Assignment 2: HTB Academy - Web Requests

Report by: Tonny Odhiambo, CS-CNS06-24028

Introduction

This module is designed to provide a comprehensive understanding of how web requests operate, their various types, and how they can be manipulated or exploited in a cybersecurity context.

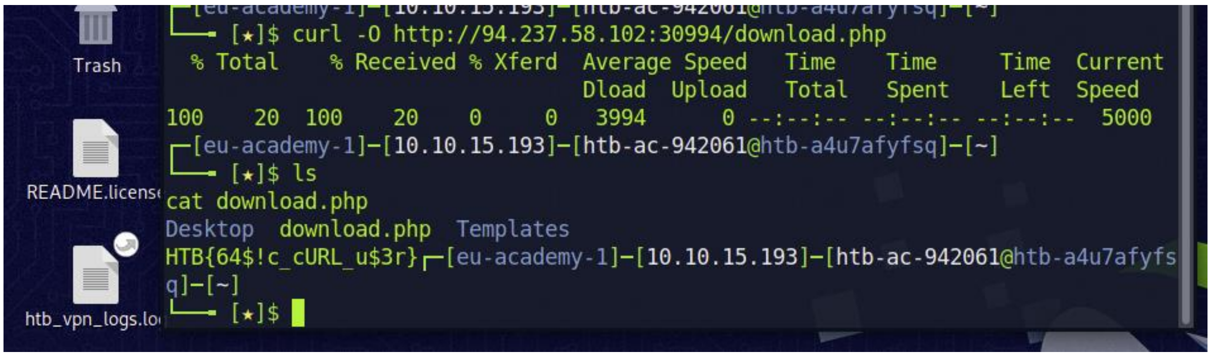
Web requests form the backbone of internet communication, allowing clients and servers to exchange information seamlessly. Understanding the intricacies of HTTP methods such as GET, POST, PUT, and DELETE, along with the role of headers, cookies, and status codes, is essential for identifying potential vulnerabilities and securing web applications. The module covers these topics in depth, offering both theoretical knowledge and practical exercises that simulate real-world scenarios.

By diving into the mechanics of web requests, I aimed to gain a robust grasp of how data is transmitted over the web and the common pitfalls that attackers exploit. This foundation is critical for anyone looking to delve deeper into web application security. Through hands-on labs and exercises, the Web Requests module provided me with the opportunity to apply these concepts in a controlled environment, reinforcing my learning and preparing me for more advanced security challenges.

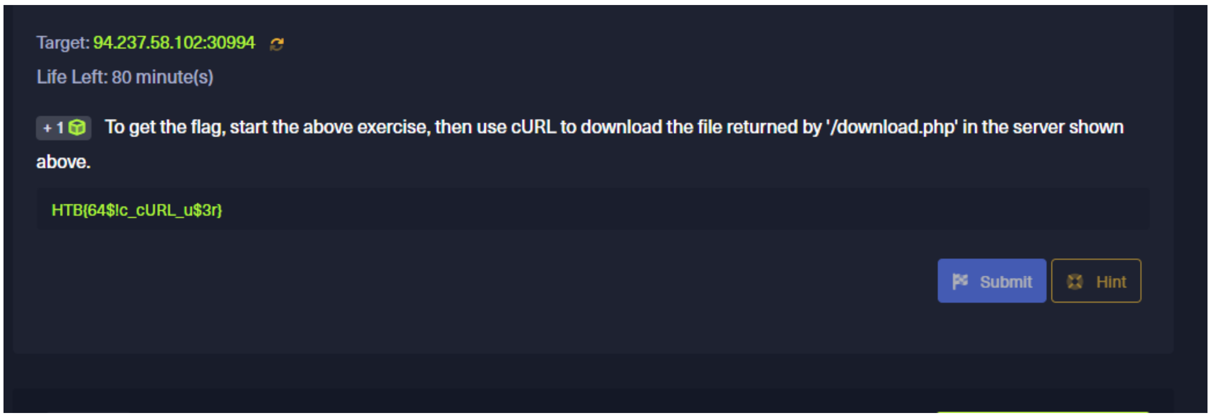
1. HyperText Transfer Protocol (HTTP)

HTTP (HyperText Transfer Protocol) is a foundational protocol used for accessing resources on the World Wide Web. It operates at the application level and involves communication between a client and a server. The client sends a request for a resource, and the server processes this request and returns the resource. HTTP typically communicates over port 80.

In this first question, I spawned the target and used the command **curl -O** http://94.237.58.102:30994/download.php to download the file, then used the commands **ls and cat download.php** to check the downloaded file and locate the flag requested in the question.



```
┌─[eu-academy-1]─[10.10.15.193]─[htb-ac-942061@htb-a4u7afyfsq]─[~]
└──╼ [★]$ curl -O http://94.237.58.102:30994/download.php
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    20  100    20    0     0   3994      0 --:--:-- --:--:-- --:--:--  5000
┌─[eu-academy-1]─[10.10.15.193]─[htb-ac-942061@htb-a4u7afyfsq]─[~]
└──╼ [★]$ ls
cat download.php
Desktop  download.php  Templates
HTB{64$!c_cURL_u$3r}┌─[eu-academy-1]─[10.10.15.193]─[htb-ac-942061@htb-a4u7afyfs
q]─[~]
└──╼ [★]$ 
```


Target: 94.237.58.102:30994

Life Left: 80 minute(s)

