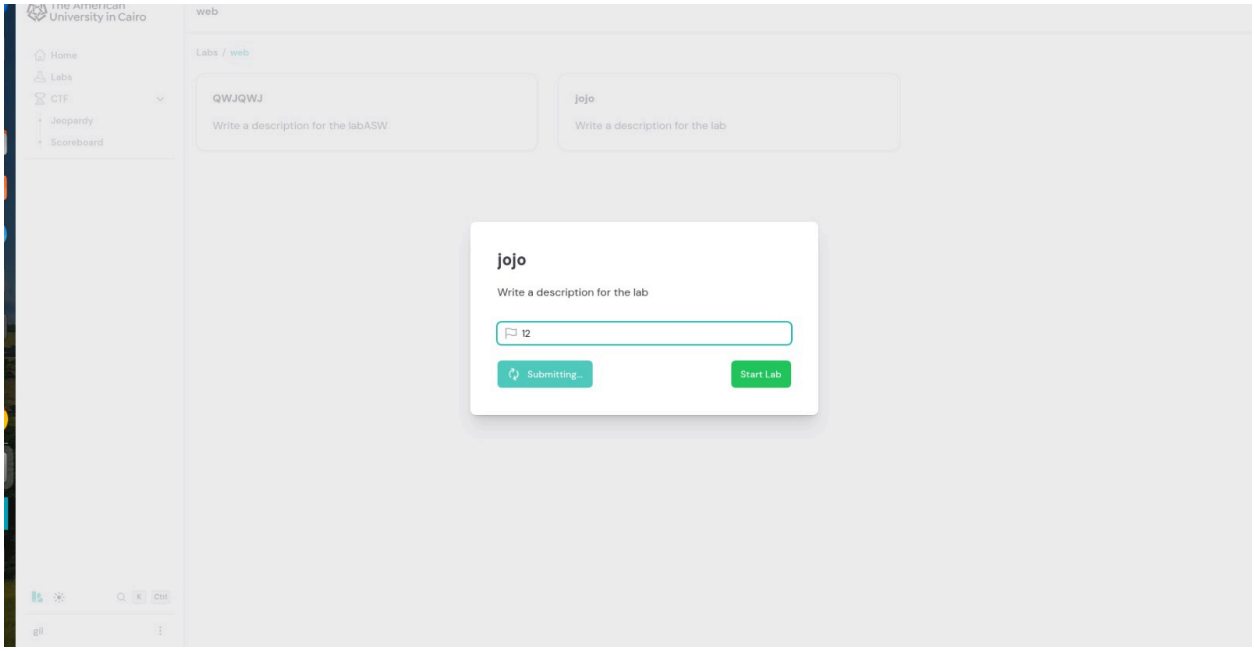
When the admin is creating the lab, he cannot change the score to negative in the frontend, but he can intercept the request and change it to a negative number.

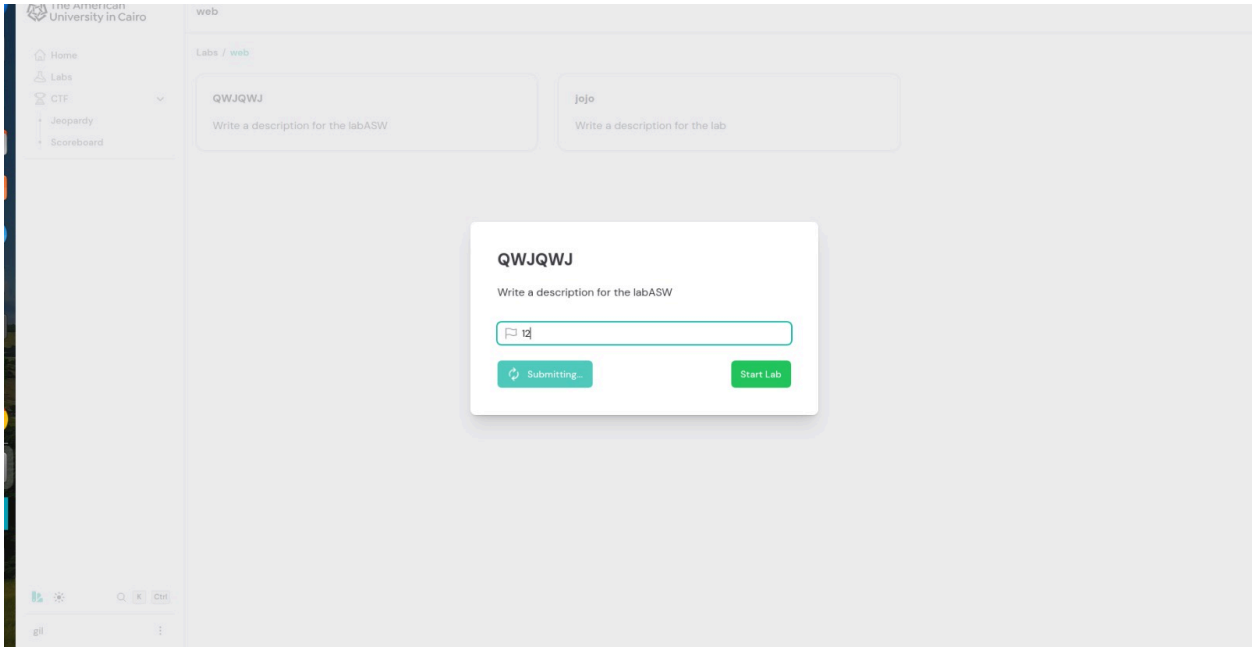
Mitigation:

