CyberDefenders

WebStrike Lab

Tool: Wireshark

Senario

You are a cybersecurity analyst working in the Security Operations Center (SOC) of BookWorld, an expansive online bookstore renowned for its vast selection of literature. BookWorld prides itself on providing a seamless and secure shopping experience for book enthusiasts around the globe. Recently, you've been tasked with reinforcing the company's cybersecurity posture, monitoring network traffic, and ensuring that the digital environment remains safe from threats. Late one evening, an automated alert is triggered by an unusual spike in database queries and server resource usage, indicating potential malicious activity. This anomaly raises concerns about the integrity of BookWorld's

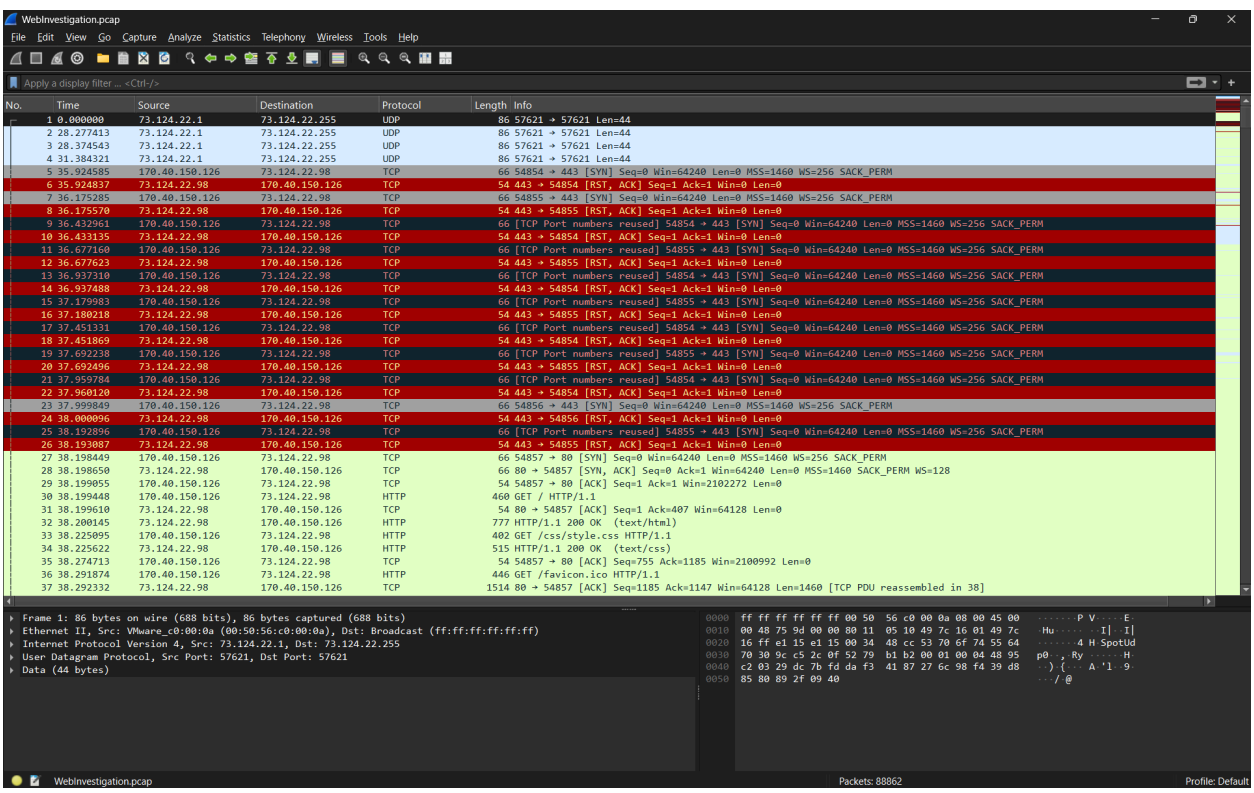
customer data and internal systems, prompting an immediate and thorough investigation. As the lead analyst in this case, you are required to analyze the network traffic to uncover the nature of the suspicious activity. Your objectives include identifying the attack vector, assessing the scope of any potential data breach, and determining if the attacker gained further access to BookWorld's internal systems.