+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}
```

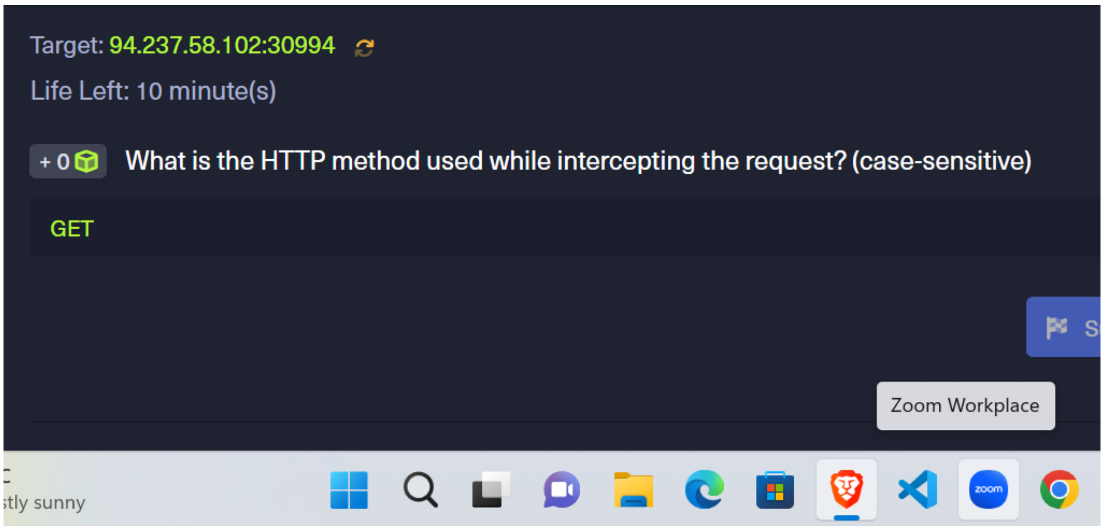
2. HTTP Requests and Response

HTTP communications are primarily composed of an HTTP request and an HTTP response. The client, such as a web browser or a command-line tool like cURL, initiates an HTTP request. This request includes essential details such as the URL, path, parameters, and any additional request data or headers. These details specify what resource the client wants to access and under what conditions.

Once the server, typically a web server, receives the HTTP request, it processes the request and generates an HTTP response. This response includes a status code indicating the outcome of the request and, if the client has the necessary permissions, the requested resource data. The status code and data help the client understand if the request was successful and retrieve the needed information.

HTTP requests and responses are fundamental to web interactions, forming the backbone of data exchange on the internet. This protocol ensures that clients can request resources and servers can deliver them efficiently.

For the questions in this section, i got their answers by running the command **curl -v** http://94.237.58.102:30994/ to send a GET request and intercept the HTTP method. The -v flag is meant to make it more verbose, showing details of the request.

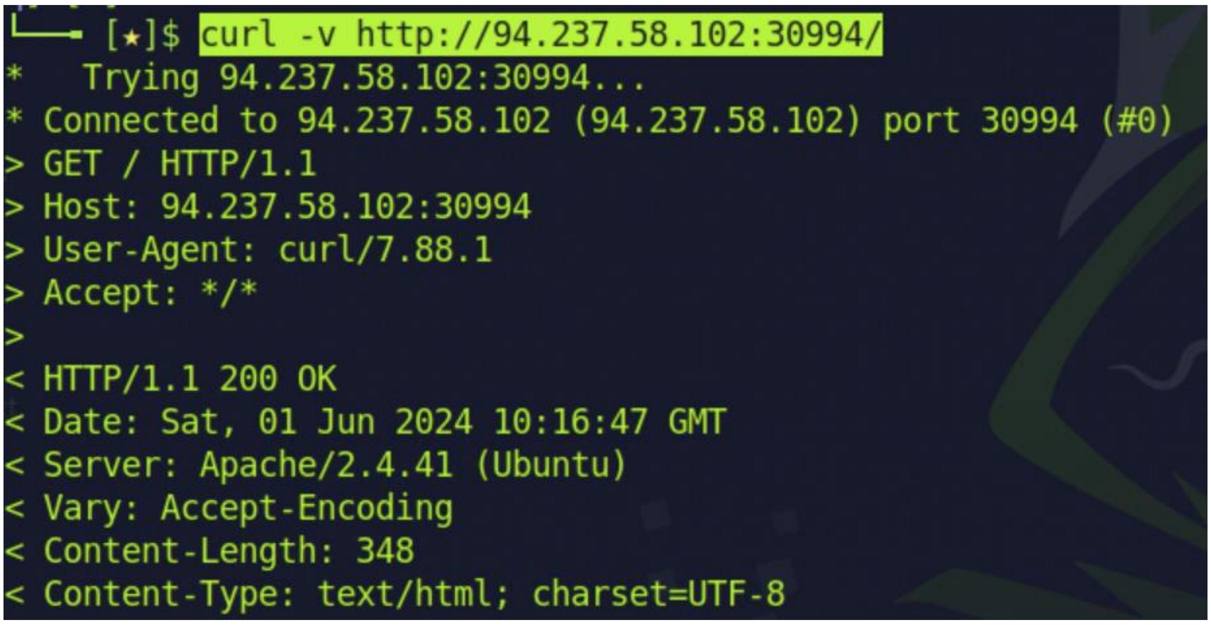


Target: 94.237.58.102:30994

Life Left: 10 minute(s)

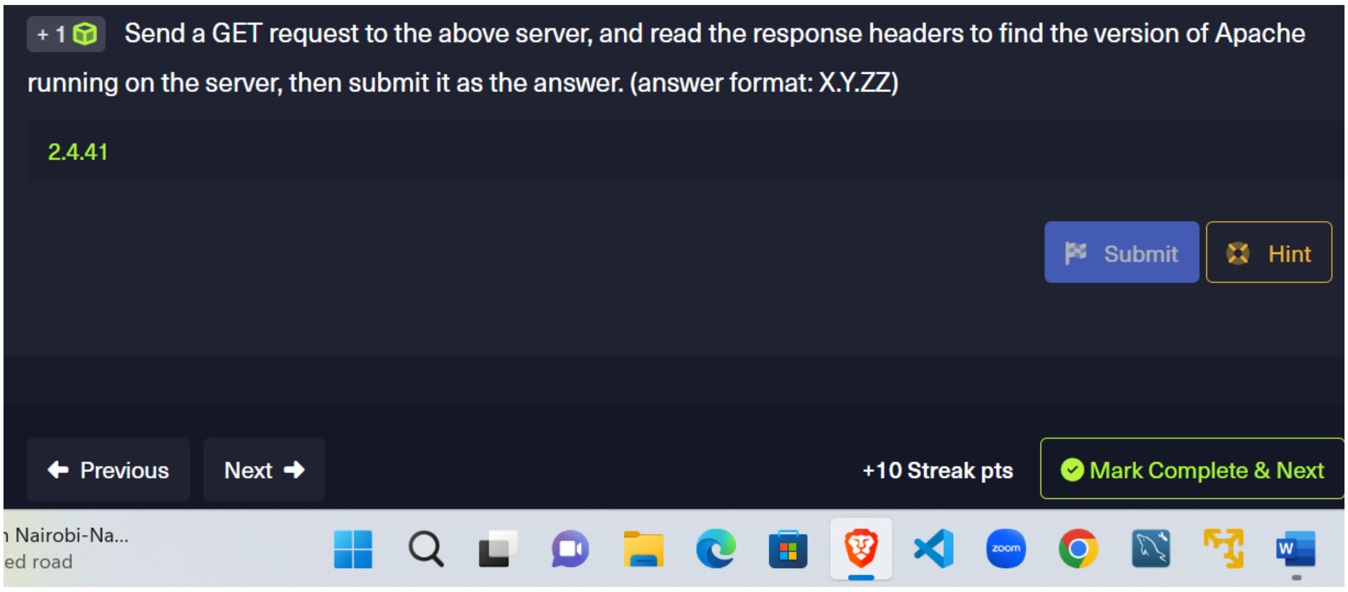
+0 What is the HTTP method used while intercepting the request? (case-sensitive)

GET



