

HTB Academy- Web Requests

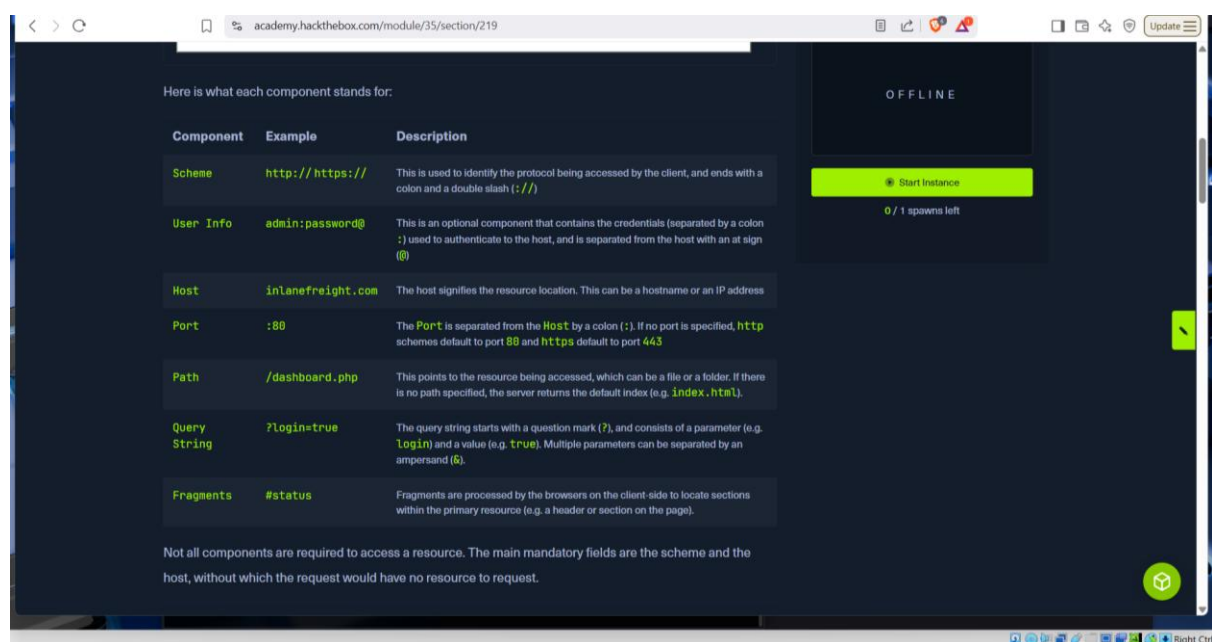
Introduction

This report explores fundamental concepts related to HTTP and web requests as covered in the HTB Academy module. It focuses on the structure and use of HTTP and HTTPS protocols, the behavior of different HTTP methods, and how to inspect requests and responses using tools such as curl and browser developer tools. The tasks provided in the module aim to reinforce an understanding of web communication, especially from a security testing perspective. By engaging in practical exercises, I was able to apply and analyze HTTP behaviors that are commonly encountered in real-world web environments.

HyperText Transfer Protocol (HTTP)

In this Section I covered the Components of a URL, http flow and the Curl Command.

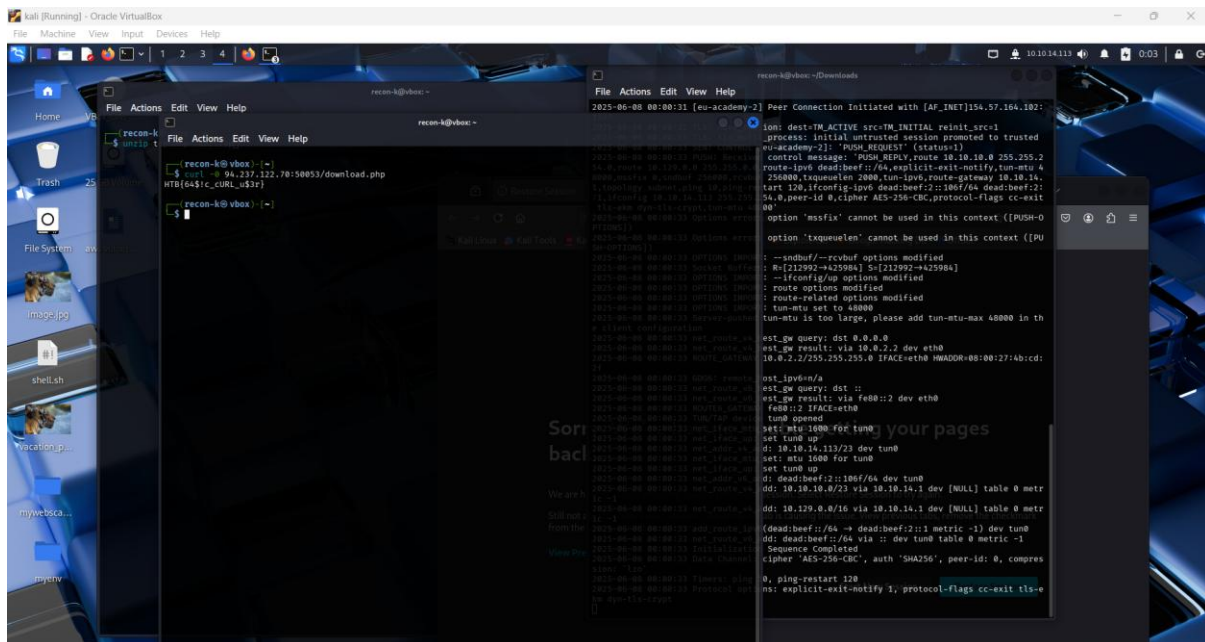
The curl -O flag is used to download a page or a file



Questions:

To get the flag, start the above exercise, then use cURL to download the file returned by `'/download.php'` in the server shown above

Flag: `HTB{64$!c_cURL_u$3r}`



Hypertext Transfer Protocol Secure (HTTPS)

This was created to counter Man in the Middle attacks as most encryption in this case are encrypted.

[academy.hackthebox.com/module/35/section/228](#)

Hypertext Transfer Protocol Secure (HTTPS)

In the previous section, we discussed how HTTP requests are sent and processed. However, one of the significant drawbacks of HTTP is that all data is transferred in clear-text. This means that anyone between the source and destination can perform a Man-in-the-middle (MitM) attack to view the transferred data.

To counter this issue, the **HTTPS (HTTP Secure) protocol** was created, in which all communications are transferred in an encrypted format, so even if a third party does 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, and HTTP is being phased out, and soon most web browsers will not allow visiting HTTP websites.

