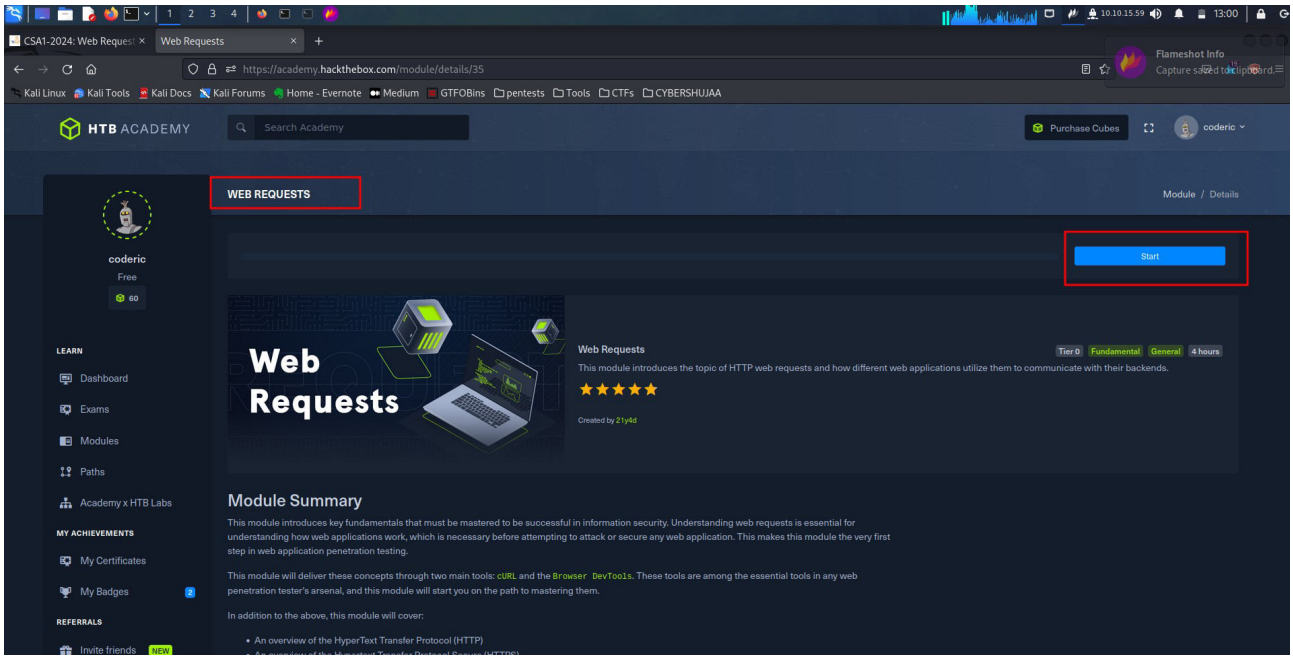
Eric Mwenda

Web Requests

https://academy.hackthebox.com/achievement/596337/35

Introduction to Web Requests.



HyperText Transfer Protocol (HTTP)

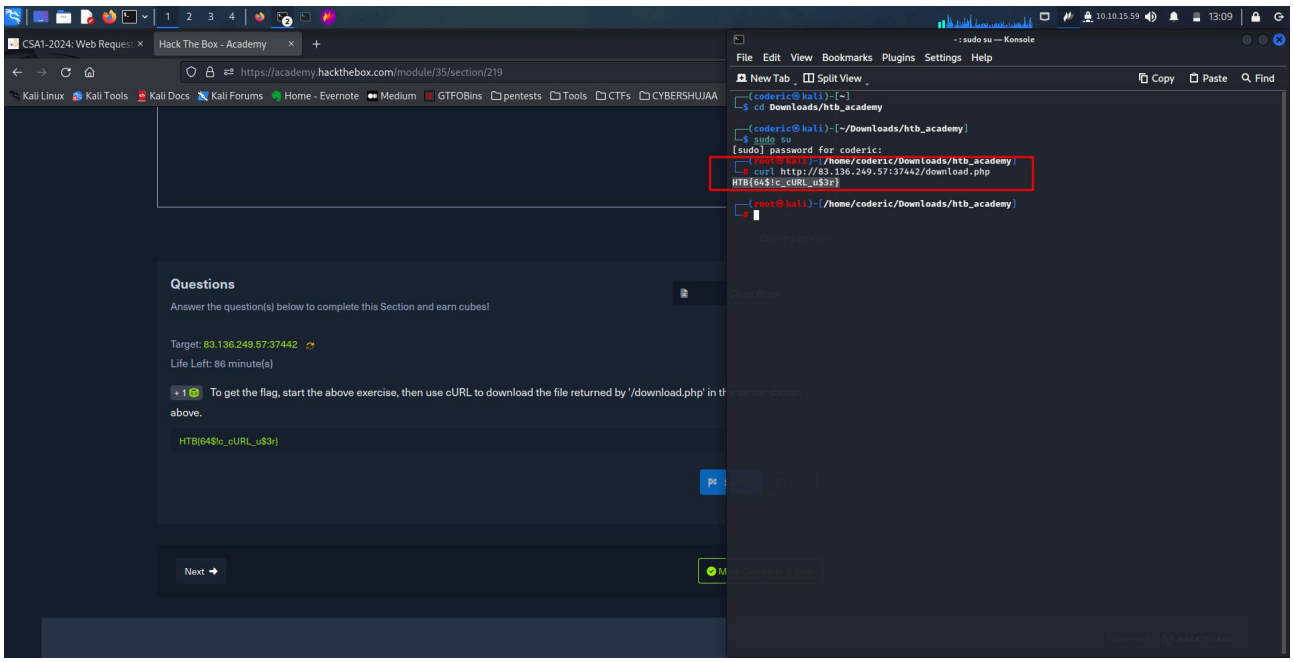
In this module it begins with explaining the HTTP protocol, which is an application-level protocol used to access the World Wide Web resources.
The default port for HTTP communication is port 80.

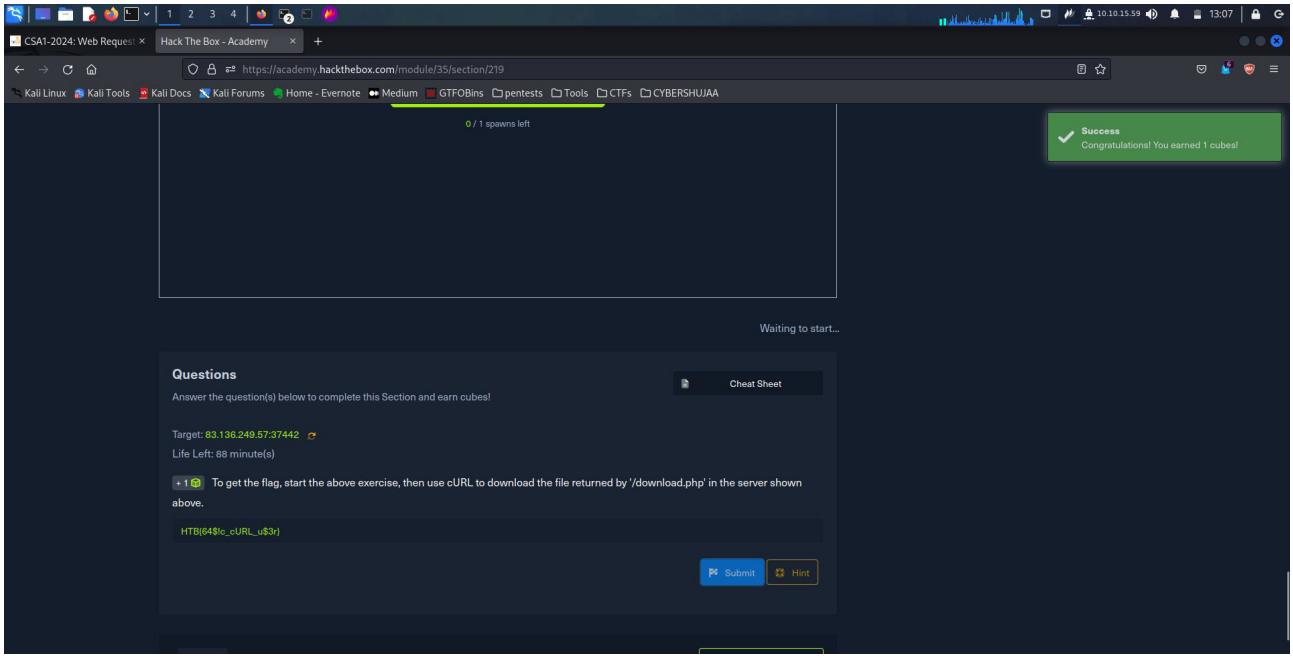
URL

Resources over HTTP are accessed via a URL, which offers many more specifications than simply specifying a website we want to visit.

Question

To get the flag, start the above exercise, then use cURL to download the file returned by '/download.php' in the server shown above. ANS: **HTB{64$!c_cURL_u$3r}**
First I opened the VPN file using my Local Virtual Machine using command:- sudo openvpn academy-regular.ovpn. Then using Command curl http://83.136.249.57:37442/download.php, I got the flag. cURL (client URL) is a command-line tool and library that primarily supports HTTP along with many other protocols.





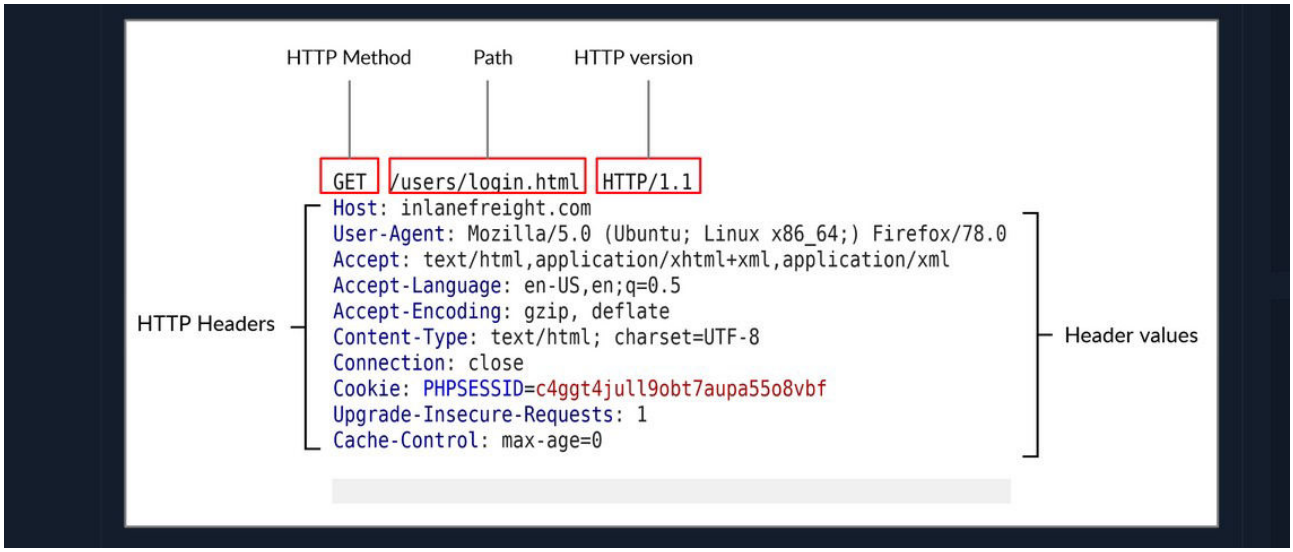
Hypertext Transfer Protocol Secure (HTTPS)

```
     7 4.573774918   192.168.0.108 192.168.0.108   TCP        76 40386 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1
     8 4.573794134   192.168.0.108 192.168.0.108   TCP        76 80 → 40386 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 S
     9 4.573806187   192.168.0.108 192.168.0.108   TCP        68 40386 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=280780439
    10 4.573966701   192.168.0.108 192.168.0.108   HTTP      640 POST /login.php HTTP/1.1  (application/x-www-form-urlencoded)
    11 4.573985767   192.168.0.108 192.168.0.108   TCP        68 80 → 40386 [ACK] Seq=1 Ack=573 Win=65024 Len=0 TSval=28078043
Frame 10: 640 bytes on wire (5120 bits), 640 bytes captured (5120 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.0.108, Dst: 192.168.0.108
Transmission Control Protocol, Src Port: 40386, Dst Port: 80, Seq: 1, Ack: 1, Len: 572
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
 ▾ Form item: "username" = "admin"
     Key: username
     Value: admin
 ▾ Form item: "password" = "password"
     Key: password
     Value: password
```

In HTTPS all communications are transferred in an encrypted format, and by this even if a third party is able to intercept the request, they would not be able to extract the data out of it.

