1. Api Security Testing

1.1 BOlA

Testing the Api security for Bola and graphQL injection and create a test environment from portswigger lab.

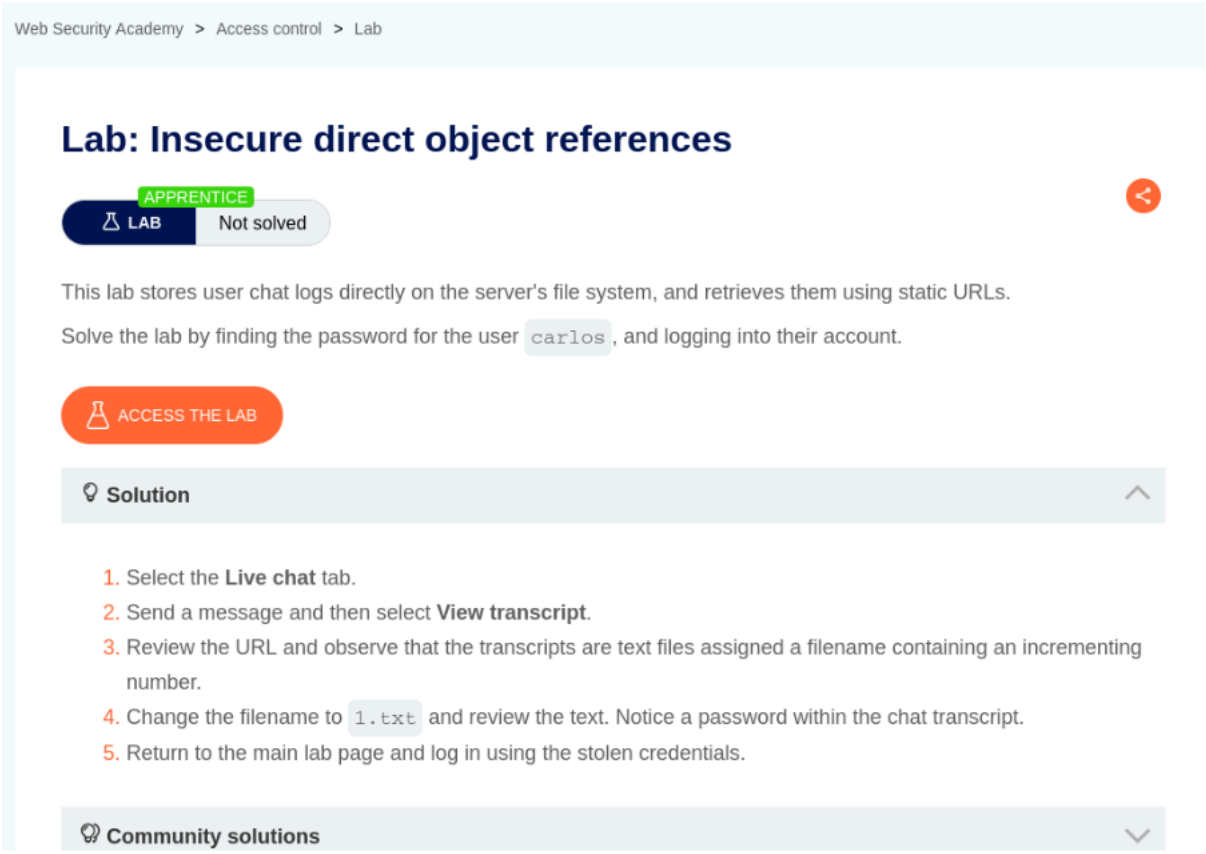
Environment setup:

1.Portswigger web security Academy Lab

2.Attacker host with Burp suite as Proxy.

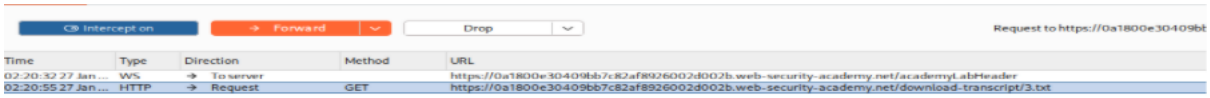
The tools which are used in this as Burp suite, sqlmap, Postman.