To mitigate this issue, implement server-side validation to enforce business rules, such as ensuring the score cannot be negative. Never rely solely on front-end controls for data validation. Reject invalid requests at the backend and log suspicious activities for monitoring and further investigation.

Bugs:

1. Lag in the flags





Description:

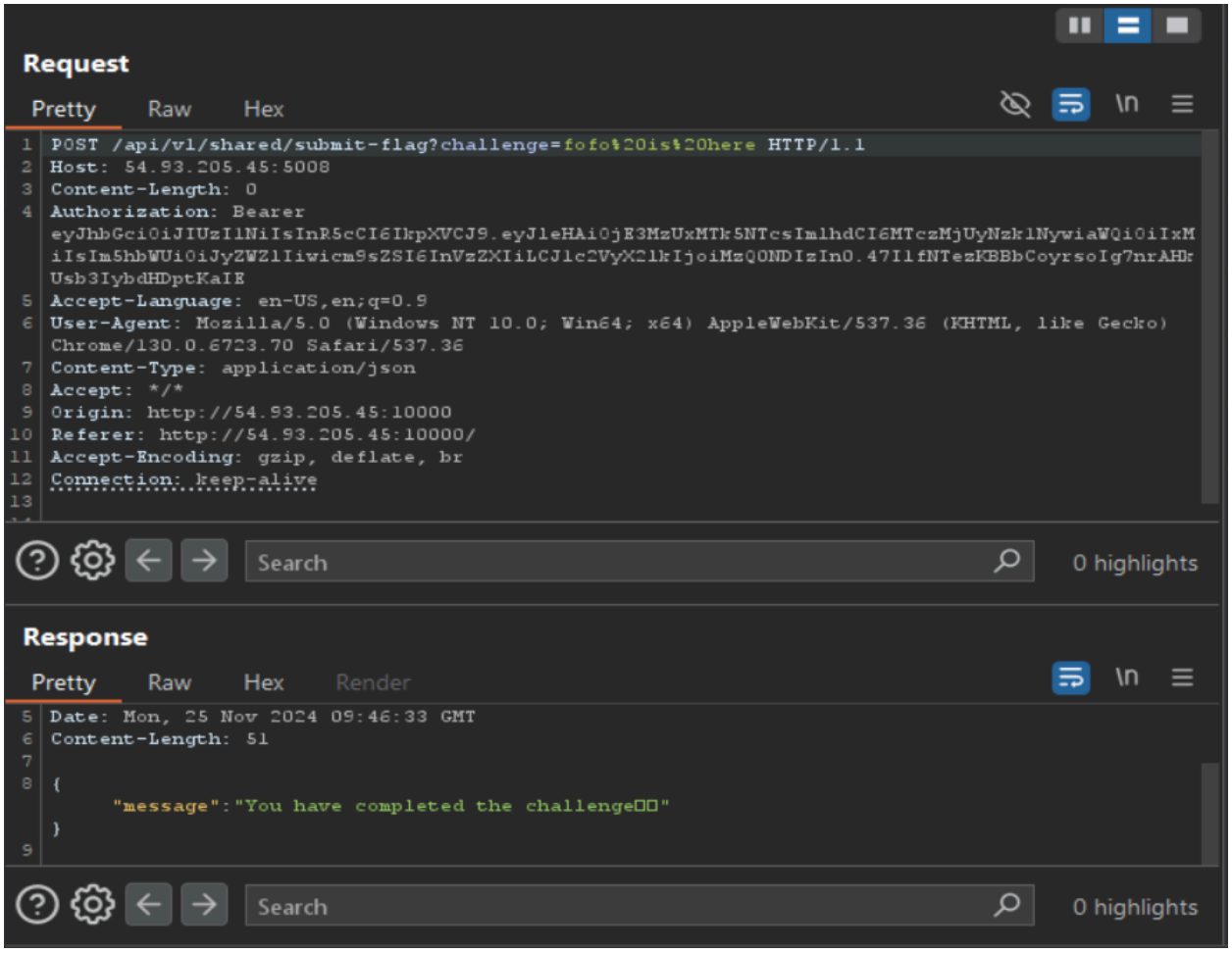
Bug of writing all the flag values at the same time in all labs.