Tasks :

Q1) By knowing the attacker's IP, we can analyze all logs and actions related to that IP and determine the extent of the attack, the duration of the attack, and the techniques used. Can you provide the attacker's IP?

📌 ***Steps to Extract the Attacker's IP:***

1️⃣ ***Load the PCAP file in Wireshark*** *to inspect the captured network traffic.*



2 ***Apply HTTP filters*** *to focus on relevant communications:*

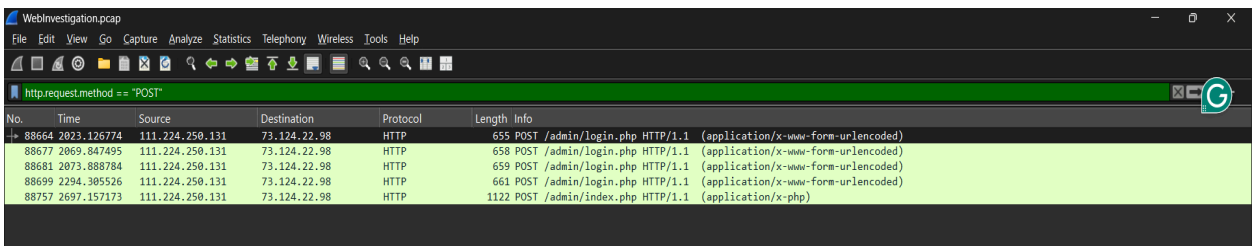
- *Since we are analyzing a* ***web application****, the attacker's activity is likely visible in* ***GET and POST requests****, as these methods are commonly used for* ***data exchange between clients and servers****.*