For this reason, HTTPS has become the mainstream scheme for websites on the internet. HTTPS is more secure than HTTP.

HTTP Requests and Responses

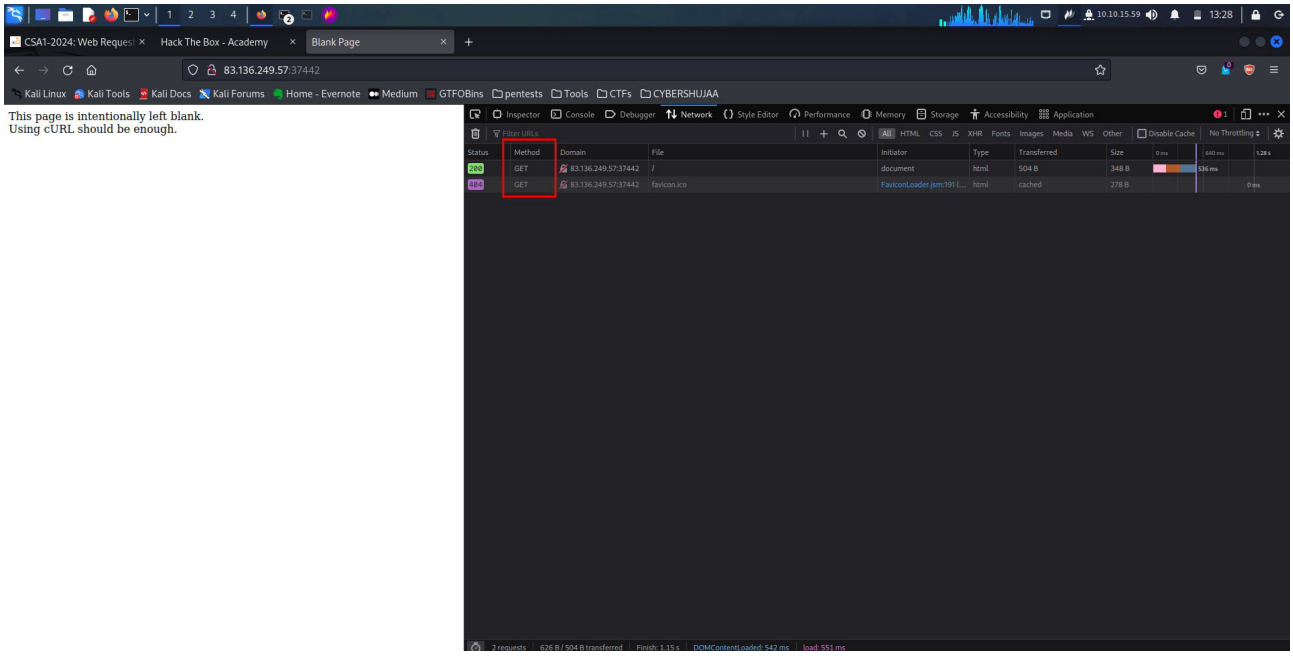


HTTP Requests are made by the client which are then processed by the server. This requests have all details a client requires from the server.

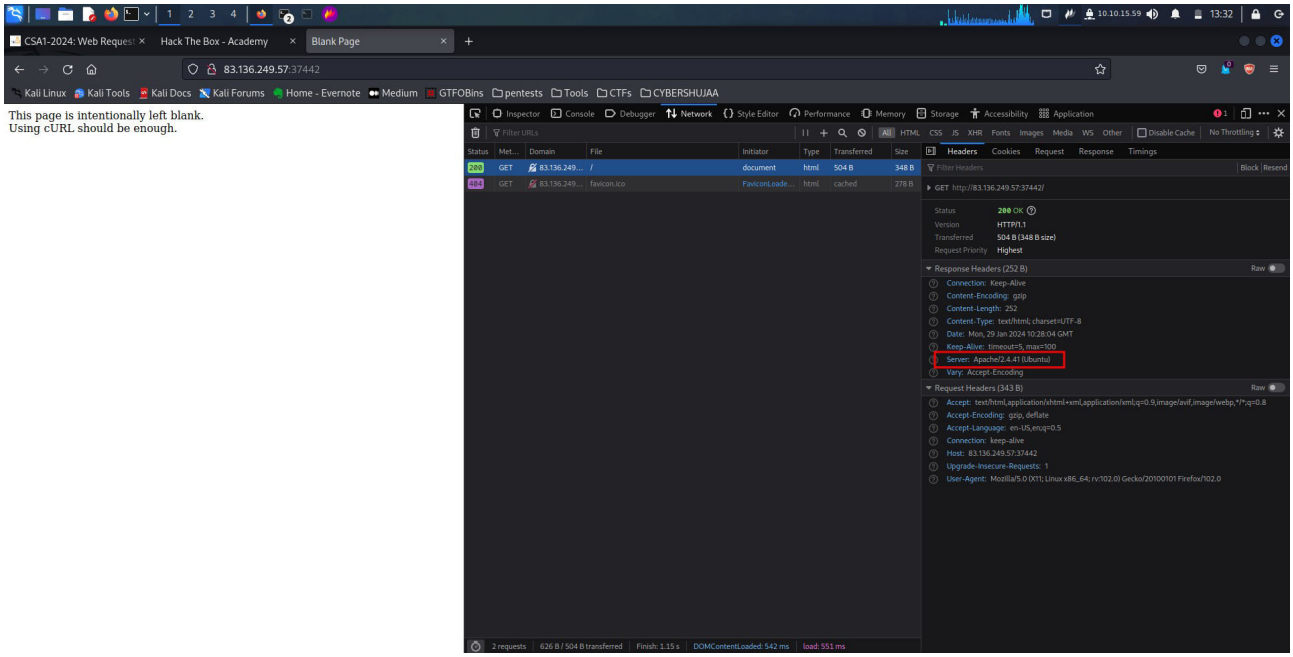
HTTP Responses are the processed response to a request containing the response code. Response codes are used to determine the request's status.

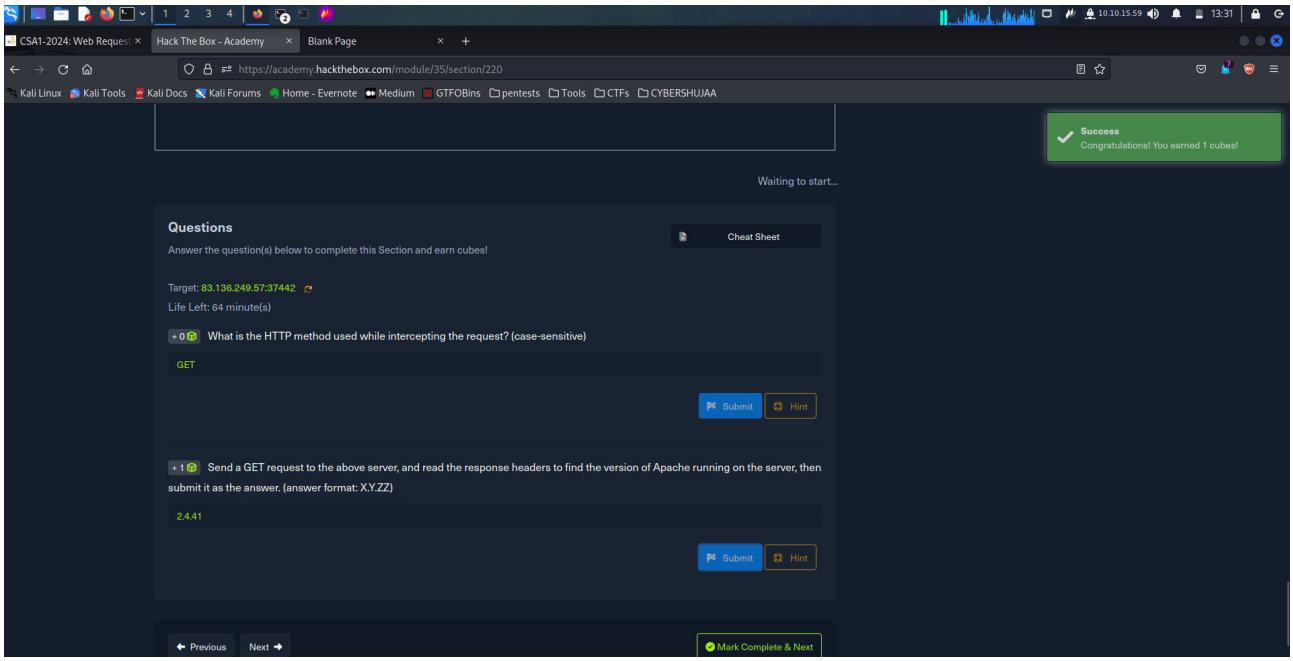
Question

What is the HTTP method used while intercepting the request? (case-sensitive) ANS: **GET**



Question
Send a GET request to the above server, and read the response headers to find the version of Apache running on the server, then submit it as the answer. (answer format: X.Y.ZZ) ANS: **2.4.41**





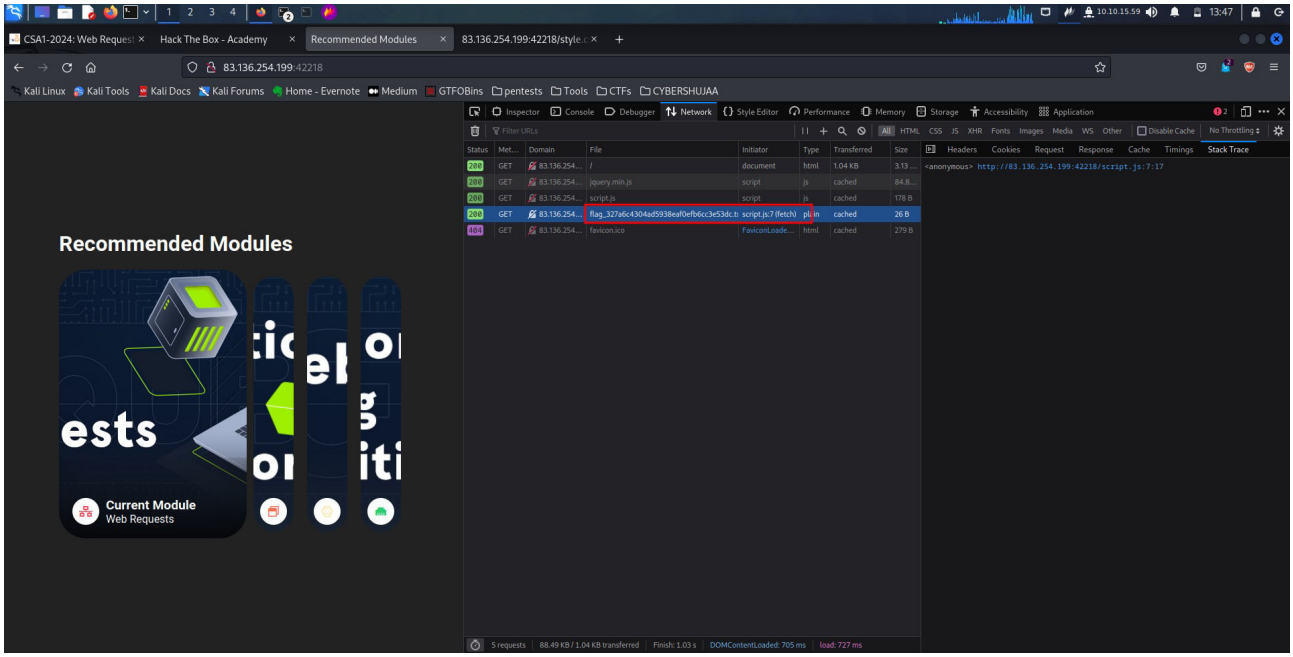
HTTP Headers.

Some HTTP headers are used to pass information between the client and the server, other headers are only used with either requests or responses, while some other general headers are common to both.

