ASSIGNMENT 3: WEB REQUESTS WRITEUP

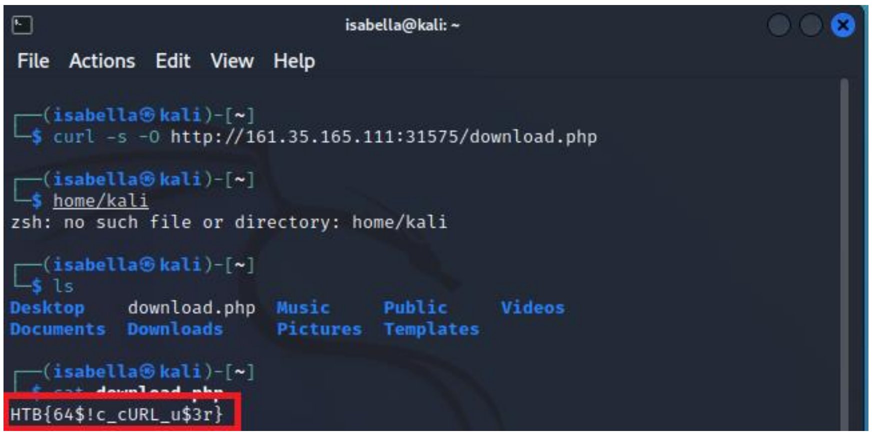
Introduction

In this module, it was divided into two parts, part one being HTTP fundamentals where we explored the fundamentals of working with HTTP requests and responses, including an understanding of headers that carry important metadata. Part two, we dived into common HTTP methods and response codes, essential knowledge for interacting with APIs. Additionally, we will learn how to leverage powerful tools like cURL and Browser DevTools to interact with web applications effectively.

Part 1: HTTP Fundamentals

In the first section of the module, we went through HTTP which is an application-level protocol used to access the World Wide Web resources. We also went through the cURL which is a command-line tool and library that primarily supports HTTP along with many other protocols. and used to send HTTP requests.

Question 1: To get the flag, start the above exercise, then use cURL to download the file returned by '/download.php' in the server shown above. HTB{64$!c_cURL_u$3r}



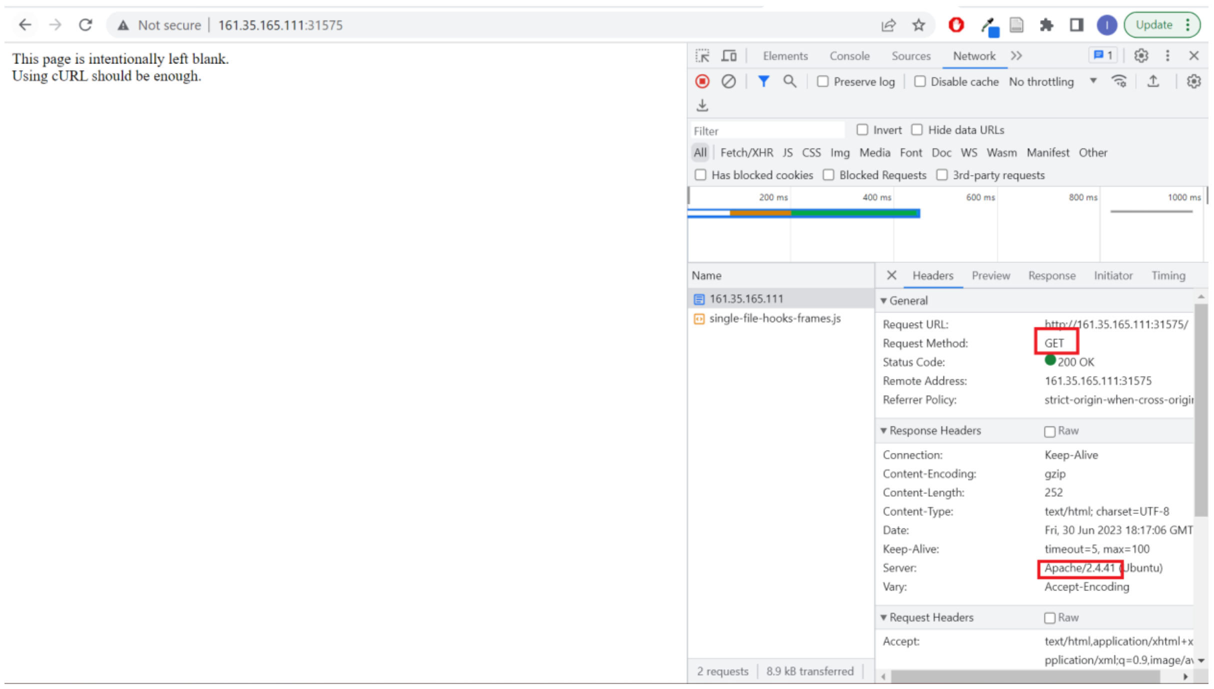
```
┌──(isabella㉿kali)-[~]
└─$ curl -s -O http://161.35.165.111:31575/download.php

┌──(isabella㉿kali)-[~]
└─$ home/kali
zsh: no such file or directory: home/kali

┌──(isabella㉿kali)-[~]
└─$ ls
Desktop    download.php  Music     Public     Videos
Documents  Downloads     Pictures  Templates

┌──(isabella㉿kali)-[~]
└─$ cat download.php
HTB{64$!c_cURL_u$3r}
```