```
└──╼ [★]$ curl -v http://94.237.58.102:30994/
*   Trying 94.237.58.102:30994...
* Connected to 94.237.58.102 (94.237.58.102) port 30994 (#0)
> GET / HTTP/1.1
> Host: 94.237.58.102:30994
> User-Agent: curl/7.88.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 01 Jun 2024 10:16:47 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 348
< Content-Type: text/html; charset=UTF-8
```

For the second question on the version of Apache running on the server, The version is given as **2.4.41** as shown in the screenshot above.


+ 1 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)

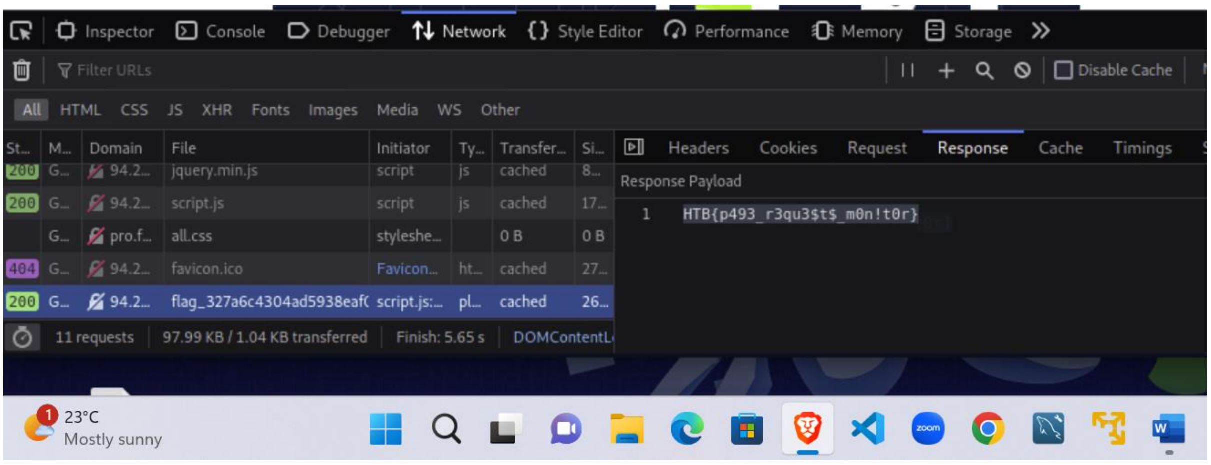
2.4.41

3. **HTTP Headers**

HTTP headers are vital in web communications, enabling data exchange between clients and servers. They are categorized as follows:

- **General Headers**: Used in both requests and responses, providing contextual information about the message, such as the Date and Connection headers.
- **Entity Headers**: Describe the content being transferred, commonly found in responses and POST or PUT requests. Key headers include Content-Type and Content-Length.
- **Request Headers**: Sent by the client to provide additional context about the request. Important headers include Host, User-Agent, and Authorization.
- **Response Headers**: Sent by the server to provide information about the server and the request's status. Common headers include Server and Set-Cookie.
- **Security Headers**: Enhance security by specifying rules and policies for the browser. Examples include Content-Security-Policy and Strict-Transport-Security.

For the question in this section, I spawned the target and used Mozilla browser to access it the used the developer tools specifically the network section to see what requests were made to the page and also to find the request to the flag as shown in the screenshot below.

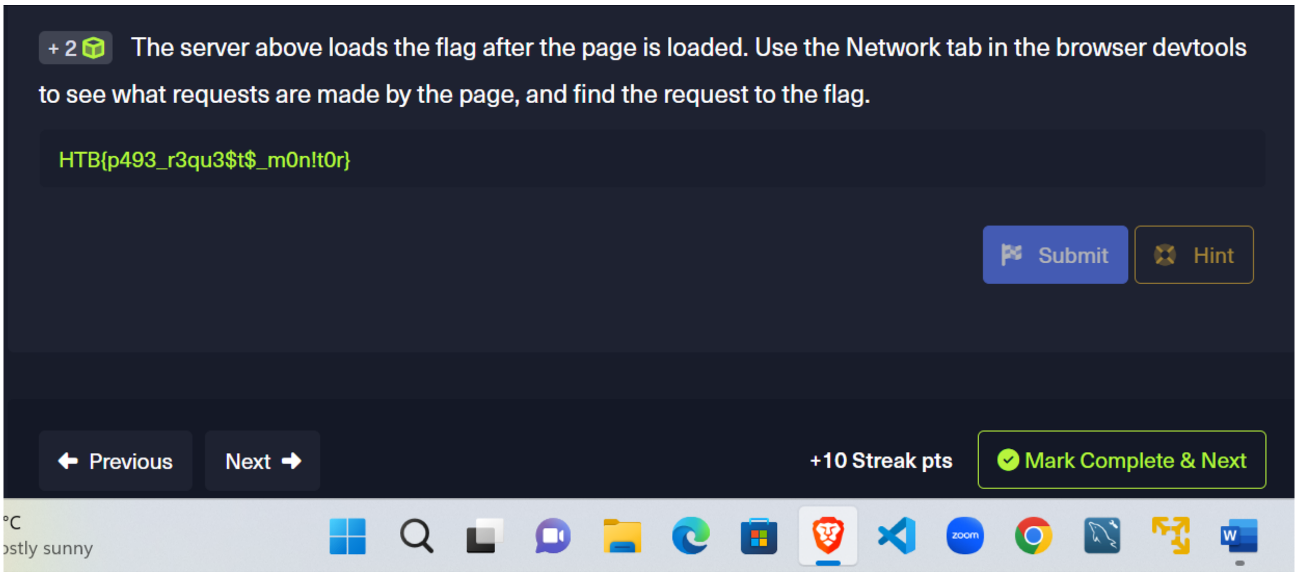


St...	M...	Domain	File	Initiator	Ty...	Transfer...	Si...
200	G...	94.2...	jquery.min.js	script	js	cached	8...