Using the ***Wireshark filter****:*

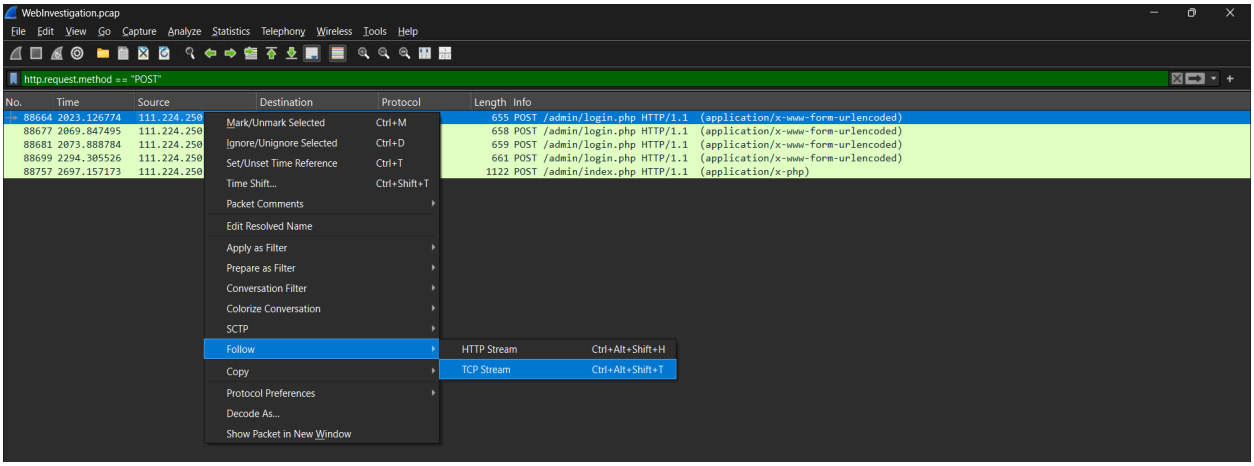
```
http.request.method == "GET"
```

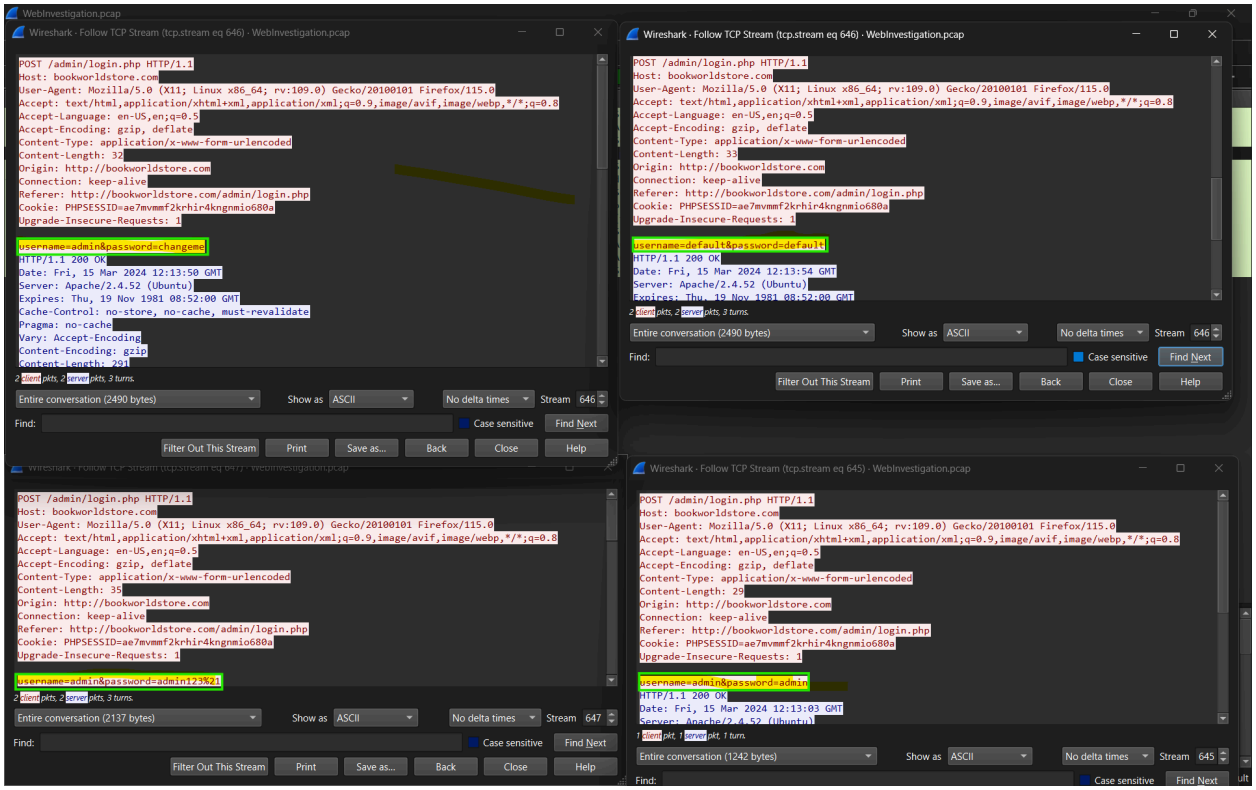
```
http.request.method == "POST"
```

- *This helps isolate HTTP requests that might reveal* ***suspicious login attempts, data exfiltration, or unauthorized access.***



- *We found some ips and got to see some sort of* ***admin/login attempts .***
- *We then tried to inspect it we get the following information*