In the next section, we went through the HTTP requests which are made by the client (e.g. cURL/browser), and is processed by the server (e.g. web server) which contains all the details one may require from a server. Once the server receives the HTTP request, it processes it and responds by sending the HTTP response, which contains the response code.

Questions

Question 1: What is the HTTP method used while intercepting the request? (case-sensitive) GET

Question 2: 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) 2.4.41



This page is intentionally left blank.
Using cURL should be enough.

161.35.165.111
single-file-hooks-frames.js

▾ General

Request URL:	http://161.35.165.111:31575/
Request Method:	GET
Status Code:	● 200 OK
Remote Address:	161.35.165.111:31575
Referrer Policy:	strict-origin-when-cross-origi

▾ Response Headers

Connection:	Keep-Alive
Content-Encoding:	gzip
Content-Length:	252
Content-Type:	text/html; charset=UTF-8
Date:	Fri, 30 Jun 2023 18:17:06 GMT
Keep-Alive:	timeout=5, max=100
Server:	Apache/2.4.41 (Ubuntu)
Vary:	Accept-Encoding

▾ Request Headers

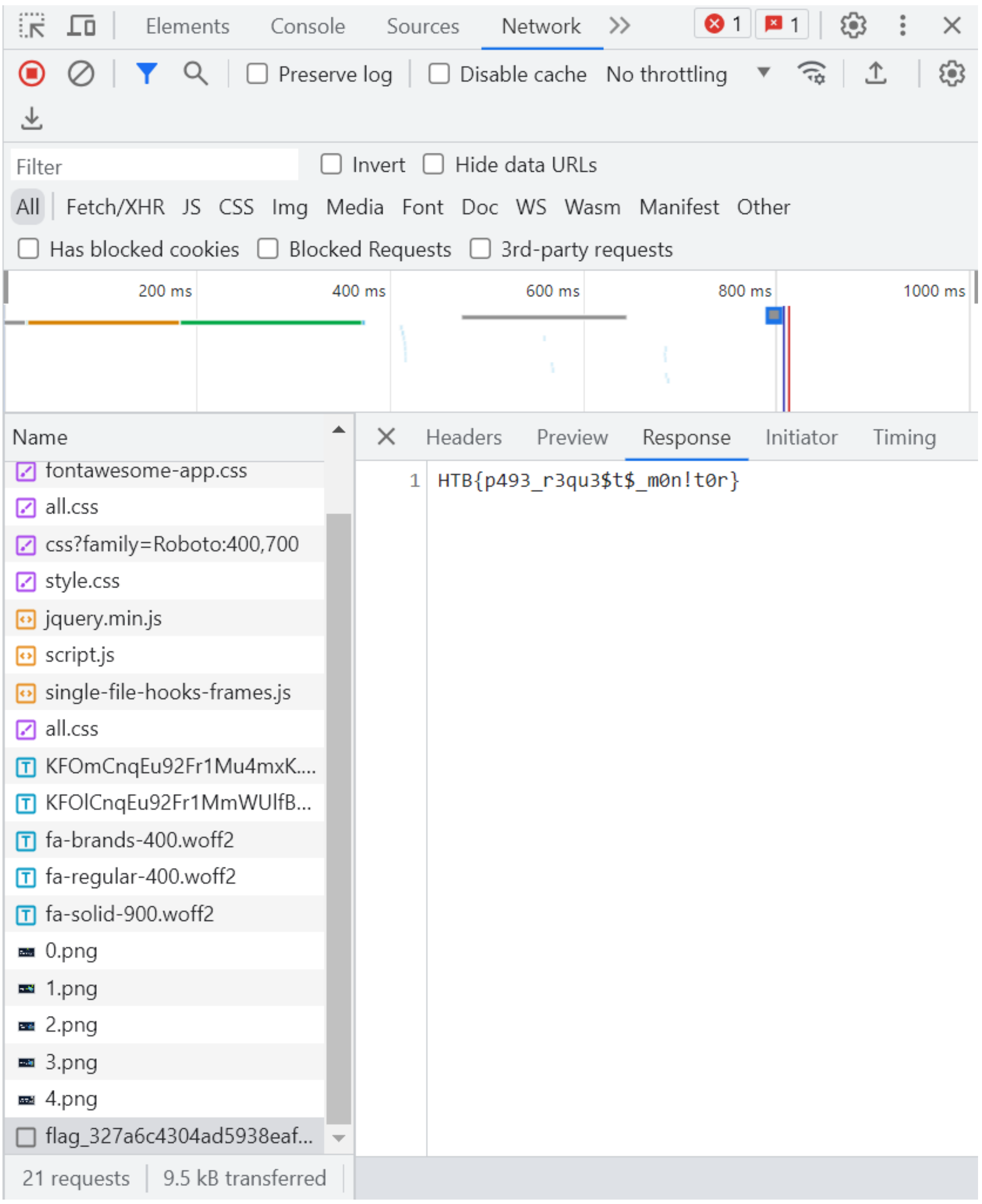
Accept:	text/html,application/xhtml+x pplication/xml;q=0.9,image/av

Next section we look at HTTP headers which are additional pieces of information sent along with an HTTP request or response. They provide important metadata about the request or response, such as the content type, cache control directives, authentication credentials, and more. Headers play a crucial role in facilitating communication between clients and servers, enabling them to exchange data and carry out various functionalities of the HTTP protocol.

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.

HTB{p493_r3qu3$t$_m0n!t0r}



```
fontawesome-app.css
all.css
css?family=Roboto:400,700
style.css
jquery.min.js
script.js
single-file-hooks-frames.js
all.css
KFOmCnqEu92Fr1Mu4mxK....
KFOlCnqEu92Fr1MmWUlfB...
fa-brands-400.woff2
fa-regular-400.woff2
fa-solid-900.woff2
0.png
1.png
2.png
3.png
4.png
flag_327a6c4304ad5938eaf...
21 requests | 9.5 kB transferred
```

```
1  HTB{p493_r3qu3$t$_m0n!t0r}
```

