COURSE: CLOUD AND NETWORK SECURITY _C1_2025

STUDENT NAME: DIANA ROSE OGUDA

STUDENT NUMBER: CS-CNS09-25172

TUESDAY ,06 JUNE,2025

WEEK 3 ASSIGNMENT 2

HTB ACADEMY - WEB REQUESTS- ASSIGNMENT REPORT

1. Introduction

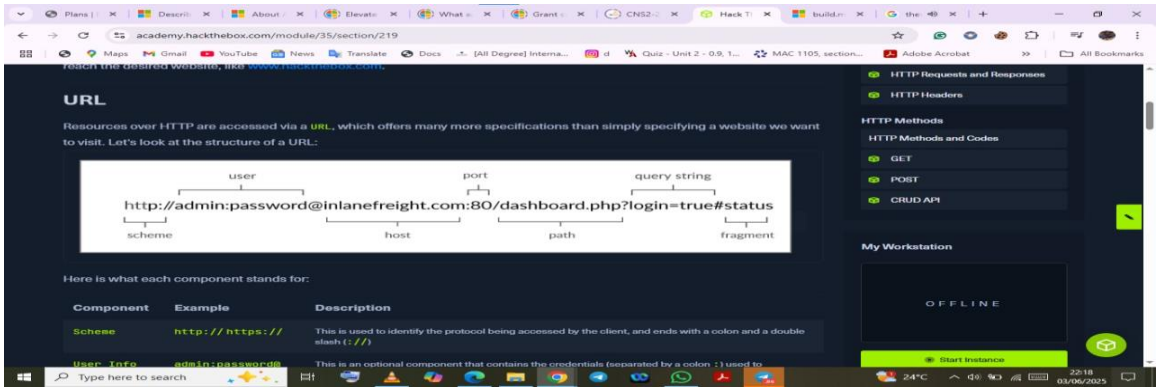
This report documents my learning experience from the "Web Requests" module on Hack The Box Academy. The module introduces key concepts in how web clients and servers communicate over the internet. I explored the mechanics of HTTP and HTTPS, the structure of requests and responses, and how APIs function in web applications. Practical tools such as cURL and browser Developer Tools were used to interact with and analyze web traffic.

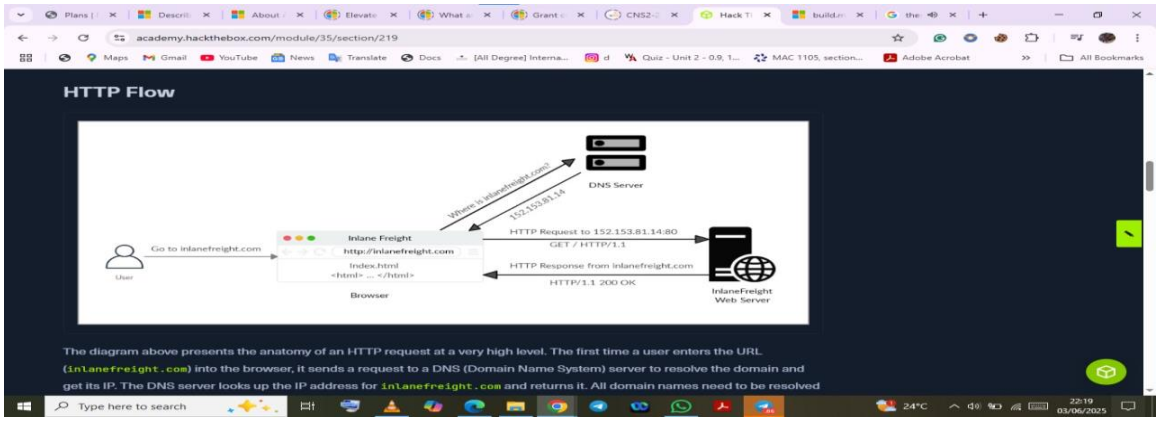
The knowledge gained from this module is foundational to anyone interested in web development, cybersecurity, or penetration testing, as it provides a clear understanding of how information is transferred and manipulated across the web.

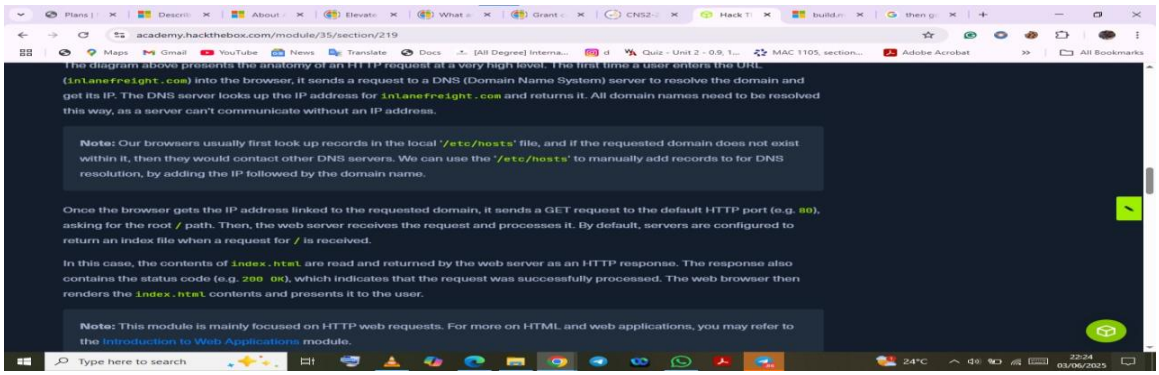
2. Module Experience and Learning Highlights

HTTP and HTTPS

The module began with an overview of HTTP—the Hypertext Transfer Protocol—and its secure counterpart, HTTPS. I learned how HTTP facilitates data exchange between clients and servers, while HTTPS adds an encryption layer using TLS to secure this data. The differences between them were illustrated in terms of data confidentiality, integrity, and authentication, which are essential for securing web traffic.