IDOR (Insecure Design Object Reference) is also known as Bola

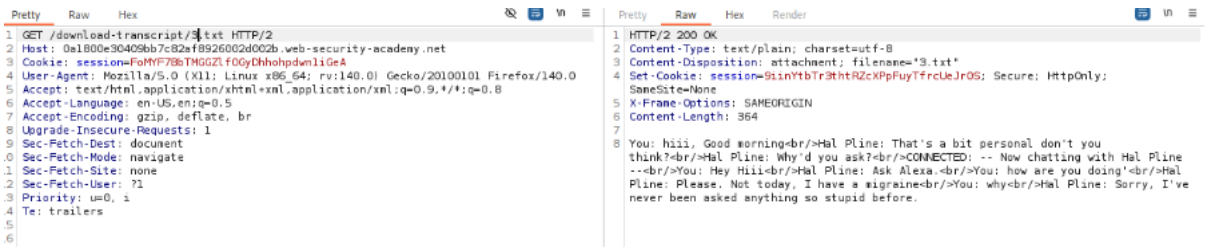


After setting the lab, we have to use burp suite to start the proxy. After that you have to do chat in Live chat than select a view transcript in the webpage to download the text.

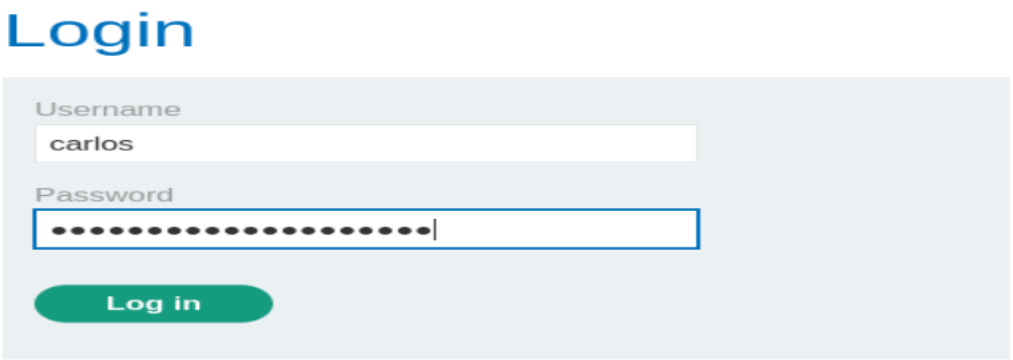
And, capture the request to the burp proxy and then send the request to the repeater.



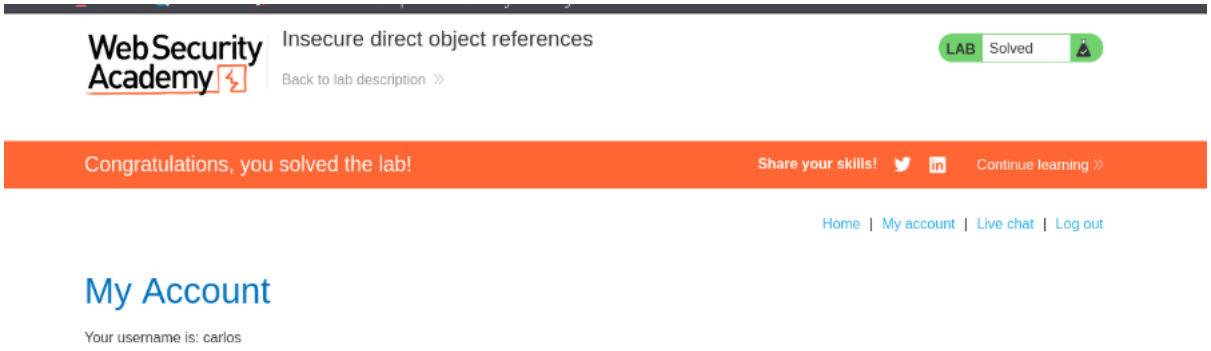
After sending the request to the repeater change the file name from 3.txt to 1. txt to get the credentials.