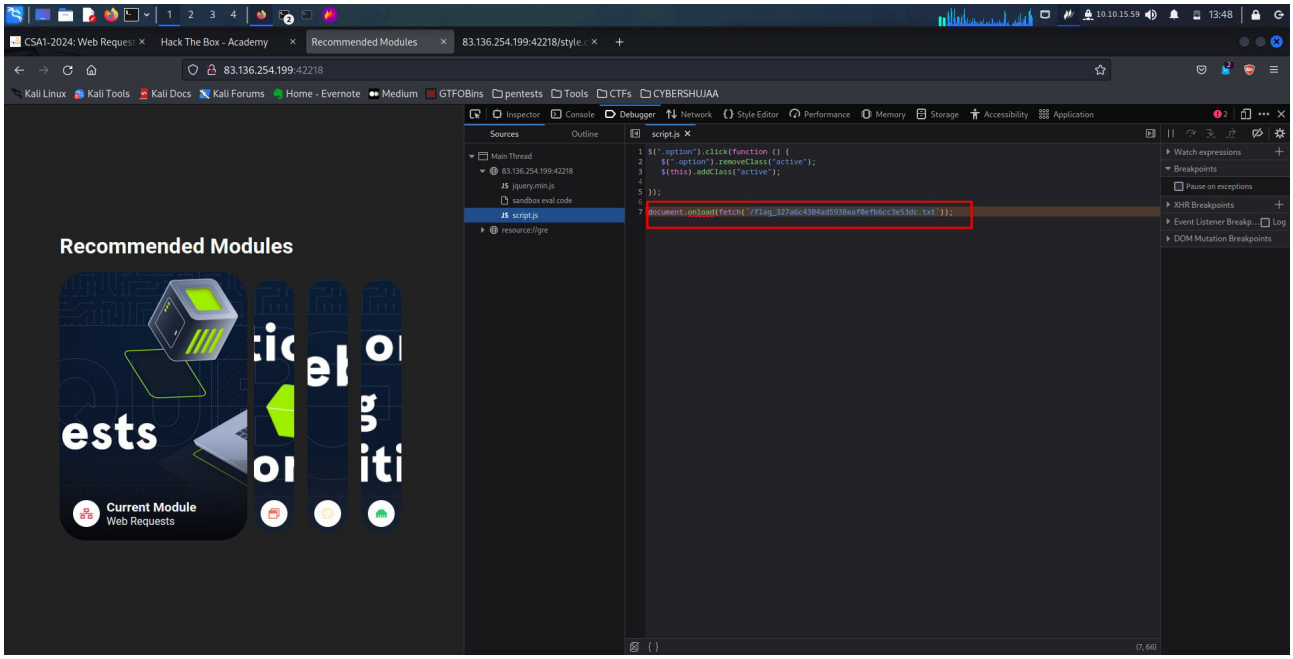
Question

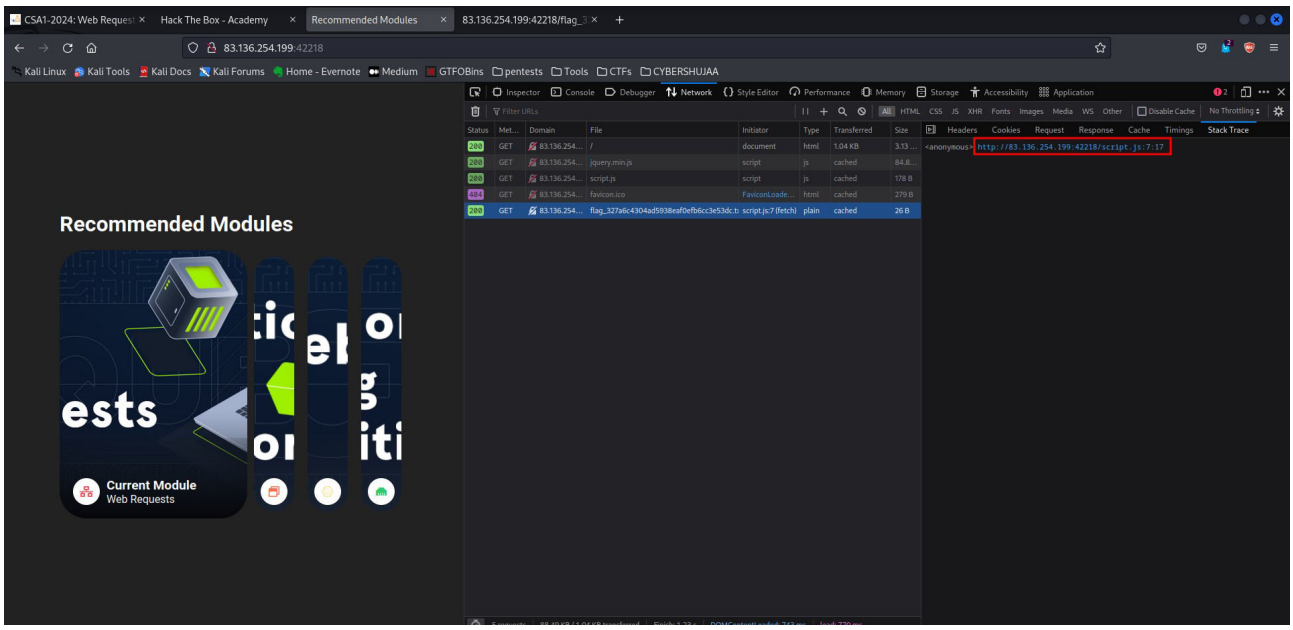
The server above loads the flag after the page is loaded. Use the Network tab in the browser devtools to see what requests are made by the page, and find the request to the flag. ANS: **HTB{p493_r3qu3t_m0n!t0r}**

First I opened the IP address provided on my browser then opened the devtools / Inspect(Q) option to take a look at the network tab.



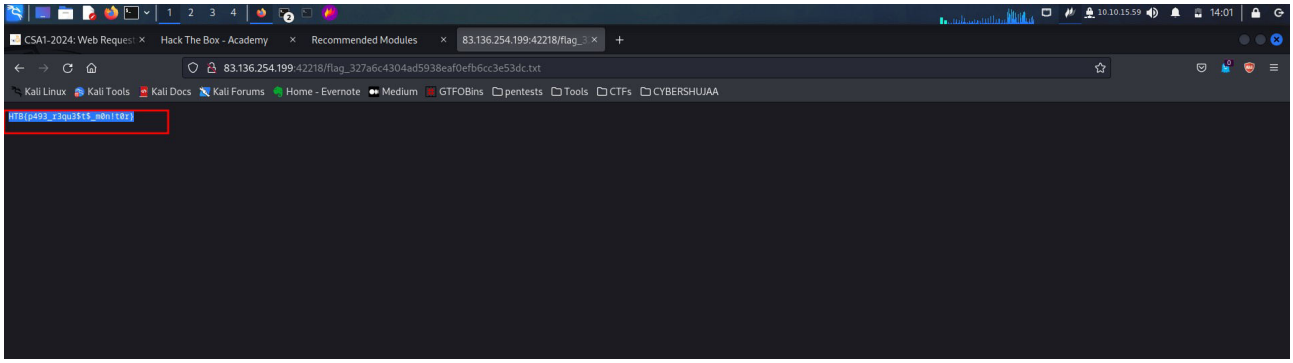
On looking at the network tab as instructed I was able to see a plain-text file that seems to be appealing, On the left side there is also a link that is called anonymous, for me this was quite interesting so I had to take a look at the link. On clicking a js code, there was another reference to a directory which I noticed it was storing a .txt file.

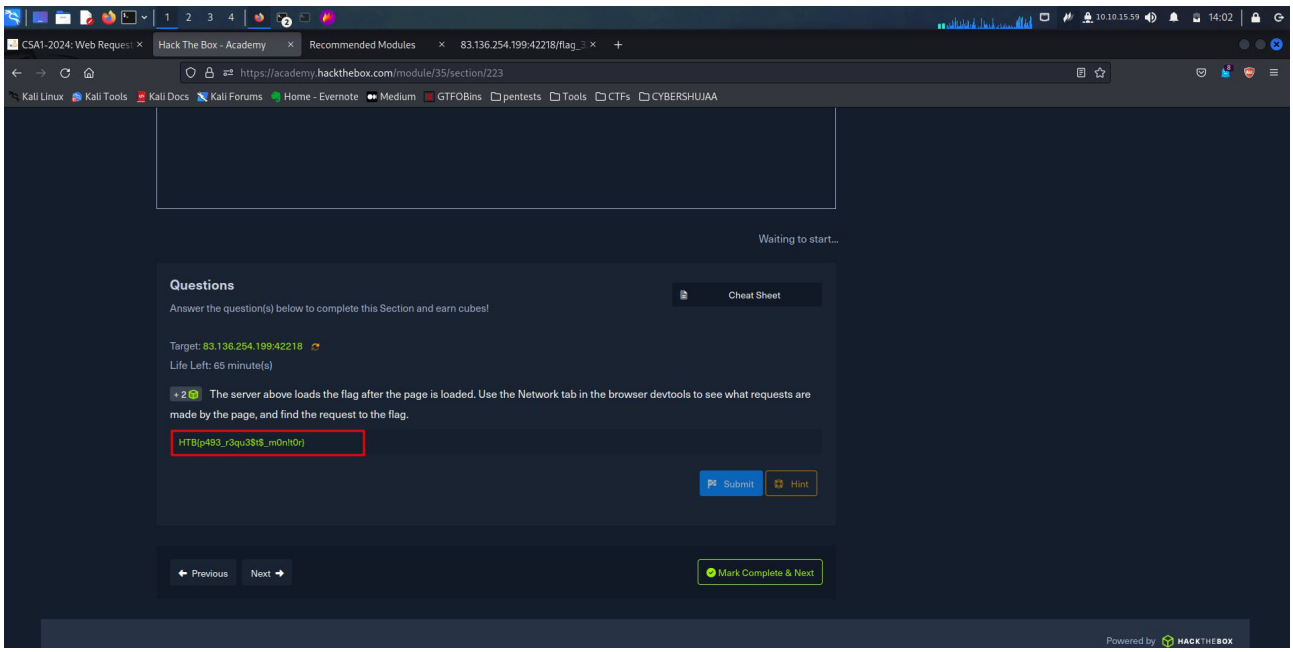




With the knowledge of having an anonymous link with a directory specified in the link's js code I decided to join this two information gathered and see what I could find.
http://83.136.254.199:42218/ - The address provided.
/flag_327a6c4304ad5938eaf0efb6cc3e53dc.txt – The directory specified in the js code.
Combination of the two:-
http://83.136.254.199:42218/flag_327a6c4304ad5938eaf0efb6cc3e53dc.txt

Well I was right. Here is the flag.

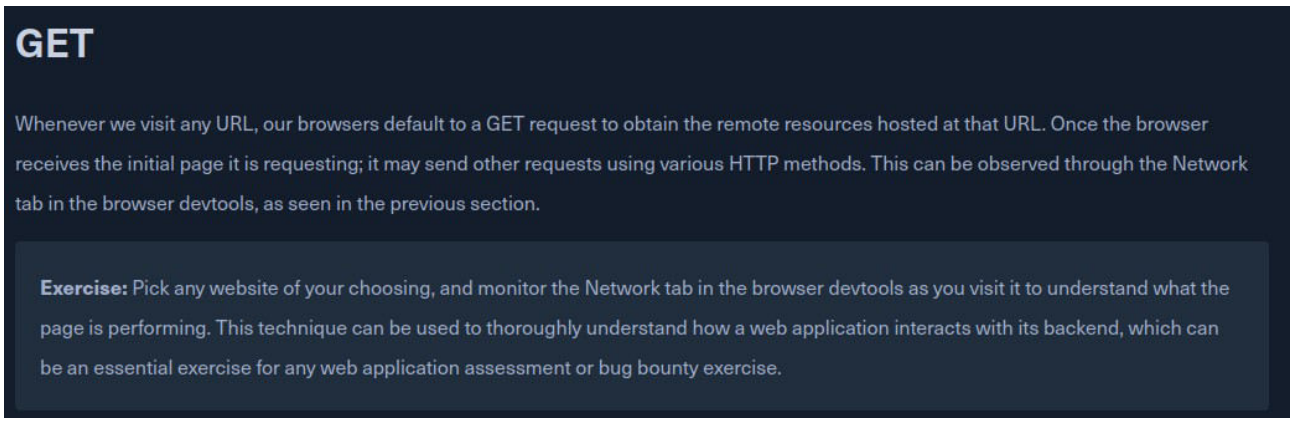




HTTP Methods and Codes

I was able to learn from this section that HTTP supports multiple methods for accessing a resource. Also learnt that most modern web applications mainly rely on the GET and POST but also any web application that utilizes REST APIs also rely on PUT and DELETE, to update and delete data on the API endpoint.
Commonly used request methods are:- GET, POST, HEAD, PUT, PATCH, OPTIONS and DELETE.
Commonly used response code in HTTP are:- 200 OK, 302 FOUND, 403 FORBIDDEN, 404 NOT FOUND etc.
Response codes used in HTTP server are:- 1xx, 2xx, 3xx, 4xx and 5xx.