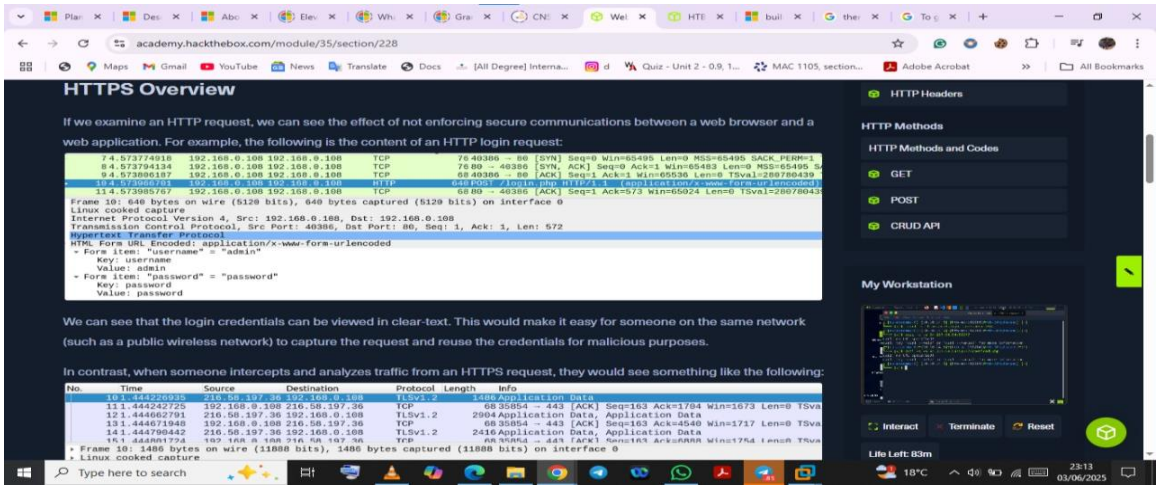




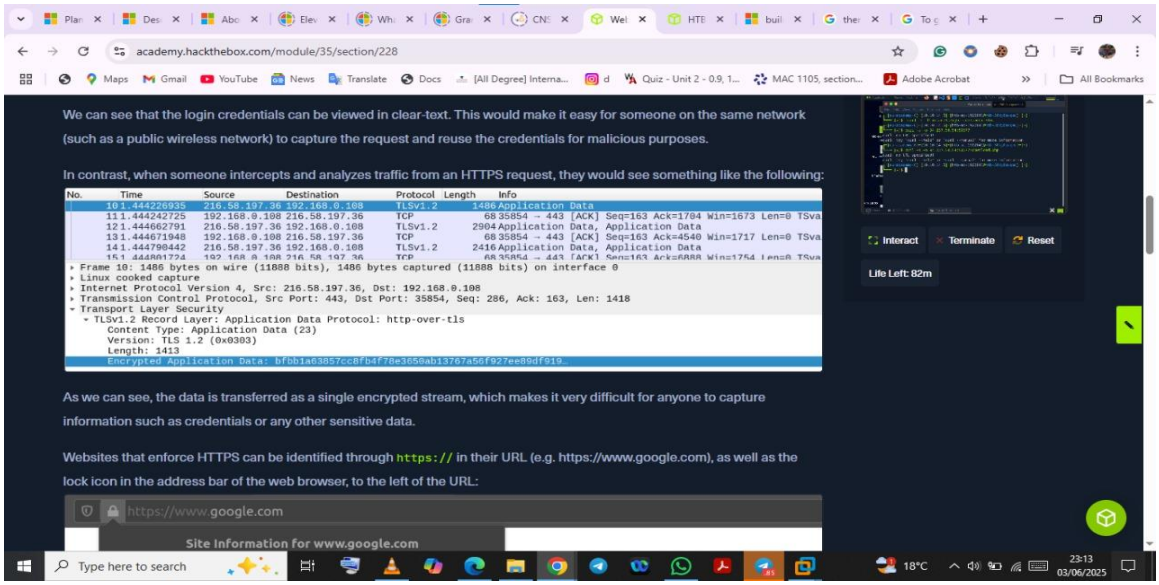


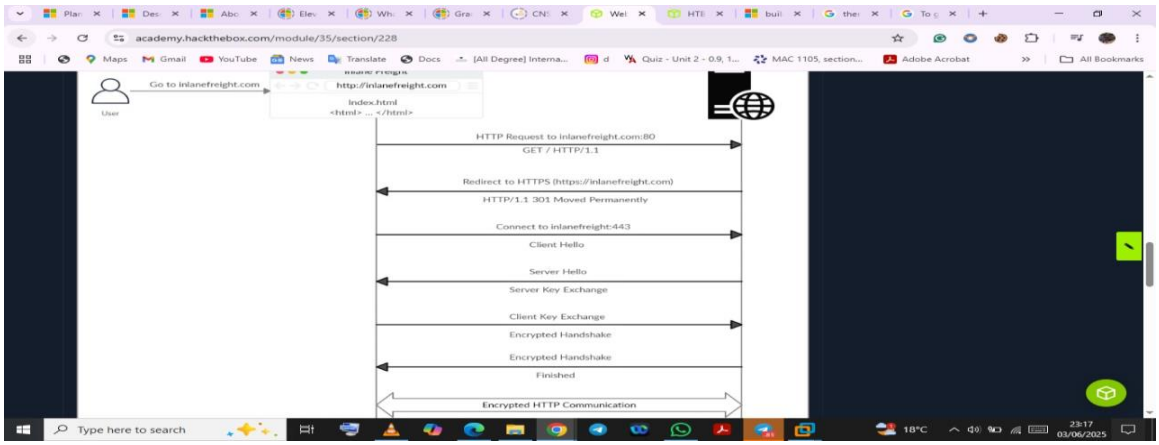
HTTPS

Seeing weakness of HTTP



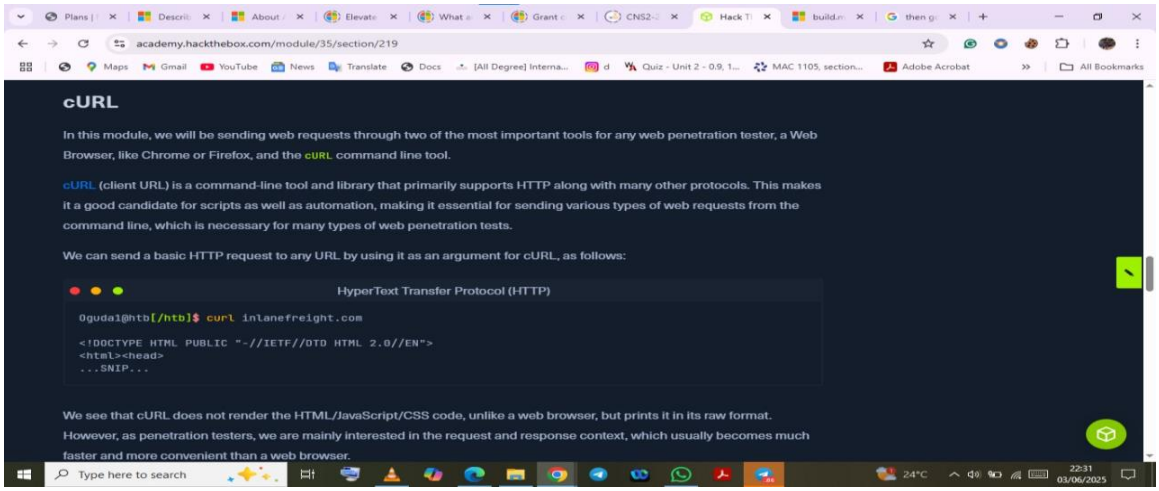
HTTPS added security

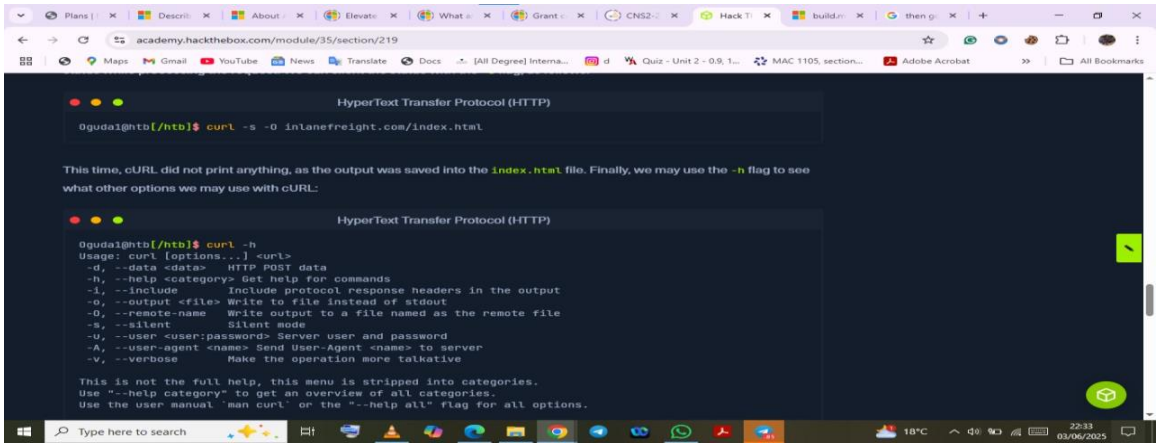




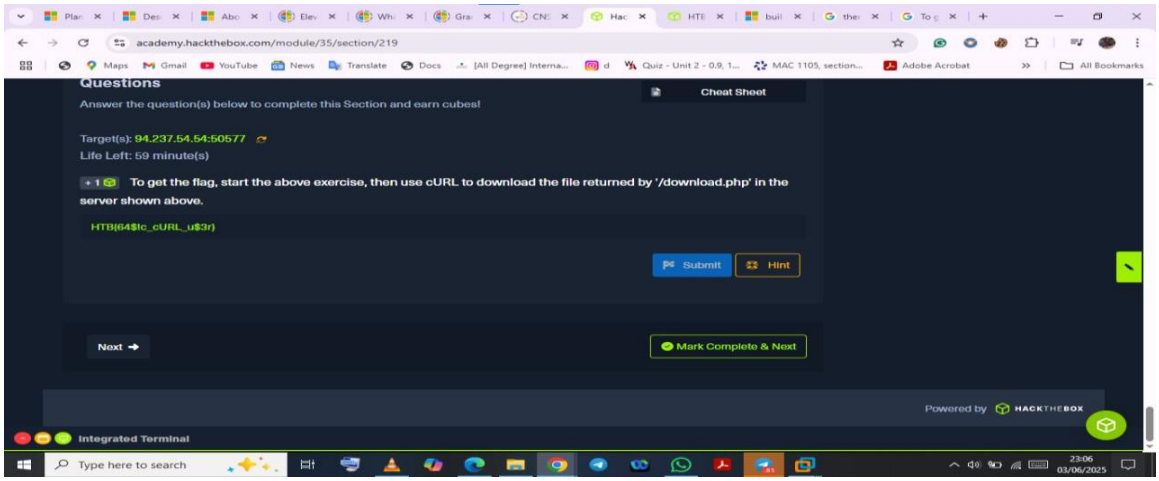
cURL

Using the command-line tool cURL, I practiced sending requests to web servers without using a browser. This included GET, POST, PUT, and DELETE requests. These tasks demonstrated how to structure HTTP requests manually, analyze server responses, and test endpoints. It also gave insight into how attackers and security professionals use such tools during reconnaissance and testing.