After getting the credentials, use login page to login with the identified credentials



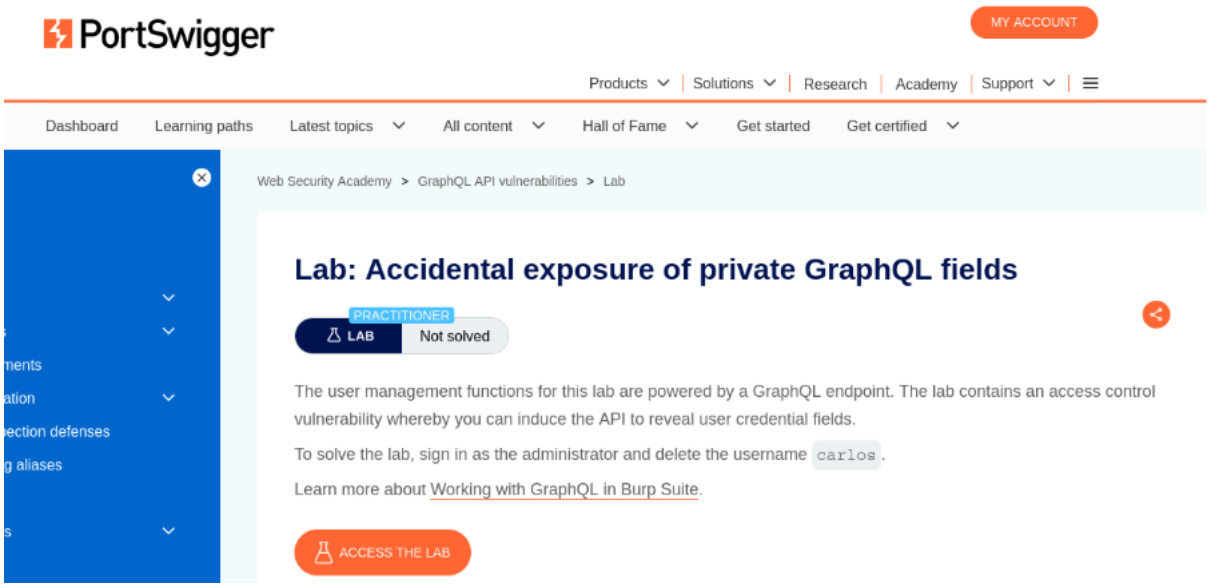
Then, it will logged in and the lab is solved.



1.2 GraphicQL Injection

First, we have to setup the test environment for GraphicQL injection.

Setup Portswigger Web Applivation Security Lab and use Attacker host with Burp suite Proxy.

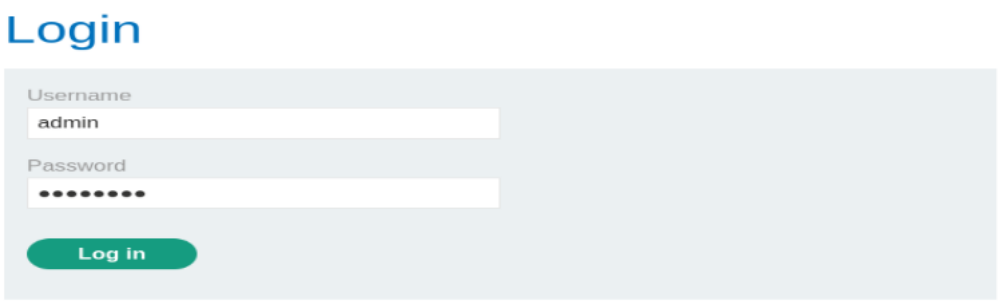


Access the lab and capture the request to Burp suite proxy and navigate to the My account tab in the lab and enter the default credentials as admin/password

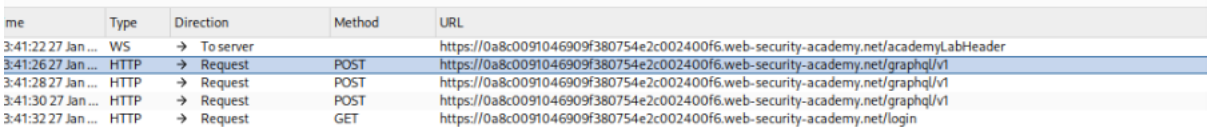
Web Security Academy

Accidental exposure of private GraphQL fields

Back to lab description >>



Capture the request and send the request to the repeater



me	Type	Direction	Method	URL
3:41:22 27 Jan ...	WS	→ To server		https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/academyLabHeader
3:41:26 27 Jan ...	HTTP	→ Request	POST	https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/graphql/v1
3:41:28 27 Jan ...	HTTP	→ Request	POST	https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/graphql/v1
3:41:30 27 Jan ...	HTTP	→ Request	POST	https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/graphql/v1
3:41:32 27 Jan ...	HTTP	→ Request	GET	https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/login