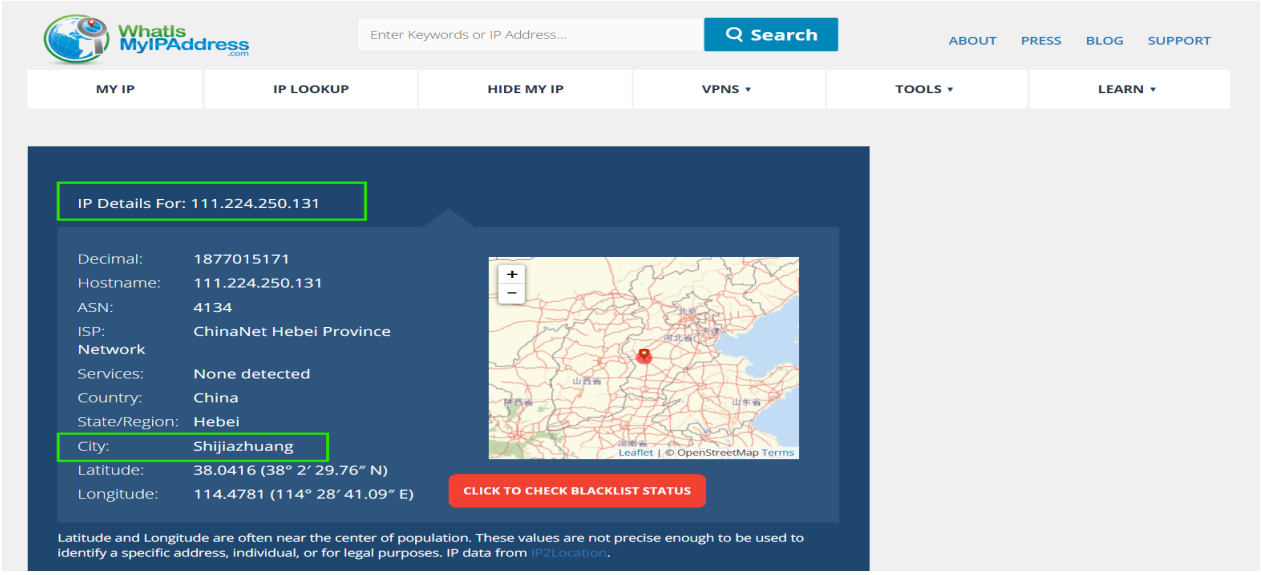




- *Just by following the set of tcp access mentioned above we get to see that there's been multiple attempts to get access.*
- *So our suspicious IP address we are searching for is* ***111.224.250.131***

Q2) If the geographical origin of an IP address is known to be from a region that has no business or expected traffic with our network, this can be an indicator of a targeted attack. Can you determine the origin city of the attacker?

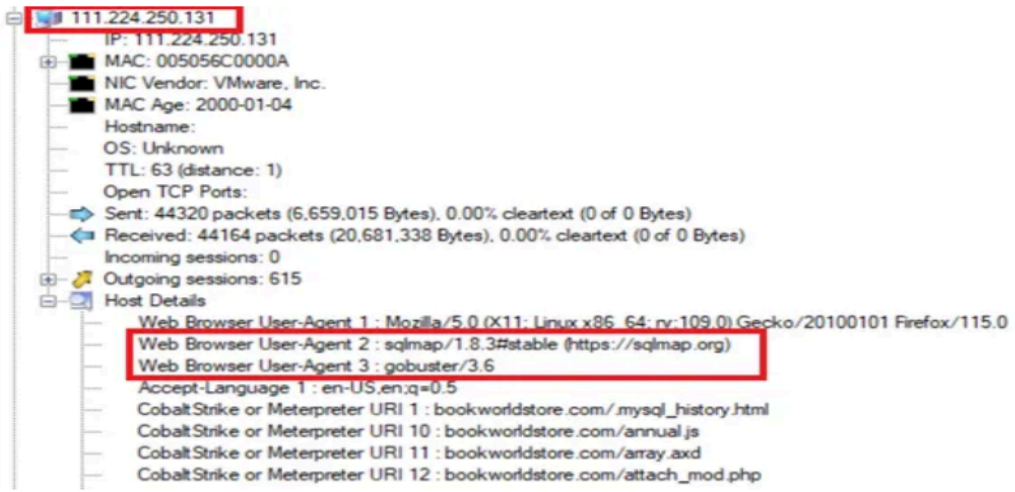
- *Now as we found the ip address we will use the osint tools such as WhatismyIpaddress.com .*
- We got the mentioned below result have a look:

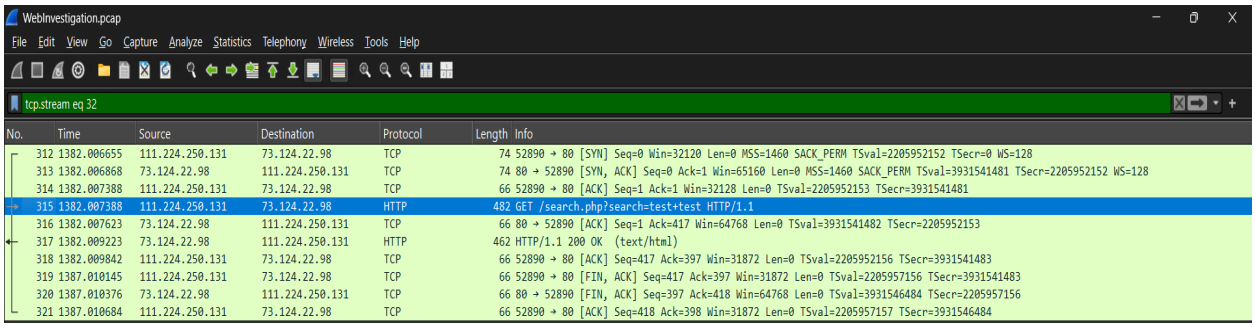


- So our city that we need to find is **Shijiazhuang (China)** .

Q3) *Identifying the exploited script allows security teams to understand exactly which vulnerability was used in the attack. This knowledge is critical for finding the appropriate patch or workaround to close the security gap and prevent future exploitation. Can you provide the vulnerable PHP script name?*

In Network Miner I saw the host details and I found that he used malicious tools that appeared in the user agent section such as go buster and SQLMap.