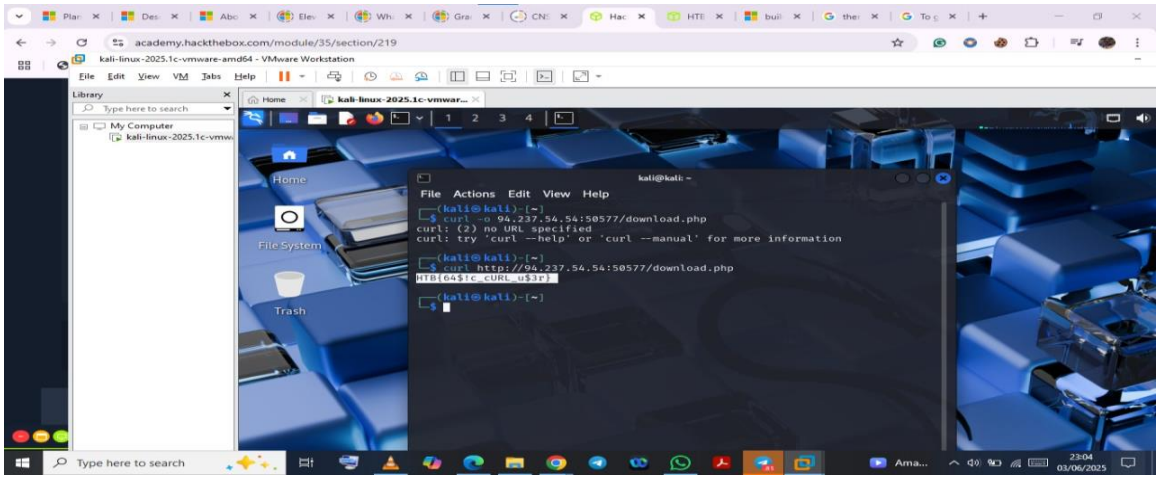




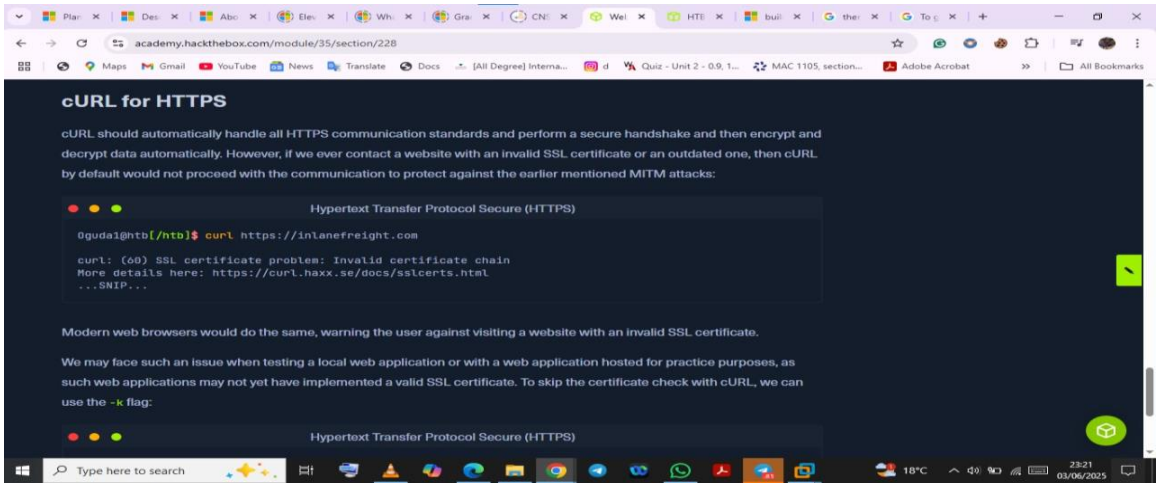
Question

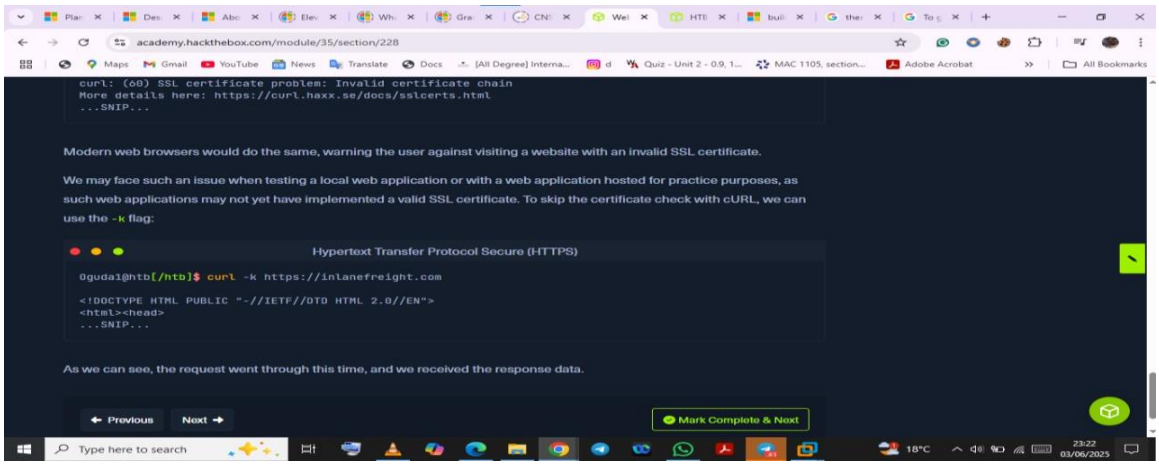


Answer



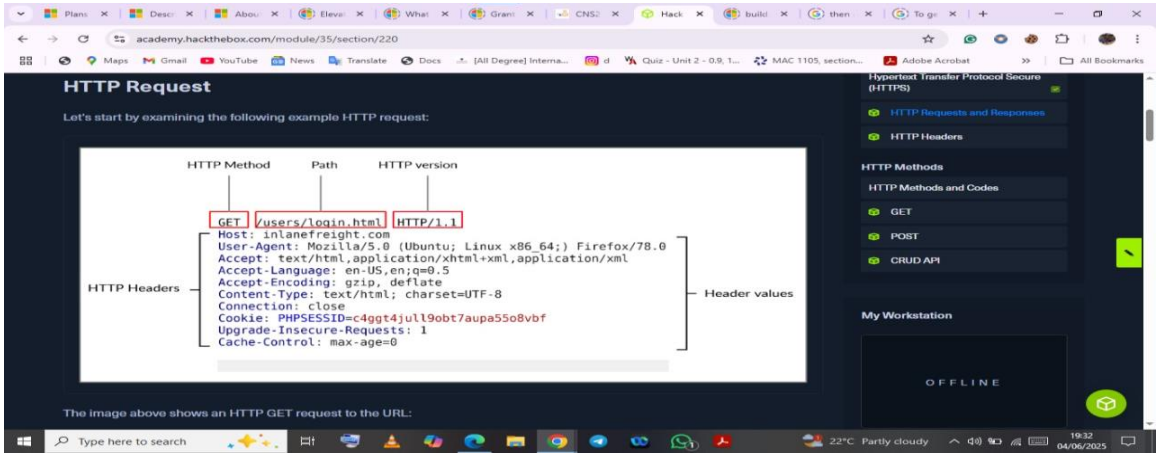
cURL for HTTPS

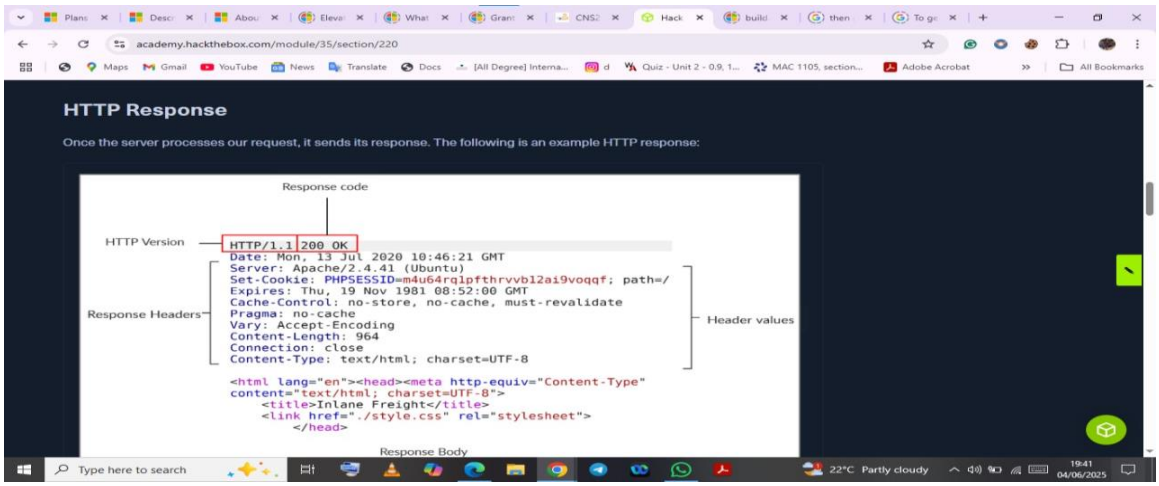


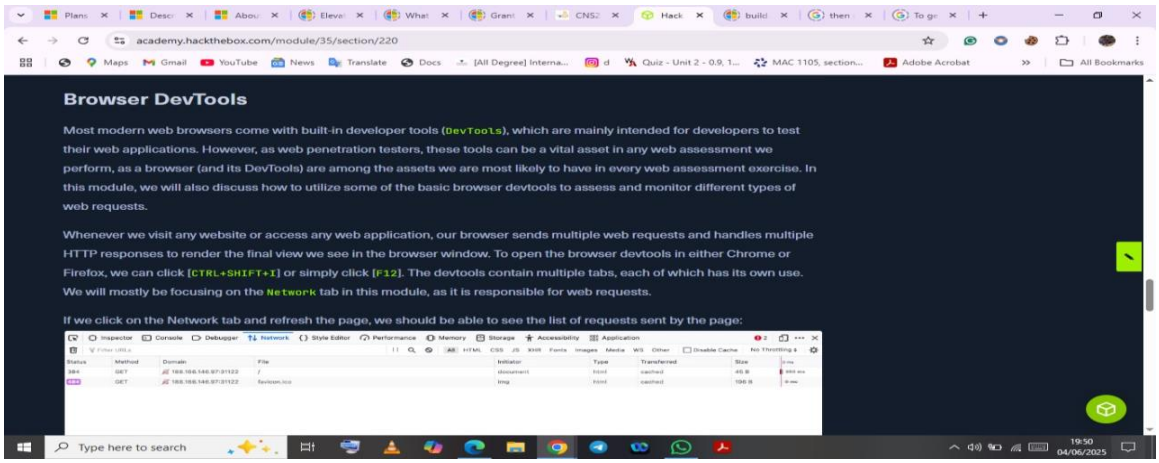


HTTP Requests and Responses

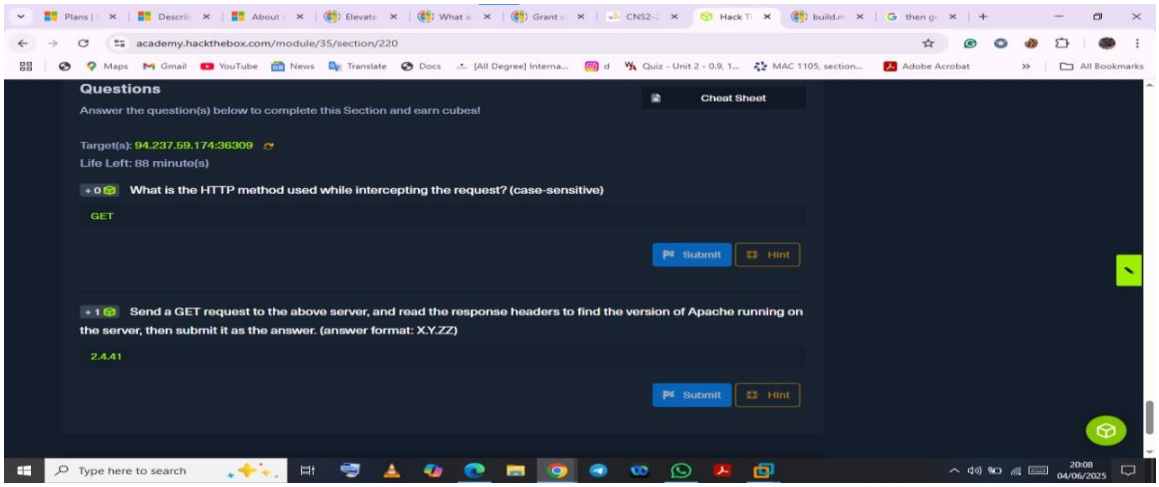
This section covered the anatomy of an HTTP request and the structure of a server's response. I learned how browsers and tools like cURL generate requests containing methods, headers, URLs, and optional payloads, and how responses include status codes, headers, and content. Understanding this flow helped demystify the client-server communication model.