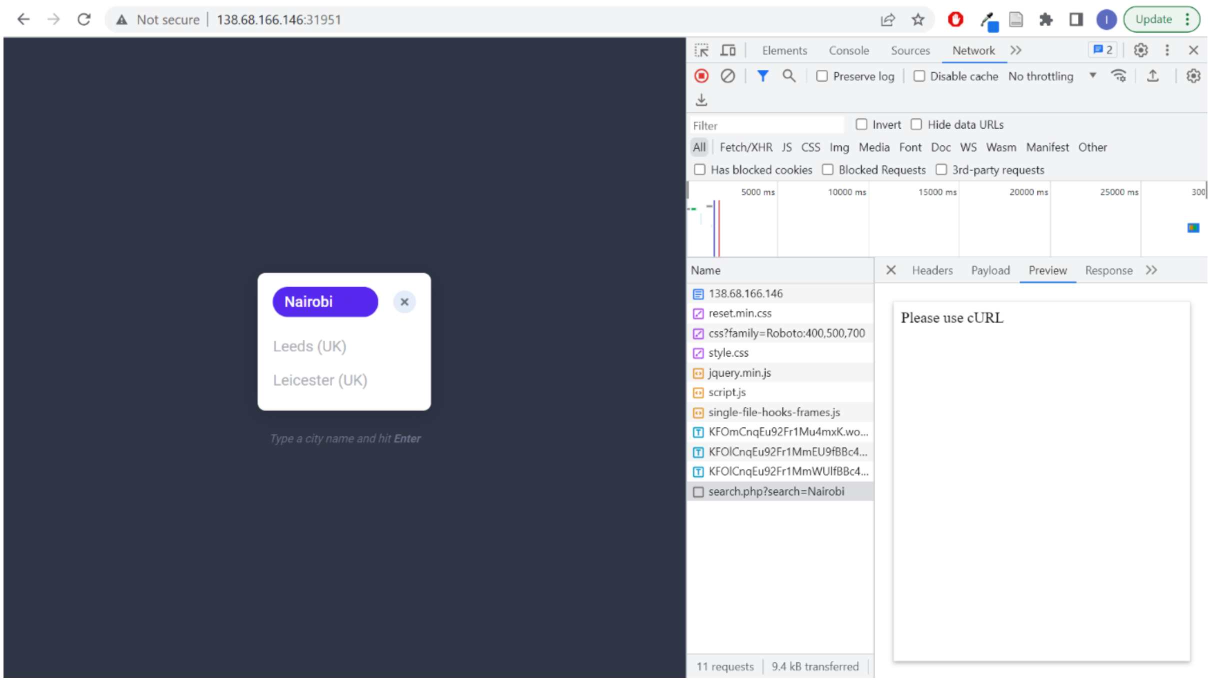
Part 2: HTTP Methods

In the first section, we examined how GET requests may be used by web applications for functionalities like search and accessing pages.

Questions

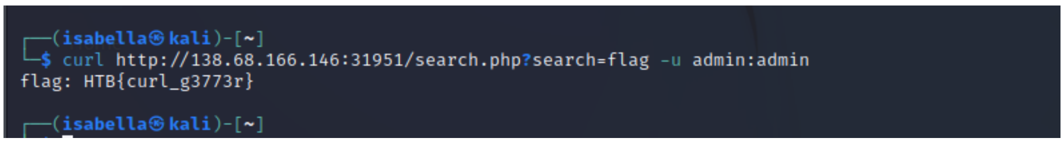
The exercise above seems to be broken, as it returns incorrect results. Use the browser devtools to see what is the request it is sending when we search, and use cURL to search for 'flag' and obtain the flag?

HTB{curl_g3773r}


Not secure | 138.68.166.146:31951
Nairobi
Leeds (UK)
Leicester (UK)
Type a city name and hit Enter
Elements Console Sources Network
Name
138.68.166.146
reset.min.css
css?family=Roboto:400,500,700
style.css
jquery.min.js
script.js
single-file-hooks-frames.js
search.php?search=Nairobi
Headers Payload Preview Response
11 requests | 9.4 kB transferred

Please use cURL

Then went to the terminal and runned curl but using the -u because the page was authenticated.