equest

Pretty Raw Hex GraphQL

```
POST /graphql/v1 HTTP/2
Host: 0a8c0091046909f380754e2c002400f6.web-security-academy.net
Cookie: session=5Xf70TdUd6y5YnBXnD6vb9tn0PKFKCi5
Content-Length: 234
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Accept: application/json
Sec-Ch-Ua: "Not_A Brand";v="99", "Chromium";v="142"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Origin: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

In repeater, analyze the code and navigate to GaphQL to view the source code for credentials.

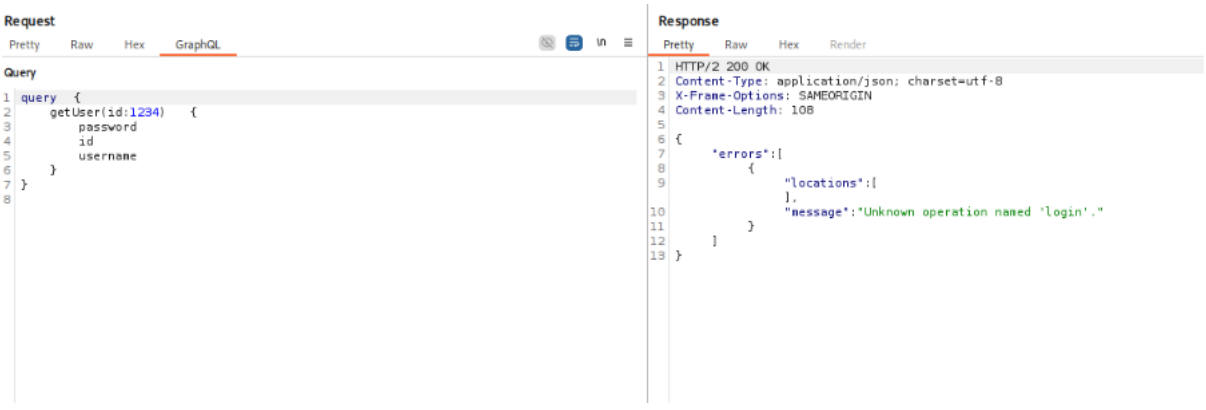
Request

Pretty Raw Hex GraphQL