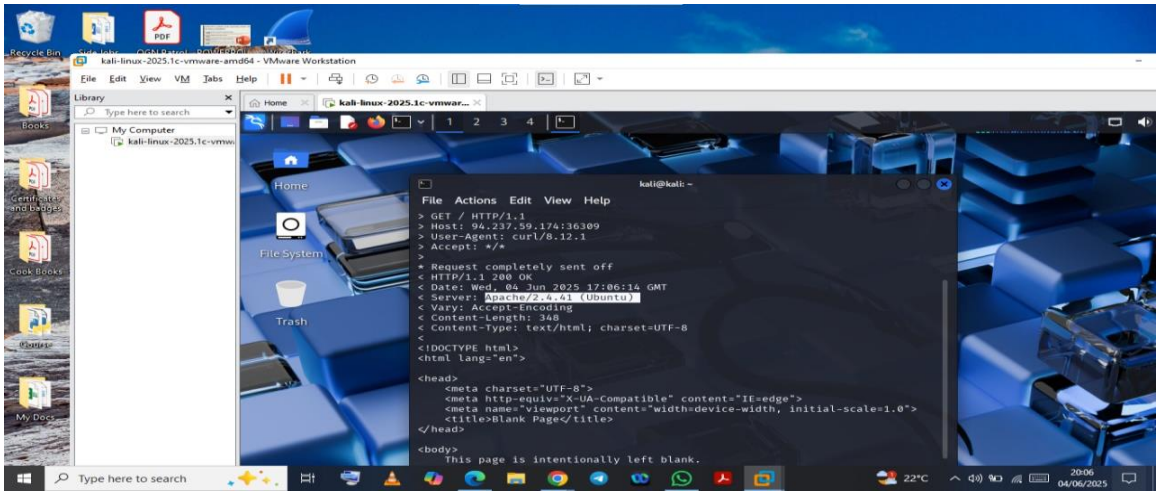




Questions

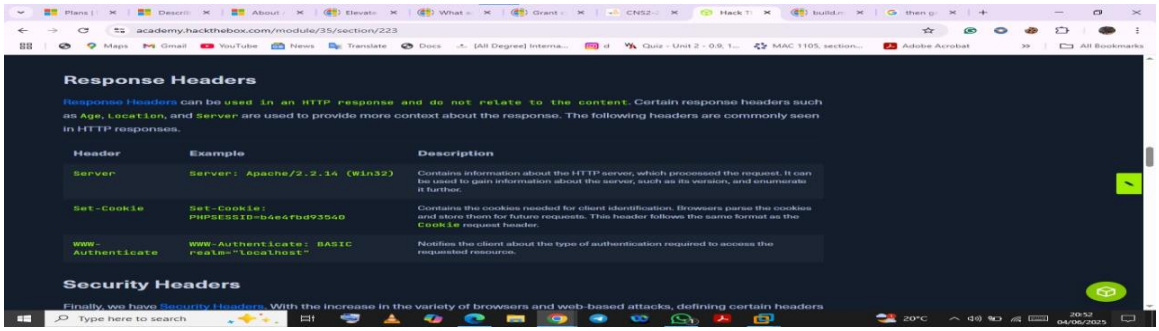


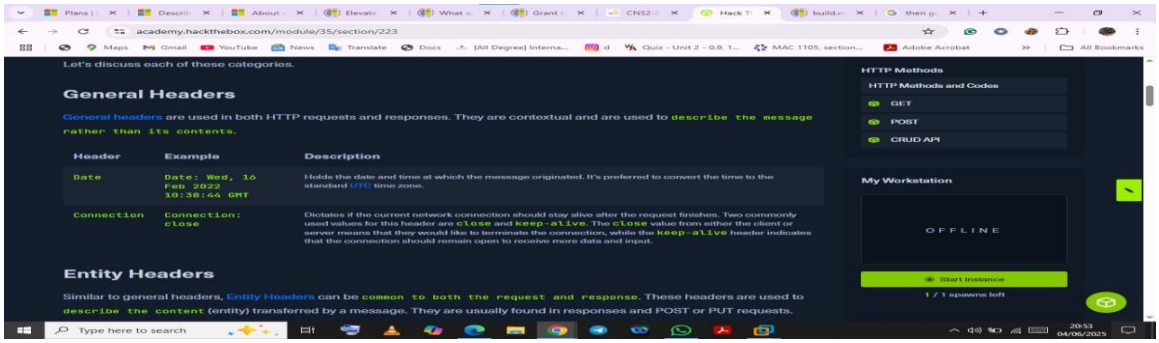
Answer

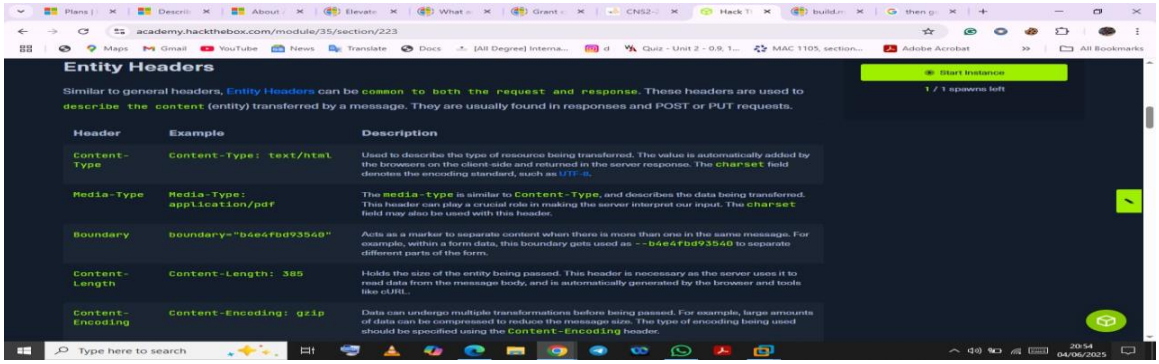


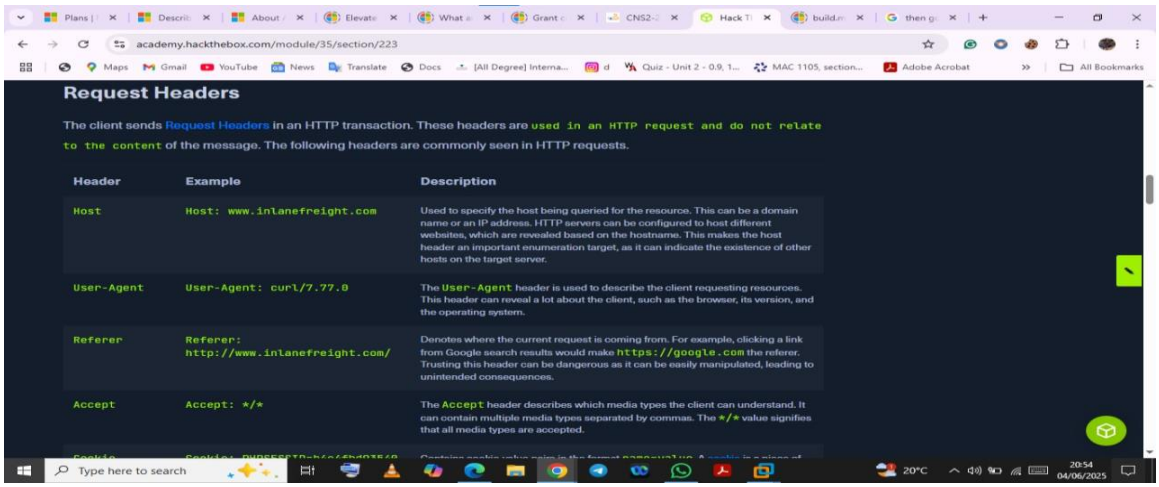
HTTP Headers

Headers in both requests and responses play a crucial role in how data is exchanged and interpreted. I studied key headers like Content-Type, User-Agent, Authorization, Set-Cookie, and Cache-Control, understanding their purpose and implications. This section highlighted how headers can influence behavior, control access, and impact security.





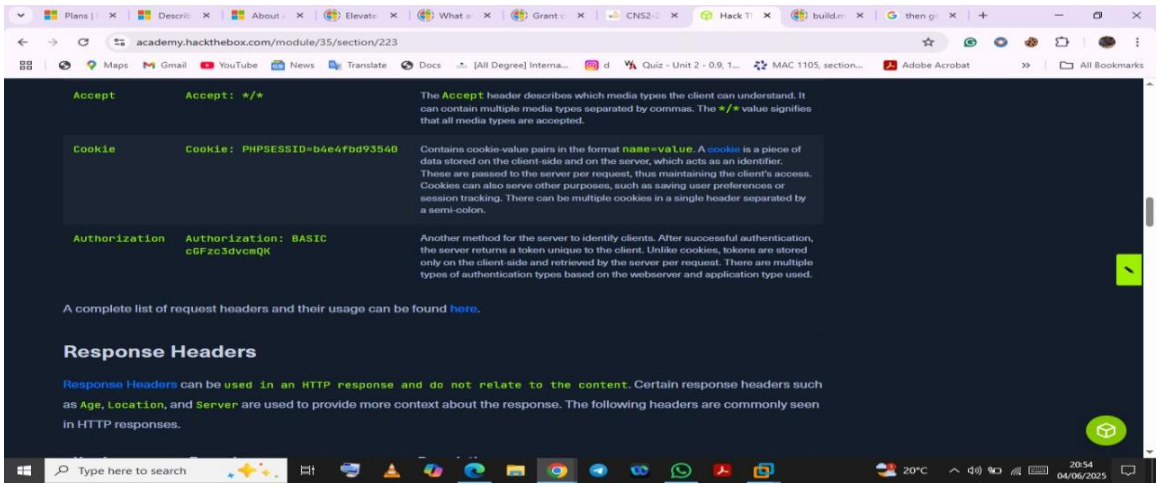




Request Headers

The client sends Request Headers in an HTTP transaction. These headers are used in an HTTP request and do not relate to the content of the message. The following headers are commonly seen in HTTP requests.

Header	Example	Description
Host	Host: www.inlanefreight.com	Used to specify the host being queried for the resource. This can be a domain name or an IP address. HTTP servers can be configured to host different websites, which are revealed based on the hostname. This makes the host header an important enumeration target, as it can indicate the existence of other hosts on the target server.
User-Agent	User-Agent: curl/7.77.0	The User-Agent header is used to describe the client requesting resources. This header can reveal a lot about the client, such as the browser, its version, and the operating system.
Referer	Referer: http://www.inlanefreight.com/	Denotes where the current request is coming from. For example, clicking a link from Google search results would make https://google.com the referer. Trusting this header can be dangerous as it can be easily manipulated, leading to unintended consequences.
Accept	Accept: */*	The Accept header describes which media types the client can understand. It can contain multiple media types separated by commas. The */* value signifies that all media types are accepted.