HTTPS Overview

If we examine an HTTP request, we can see the effect of not enforcing secure communications between a web browser and a web application. For example, the following is the content of an HTTP login request:

```

74.573774918 192.168.0.108 192.168.0.108 TCP 78 48388 - 80 [SYN] Seq=0 Win=0 Len=0 MSS=65535 SACK_PERM=1
84.573784134 192.168.0.108 192.168.0.108 TCP 78 0 - 48388 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=65535 SACK_PERM=1
94.573788187 192.168.0.108 192.168.0.108 TCP 68 48388 - 80 [ACK] Seq=1 Ack=1 Win=0 Len=0 MSS=65535 SACK_PERM=1
114.573791672 192.168.0.108 192.168.0.108 HTTP 646 POST /login HTTP/1.1 646 192.168.0.108 192.168.0.108 646 192.168.0.108 192.168.0.108
Host: 192.168.0.108
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:1.9.0.1) Gecko/20100101 Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
Origin: http://192.168.0.108
Cache-Control: no-cache
Connection: close
Cookie: PHPSESSID=192.168.0.108

```

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: 48388, Dst Port: 80, Seq: 1, Ack: 1, Len: 572
Hypertext Transfer Protocol
Host: 192.168.0.108
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:1.9.0.1) Gecko/20100101 Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
Origin: http://192.168.0.108
Cache-Control: no-cache
Connection: close
Cookie: PHPSESSID=192.168.0.108

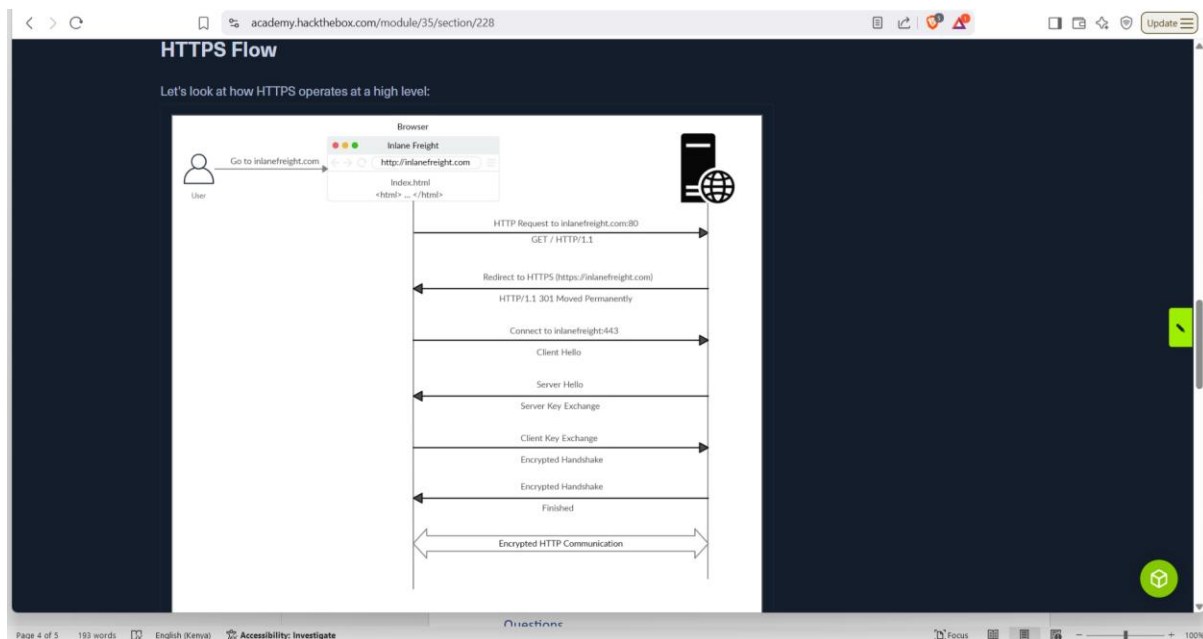
Form item: "username" = "admin"
Key: username
Value: admin
Form item: "password" = "password"
Key: password
Value: password

Cheat Sheet

Table of Contents

- HTTP Fundamentals
 - HyperText Transfer Protocol (HTTP) ☒
 - Hypertext Transfer Protocol Secure (HTTPS) ☒
 - HTTP Requests and Responses ☒
 - HTTP Headers ☒
- HTTP Methods
 - HTTP Methods and Codes ☒
 - GET ☒
 - POST ☒
 - CRUD API ☒
- My Workstation ☒

HTTPS FLOW:



The -k flag on curl command can be used to ignore outdated SSL certificates as curl by default supports https.

HTTP Requests and Responses

I started by understanding the different parts of a Http requests:

which contains the response code, as discussed in a later section, and may contain the resource data if the requester has access to it.

HTTP Request

Let's start by examining the following example HTTP request:

```

GET /users/login.html HTTP/1.1
Host: inlanefreight.com
User-Agent: Mozilla/5.0 (Ubuntu; Linux x86_64;) Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/html; charset=UTF-8
Connection: close
Cookie: PHPSESSID=c4pgt4jull9obt7aup55o8vbf
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
  
```

The image above shows an HTTP GET request to the URL:

- <http://inlanefreight.com/users/login.html>

The first line of any HTTP request contains three main fields 'separated by spaces':

- HTTP Method
- Path
- HTTP version

The headers are separated by a blank line from the request line. The headers are separated by a blank line from the body.

HTTP Headers

- Host: inlanefreight.com
- User-Agent: Mozilla/5.0 (Ubuntu; Linux x86_64;) Firefox/78.0
- Accept: text/html,application/xhtml+xml,application/xml
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- Content-Type: text/html; charset=UTF-8
- Connection: close
- Cookie: PHPSESSID=c4pgt4jull9obt7aup55o8vbf
- Upgrade-Insecure-Requests: 1
- Cache-Control: max-age=0

HTTP Methods

- GET
- POST
- CRUD API

My Workstation

OFFLINE

Start Instance

0 / 1 spawns left

Explanation:

The first line of any HTTP request contains three main fields 'separated by spaces':

Field	Example	Description
Method	GET	The HTTP method or verb, which specifies the type of action to perform.
Path	/users/login.html	The path to the resource being accessed. This field can also be suffixed with a query string (e.g. ?username=user).
Version	HTTP/1.1	The third and final field is used to denote the HTTP version.

The next set of lines contain HTTP header value pairs, like **Host**, **User-Agent**, **Cookie**, and many other possible headers. These headers are used to specify various attributes of a request. The headers are terminated with a new line, which is necessary for the server to validate the request. Finally, a request may end with the request body and data.

Note: HTTP version 1.X sends requests as clear-text, and uses a new-line character to separate different fields and different requests. HTTP version 2.X, on the other hand, sends requests as binary data in a dictionary form.

I then proceeded to http response

Once the server processes our request, it sends its response. The following is an example HTTP response:

```

HTTP/1.1 200 OK
Date: Mon, 13 Jul 2020 10:46:21 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: PHPSESSID=m4u64rqipfthrvbl2ai9voqqf; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 964
Connection: close
Content-Type: text/html; charset=UTF-8

<html lang="en"><head><meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<title>Inline Freight</title>
<link href="/style.css" rel="stylesheet">
</head>

```

The first line of an HTTP response contains two fields separated by spaces. The first being the **HTTP version** (e.g. HTTP/1.1), and the second denotes the **HTTP response code** (e.g. 200 OK).

Response codes are used to determine the request's status, as will be discussed in a later section. After the first line, the response lists its headers, similar to an HTTP request. Both request and response headers are discussed in the next section.

Explanation:

The first line of an HTTP response contains two fields separated by spaces. The first being the **HTTP version** (e.g. **HTTP/1.1**), and the second denotes the **HTTP response code** (e.g. **200 OK**).

Response codes are used to determine the request's status, as will be discussed in a later section. After the first line, the response lists its headers, similar to an HTTP request. Both request and response headers are discussed in the next section.