GET

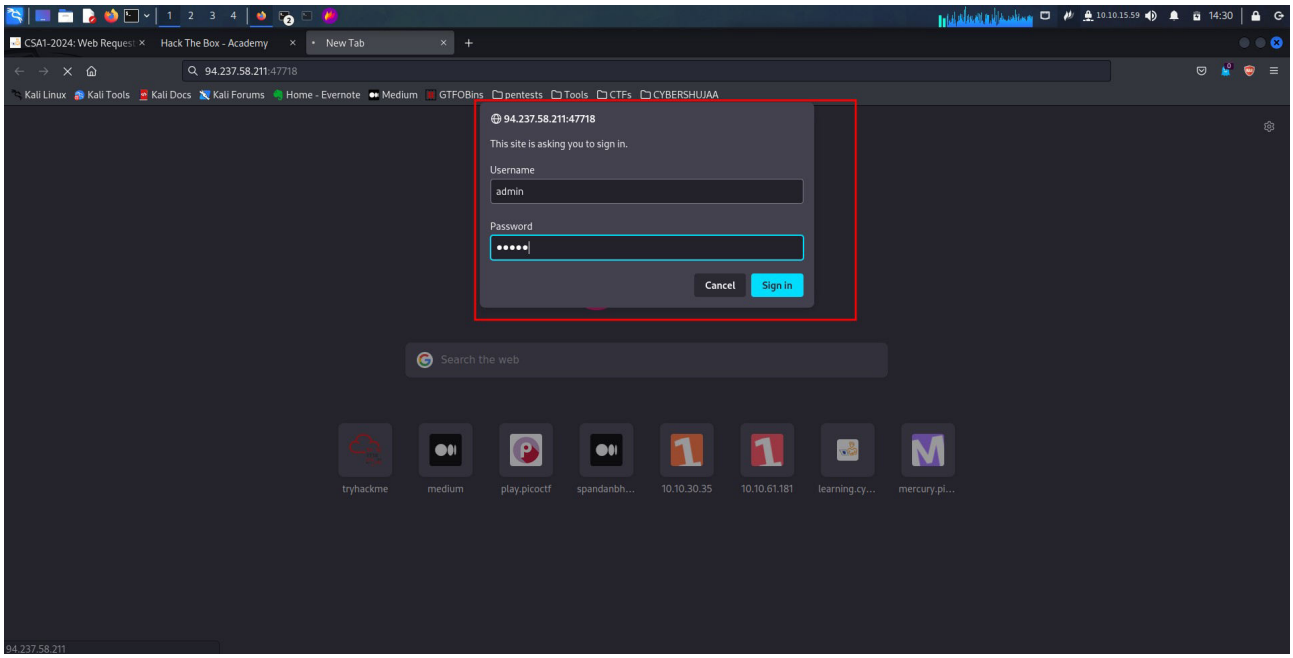


I am able to understand that whenever I visit any URL in my browsers, by default they send GET request to obtain the remote resources hosted at that URL.

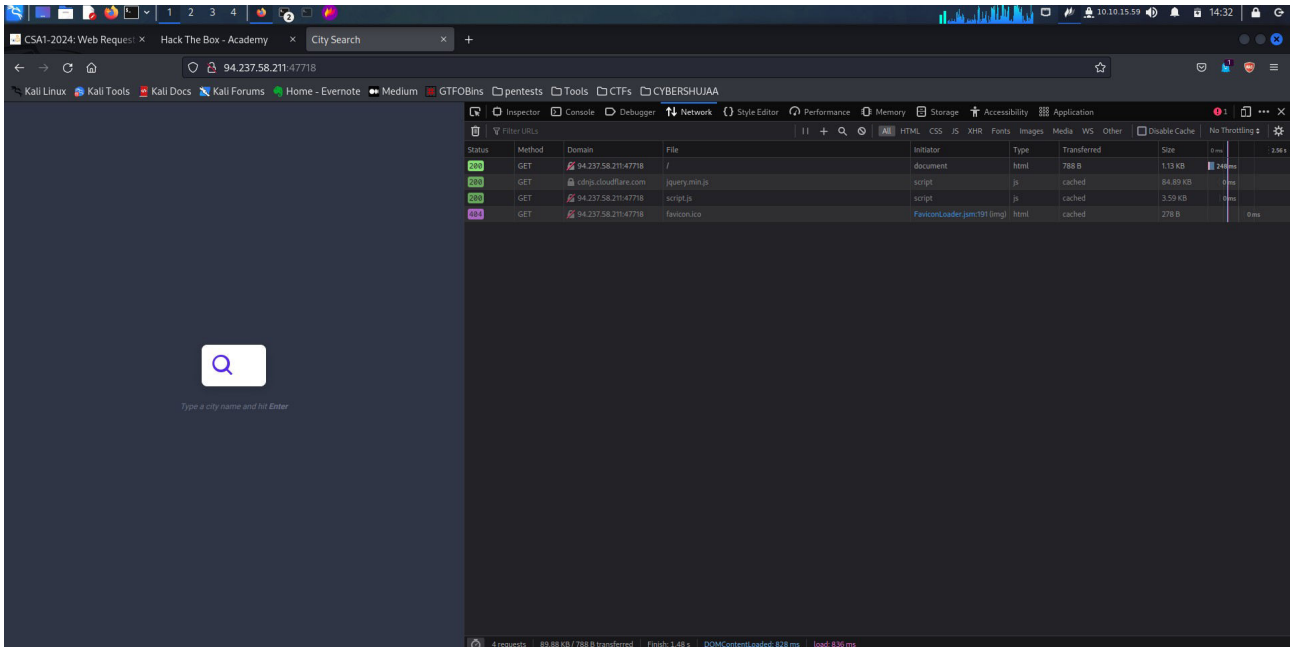
Question

he exercise above seems to be broken, as it returns incorrect results. Use the browser devtools to see what is the request it is sending when we search, and use cURL to search for 'flag' and obtain the flag. ANS: **HTB{curl_g3773r}**

Once I had the IP address for my target I opened it in my browser which prompted a username and password, good enough from the question this information is provided for both as “admin”.

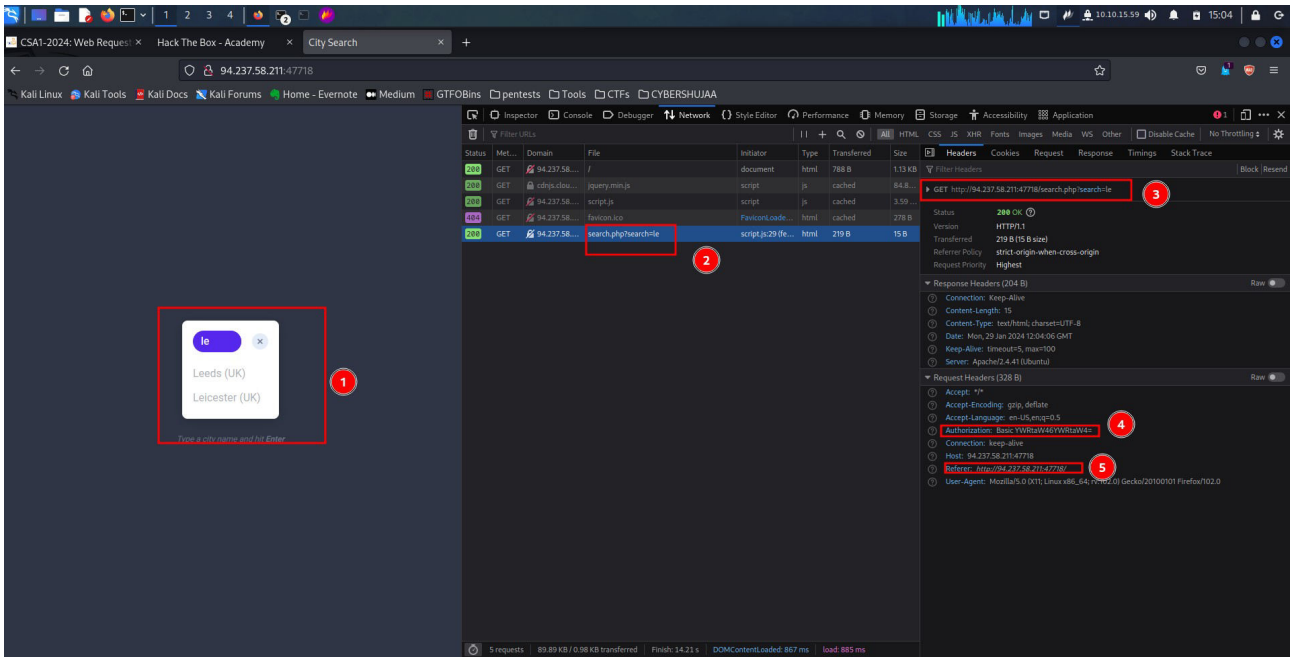


I then took a look at the network tab

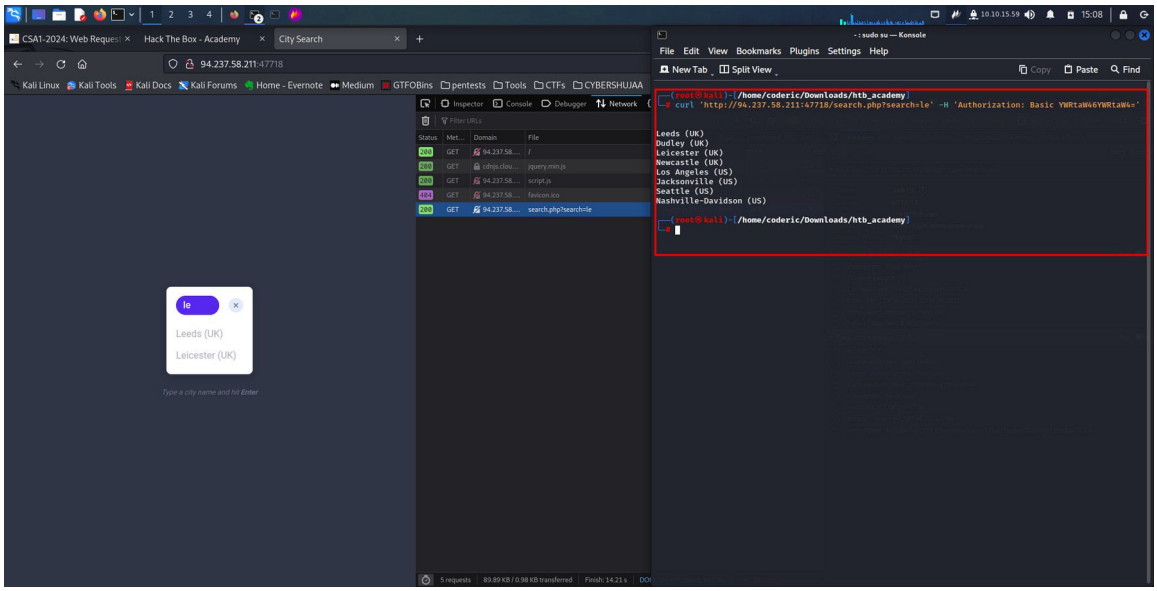


On the search bar I searched for a random search but I prefer using “le” as in the example notes, which returns an HTTP response code 200 meaning the search was sent successful. With it there is an attaches search.php. I therefore clicked the link to find more information about the search.php file.