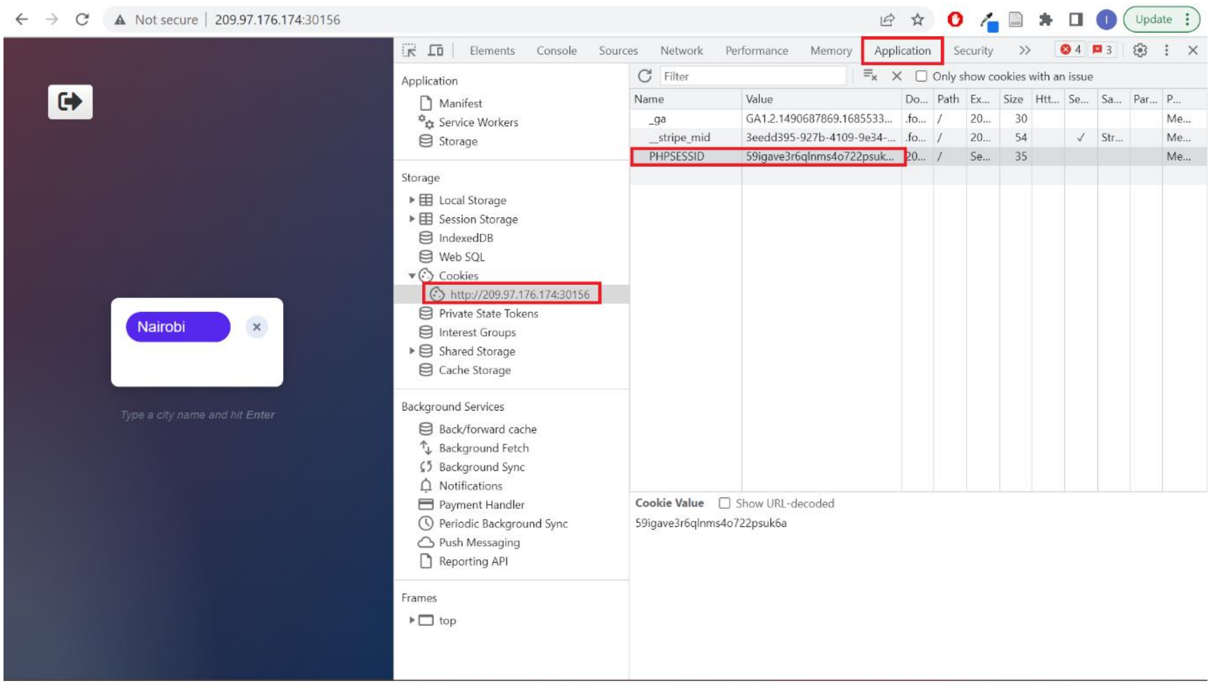
```
┌──(isabella㉿kali)-[~]
└─$ curl http://138.68.166.146:31951/search.php?search=flag -u admin:admin
flag: HTB{curl_g3773r}

┌──(isabella㉿kali)-[~]
```

In the next section, we explored the POST request which is essentially for transferring files or moving user parameters from the URL. POST places user parameters within the HTTP request body. Three main benefits: Lack of logging, Less encoding requirement and more data can be sent - maximum URL length varies between browsers, servers and content delivery networks.

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' HTB{p0$t_r3p34t3r}

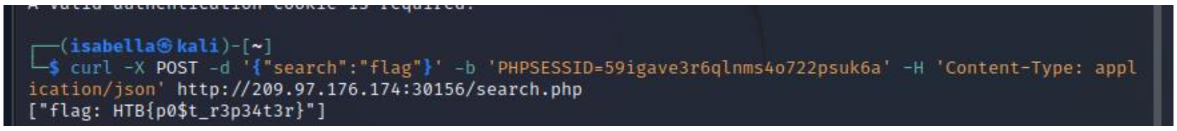
Several step were made in order to get the flag first, go the website under the devtools under the storage section to get the cookie as shown below



Name	Value	Do...	Path	Ex...	Size	Htt...	Se...	Sa...	Par...	P...
_ga	GA1.2.1490687869.1685533...	.fo...	/	20...	30					Me...
__stripe_mid	3eedd395-927b-4109-9e34-...	.fo...	/	20...	54		✓	Str...		Me...
PHPSESSID	59igave3r6qlnms4o722psuk...	20...	/	Se...	35					Me...