```
POST /graphql/v1 HTTP/2
Host: 0a8c0091046909f380754e2c002400f6.web-security-academy.net
Cookie: session=5Xf70TdUd6y5YnBXnD6vb9tn0PKFKCi5
Content-Length: 202
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Accept: application/json
Sec-Ch-Ua: "Not_A Brand";v="99", "Chromium";v="142"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/142.0.0.0 Safari/537.36
Origin: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{
      "query":
      "query\t{\n    getUser(id:1234)\t{\n        password\n        id\n        usern
      ame\n    }\n}\n        ",
      "operationName":"login",
      "variables":{
            "input":{
                  "username":"admin",
                  "password":"pass@123"
            }
      }
}
```

After, send the request and analyze the results in the below screenshot. It was ok but the success is failure. So, you have to change the code and navigate to the Inql scanner and copy the captured request and past in the Inql Scanner.



Then, we have got some results as different queries, in that use the login query and copy that and past in the GaphQL tabl to get the credentails. Which you have seen in the below.

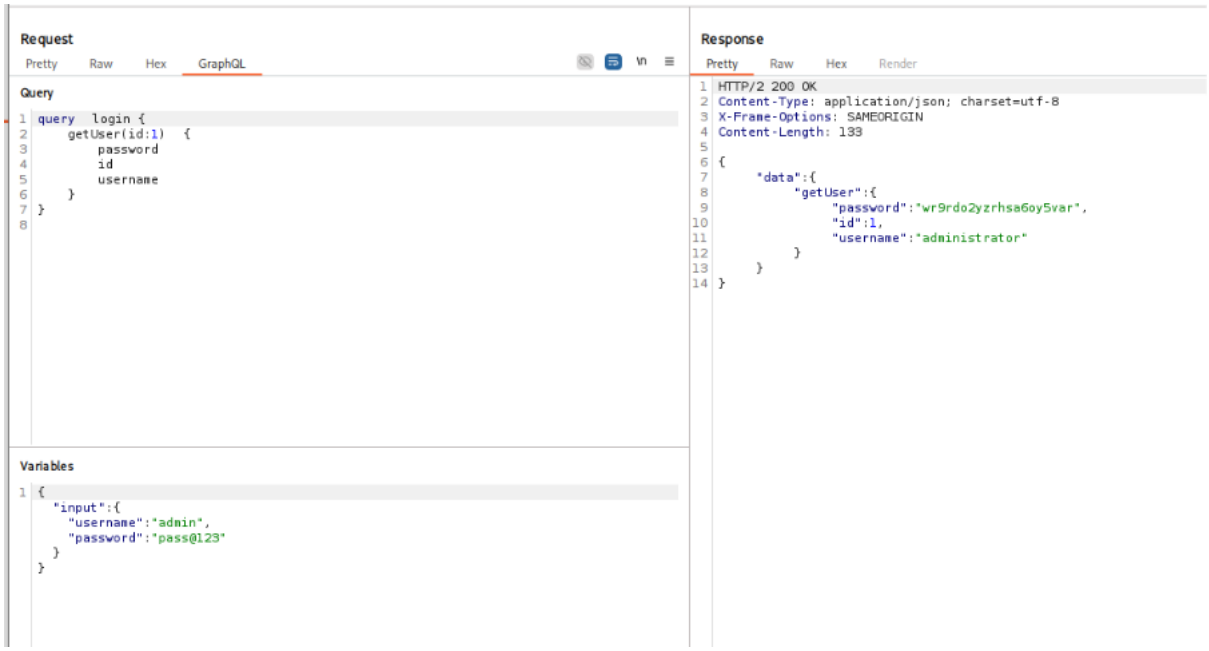
Request

Pretty Raw Hex GraphQL

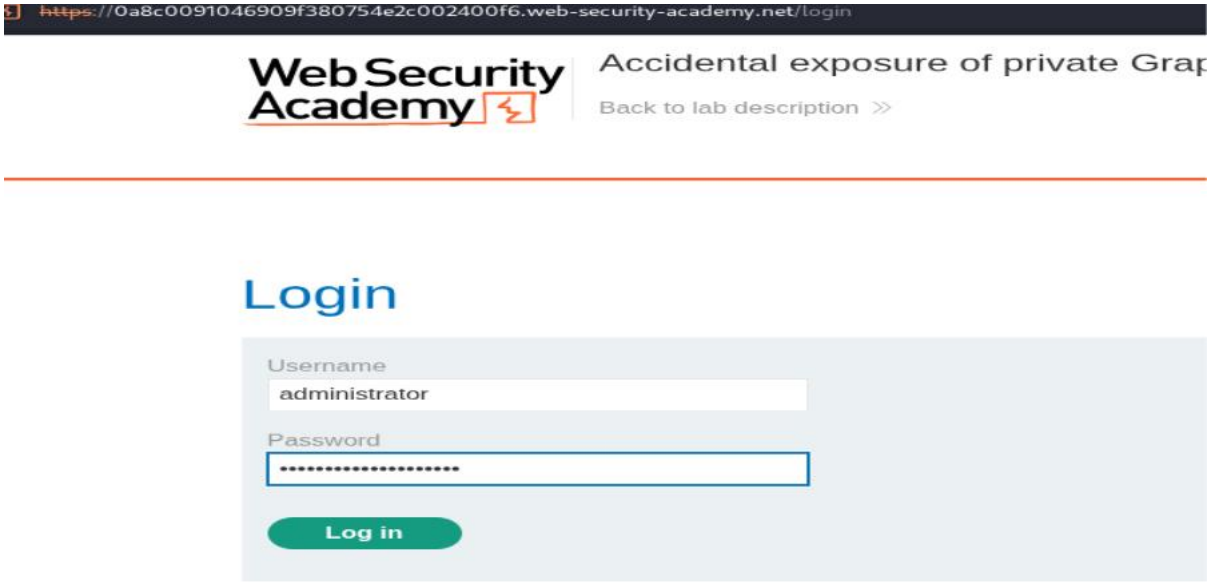
Query