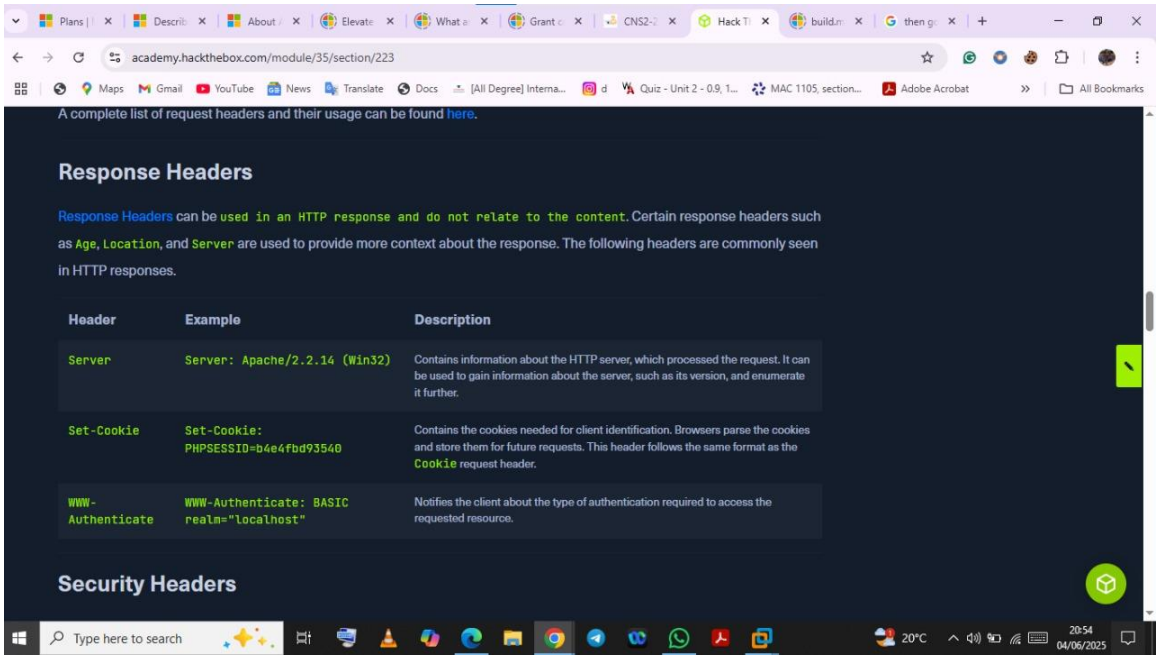


Accept	Accept: */*	The Accept header describes which media types the client can understand. It can contain multiple media types separated by commas. The */* value signifies that all media types are accepted.
Cookie	Cookie: PHPSESSID=b4e4fbd93540	Contains cookie-value pairs in the format name=value. A cookie is a piece of data stored on the client-side and on the server, which acts as an identifier. These are passed to the server per request, thus maintaining the client's access. Cookies can also serve other purposes, such as saving user preferences or session tracking. There can be multiple cookies in a single header separated by a semi-colon.
Authorization	Authorization: BASIC cGFzc3dvcmQK	Another method for the server to identify clients. After successful authentication, the server returns a token unique to the client. Unlike cookies, tokens are stored only on the client-side and retrieved by the server per request. There are multiple types of authentication types based on the webserver and application type used.

A complete list of request headers and their usage can be found here.

Response Headers

Response Headers can be used in an HTTP response and do not relate to the content. Certain response headers such as Age, Location, and Server are used to provide more context about the response. The following headers are commonly seen in HTTP responses.



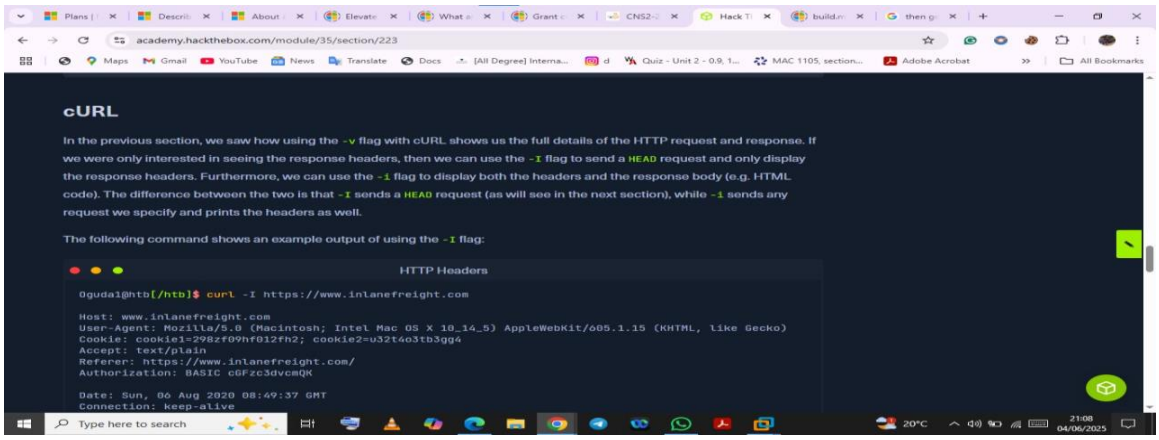
A complete list of request headers and their usage can be found here.

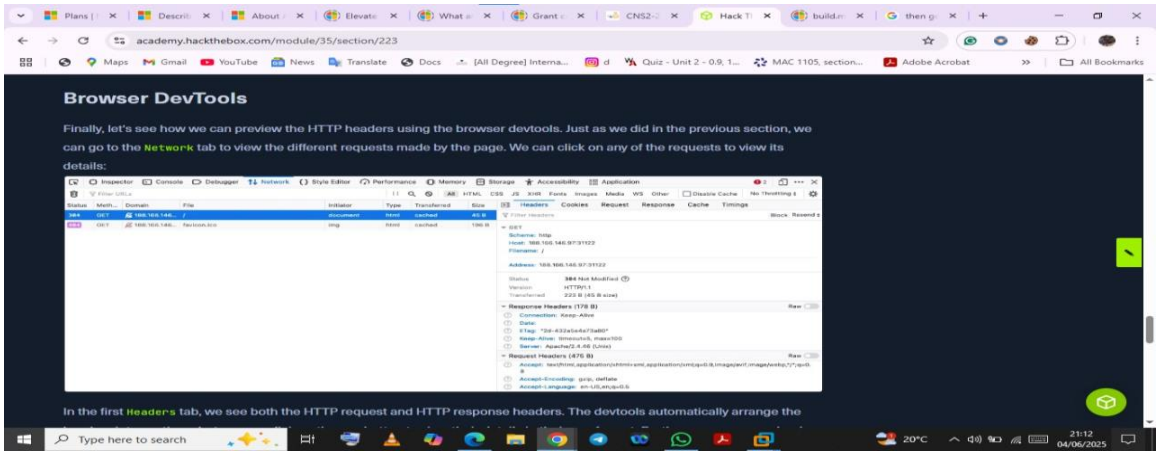
Response Headers

Response Headers can be used in an HTTP response and do not relate to the content. Certain response headers such as Age, Location, and Server are used to provide more context about the response. The following headers are commonly seen in HTTP responses.

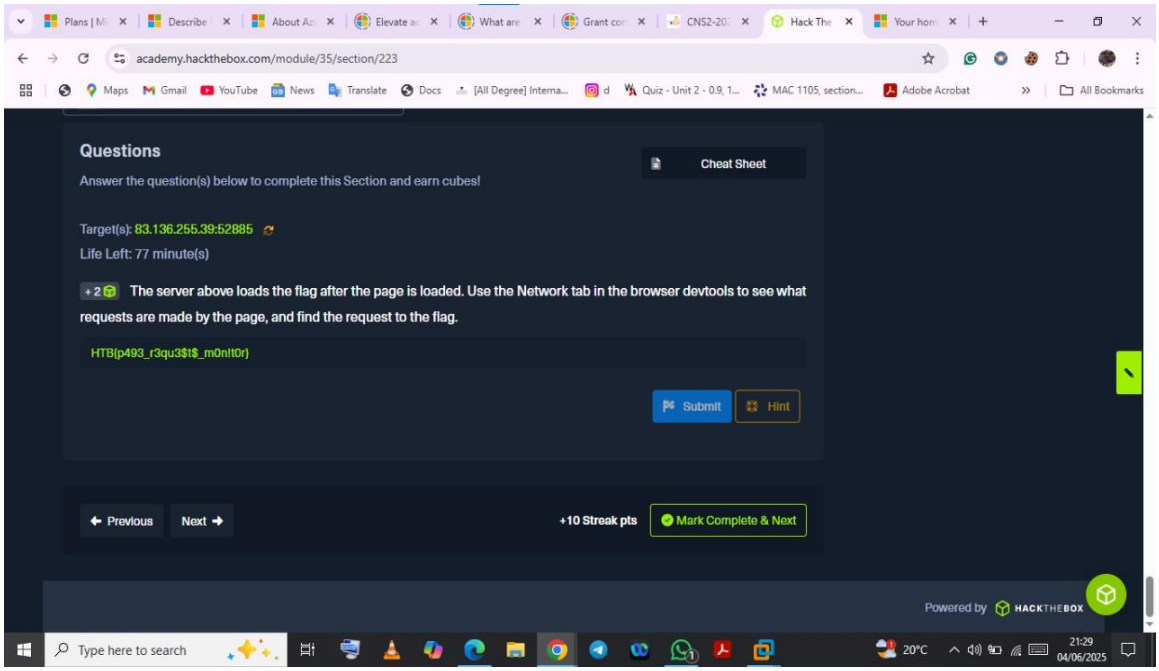
Header	Example	Description
Server	Server: Apache/2.2.14 (Win32)	Contains information about the HTTP server, which processed the request. It can be used to gain information about the server, such as its version, and enumerate it further.
Set-Cookie	Set-Cookie: PHPSESSID=b4e4fbd93540	Contains the cookies needed for client identification. Browsers parse the cookies and store them for future requests. This header follows the same format as the Cookie request header.
WWW-Authenticate	WWW-Authenticate: BASIC realm="localhost"	Notifies the client about the type of authentication required to access the requested resource.