200	G...	94.2...	script.js	script	js	cached	17...
	G...	pro.f...	all.css	styleshe...		0 B	0 B
404	G...	94.2...	favicon.ico	Favicon...	ht...	cached	27...
200	G...	94.2...	flag_327a6c4304ad5938eaf(	script.js:...	pl...	cached	26...

11 requests | 97.99 KB / 1.04 KB transferred | Finish: 5.65 s | DOMContentL

Response Payload

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



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_r3qu3t_m0n!t0r}

HTTP Methods and Codes

HTTP methods and status codes are essential for web communication. Common request methods include:

- **GET**: Requests a specific resource.
- **POST**: Sends data to the server.
- **HEAD**: Requests headers without the body.
- **PUT**: Creates or updates a resource.
- **DELETE**: Deletes a resource.
- **OPTIONS**: Returns supported HTTP methods.
- **PATCH**: Applies partial modifications.

HTTP status codes indicate the request's result:

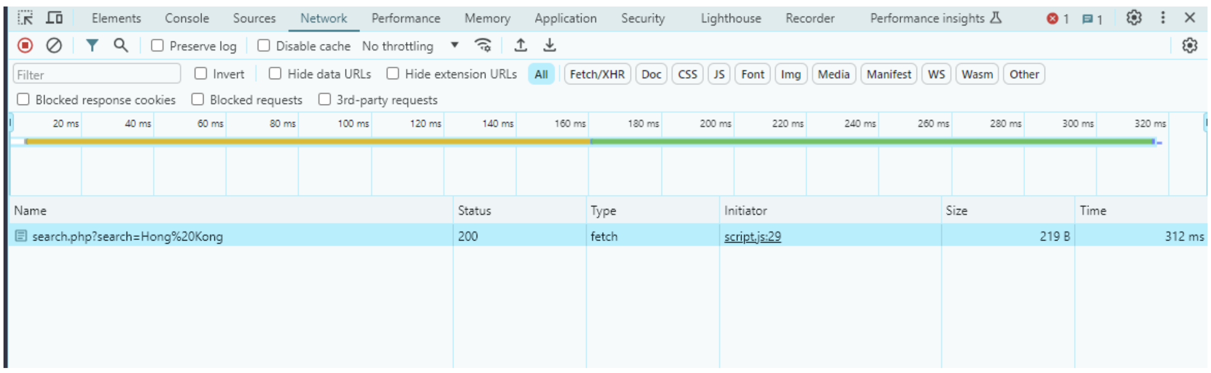
- **1xx**: Informational.
- **2xx**: Success (e.g., 200 OK).
- **3xx**: Redirection (e.g., 302 Found).
- **4xx**: Client errors (e.g., 404 Not Found).
- **5xx**: Server errors (e.g., 500 Internal Server Error).

Modern web applications primarily use GET and POST methods.

a) GET

In this section, the first question required accessing a web application and performing a search of any city and using the Network tab of the developer tools to identify the search request as shown below.

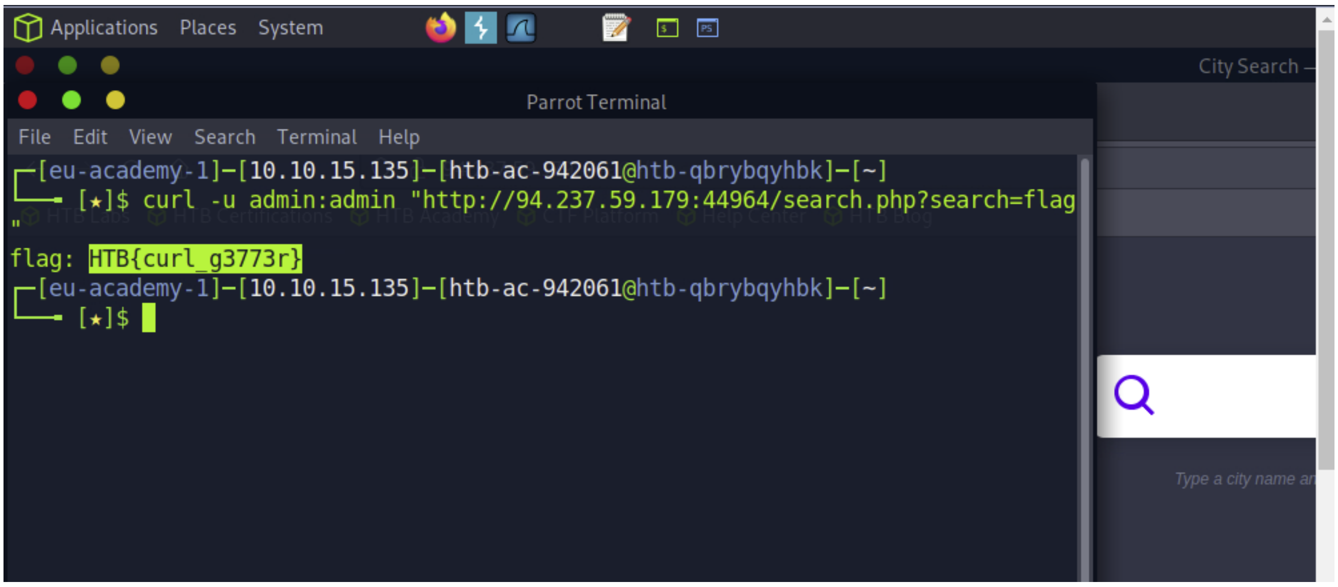
The search request url was http://94.237.59.179:44964/search.php?search=Hong%20Kong since the city that i searched was Hong Kong.



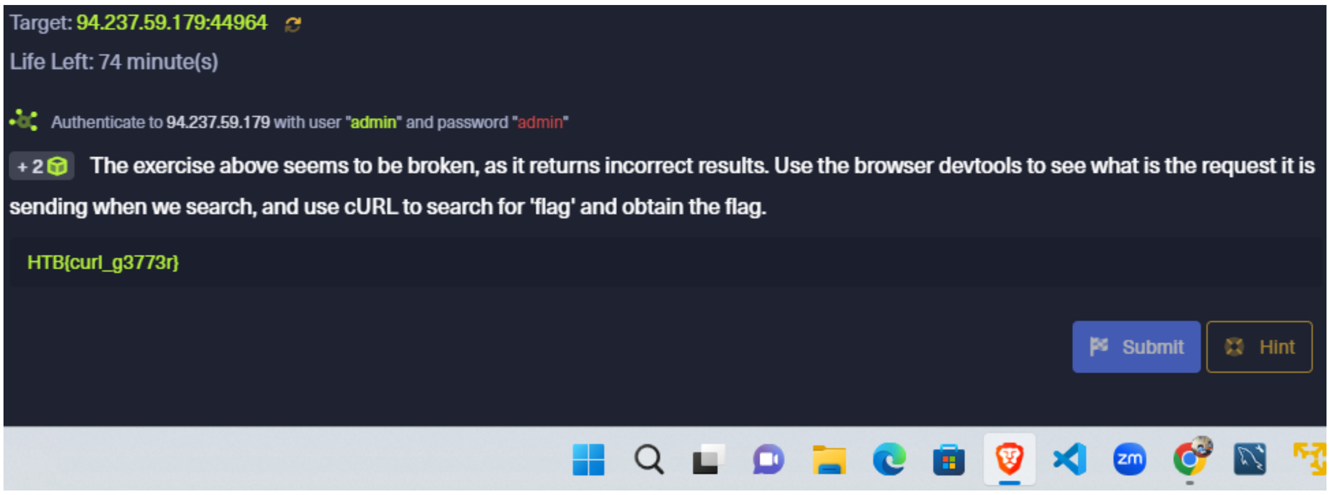
Name	Status	Type	Initiator	Size	Time
search.php?search=Hong%20Kong	200	fetch	script.js:29	219 B	312 ms

After using the devtools to determine the search request, I then used **curl** to search and find the flag needed in the question.

To get the flag I replaced the name of the city with ‘flag’ in my terminal and used the curl to search.