Finally, the response may end with a response body, which is separated by a new line after the headers. The response body is usually defined as **HTML** code. However, it can also respond with other code types such as **JSON**, website resources such as images, style sheets or scripts, or even a document such as a PDF document hosted on the webserver.

To view the full HTTP request and response, we can simply add the **-v** verbose flag on curl.

Browser dev tools can also be used for web assessment during pentesting.

To open the browser devtools in either Chrome or Firefox, we can click [CTRL+SHIFT+I] or simply click [F12]. The devtools contain multiple tabs, each of which has its own use

Questions

What is the HTTP method used while intercepting the request? GET

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

```

recon-k@vbox: ~
$ curl -v 94.237.122.70:50053
* Trying 94.237.122.70:50053...
* Connected to 94.237.122.70 (94.237.122.70) port 50053
* using HTTP/1.x
> GET / HTTP/1.1
> Host: 94.237.122.70:50053
> User-Agent: curl/8.13.0
> Accept: */*
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Sat, 07 Jun 2025 21:38:10 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 348
< Content-Type: text/html; charset=UTF-8
< (IDCTYPE html)
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Blank Page</title>
</head>
<body>
This page is intentionally left blank.
<br>
Using curl should be enough.
</body>
</html>
* Connection #0 to host 94.237.122.70 left intact
recon-k@vbox: ~

```


HTTP Headers

Headers can have one or multiple values, appended after the header name and separated by a colon. We can divide headers into the following categories:

1. General Headers
2. Entity Headers
3. Request Headers
4. Response Headers
5. Security Headers

Security Headers

Finally, we have **Security Headers**. With the increase in the variety of browsers and web-based attacks, defining certain headers that enhanced security was necessary. HTTP Security headers are **a class of response headers used to specify certain rules and policies** to be followed by the browser while accessing the website.

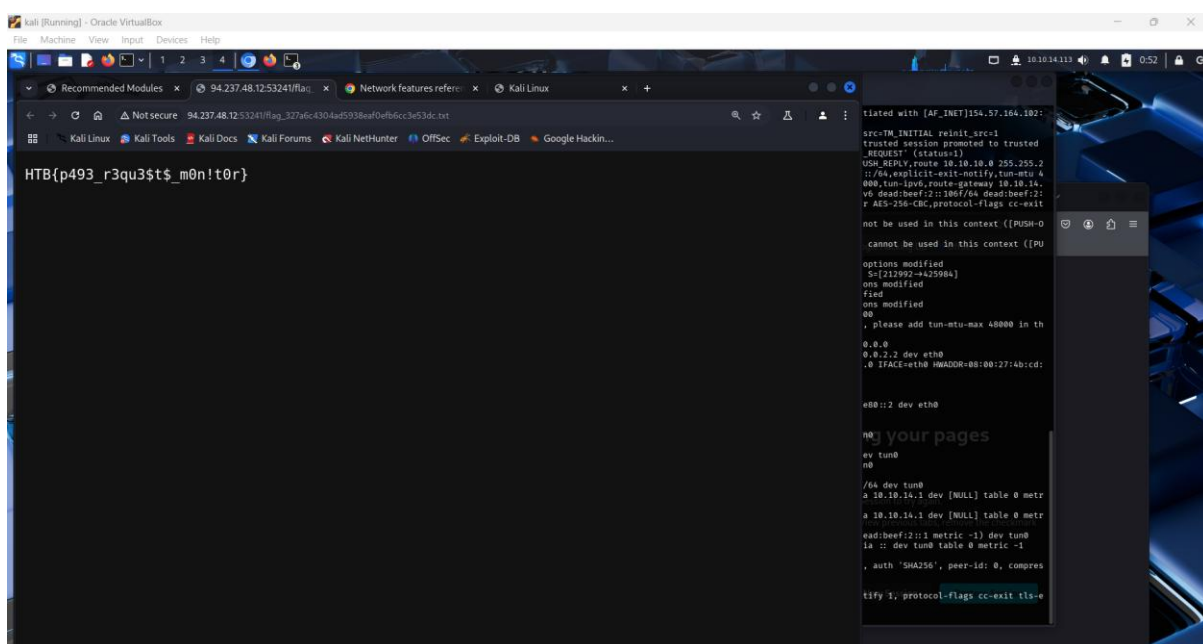
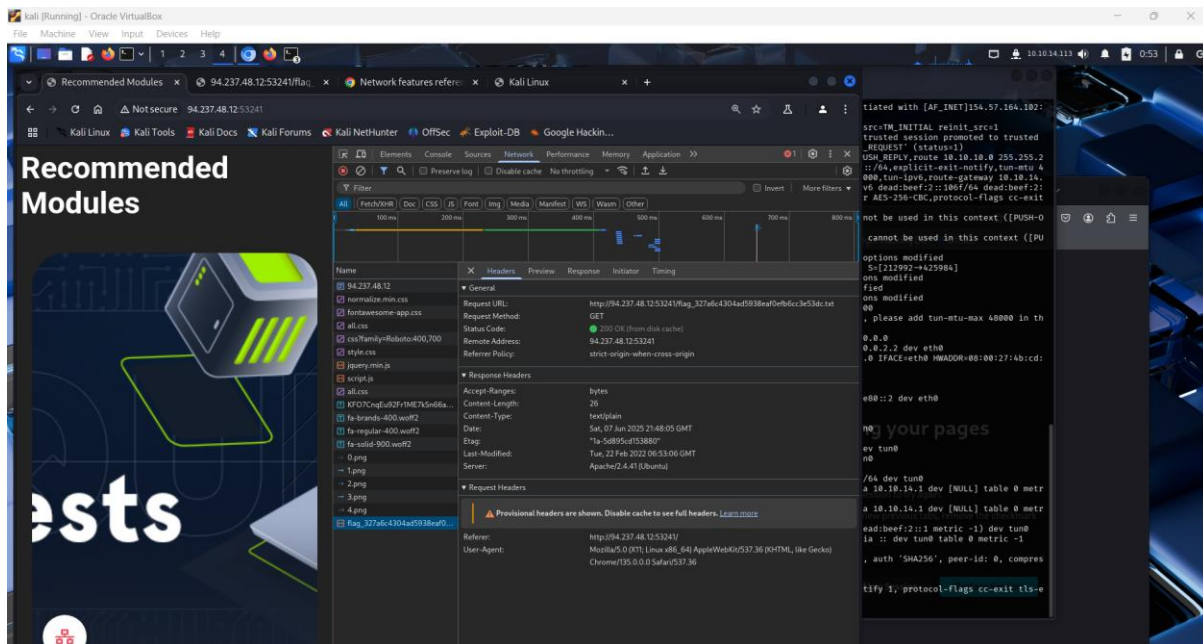
Header	Example	Description
Content-Security-Policy	Content-Security-Policy: script-src 'self'	Dictates the website's policy towards externally injected resources. This could be JavaScript code as well as script resources. This header instructs the browser to accept resources only from certain trusted domains, hence preventing attacks such as Cross-site scripting (XSS) .
Strict-Transport-Security	Strict-Transport-Security: max-age=31536000	Prevents the browser from accessing the website over the plaintext HTTP protocol, and forces all communication to be carried over the secure HTTPS protocol. This prevents attackers from sniffing web traffic and accessing protected information such as passwords or other sensitive data.
Referrer-Policy	Referrer-Policy: origin	Dictates whether the browser should include the value specified via the Referer header or not. It can help in avoiding disclosing sensitive URLs and information while browsing the website.