Security Headers

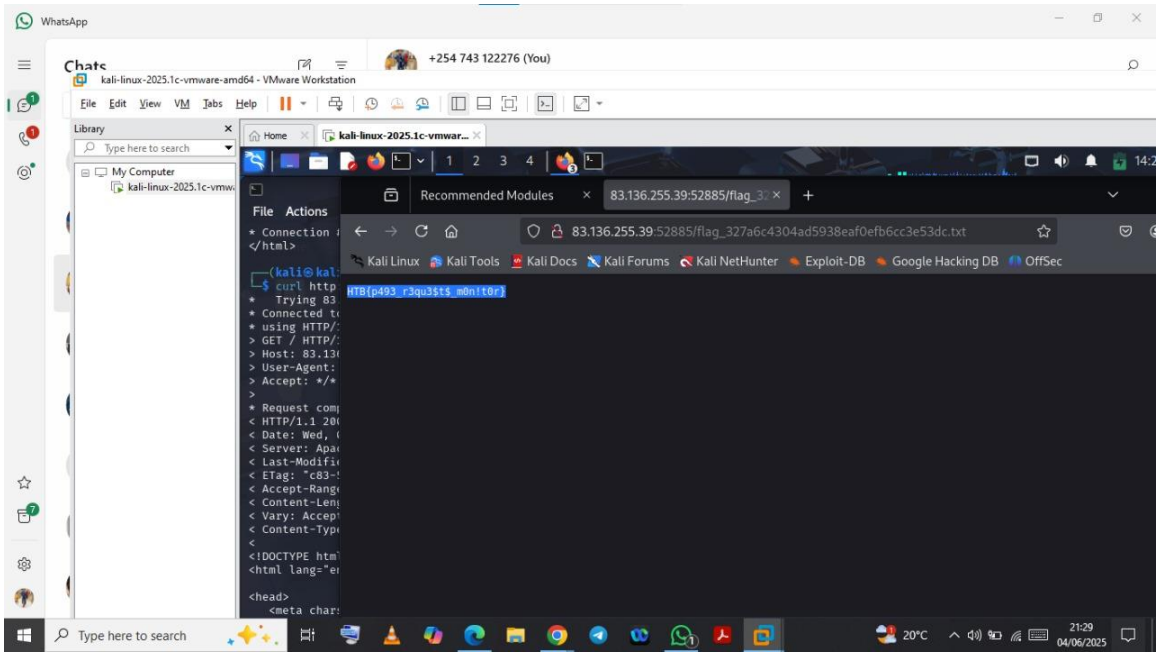

cURL
In the previous section, we saw how using the -v flag with cURL shows us the full details of the HTTP request and response. If we were only interested in seeing the response headers, then we can use the -I flag to send a HEAD request and only display the response headers. Furthermore, we can use 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.
The following command shows an example output of using the -I flag:
HTTP Headers
0guda1@htb[/htb]$ curl -I https://www.inlanefreight.com
Host: www.inlanefreight.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/605.1.15 (KHTML, like Gecko)
Cookie: cookie1=298zf09hf012fh2; cookie2=u32t4o3tb3gg4
Accept: text/plain
Referer: https://www.inlanefreight.com/
Authorization: BASIC cGFzc3dvcmQK
Date: Sun, 06 Aug 2020 08:49:37 GMT
Connection: keep-alive


Browser DevTools
Finally, let's see how we can preview the HTTP headers using the browser devtools. Just as we did in the previous section, we can go to the Network tab to view the different requests made by the page. We can click on any of the requests to view its details:
In the first Headers tab, we see both the HTTP request and HTTP response headers. The devtools automatically arrange the

Question


Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
Target(s): 83.136.255.39:52885
Life Left: 77 minute(s)
+2 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_m0nlt0r}
Submit
Hint
Previous
Next
+10 Streak pts
Mark Complete & Next
Powered by HACKTHEBOX

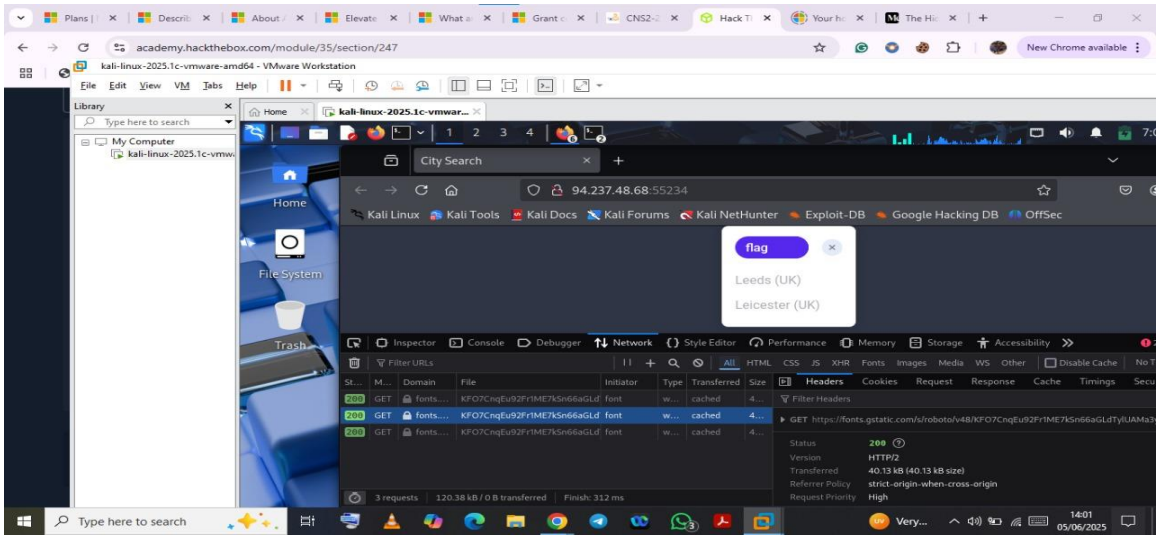
Answer



HTTP Methods and Status Codes

This part of the module detailed the standard HTTP methods—GET, POST, PUT, DELETE—and how they are used in resource manipulation. Each method has a specific role in interacting with server resources. I also explored HTTP status codes, such as 200 (OK), 301 (Moved Permanently), 403 (Forbidden), 404 (Not Found), and 500 (Internal Server Error). Learning to interpret these codes is essential for debugging and understanding server behavior.

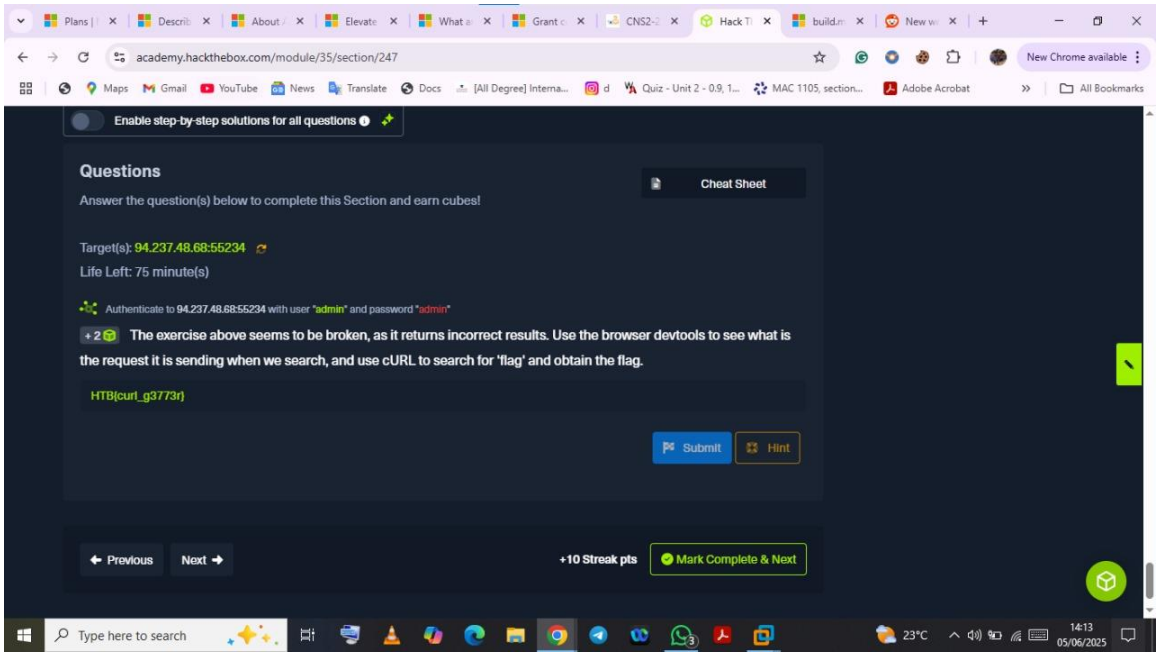
200 Code



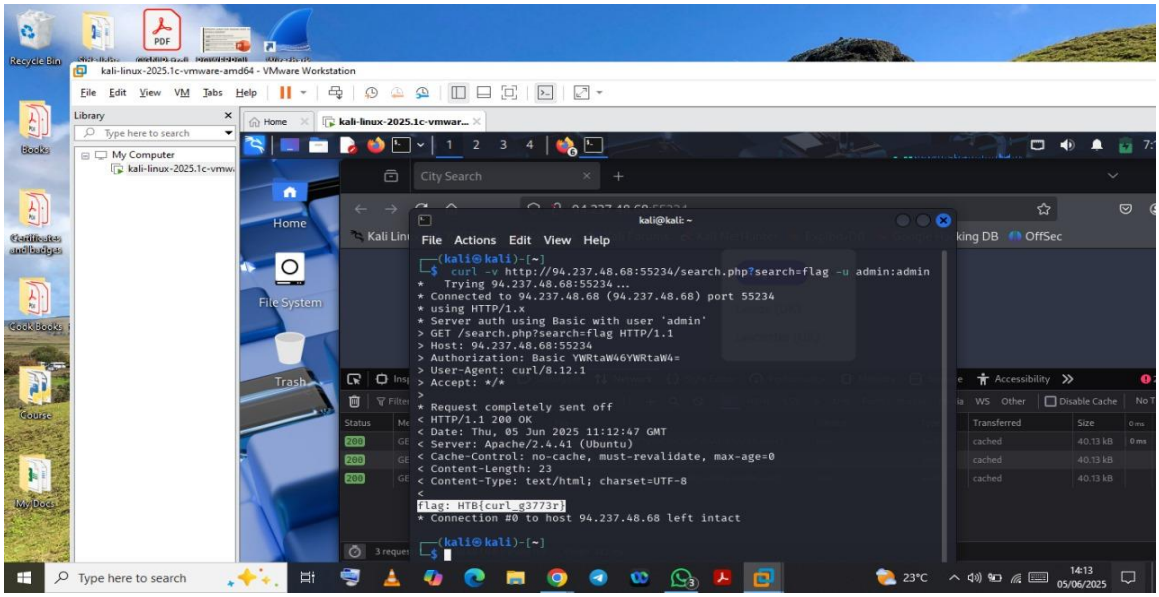
GET Requests

I learned that GET requests are used to retrieve data from a server without altering its state. Using both cURL and browser DevTools, I observed how GET requests are used to access resources such as HTML pages, images, and JSON responses from APIs. These requests typically include parameters in the URL and are idempotent, meaning they can be repeated without side effects.