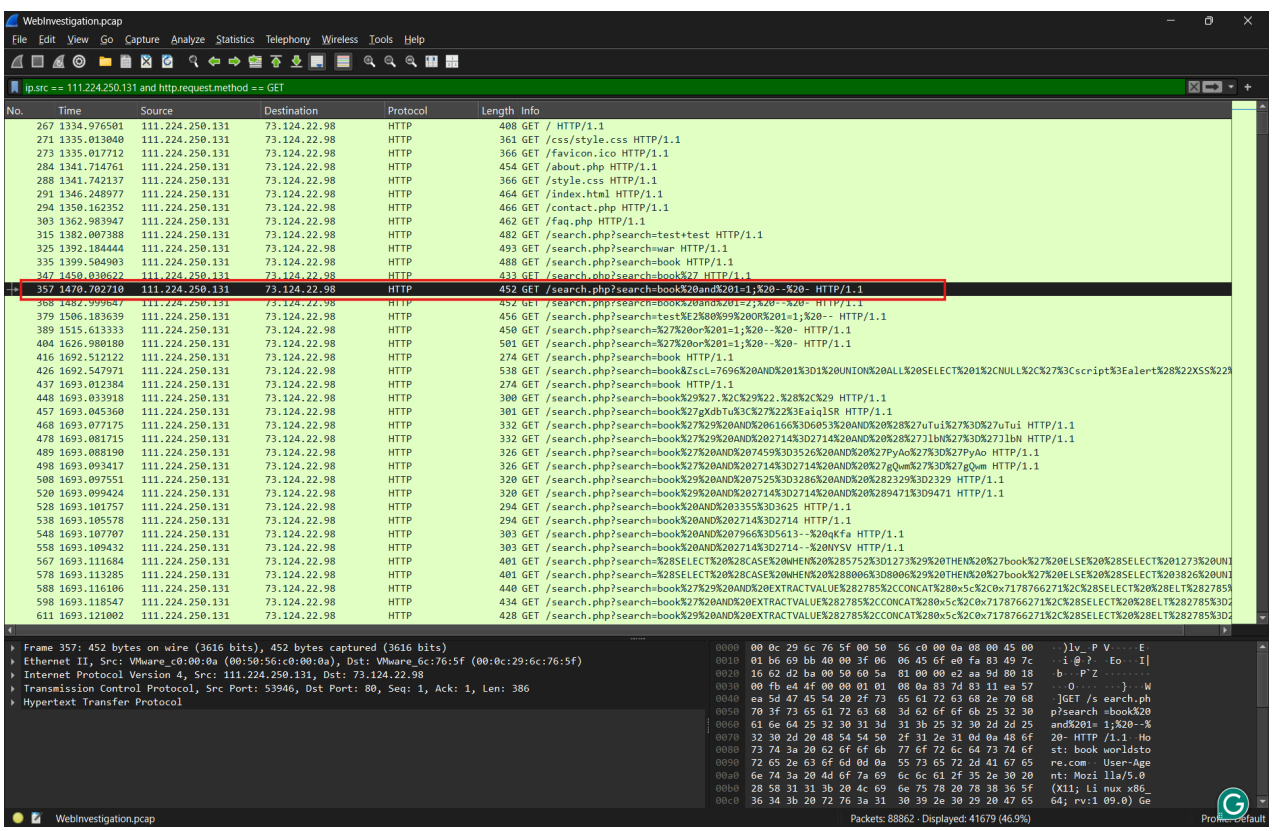


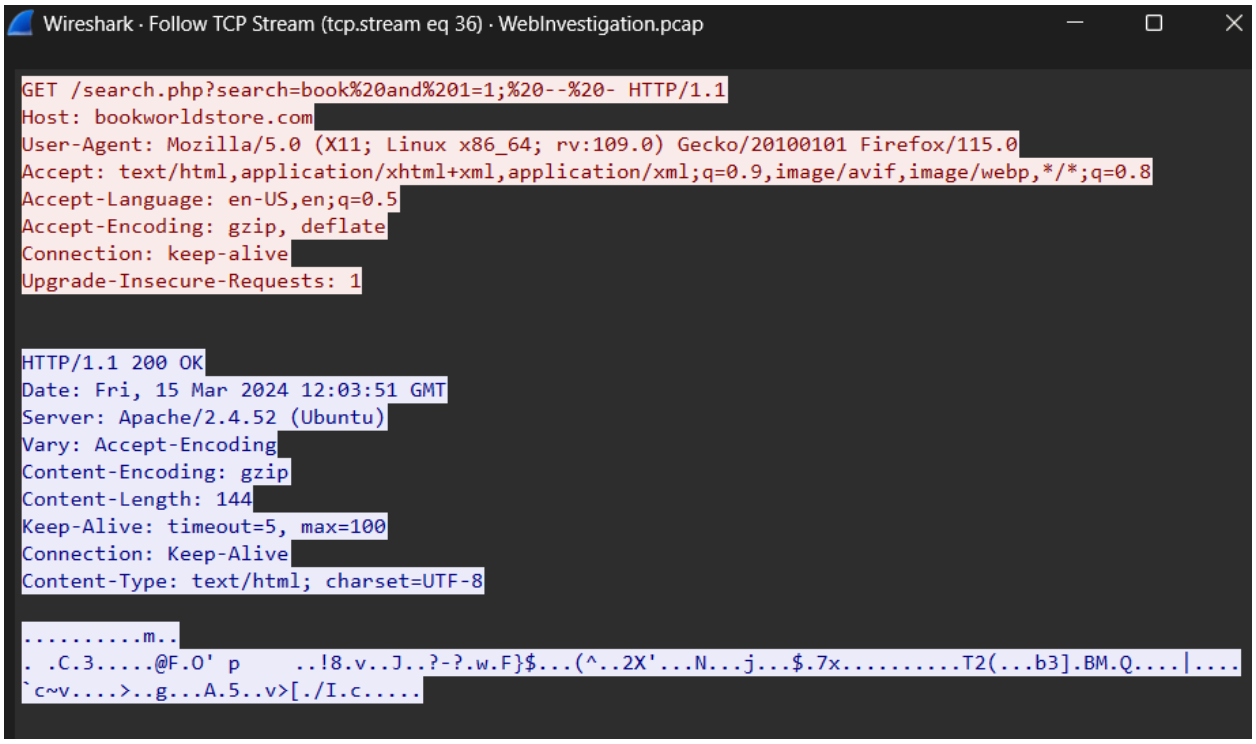


- We found the php script as **/search.php**.

Q4) Establishing the timeline of an attack, starting from the initial exploitation attempt, What's the complete request URL of the first SQLi attempt by the attacker?