Note: This section only mentions a small subset of commonly seen HTTP headers. There are many other contextual headers that can be used in HTTP communications. It's also possible for applications to define custom headers based on their requirements. A complete list of standard HTTP headers can be found [here](#).

In curl, If we were only interested in seeing the response headers, then we can use the -I flag to send a HEAD request and only display the response headers. Furthermore, we can use the -i flag to display both the headers and the response body (e.g. HTML code). The difference between the two is that -I sends a HEAD request (as will see in the next section), while -i sends any request we specify and prints the headers as well.

Questions

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.



HTTP Methods and Codes

HTTP supports multiple methods for accessing a resource. In the HTTP protocol, several request methods allow the browser to send information, forms, or files to the server. These methods are used, among other things, to tell the server how to process the request we send and how to reply.

also contain the HTTP response code, which states the status of processing our HTTP request.

Request Methods

The following are some of the commonly used methods:

Method	Description
GET	Requests a specific resource. Additional data can be passed to the server via query strings in the URL (e.g. <code>?param=value</code>).
POST	Sends data to the server. It can handle multiple types of input, such as text, PDFs, and other forms of binary data. This data is appended in the request body present after the headers. The POST method is commonly used when sending information (e.g. forms/logins) or uploading data to a website, such as images or documents.
HEAD	Requests the headers that would be returned if a GET request was made to the server. It doesn't return the request body and is usually made to check the response length before downloading resources.
PUT	Creates new resources on the server. Allowing this method without proper controls can lead to uploading malicious resources.
DELETE	Deletes an existing resource on the webserver. If not properly secured, can lead to Denial of Service (DoS) by deleting critical files on the web server.
OPTIONS	Returns information about the server, such as the methods accepted by it.
PATCH	Applies partial modifications to the resource at the specified location.

The list only highlights a few of the most commonly used HTTP methods. The availability of a particular method depends on the server as well as the application configuration. For a full list of HTTP methods, you

HTTP Requests and Responses

HTTP Headers

HTTP Methods

HTTP Methods and Codes

GET

POST

CRUD API

My Workstation

OFFLINE

Start Instance

0 / 1 spawns left

Page 9 of 10 493 words English (Kenya) Accessibility: Investigate

Response codes

Response Codes

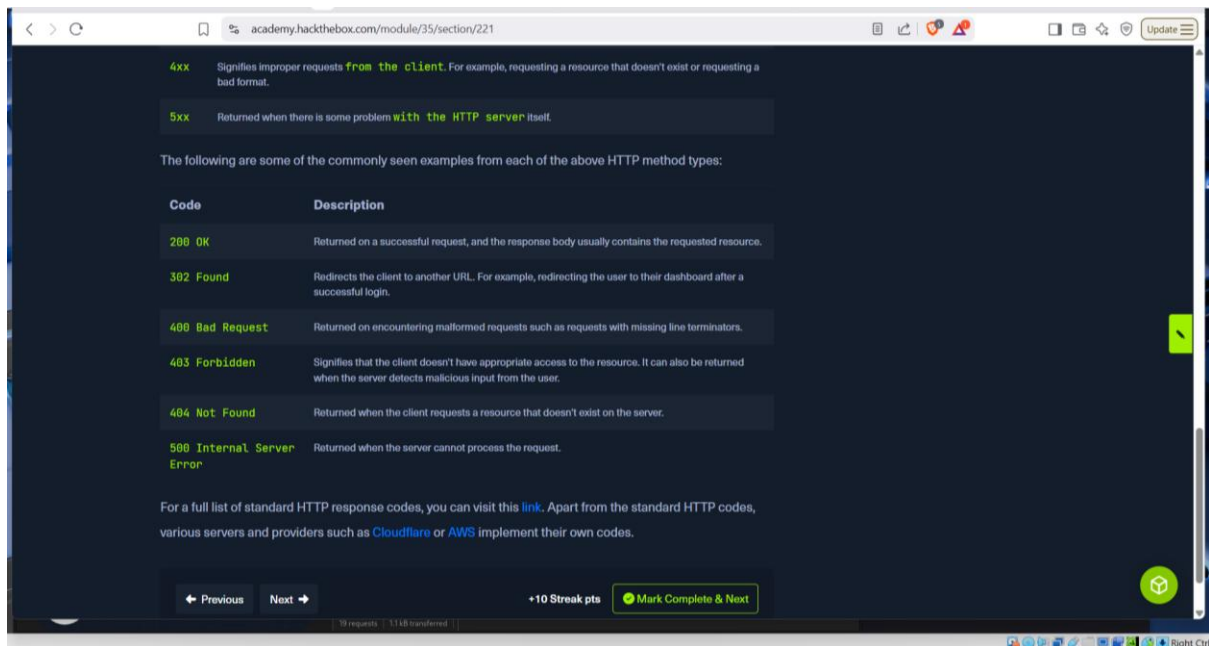
HTTP status codes are used to tell the client the status of their request. An HTTP server can return five types of response codes:

Type	Description
1xx	Provides information and does not affect the processing of the request.
2xx	Returned when a request succeeds.
3xx	Returned when the server redirects the client.
4xx	Signifies improper requests from the client. For example, requesting a resource that doesn't exist or requesting a bad format.
5xx	Returned when there is some problem with the HTTP server itself.

The following are some of the commonly seen examples from each of the above HTTP method types:

Code	Description
200 OK	Returned on a successful request, and the response body usually contains the requested resource.
302 Found	Redirects the client to another URL. For example, redirecting the user to their dashboard after a successful login.
400 Bad Request	Returned on encountering malformed requests such as requests with missing line terminators.

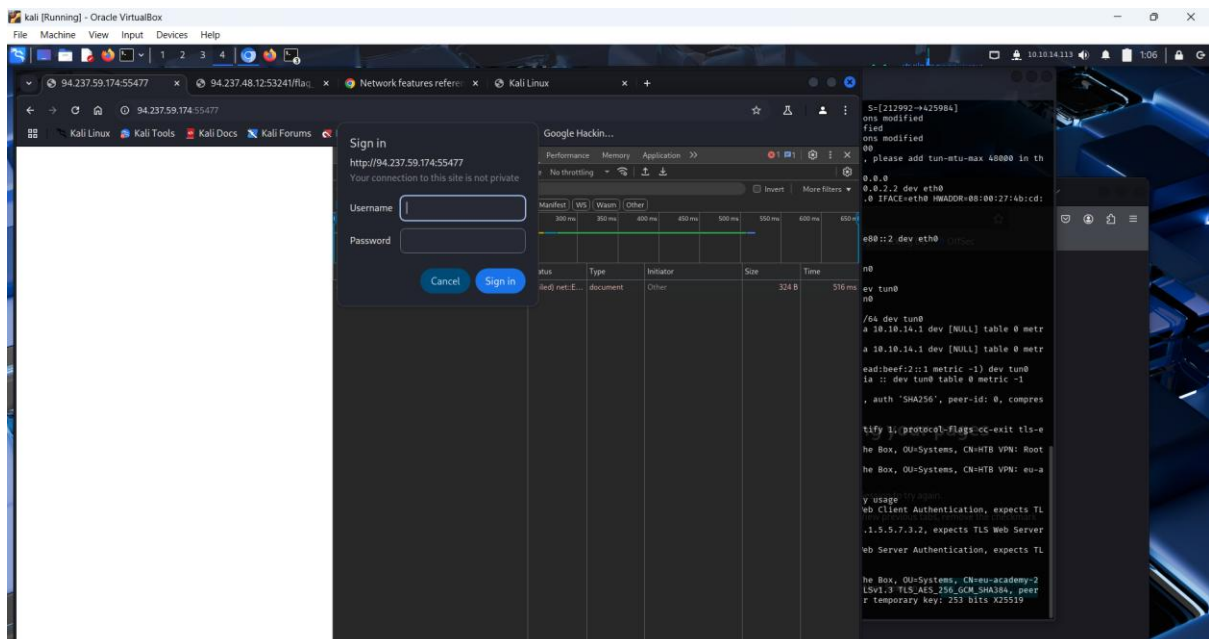
Page 10 of 10 542 words English (Kenya) Accessibility: Investigate