```
┌─[eu-academy-1]─[10.10.15.135]─[htb-ac-942061@htb-qbrybqyhbk]─[~]
└──╼ [★]$ curl -u admin:admin "http://94.237.59.179:44964/search.php?search=flag
"
flag: HTB{curl_g3773r}
┌─[eu-academy-1]─[10.10.15.135]─[htb-ac-942061@htb-qbrybqyhbk]─[~]
└──╼ [★]$
```



Target: 94.237.59.179:44964

Life Left: 74 minute(s)

Authenticate to 94.237.59.179 with user "admin" and password "admin"

+2 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}

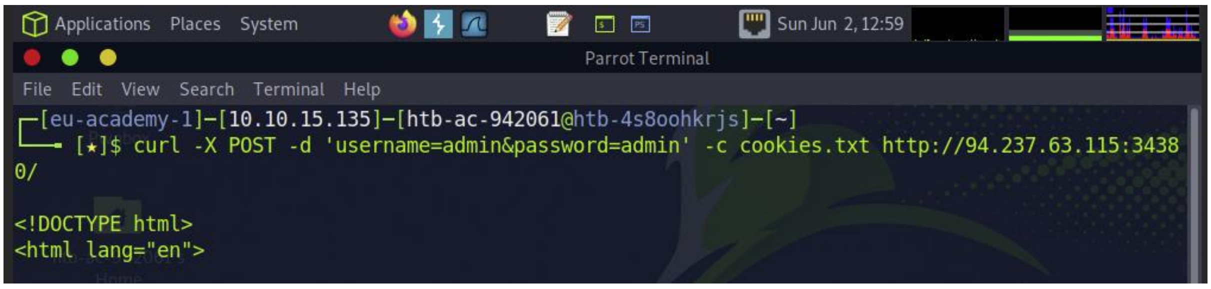
b) POST

In the POST section, I explored how web applications utilize POST requests for transferring user parameters securely, especially when handling large files or sensitive data. I demonstrated how to interact with login forms, authenticate users, and obtain session cookies using tools like cURL and browser devtools. Furthermore, I examined how to craft JSON POST requests to interact directly with web application functionalities, such as search functions, without requiring frontend interaction.

First to authenticate and obtain the session cookie, I used the command:

curl -X POST -d 'username=admin&password=admin' -c cookies.txt http://94.237.63.115:34380/

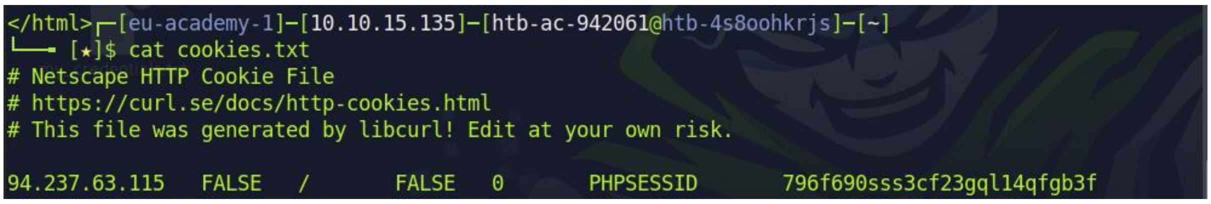
This command sends a POST request to the login endpoint with the provided username and password. It saves the received cookies in a file named cookies.txt as shown below.


Applications Places System
Sun Jun 2, 12:59
Parrot Terminal
File Edit View Search Terminal Help