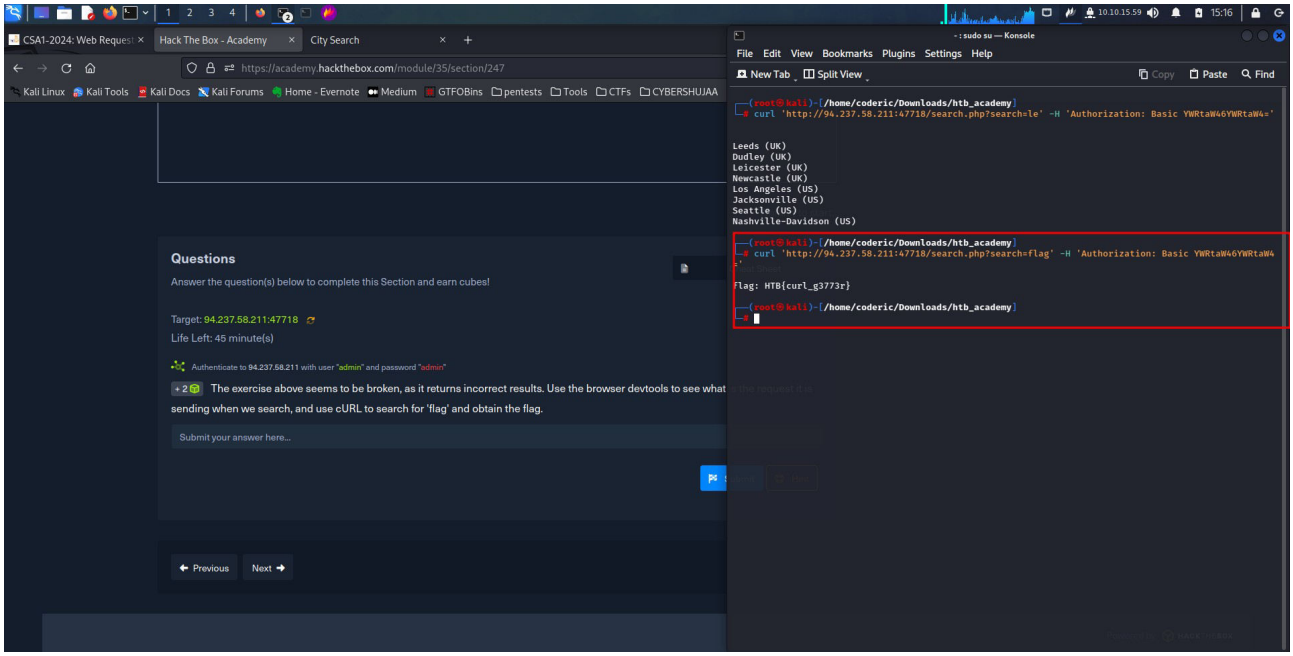
On clicking the search.php, with it I find the GET parameter search=le used in the URL. From the notes section we are told that the search function (search=le) requests another page for the results.



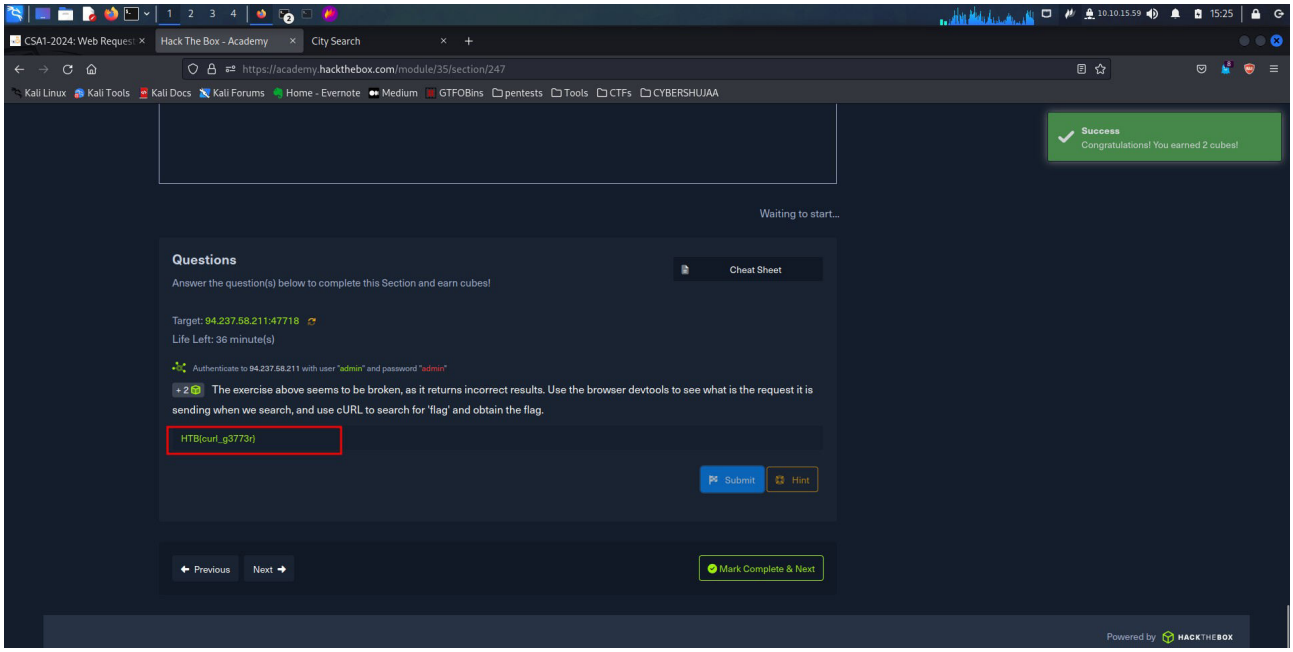
We are advised to use Curl, to send GET Requests, so I sent a Get request hoping to get useful information. Command used:-
curl 'http://http://94.237.58.211:47718/search.php?search=le' -H 'Authorization: Basic YWRtaW46YWRtaW4='



It seems what this GET Request does, it returns all values with keyword “le” .
This was promising therefore I made a search using keyword flag, and there was the flag.



Command used:- curl 'http://http://94.237.58.211:47718/search.php?search=flag' -H 'Authorization: Basic YWRtaW46YWRtaW4='



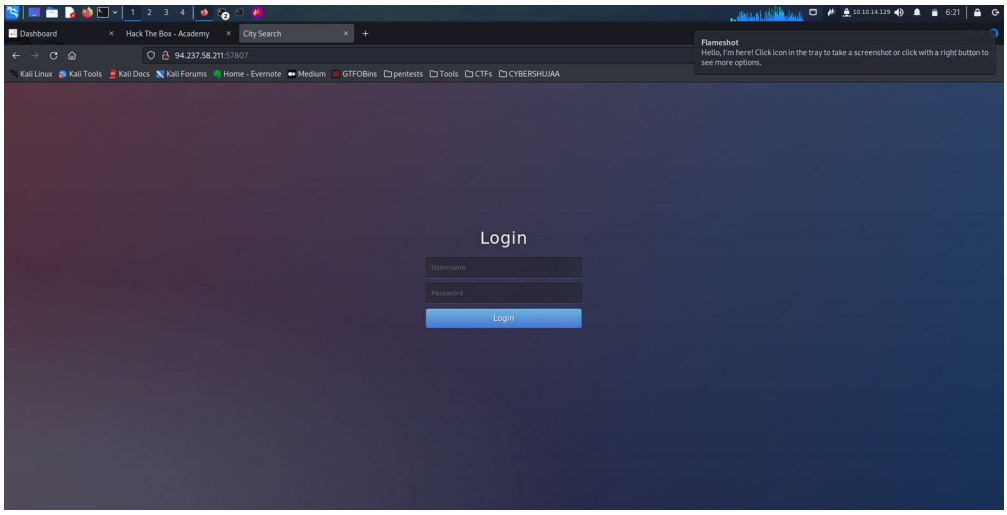
POST

POST is utilized in web applications where transfer of files or moving the user parameters from the URL is needed.
POST has 3 main benefits:-
1. Lesser Encoding Requirements.
2. Lack of Logging.
3. More data can be sent.

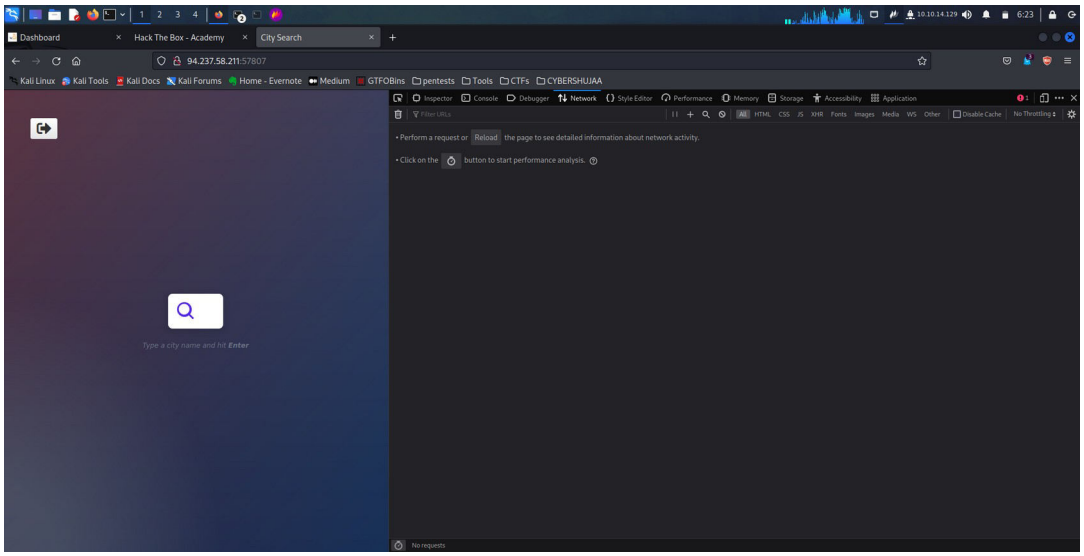
Question

Obtain a session cookie through a valid login, and then use the cookie with cURL to search for the flag through a JSON POST request to '/search.php' ANS: **HTB{p0$t_r3p34t3r}**

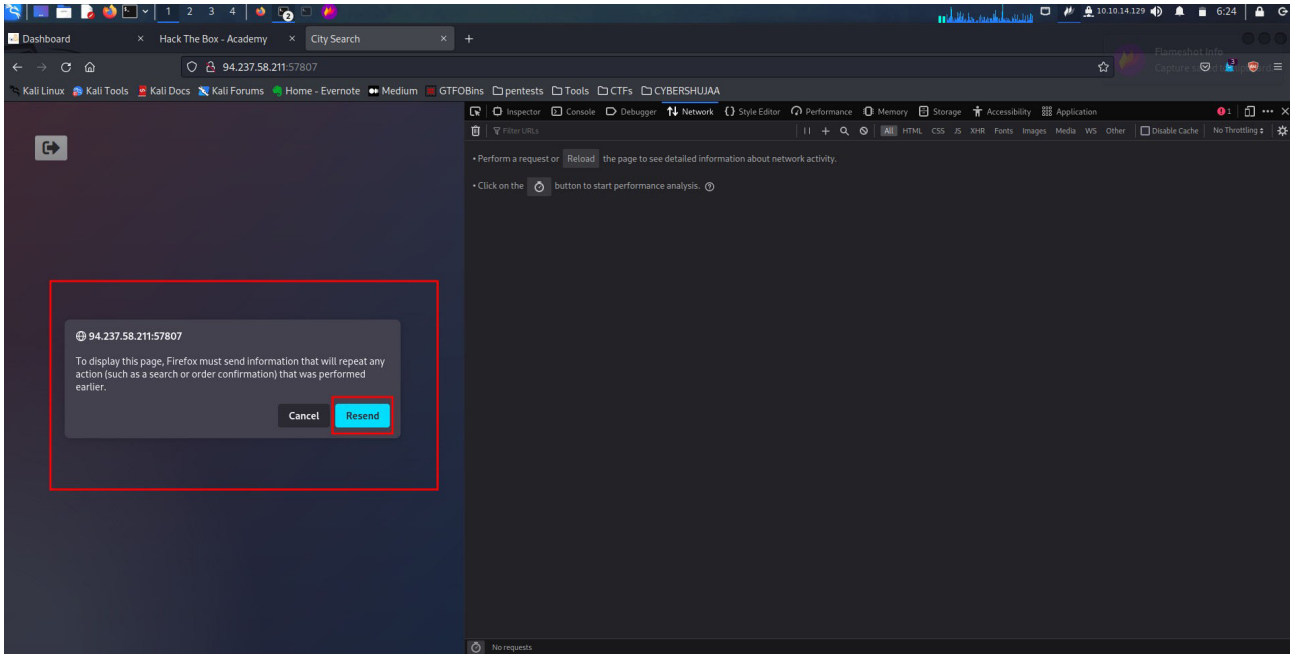
On receiving my target IP address I opened it in my browser which lead me to a login page.
Username: admin
Password: admin