Question



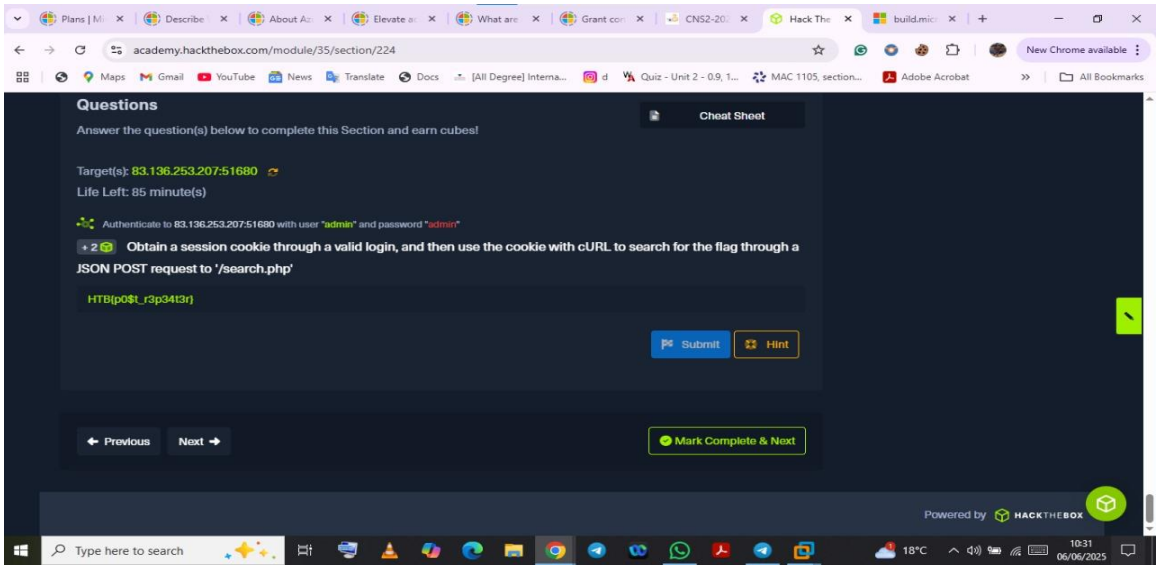
Answer



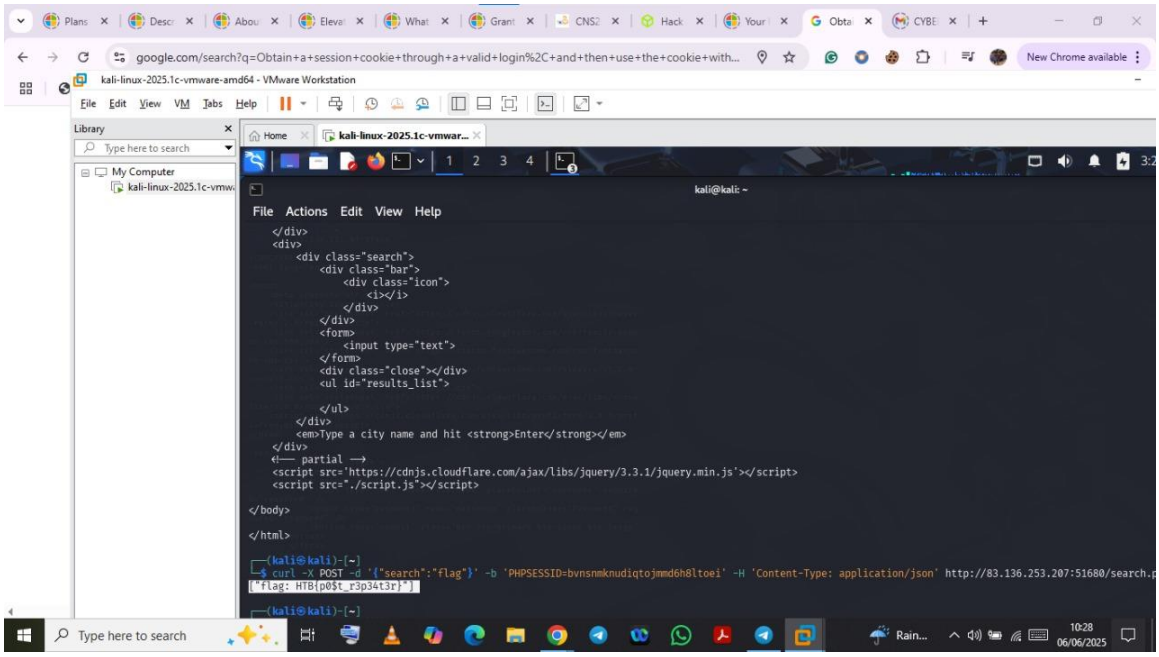
POST Requests

POST requests are used to send data to a server to create or process resources. I practiced crafting POST requests to simulate form submissions and API calls. Unlike GET, POST includes the data in the request body rather than the URL and is used when the client intends to change the server state. I learned to view these requests and their payloads using DevTools' Network tab and to replicate them using cURL.