```
┌─[eu-academy-1]─[10.10.15.135]─[htb-ac-942061@htb-4s8oohkrjs]─[~]
└──╼ [★]$ curl -X POST -d 'username=admin&password=admin' -c cookies.txt http://94.237.63.115:3438
0/

<!DOCTYPE html>
<html lang="en">
```

I then used the cat command to display the contents of the file in the terminal to ensure the session cookie was obtained and stores in the cookies.txt file: **cat cookies.txt**



```
</html>┌─[eu-academy-1]─[10.10.15.135]─[htb-ac-942061@htb-4s8oohkrjs]─[~]
└──╼ [★]$ cat cookies.txt
# Netscape HTTP Cookie File
# https://curl.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

94.237.63.115   FALSE   /       FALSE   0       PHPSESSID       796f690sss3cf23gql14qfgb3f
```

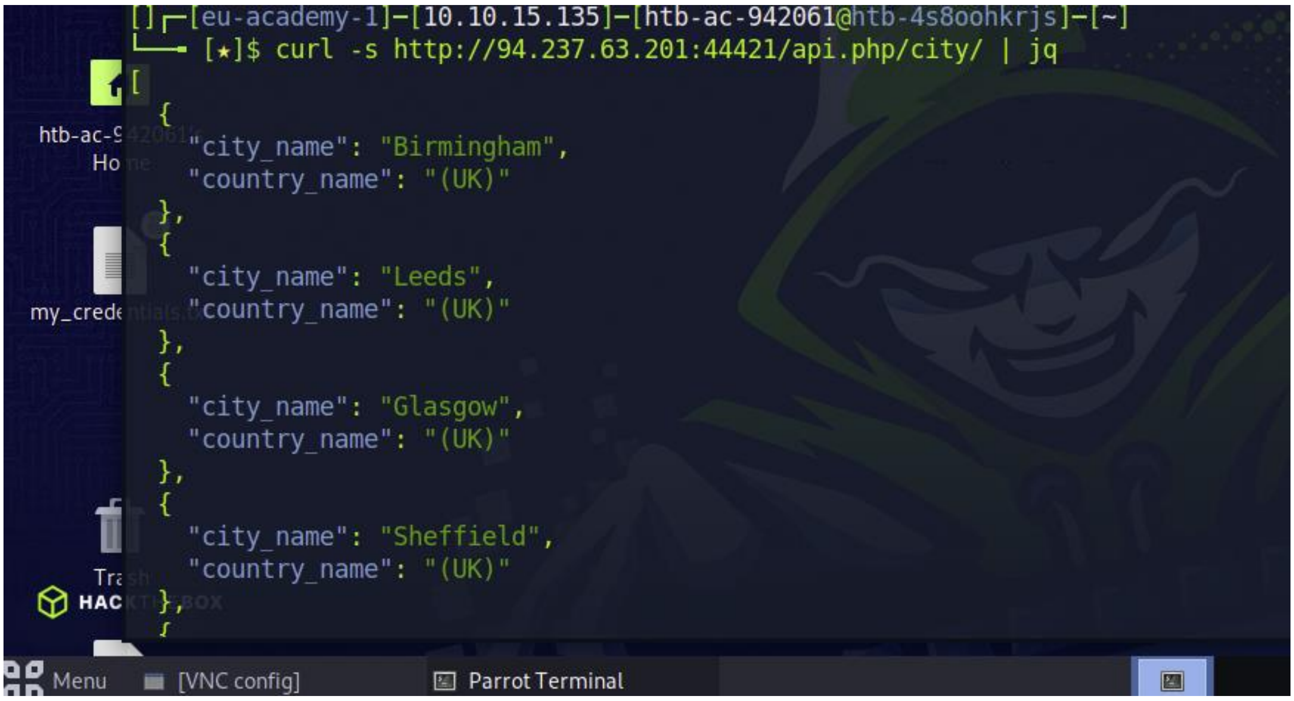
I then used the obtained session cookie to search for the flag using the command:

curl -X POST -d '{"search":"flag"}' -b cookies.txt -H 'Content-Type: application/json' http://94.237.63.115:34380/search.php

c) CRUD API

In this CRUD API exercise, I interacted with a web application's API endpoints to perform basic CRUD (Create, Read, Update, Delete) operations on a city database. Using curl commands, I demonstrated how to retrieve city information, add new cities, update existing cities, and delete cities.