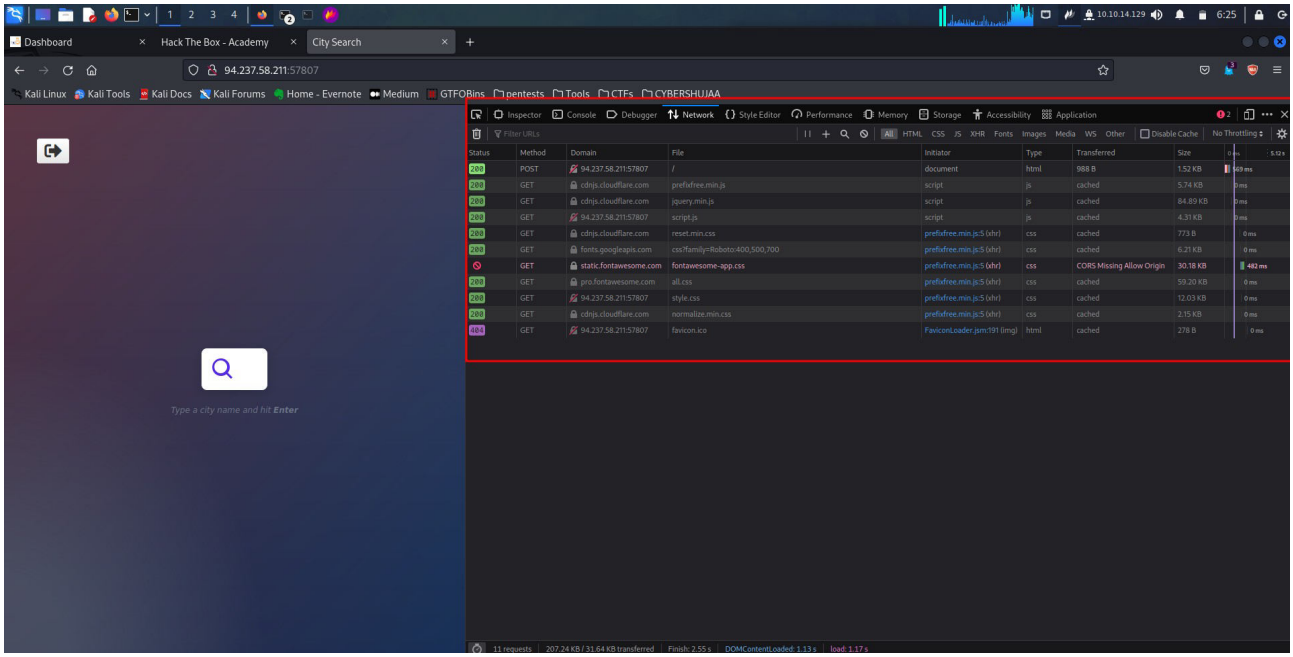
On looking at the devtools on the first login I was not able to see the requests being sent therefore I refreshed the city page once more to view requests being sent.



Refreshing my browser to view the requests being sent.



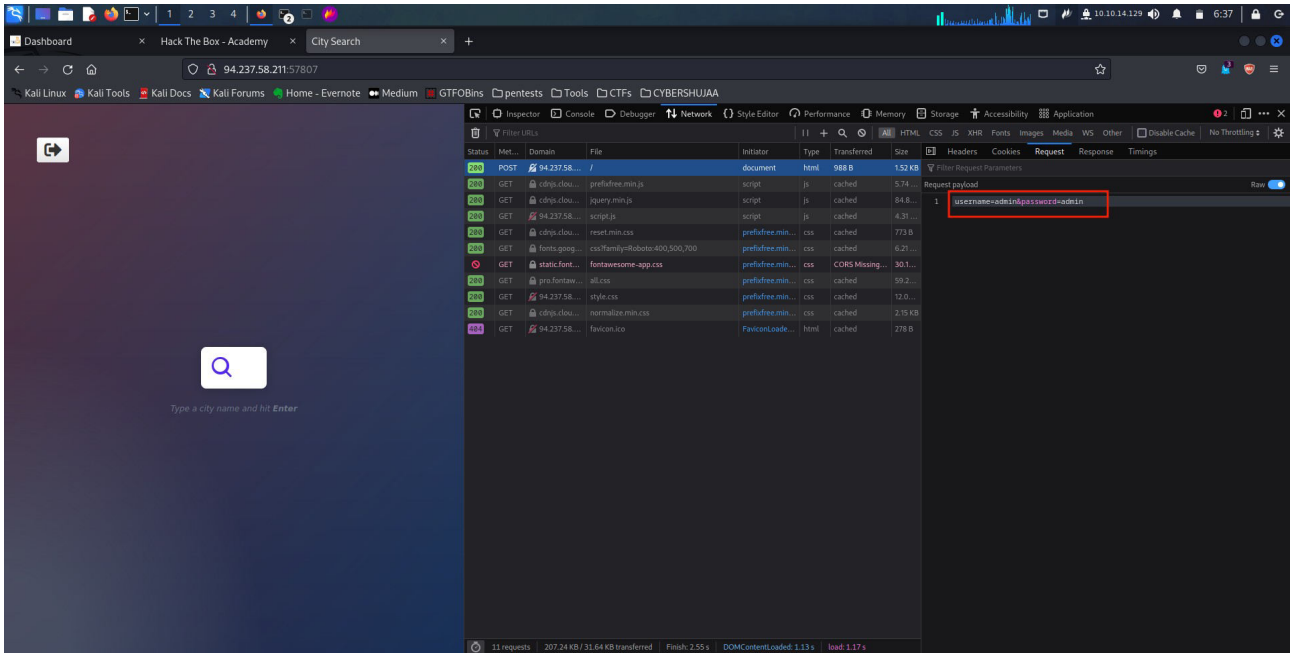
Well with a refresh to the city page here are the sent requests.



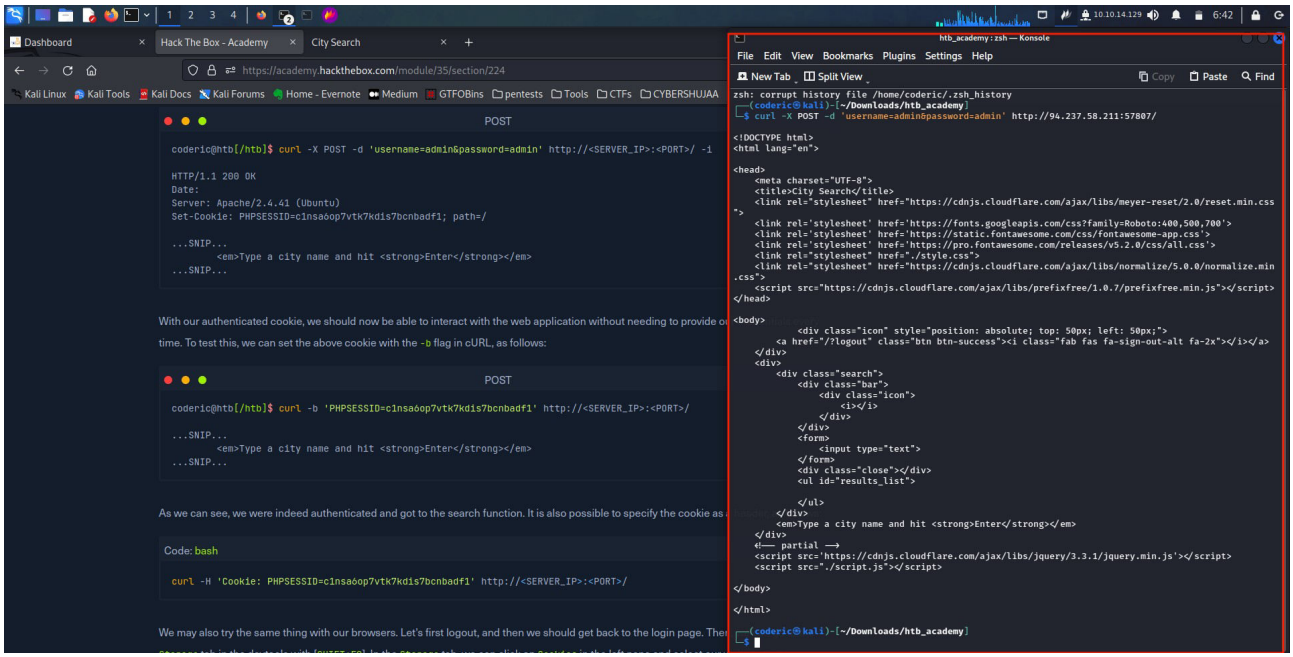
One of the requests is sent in POST method so I clicked on it to see more information about the method.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	94.237.58.211:57807	/	document	html	988 B	1.52 KB

I then clicked on the Request tab then clicked on the Raw button to show the raw request data. The following data is being sent as the POST request data:



With the request data at hand, I tried to send a similar request with cURL, to see whether this would allow us to login as well.
I used the -X POST flag to send a POST request and -d flag to add our POST data \ above data.
Command used:- curl -X POST -d 'username=admin&password=admin' http://94.237.58.211:57807/



The above data represents the HTML code for the login page. It also shows we had a successful login.

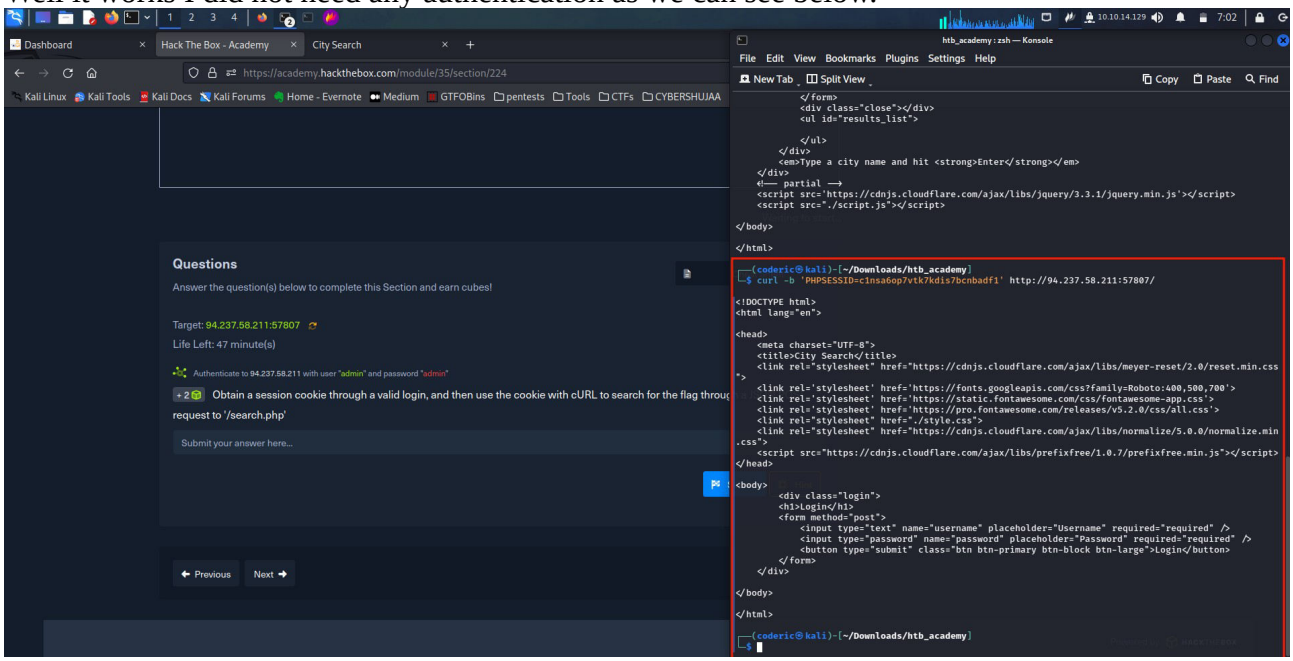
Since we have been successfully authenticated, a cookie also is sent in our browser to persist our authentication so that we don't need to login every time we visit the page.
We can use the -v or -i flags to view the response, which should contain the Set-Cookie header with our authenticated cookie:
Command used:- curl -X POST -d 'username=admin&password=admin' http://94.237.58.211:57807/ -i
Here is the response given with a Set-Cookie header in the authentication code.

Set Cookie= 'PHPSESSID=c1nsa6op7vtk7kdis7bcnbadf1

With our authenticated cookie, we should now be able to interact with the web application without needing to provide our credentials every time. To test this, we can set the above cookie with the -b flag in cURL.