2. Error Fetching Space at the end

Description:

The other Bug is that when the lab has a space at the end when it is retrieved to be deleted in the Frontend it will not be deleted; however, it will return deleted successfully.

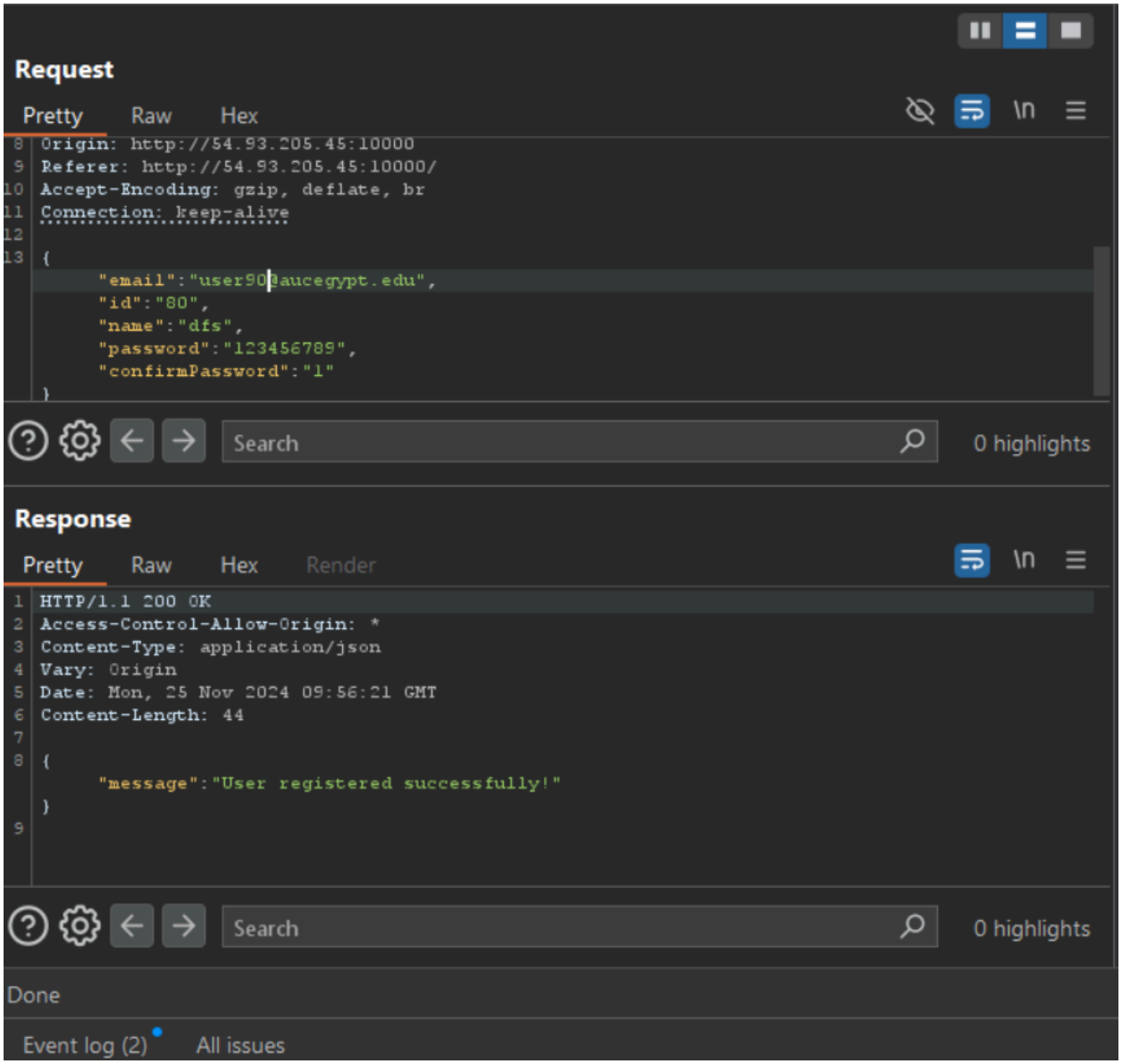
3. Submit the lab without a flag



Description:

I created a challenge that when I submit without the flag it returns that “You have completed the challenge🎉”; however, it is not reflected in the scoreboard.