Cookie Value
59igave3r6qlnms4o722psuk6a

Then used the curl command with search as flag and with the cookie session id as shown in the diagram below to obtain the flag.



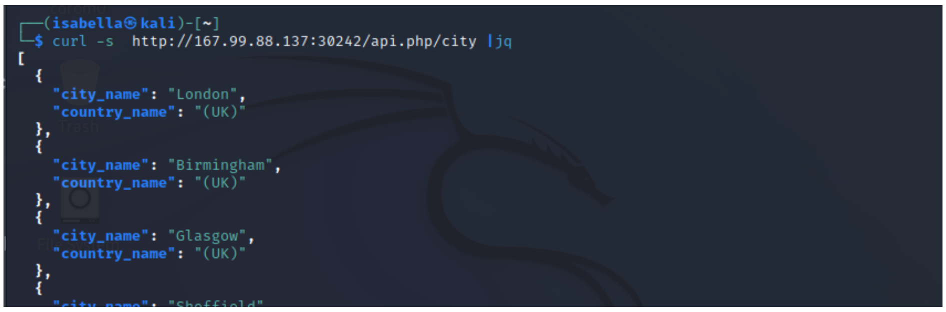
```
┌──(isabella㉿kali)-[~]
└─$ curl -X POST -d '{"search":"flag"}' -b 'PHPSESSID=59igave3r6qlnms4o722psuk6a' -H 'Content-Type: appl
ication/json' http://209.97.176.174:30156/search.php
["flag: HTB{p0$t_r3p34t3r}"]
```

To conclude the module, we lastly examined the CRUD API which is an API that supports the basic CRUD operations: Create, Read, Update, and Delete. It allows clients to interact with a server and perform actions like creating new resources, retrieving existing resources, updating resource data, and deleting resources. By defining endpoints and specifying HTTP methods like POST, GET, PUT, and DELETE, a CRUD API provides a standardized way to manipulate data over HTTP.

Question

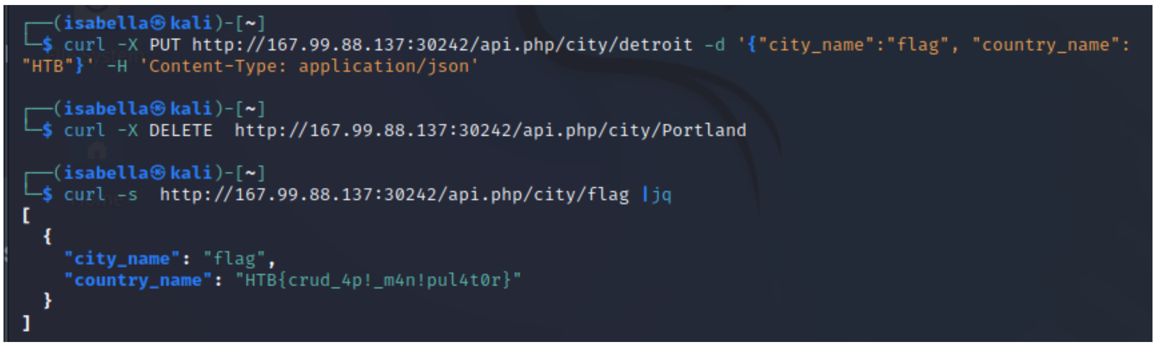
First, try to update any city's name to be 'flag'. Then, delete any city. Once done, search for a city named 'flag' to get the flag.