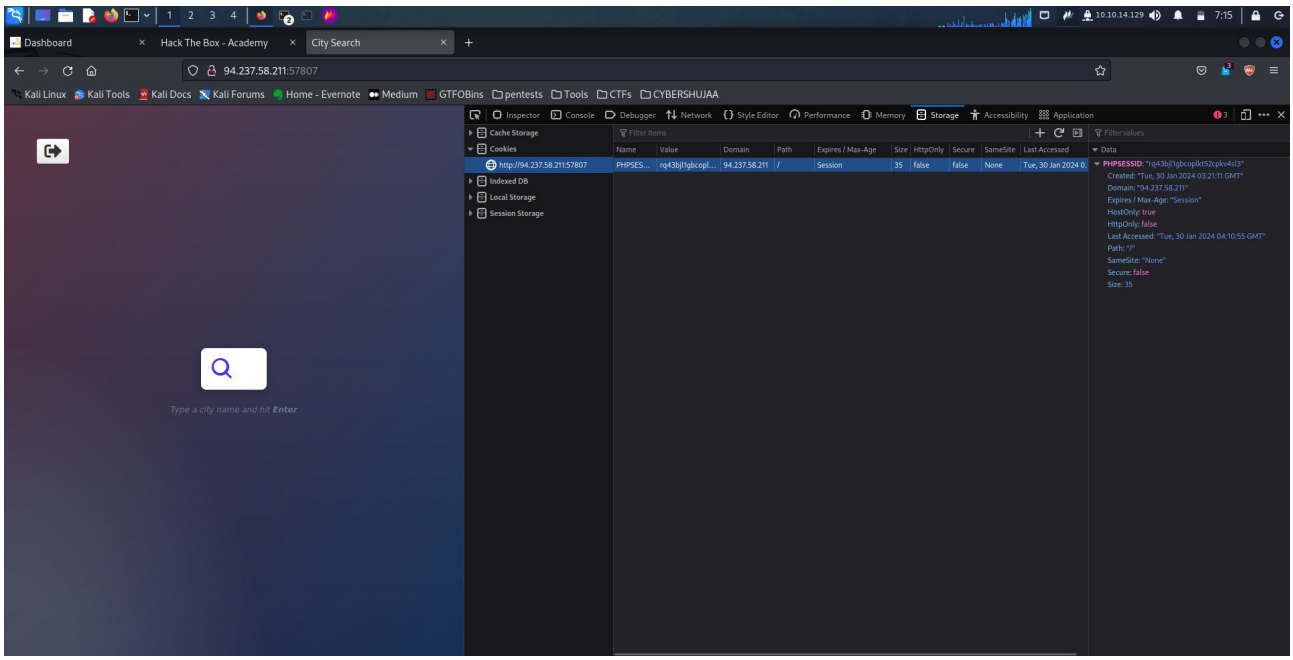
Command used:- curl -b 'PHPSESSID=c1nsa6op7vtk7kdis7bcnbadf1' http://94.237.58.211:57807/

Well it works I did not need any authentication as we can see below.

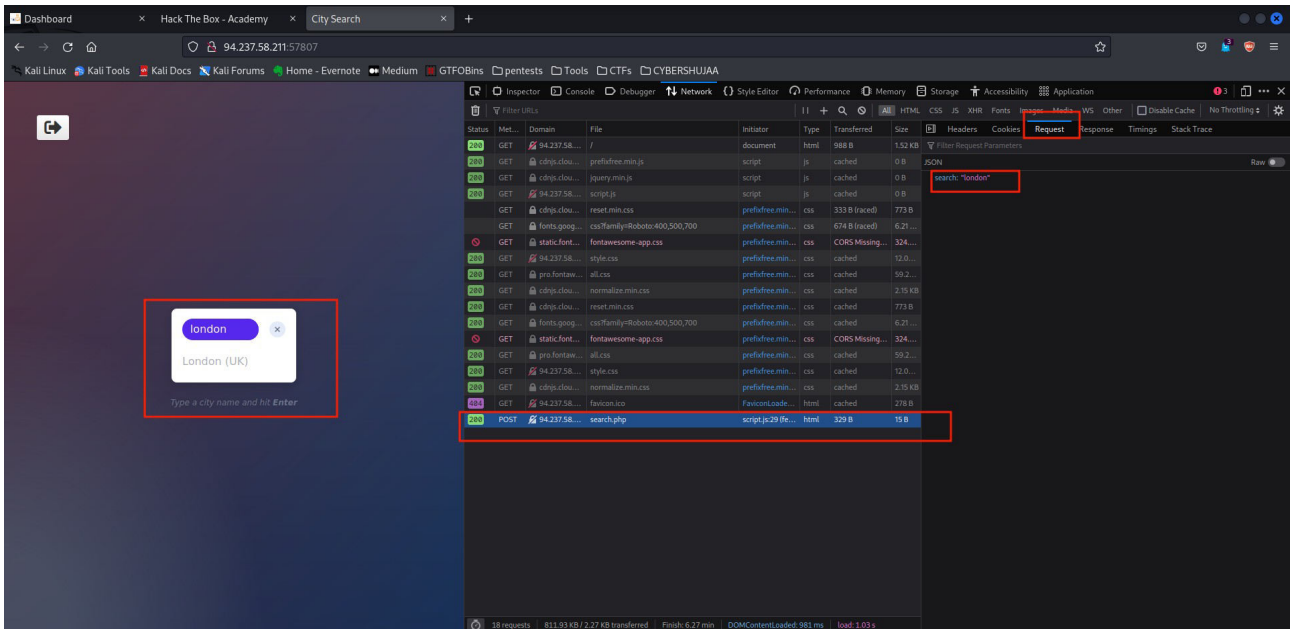


Now having this knowledge, I will then tried logging out and see if I could login without needing authentication credentials.
To do so, I simply am required to replace the cookie value with our own on the login page then refresh the page.
My Set Cookie = PHPSESSID=c1nsa6op7vtk7kdis7bcnbadf1

Here is a direct login without credentials by just setting the cookie.



Next I interacted with the City Search function to see what requests get sent. On the network tab we can see, the search form sends a POST request to search.php, with “ **{"search":"london"}** ” data.



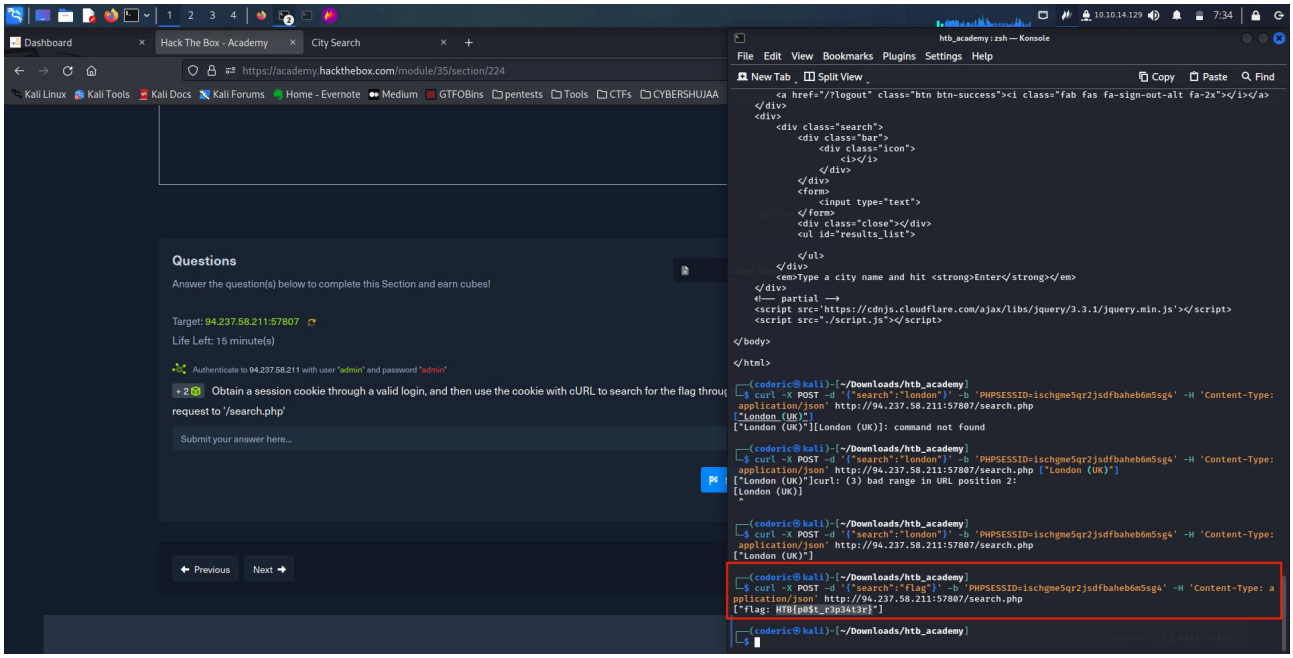
With the knowledge data gathered to this point I then tried to replicate this request as I had done earlier, but this time include both the cookie and content-type headers, and send our request to search.php

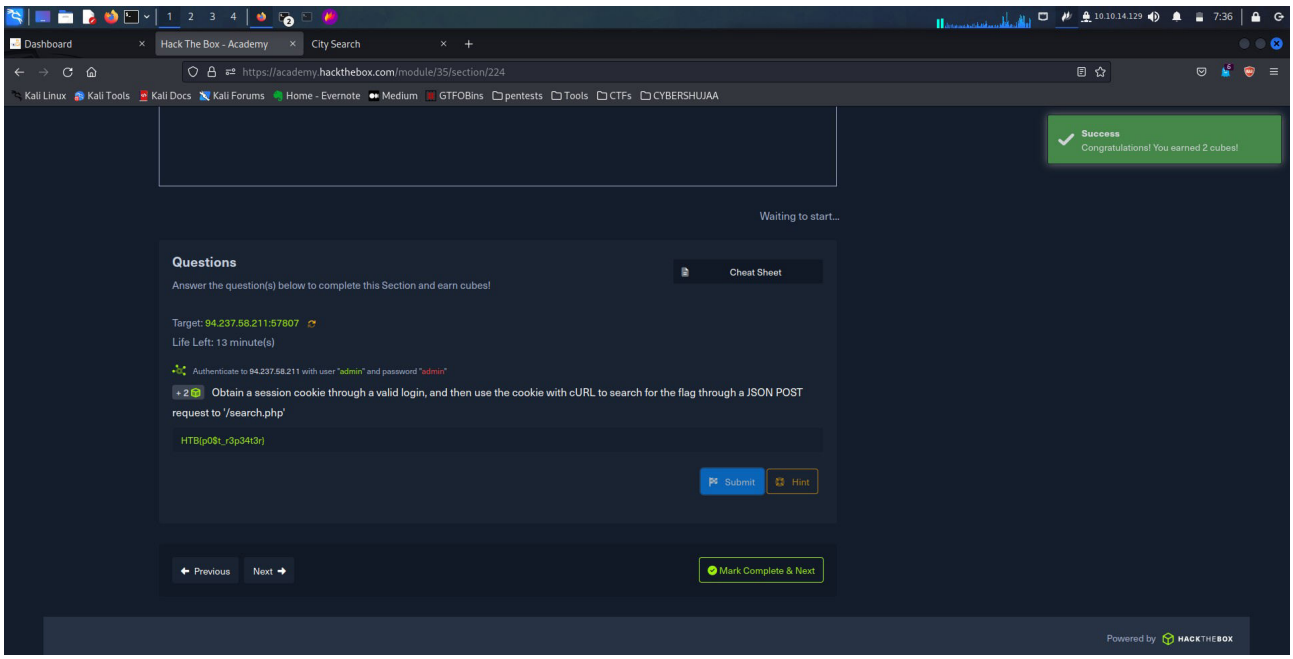
command used: curl -X POST -d '{"search":"london"}' -b 'PHPSESSID=ischgme5qr2jsdfbaheb6m5sg4' -H 'Content-Type: application/json' http://94.237.58.211:57807/search.php

As we can see we have an output **[“london (UK)”]**

I then tried to search for the flag using this method all I had to do was replace my search value with the keyword flag and there it was;

Command used:- url -X POST -d '{"search":"flag"}' -b 'PHPSESSID=ischgme5qr2jsdfbaheb6m5sg4' -H 'Content-Type: application/json' http://94.237.58.211:57807/search.php