GET

Whenever we visit any URL, our browsers default to a GET request to obtain the remote resources hosted at that URL. Once the browser receives the initial page it is requesting; it may send other requests using various HTTP methods.

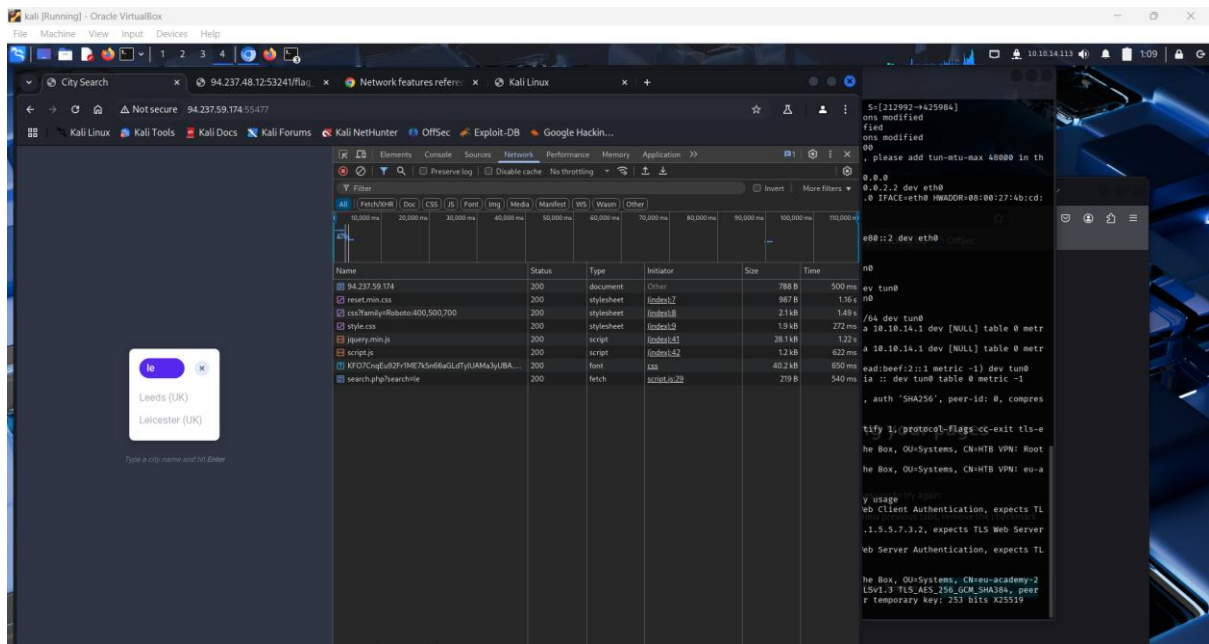
HTTP basic Auth:



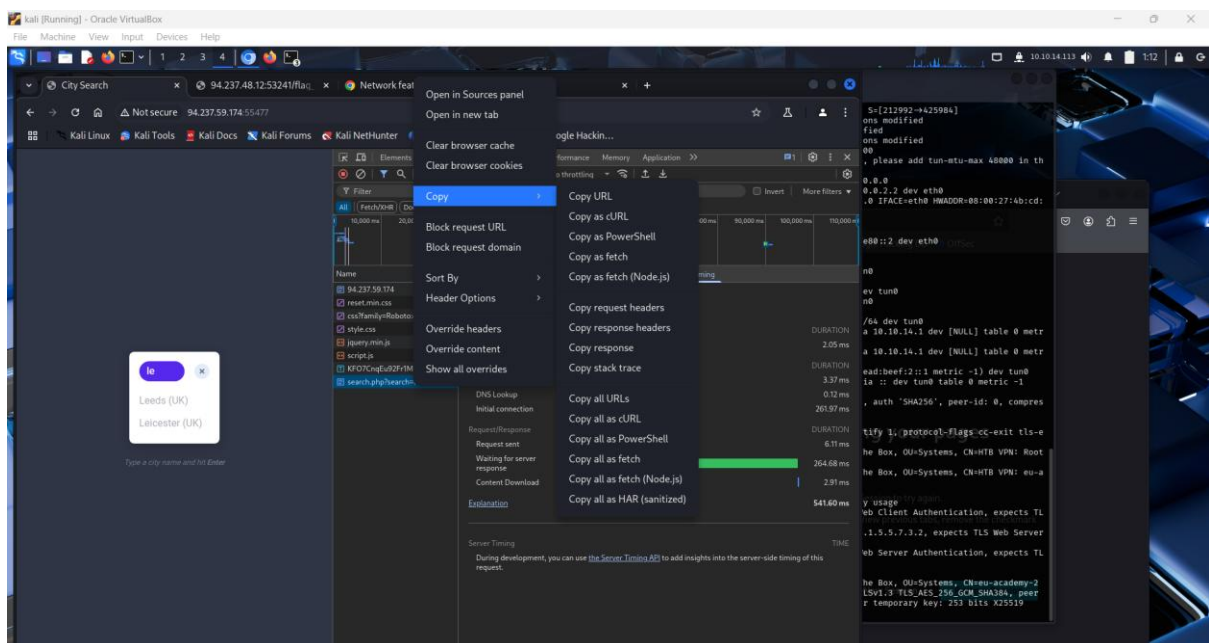
Question:

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}

To do this i entered the ip and port number and went to the developer tools in the network tab