Question



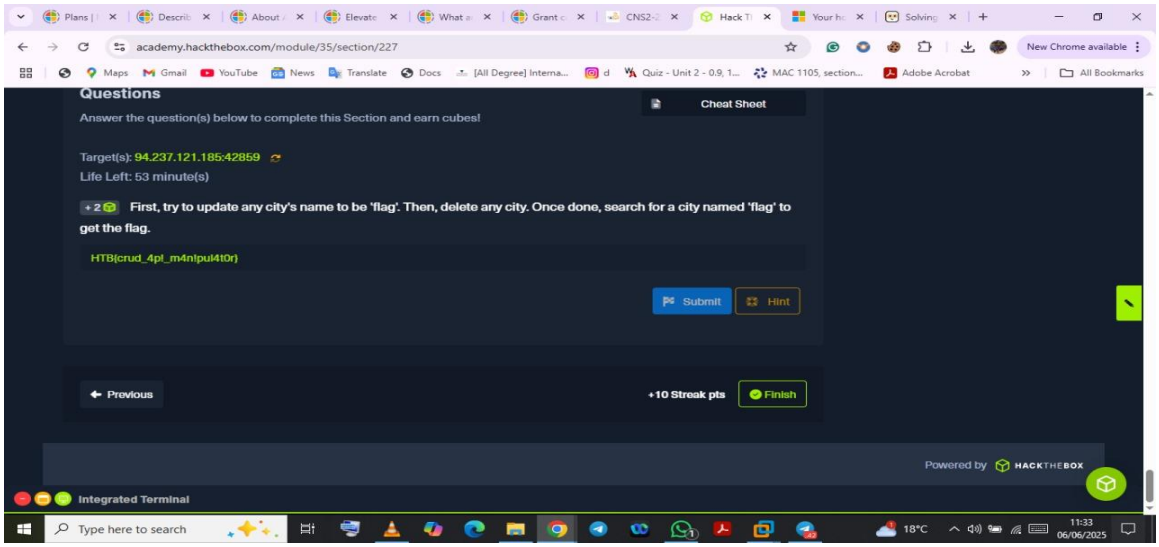
Answer



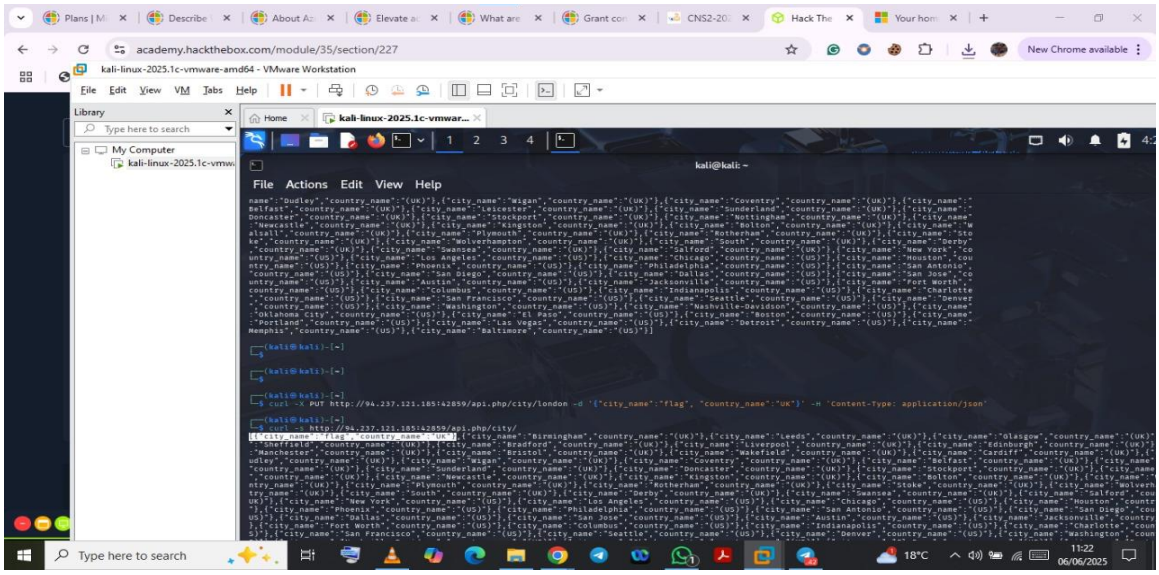
CRUD Operations on APIs

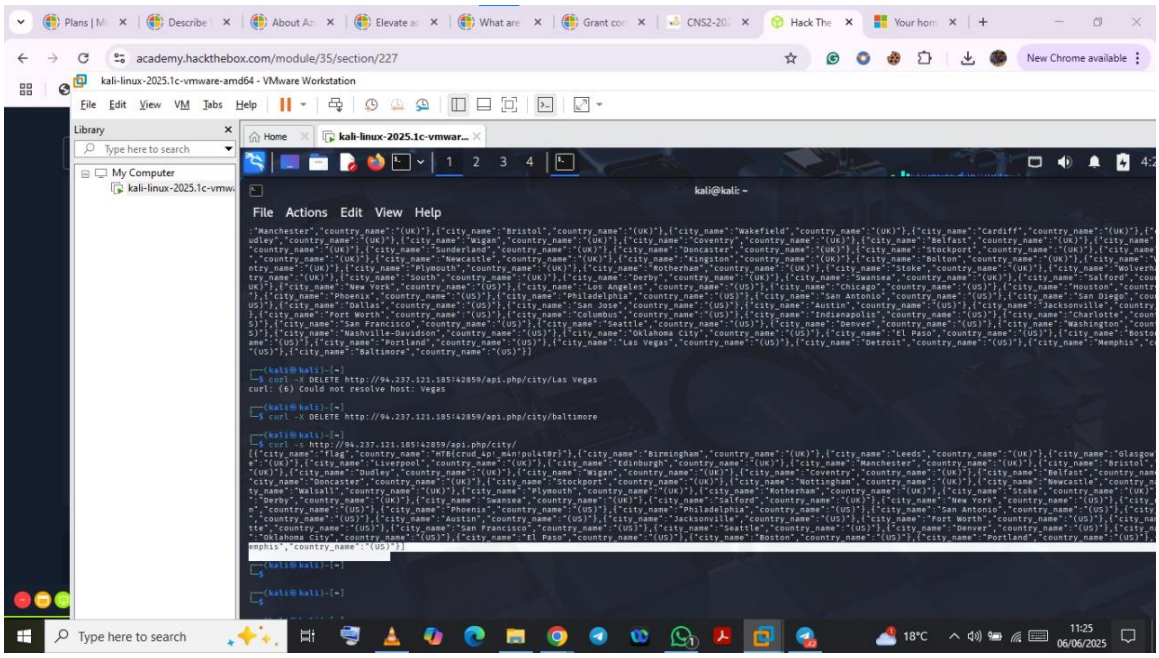
The final segment focused on RESTful APIs and how CRUD operations—Create, Read, Update, Delete—are implemented using HTTP methods. I practiced interacting with API endpoints using POST for creating new resources, GET for reading data, PUT for updating existing resources, and DELETE for removing records. These tasks deepened my understanding of how web applications manage data behind the scenes and how structured, predictable HTTP communication powers API functionality.

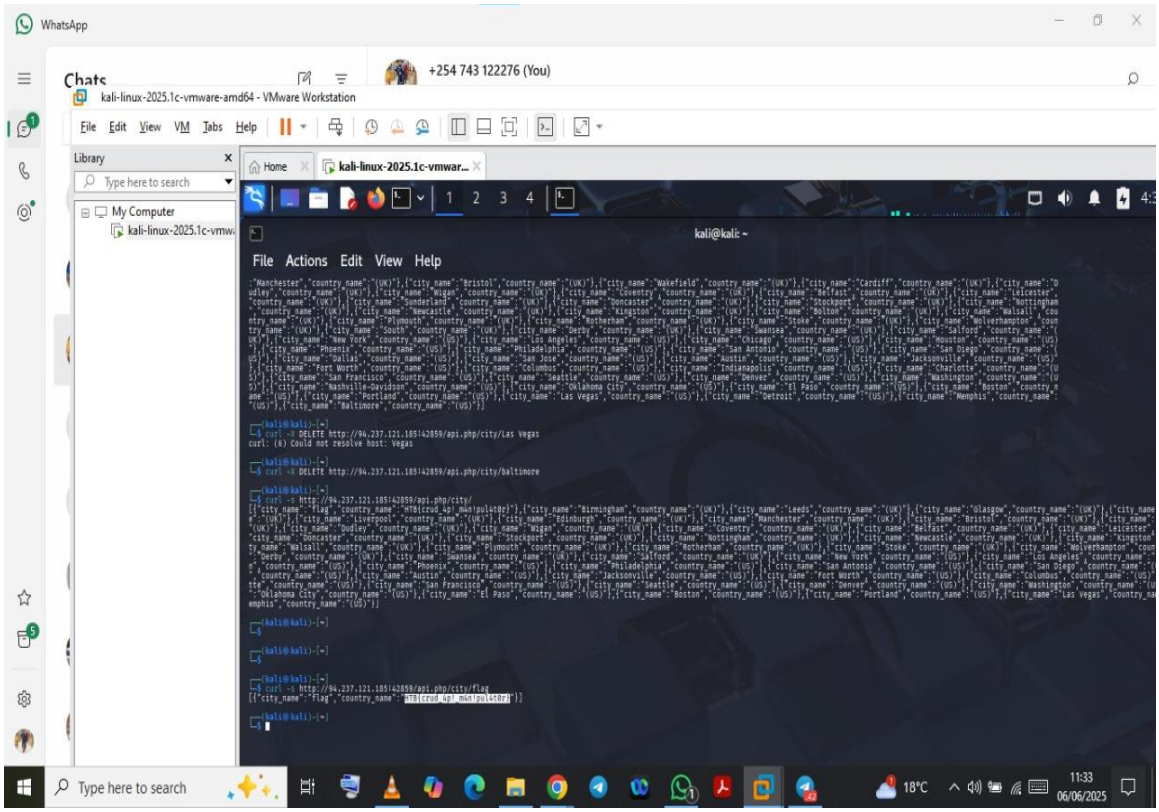
Question



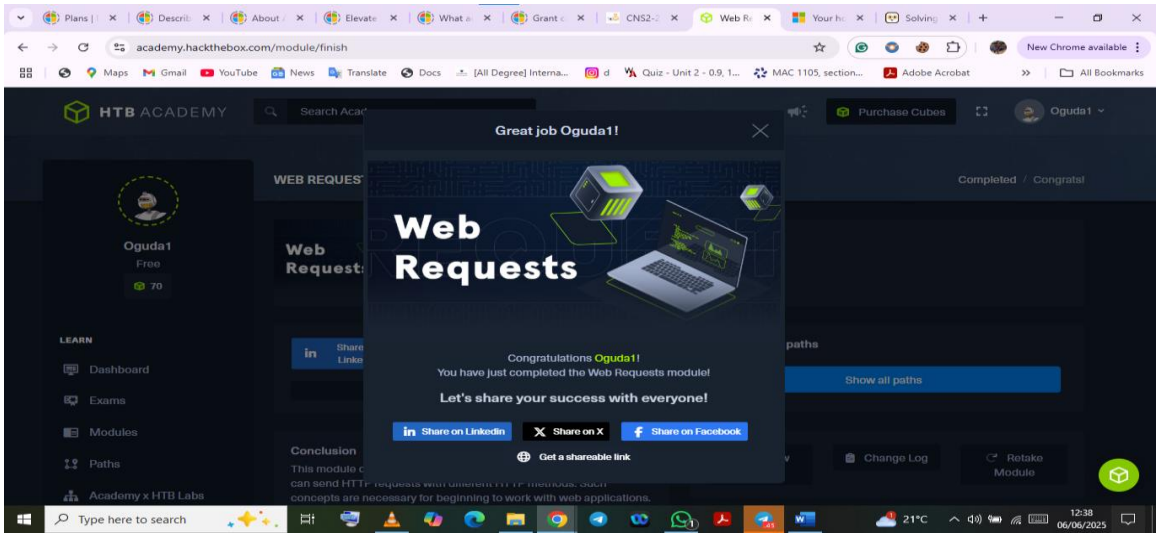
Answer




academy.hackthebox.com/module/35/section/227
kali-linux-2025.1c-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
kali-linux-2025.1c-vmwa
Home
kali-linux-2025.1c-vmwar...
kali@kali: ~
File Actions Edit View Help
curl -X DELETE http://94.237.121.185:42859/api.php/city/Las Vegas
curl: (6) Could not resolve host: Vegas
curl -X DELETE http://94.237.121.185:42859/api.php/city/baltimore
curl -s http://94.237.121.185:42859/api.php/city/
Type here to search
18°C
11:25
06/06/2025


WhatsApp
Chats
+254 743 122276 (You)
kali-linux-2025.1c-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
kali-linux-2025.1c-vmwa
Home
kali-linux-2025.1c-vmwar...
kali@kali: ~
File Actions Edit View Help
curl -X DELETE http://94.237.121.185:42859/api.php/city/Las Vegas
curl: (6) Could not resolve host: Vegas
curl -X DELETE http://94.237.121.185:42859/api.php/city/baltimore
curl -s http://94.237.121.185:42859/api.php/city/
curl -s http://94.237.121.185:42859/api.php/city/flag
Type here to search
18°C
11:33
06/06/2025

3. Completion Proof and Social Sharing



Shareable Link to Completion

https://academy.hackthebox.com/achievement/1922141/35

4. Conclusion

Completing the Web Requests module has been both insightful and rewarding. It offered a solid foundation in understanding how the web works from a technical perspective. I now feel more confident in analyzing HTTP traffic, using command-line tools to interact with APIs, and interpreting server responses. These skills are not only useful in development and testing environments but are also crucial for cybersecurity practices such as reconnaissance, vulnerability assessment, and penetration testing.

The hands-on exercises reinforced theoretical concepts and helped me develop a more analytical mindset when approaching web technologies. I look forward to building upon this knowledge in future modules and applying these skills in real-world security scenarios.