```
query   {
     getUser(id:1332)    {
          password
          id
          username
     }
}
```

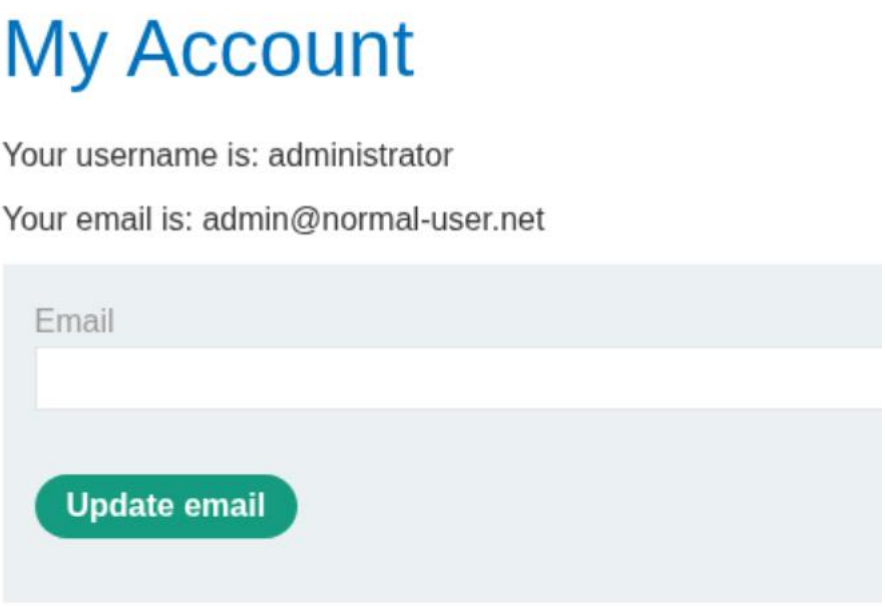
Using this code, we got the error and there is always the id number is 1 for administrator and change the id to the 1 and insert a “login” after the query and the result will the credentials.



In the above image, we got the credentials and use the credentials in the web page and login to the administartor.

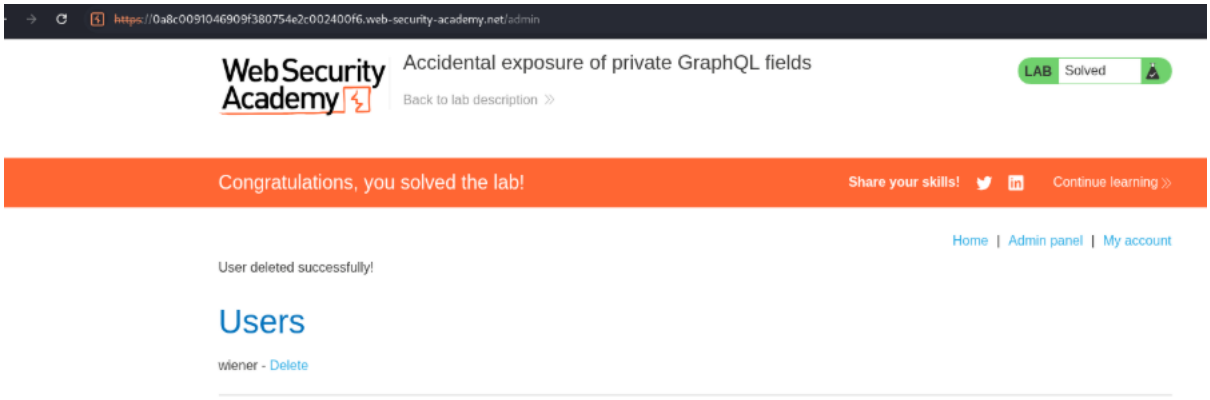


Here is the successful login to the account.



Navighate to the users and we found that how many users are there.

Then, delete the user carlos, then the lab is finished and solved.



TEST ID	VULNERABILITY	SEVERITY	TARGET ENDPOINT
1	Bola	Critical	/api/users
2	GraphicQL Injection	High	/graphql