4. Confirmed Password

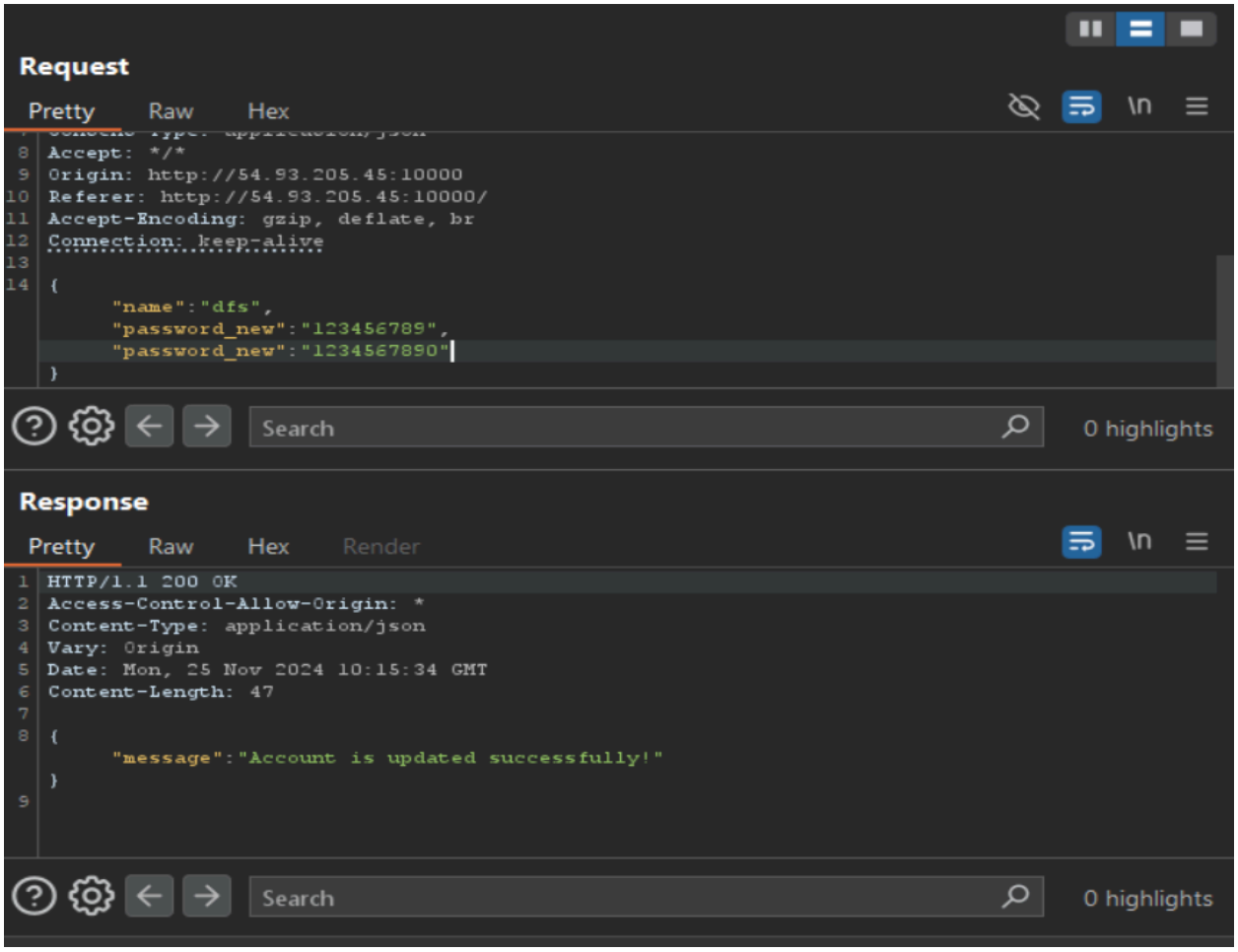


Description:

Users can make an account with a password that does not match the confirmed password.

Bonus :

1. No need to confirm the password.



Description:

The user can change his password without checking for the old password.

Mitigation:

Require the user to input their current password and validate it server-side before allowing a password change.