To get the flag, firstly we used the command below to list all the available cities.



```
┌──(isabella㉿kali)-[~]
└─$ curl -s  http://167.99.88.137:30242/api.php/city |jq
[
  {
    "city_name": "London",
    "country_name": "(UK)"
  },
  {
    "city_name": "Birmingham",
    "country_name": "(UK)"
  },
  {
    "city_name": "Glasgow",
    "country_name": "(UK)"
  },
  {
    "city_name": "Sheffield"
```

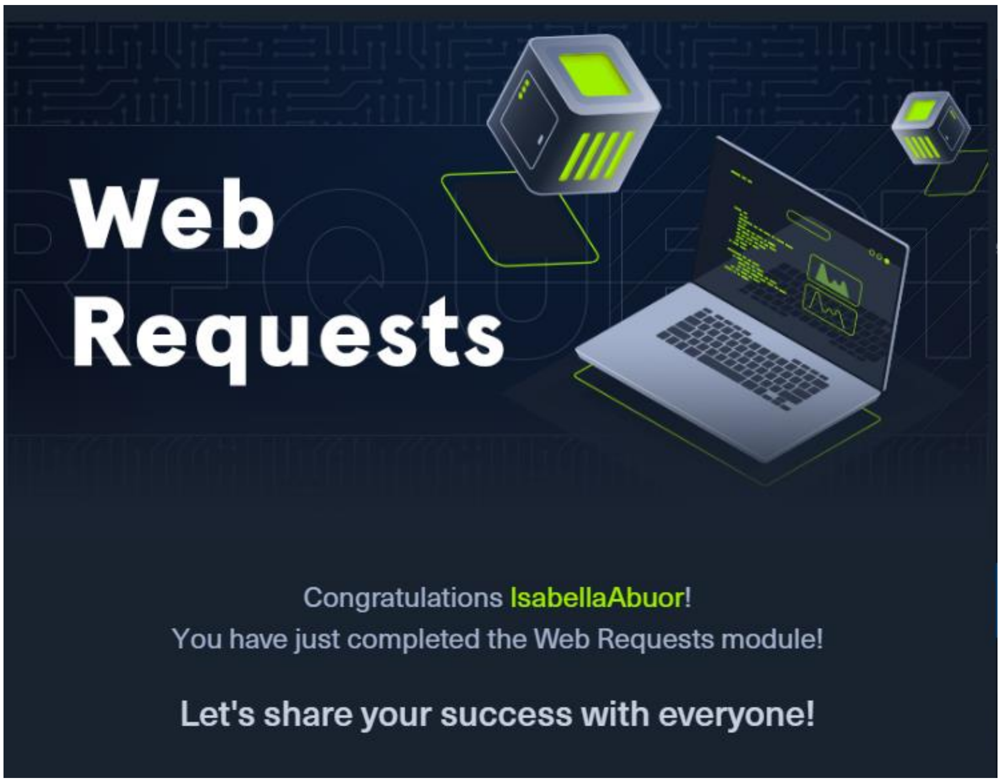
Then we used the PUT command to update the detroit URL, then used DELETE command to remove Portland and finally used the Print the api.php/city/flag url with " | jq" to list the flag as shown below



```
┌──(isabella㉿kali)-[~]
└─$ curl -X PUT http://167.99.88.137:30242/api.php/city/detroit -d '{"city_name":"flag", "country_name":
"HTB"}' -H 'Content-Type: application/json'

┌──(isabella㉿kali)-[~]
└─$ curl -X DELETE  http://167.99.88.137:30242/api.php/city/Portland

┌──(isabella㉿kali)-[~]
└─$ curl -s  http://167.99.88.137:30242/api.php/city/flag |jq
[
  {
    "city_name": "flag",
    "country_name": "HTB{crud_4p!_m4n!pul4t0r}"
  }
]
```

Conclusion

In conclusion, this module was more challenging than I expected it to be but the exhilarating feeling of overcoming each hurdle and finally reaching and getting the flag was incredibly rewarding. This module provided a comprehensive exploration of the HTTP protocol and its fundamental aspects. In part one, we covered the essentials of working with HTTP requests and responses, emphasizing the significance of headers in conveying metadata. Part two focused on understanding common HTTP methods and response codes, essential for interacting with APIs. Furthermore, we gained practical insights into leveraging tools such as cURL and Browser DevTools to enhance our interactions with web applications. With this knowledge, we are now well-equipped to navigate the intricacies of web communication and effectively engage with modern web technologies.



Congratulations IsabellaAbuor!
You have just completed the Web Requests module!

Let's share your success with everyone!