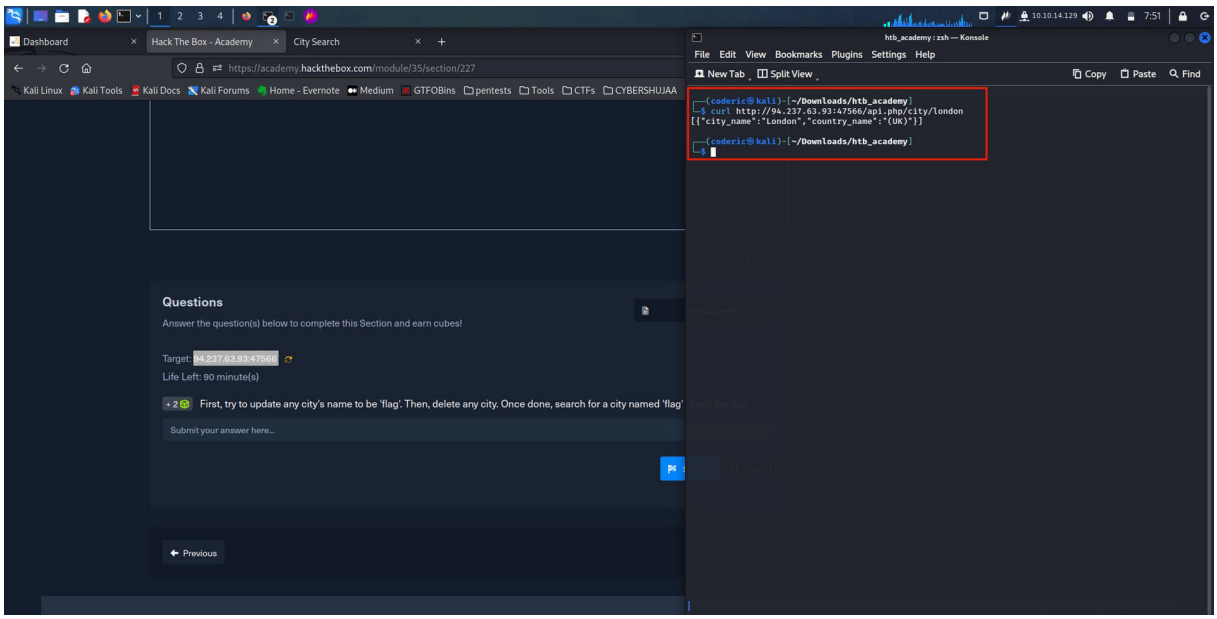


CRUD API

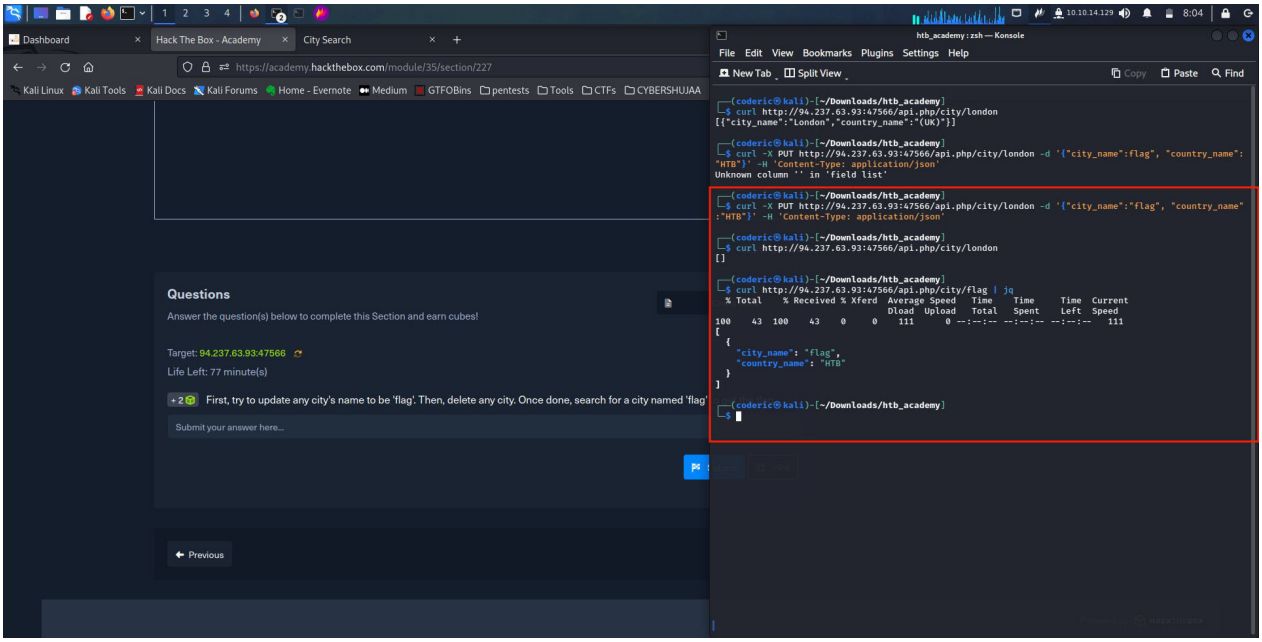
In this section I learnt on how to utilize APIs in web applications to perform the same thing as we had done earlier on the city search task using php parameter and how to directly interact with the API endpoint.
API - There are several types of APIs but many APIs are used to interact with a database in a way we can be able to specify the requested table and the requested row within our API query, and then use an HTTP method to perform the operation needed.
I also learnt CURD represents:- Create, Read, Update and Delete.

Question

First, try to update any city's name to be 'flag'. Then, delete any city. Once done, search for a city named 'flag' to get the flag. ANS: **HTB{crud_4p!_m4n!pul4t0r}**
First I tested the read command to see if its working and whether london city's name is valid in the search so that I ca use it.
Command used:- curl http://94.237.63.93:47566/api.php/city/london
Here we see city london is valid with its country being UK.



I then continued to first update the city's name "london" to be "flag" then checked that the record was updated by searching for london and for flag. Only flag search could give an output this time so I was sure my record had been successfully updated.



According to the instructions we are told to delete any city, at first I did not have any city in mind therefore I tried to find out cities in UK and came across **Leicester.** I had a city in UK but didn't know if it was in the database so I had to check if it was there firs using command:-

curl http://94.237.63.93:47566/api.php/city/Leicester

Output Results shows it is in our database.
[{"city_name":"Leicester","country_name":"(UK)"}]

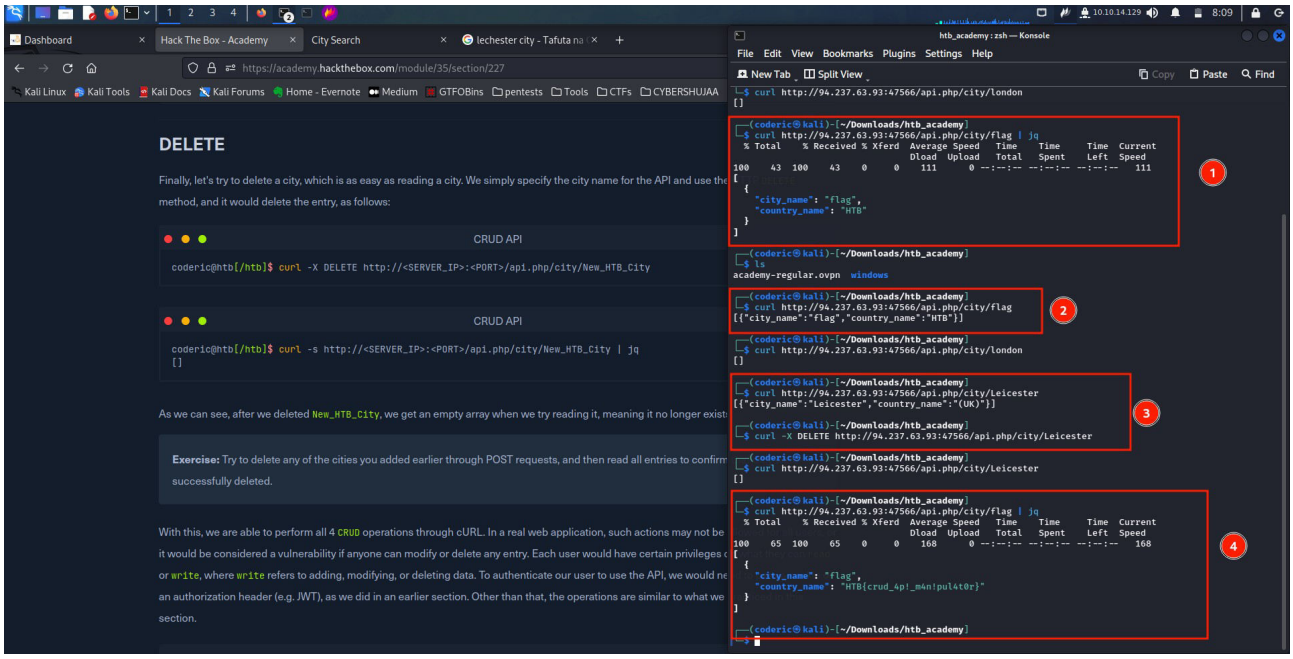
After this I had then to delete this city first before searching for my flag so I first did so.

Command used:- curl -X DELETE http://94.237.63.93:47566/api.php/city/Leicester

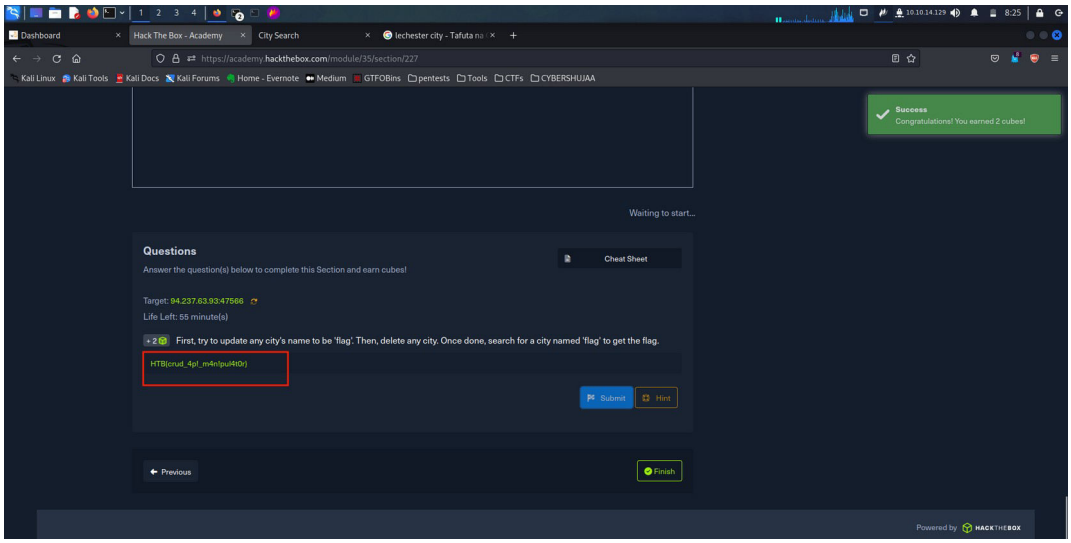
On checking again if the city was still available I got no results so I was sure this record had been deleted.

Now what was remaining is to search for the flag.
Command used:-
curl http://94.237.63.93:47566/api.php/city/flag | jq

- **| jp** – piping the output to jq utility, will format the information given it properly



The flag is:- **HTB{crud_4p!_m4n!pul4t0r}**



Conclusion

In this module I have learnt a few basics on the different web application attacks that raise concerns but also the knowledge offered in the module has helped realize some common web application security issues.
This are some of the web security aspects I have learnt:-

1. HTTP Basics – I have Understood the fundamentals of the HTTP protocol, different request methods (GET, POST, etc.), headers, and responses.

2. Web Application Architecture – I have Explored on how web applications are structured, including the client-server model, front-end, and back-end components.

3. Web Security Protocols – I have Learnt about security protocols like HTTPS and their role in securing web communication.

4. Web Vulnerabilities – I was able to Identify and exploit common web vulnerabilities such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), SQL injection, and more.

5. Authentication and Authorization – I have learnt a few on examining how web applications handle user authentication and authorization and identifying potential weaknesses in these mechanisms.

6. API Security – Lastly I have Explored security considerations related to web APIs, including proper authentication and authorization