

1. Api Security Testing

1.1 BOIA

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

Web Security Academy > Access control > Lab

Lab: Insecure direct object references

APPRENTICE

LAB

Not solved

This lab stores user chat logs directly on the server's file system, and retrieves them using static URLs. Solve the lab by finding the password for the user `carlos`, and logging into their account.

ACCESS THE LAB

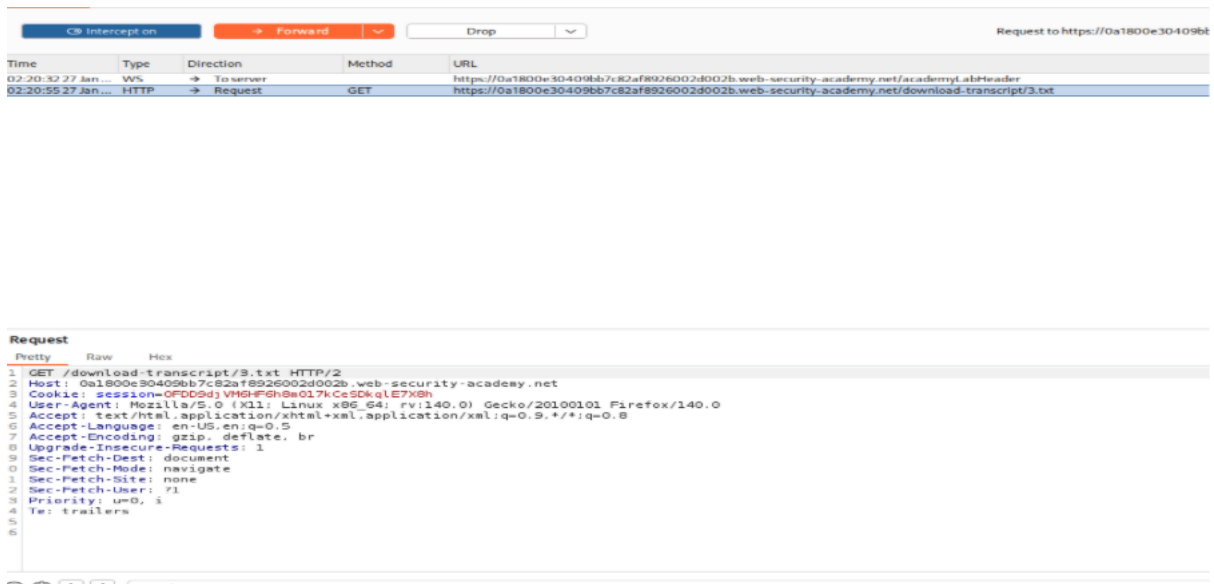
Solution

1. Select the **Live chat** tab.
2. Send a message and then select **View transcript**.
3. Review the URL and observe that the transcripts are text files assigned a filename containing an incrementing number.
4. Change the filename to `1.txt` and review the text. Notice a password within the chat transcript.
5. Return to the main lab page and log in using the stolen credentials.

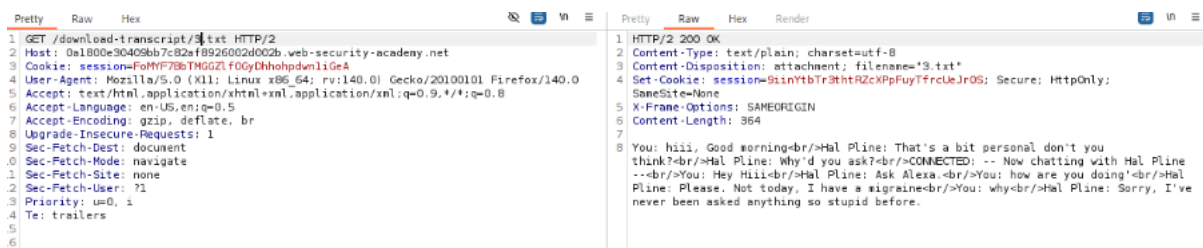
Community solutions

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

Login

Username

carlos

Password

.....

Log in

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



Insecure direct object references

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning >>](#)

[Home](#) | [My account](#) | [Live chat](#) | [Log out](#)


My Account

Your username is: carlos

1.2 GraphQL Injection

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

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



MY ACCOUNT

Products | Solutions | Research | Academy | Support

Dashboard | Learning paths | Latest topics | All content | Hall of Fame | Get started | Get certified

Web Security Academy > GraphQL API vulnerabilities > Lab

Lab: Accidental exposure of private GraphQL fields

PRACTITIONER LAB Not solved

The user management functions for this lab are powered by a GraphQL endpoint. The lab contains an access control vulnerability whereby you can induce the API to reveal user credential fields.