- *So as we inspect we found the following sqli command executed*





```
Wireshark · Follow TCP Stream (tcp.stream eq 36) · WebInvestigation.pcap

GET /search.php?search=book%20and%201=1;%20--%20- HTTP/1.1
Host: bookworldstore.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:03:51 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 144
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

..........m..
. .C.3.....@F.O' p     ..!8.v..J..?-?.w.F}$...(^..2X'...N...j...$.7x..........T2(...b3].BM.Q....|....
`c~v....>..g...A.5..v>[./I.c.....
```

- *1st sqli attempt was done on Date: Fri, 15 Mar 2024 12:03:51 GMT*
- *On the top we find in red color the GET portion: /search.php?search=book%20and%201=1;%20--%20-*

This is the desired answer for the question.

Q5) Can you provide the complete request URI that was used to read the web server's available databases?

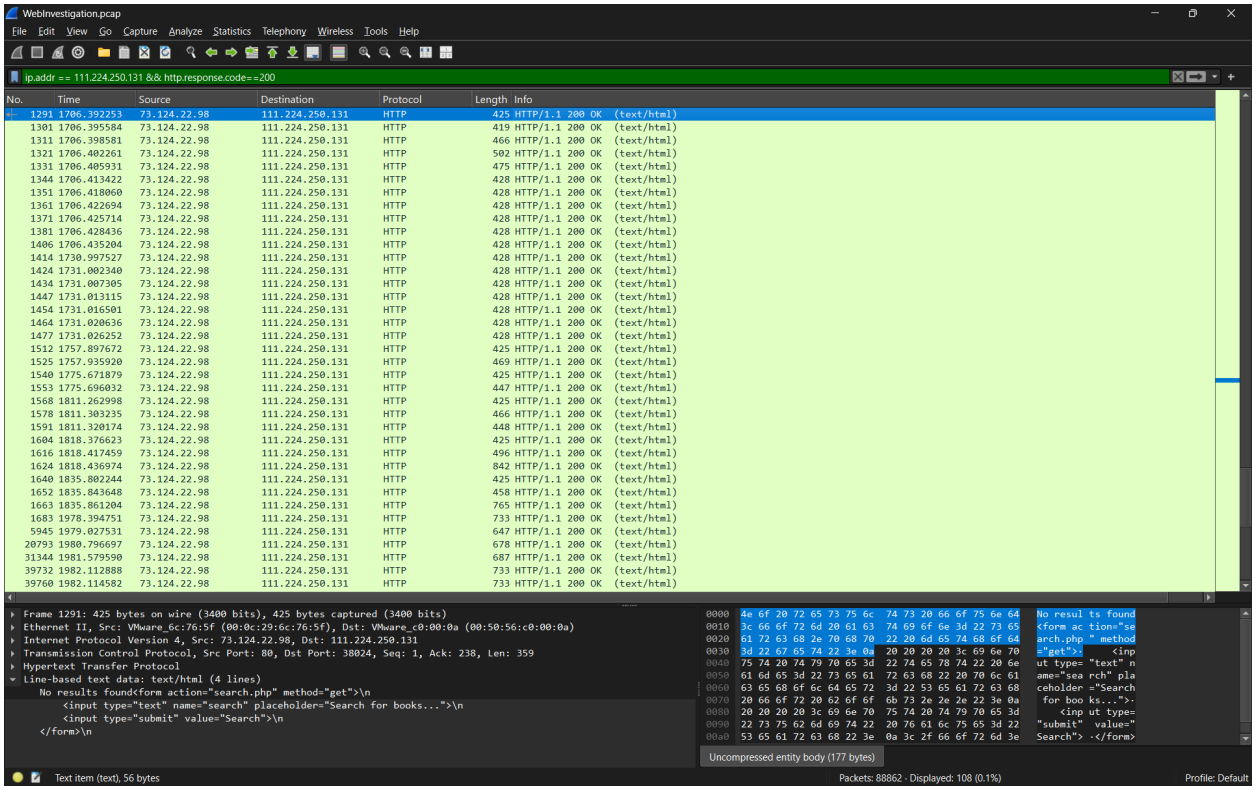
Since we have identified an SQL injection attack, the next step is to check if the website responded to it.