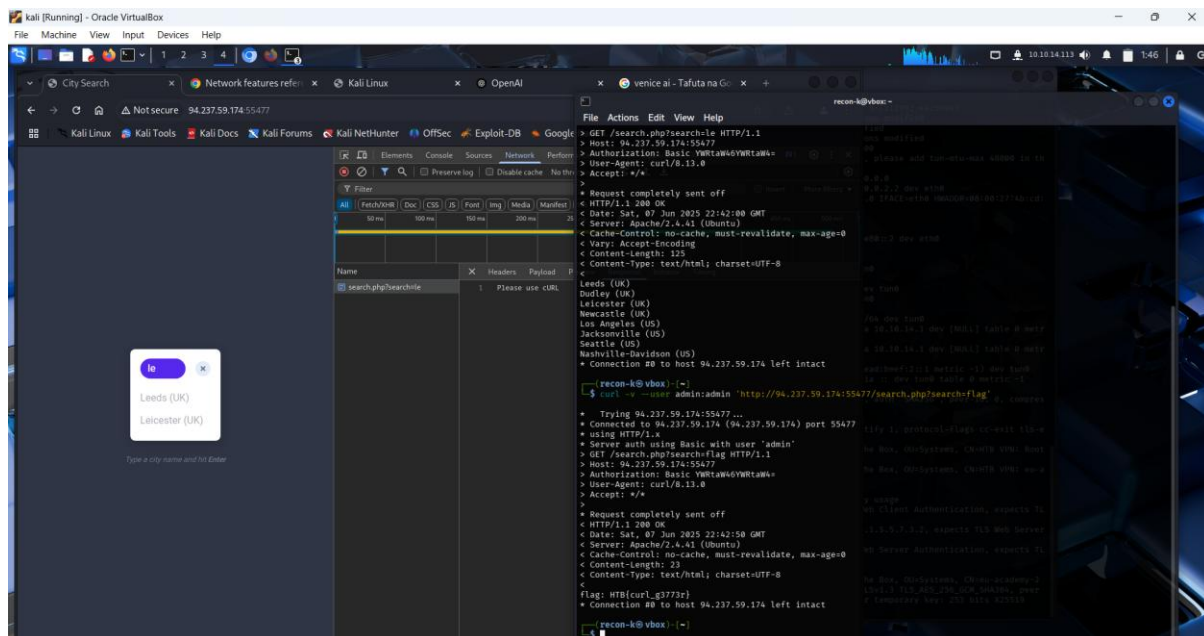


Authenticated and searched “le”, then copied the fetch request using curl



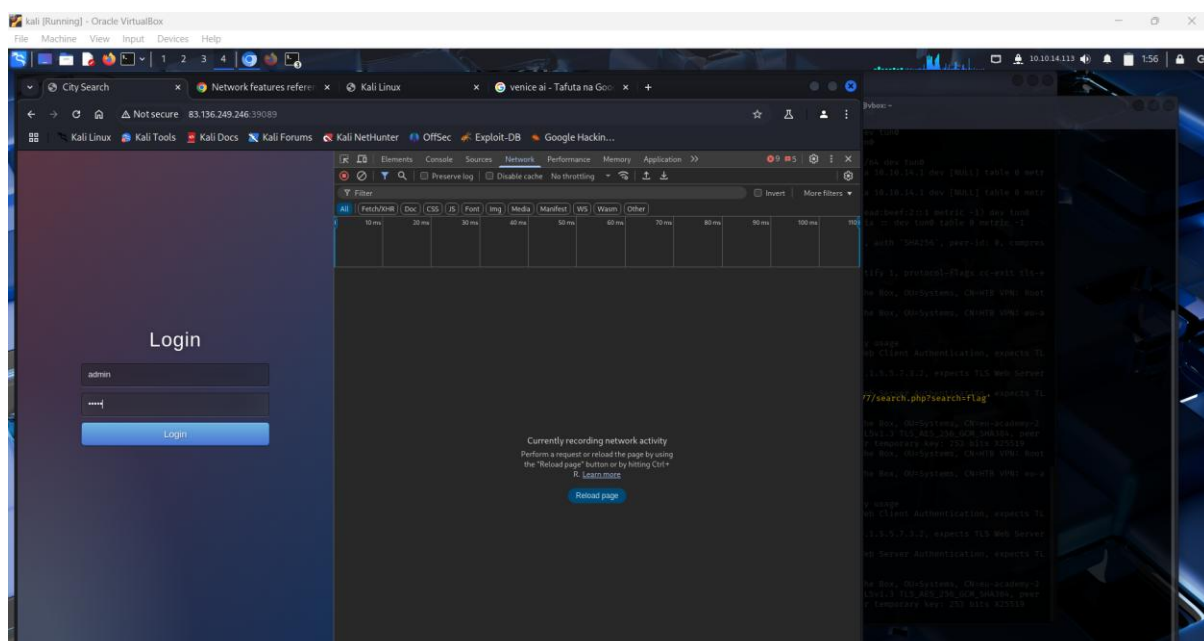
After several failed attempts using the copied curl command from the dev tools i used the basic command with authentication

curl -v --user admin:admin 'http://94.237.59.174:55477/search.php?search=le'



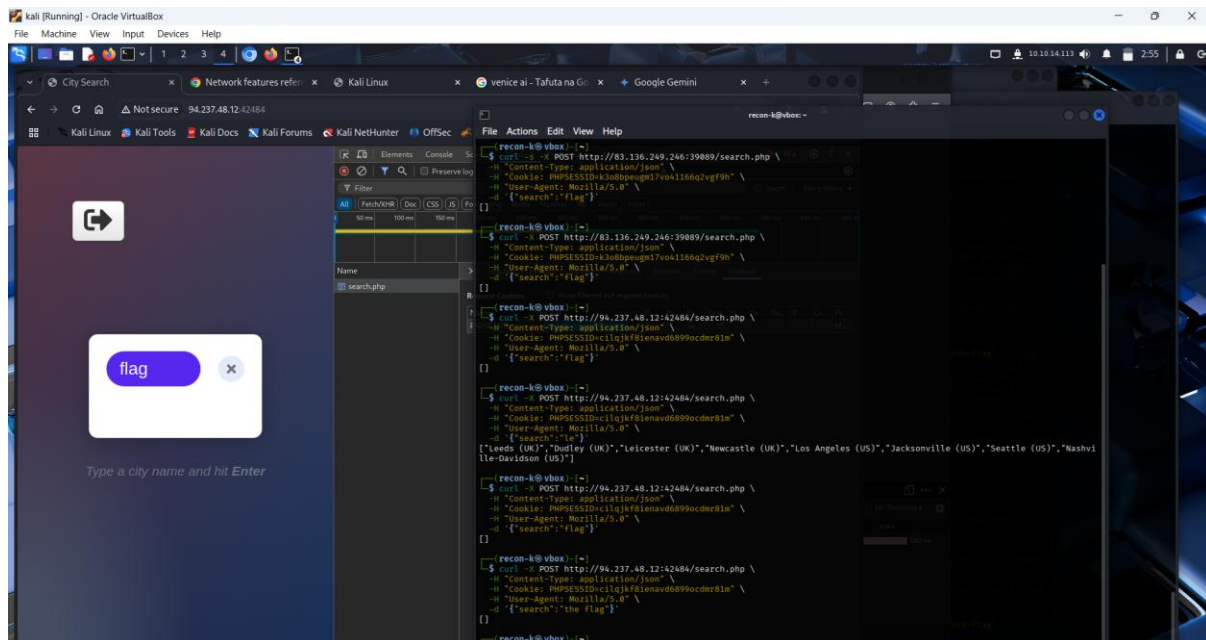
POST

Unlike HTTP GET, which places user parameters within the URL, HTTP POST places user parameters within the HTTP Request body



Questions

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'



CRUD API

CRUD stands for the 4 main operations done by APIs

Create, Read, Update, and Delete

academy.hackthebox.com/module/35/section/227

CRUD

As we can see, we can easily specify the table and the row we want to perform an operation on through such APIs. Then we may utilize different HTTP methods to perform different operations on that row. In general, APIs perform 4 main operations on the requested database entity:

Operation	HTTP Method	Description
Create	POST	Adds the specified data to the database table
Read	GET	Reads the specified entity from the database table
Update	PUT	Updates the data of the specified database table
Delete	DELETE	Removes the specified row from the database table

These four operations are mainly linked to the commonly known CRUD APIs, but the same principle is also used in REST APIs and several other types of APIs. Of course, not all APIs work in the same way, and the user access control will limit what actions we can perform and what results we can see. The [Introduction to Web Applications](#) module further explains these concepts, so you may refer to it for more details about APIs and their usage.