To solve the lab, sign in as the administrator and delete the username `carlos`.

Learn more about [Working with GraphQL in Burp Suite](#).

ACCESS THE LAB

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

Login

Username

Password

Capture the request and send the request to the repeater

Time	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

Request

Pretty Raw Hex GraphQL

```
POST /graphql/v1 HTTP/2
Host: 0a8c0091046909f380754e2c002400f6.web-security-academy.net
Cookie: session=5xf70TdUd6y5YnBXnD6vb9tn0PKFKC15
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 GraphQL to view the source code for credentials.

```
Request
Pretty Raw Hex GraphQL
1 POST /graphql/v1 HTTP/2
2 Host: 0a8c0091046909f380754e2c002400f6.web-security-academy.net
3 Cookie: session=5Xf70TdUd6y5YnBXnD6vb9tn0PKFKCi5
4 Content-Length: 202
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json
8 Sec-Ch-Ua: "Not_A Brand";v="99", "Chromium";v="142"
9 Content-Type: application/json
10 Sec-Ch-Ua-Mobile: ?0
11 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
12 Origin: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/login
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 {
  "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.

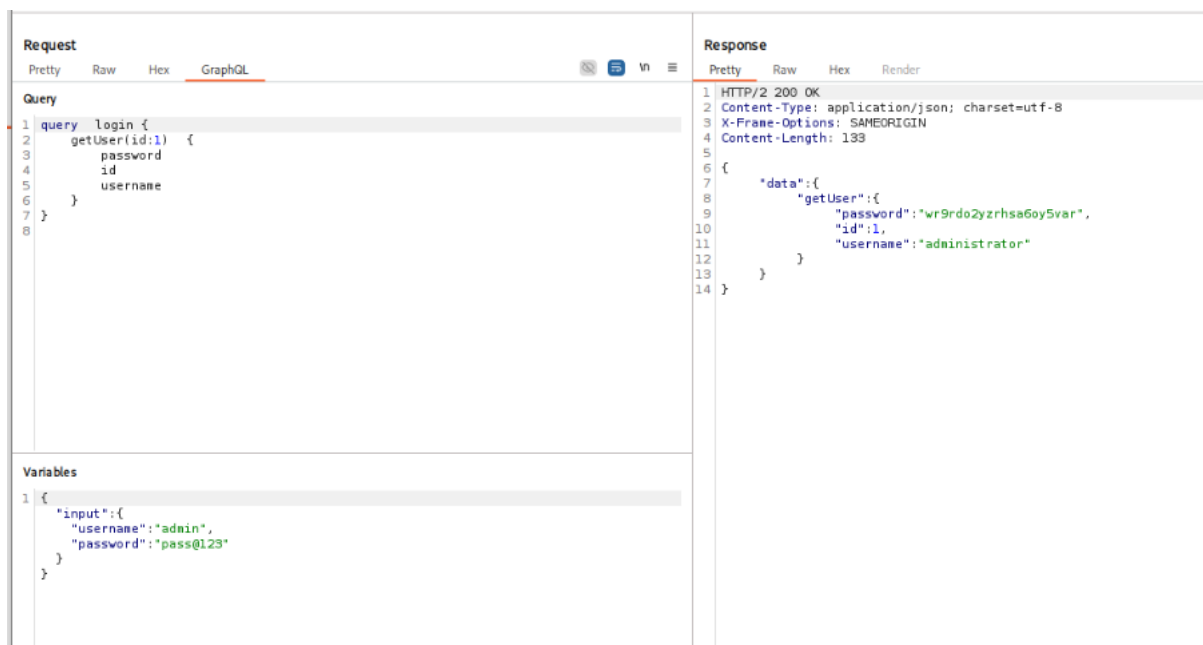
```
Request
Pretty Raw Hex GraphQL
Query
1 query {
2   getUser(id:1234) {
3     password
4     id
5     username
6   }
7 }
8

Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 108
5
6 {
7   "errors": [
8     {
9       "locations": [
10        ],
11       "message": "Unknown operation named 'login'."
12     }
13   ]
14 }
```


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.



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 be the credentials.



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

 [Accidental exposure of private GraphQL endpoint](#)
[Back to lab description >>](#)

Login

Username

administrator

Password

.....

Log in

Here is the successful login to the account.

My Account

Your username is: administrator

Your email is: admin@normal-user.net

Email

Update email

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

Users

wiener - [Delete](#)
carlos - [Delete](#)

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

→

https://0a8c0091046909f380754e2c002400f6.web-security-academy.net/admin

WebSecurity Academy

Accidental exposure of private GraphQL fields

Back to lab description >>

LAB Solved

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning >>](#)

[Home](#) | [Admin panel](#) | [My account](#)

User deleted successfully!

Users

wiener - [Delete](#)

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