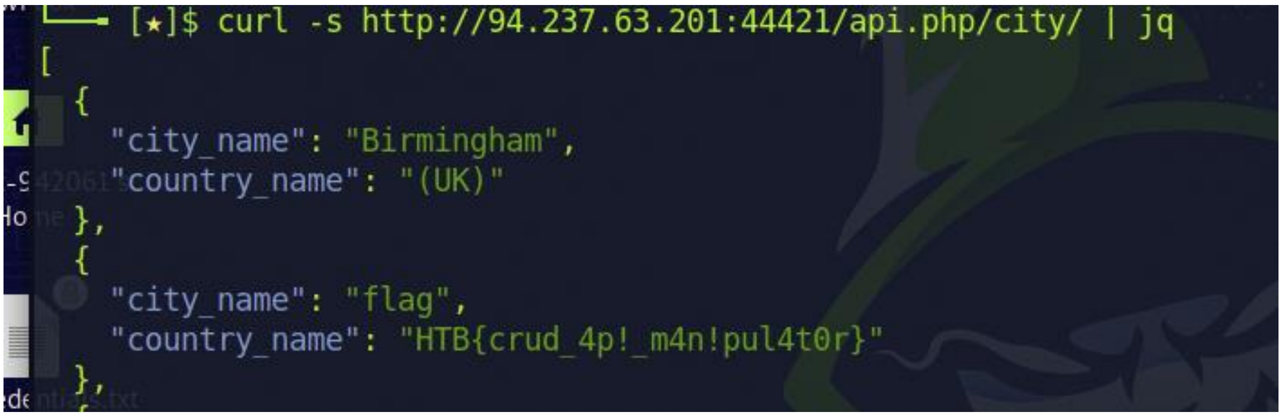
To view the city names available in the API, I performed a GET request to the /city endpoint. Using the curl command **curl -s http://94.237.63.201:44421/api.php/city/ | jq**


```
[]┌─[eu-academy-1]─[10.10.15.135]─[htb-ac-942061@htb-4s8oohkrjs]─[~]
└──╼ [★]$ curl -s http://94.237.63.201:44421/api.php/city/ | jq
[
  {
    "city_name": "Birmingham",
    "country_name": "(UK)"
  },
  {
    "city_name": "Leeds",
    "country_name": "(UK)"
  },
  {
    "city_name": "Glasgow",
    "country_name": "(UK)"
  },
  {
    "city_name": "Sheffield",
    "country_name": "(UK)"
  },
  {
```

I then updated Leeds city name to flag using the command:

curl -X PUT http://94.237.63.201:44421/api.php/city/Leeds -d '{"city_name":"flag", "country_name":"(UK)"}' -H 'Content-Type: application/json'

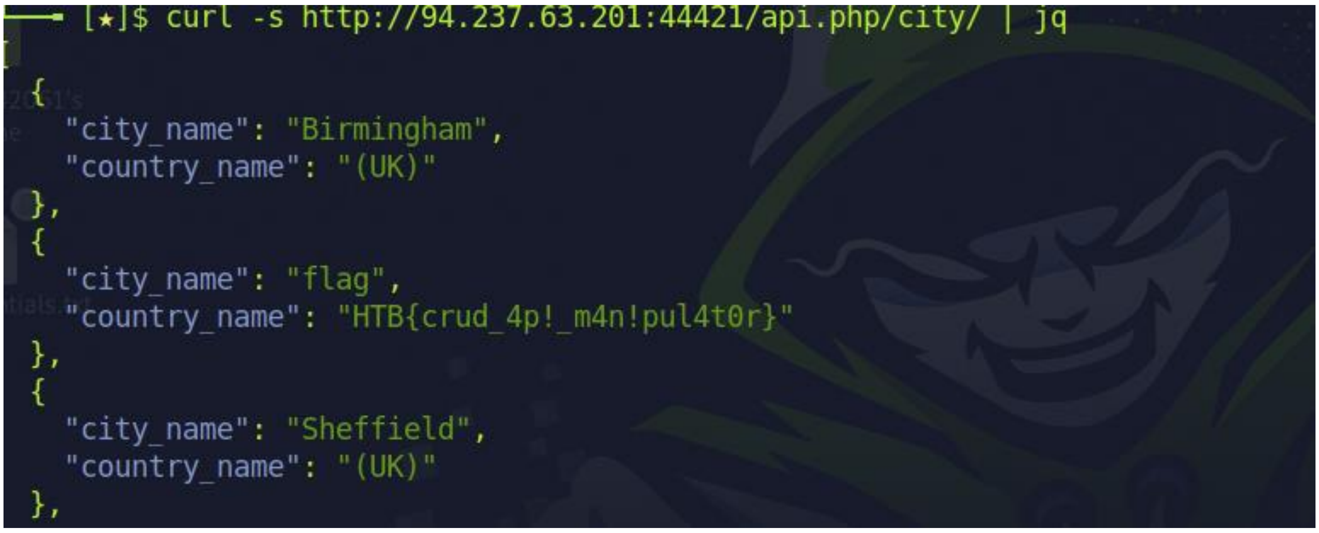
As you can now see, Leeds that was appearing at the second place has now been replaced by a city named flag.