Read

POST

CRUD API

My Workstation

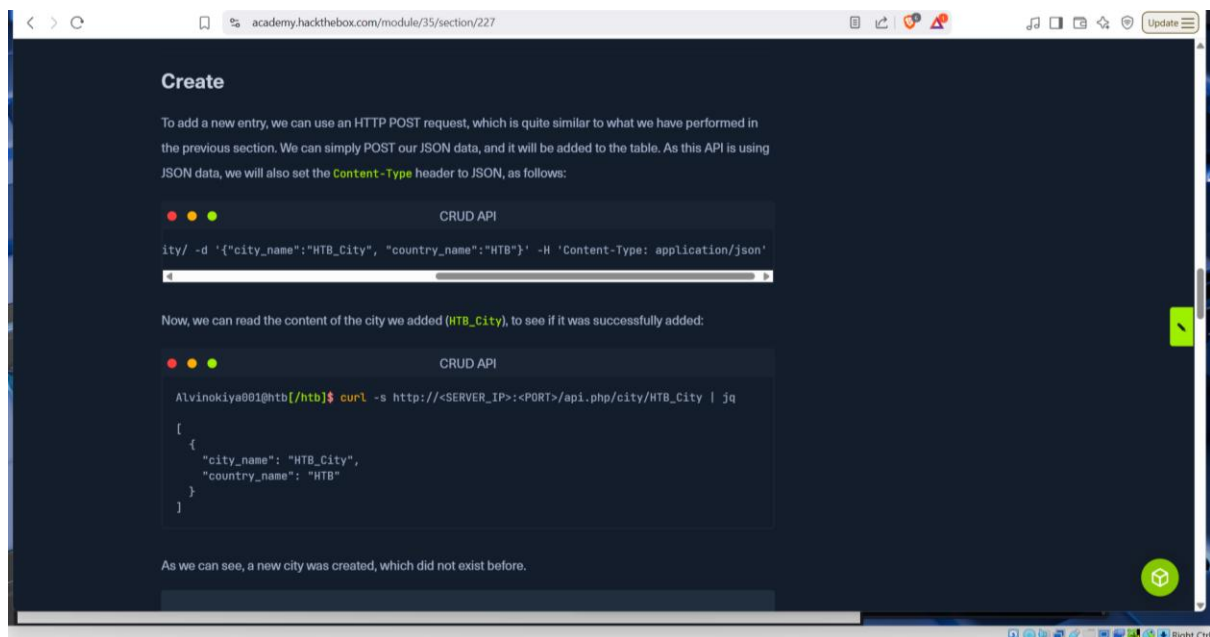
OFFLINE

Start Instance

0 / 1 spawns left

Page 14 of 14 731 words English (Kenya) Accessibility: Investigate

Create commands



Update command

```
curl -X PUT http://<SERVER_IP>:<PORT>/api.php/city/london -d
'{"city_name": "New_HTB_City", "country_name": "HTB"}' -H 'Content-Type:
application/json'
```

Delete Command

```
curl -X DELETE http://<SERVER_IP>:<PORT>/api.php/city/New_HTB_City
```

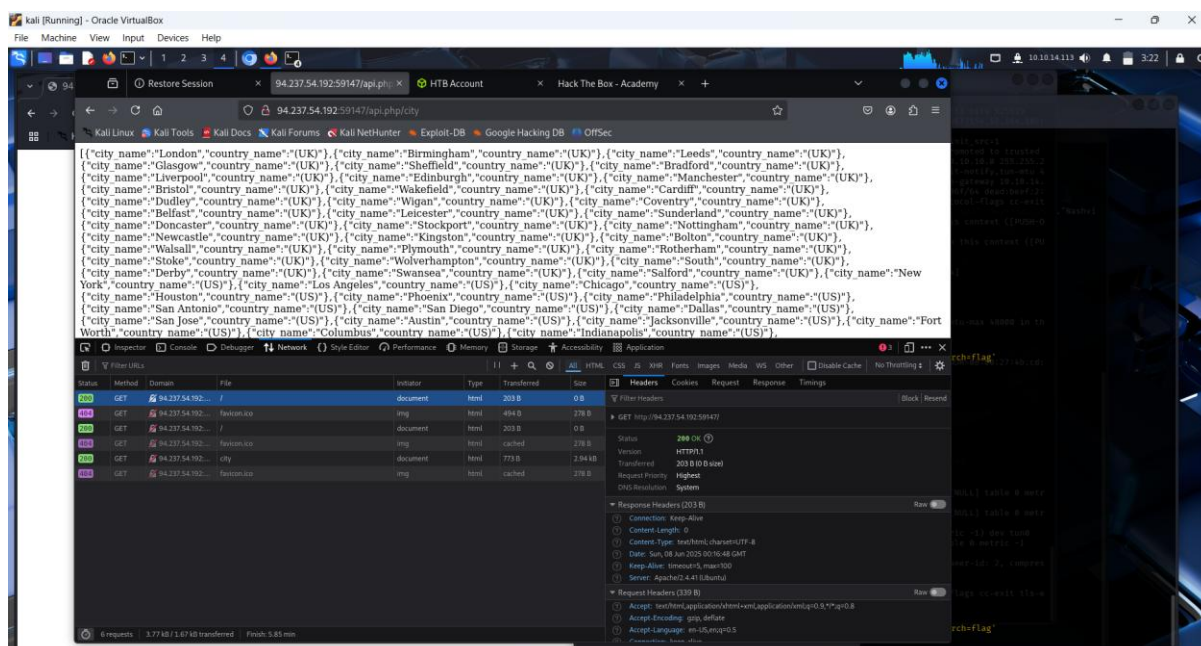
Questions

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.