```
└──╼ [★]$ curl -s http://94.237.63.201:44421/api.php/city/ | jq
[
  {
    "city_name": "Birmingham",
    "country_name": "(UK)"
  },
  {
    "city_name": "flag",
    "country_name": "HTB{crud_4p!_m4n!pul4t0r}"
  },
```

I then deleted Glasgow city which appears at number three right below Leeds using the command:

curl -X DELETE http://94.237.63.201:44421/api.php/city/Glasgow

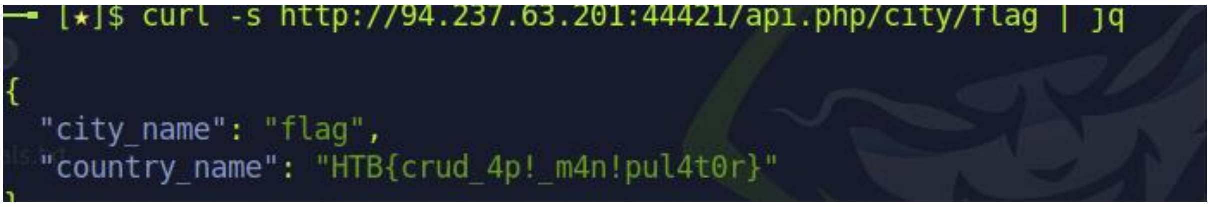
As you can now see, it is Sheffield city that appears in third place right below flag that replaced Leeds.



```
─── [★]$ curl -s http://94.237.63.201:44421/api.php/city/ | jq
[
  {
    "city_name": "Birmingham",
    "country_name": "(UK)"
  },
  {
    "city_name": "flag",
    "country_name": "HTB{crud_4p!_m4n!pul4t0r}"
  },
  {
    "city_name": "Sheffield",
    "country_name": "(UK)"
  },
```

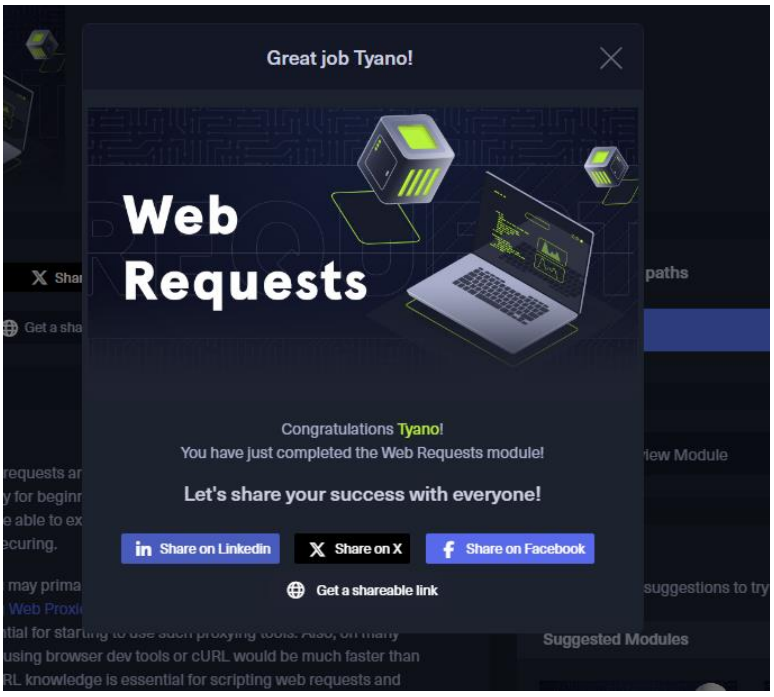
I then searched for a city named flag so that I could get the flag using the command:

curl -s http://94.237.63.201:44421/api.php/city/flag | jq



```
─── [★]$ curl -s http://94.237.63.201:44421/api.php/city/flag | jq
{
  "city_name": "flag",
  "country_name": "HTB{crud_4p!_m4n!pul4t0r}"
```

Proof of completion and here is a link to the completed module**:**
https://academy.hackthebox.com/achievement/942061/35



Great job Tyano!

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

Let's share your success with everyone!

Conclusion

Completing the Web Requests module on Hack The Box Academy has been an enlightening experience. It not only deepened my technical knowledge but also honed my practical skills in analysing and manipulating web requests. The hands-on exercises were particularly beneficial, as they provided real-world scenarios that reinforced the theoretical concepts.

This module has equipped me with the necessary tools to better understand and secure web applications, making me more confident in my ability to detect and respond to potential security threats. As I continue my journey in cybersecurity, the insights and skills gained from this module will undoubtedly be invaluable, laying a solid foundation for more advanced studies and professional endeavours in web security.