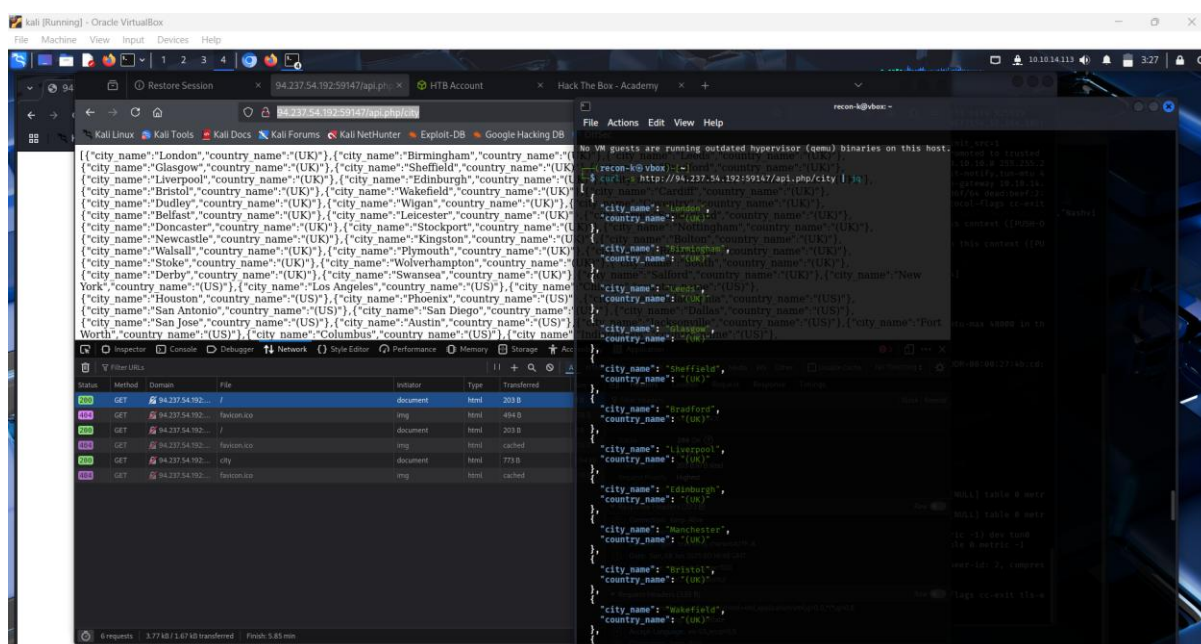
```
HTB{crud_4p!_m4n!pul4t0r}
```

Practical:

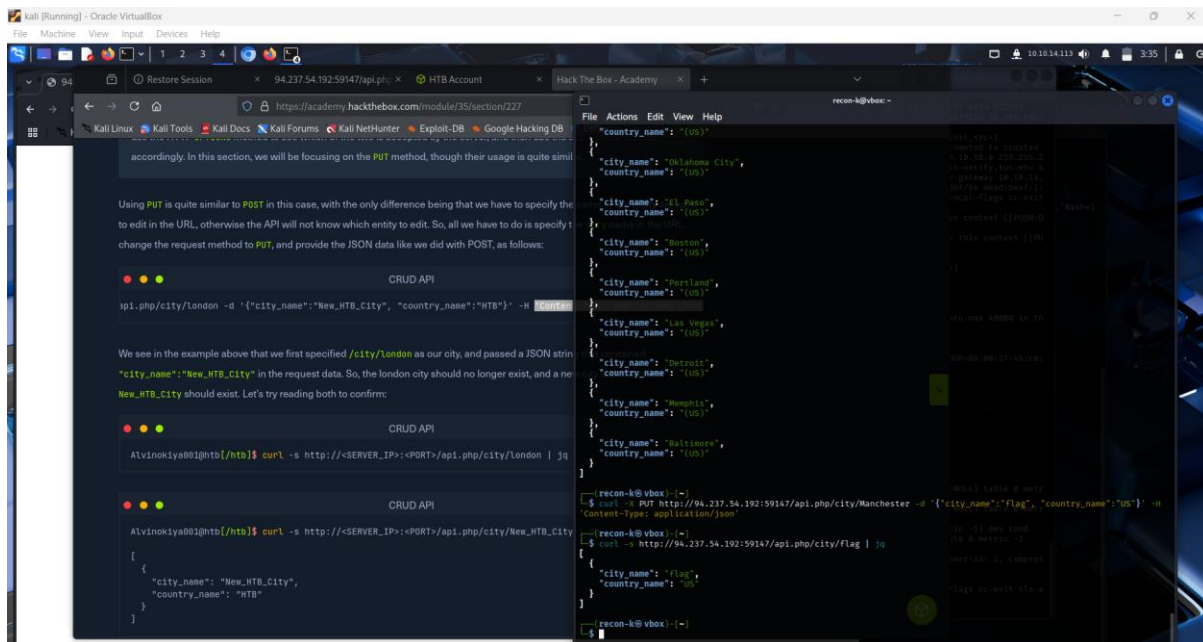
To solve the question, I started by opening the endpoint in a browser, and viewing the cities



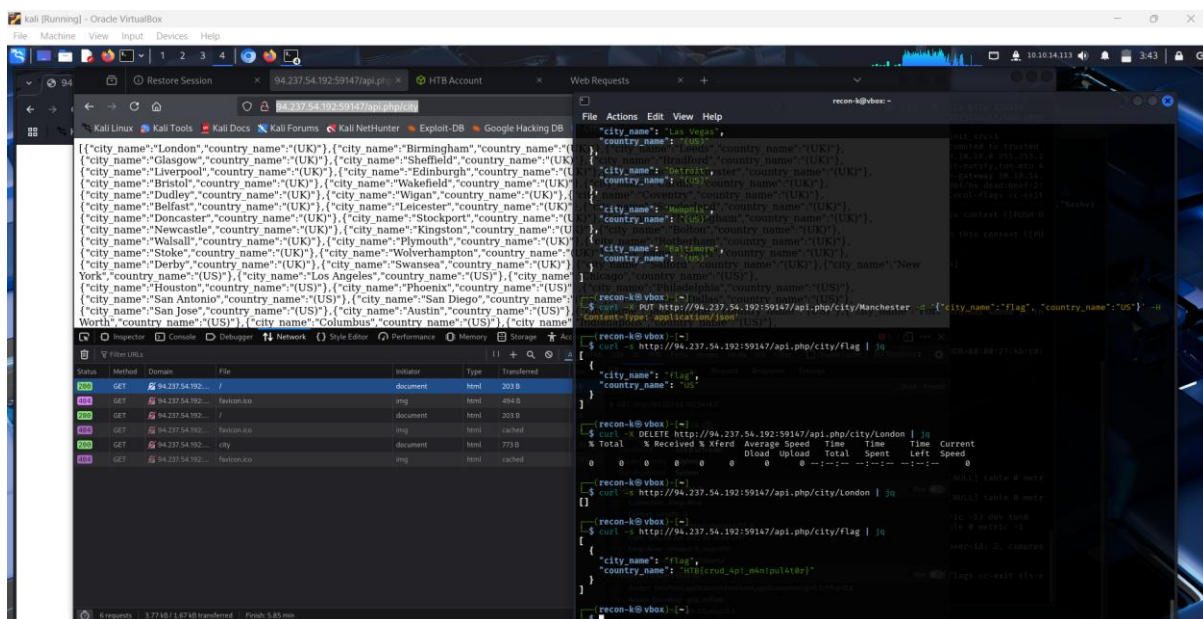
Then using the curl command with jq



I then updated the Manchester city name as flag



After that i Deleted London using the DELETE method and verified whether it exists to get my flag



Conclusion

This assignment provided valuable hands-on experience in understanding how web requests function and how they can be inspected and manipulated for testing purposes. I learned how different HTTP methods work, how headers and status codes communicate important information, and how to use tools like curl and browser dev tools effectively. These skills are foundational for web application security testing and will serve as a critical component in performing future assessments, especially when identifying vulnerabilities and analyzing web server behaviors.