A successful SQL injection attempt would generate a valid response from the server. To confirm this, we can filter the network traffic in Wireshark using the attacker's IP address along with an ***HTTP 200 response code****,* ***which indicates a successful request****.*

Using the following :-

Wireshark filter: ip.addr == 111.224.250.131 && http.response.code == 200

This filter helps us identify responses from the server to the attacker's requests, confirming that the SQL injection attempt was processed successfully.



Narrowing Down the Results as the above filter returns a large number of results. To extract relevant information related to SQL Injection, we need to refine our search further.

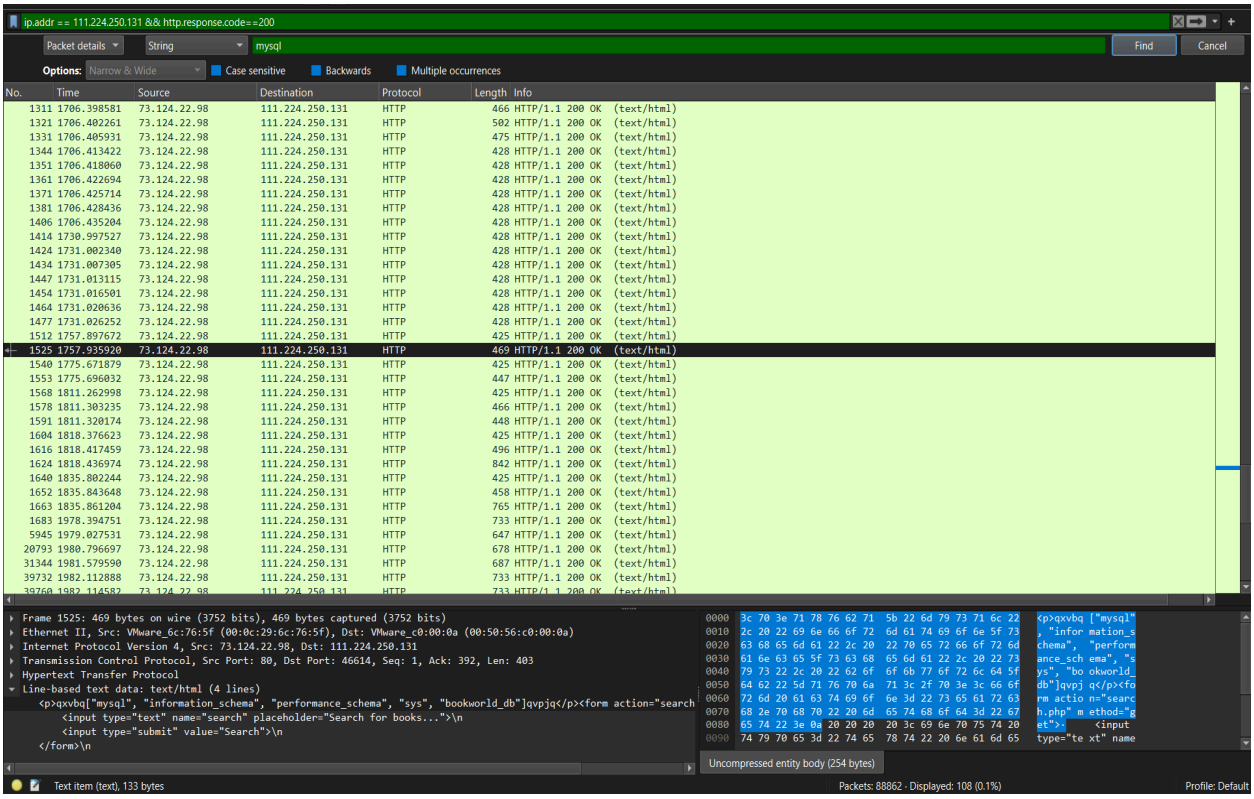
Since SQL Injection often exposes database-related information, we will search for keywords commonly associated with database interactions.

Searching for SQL-related Data

*Wireshark provides a **Find** feature that allows us to search for specific strings within the captured network traffic.*

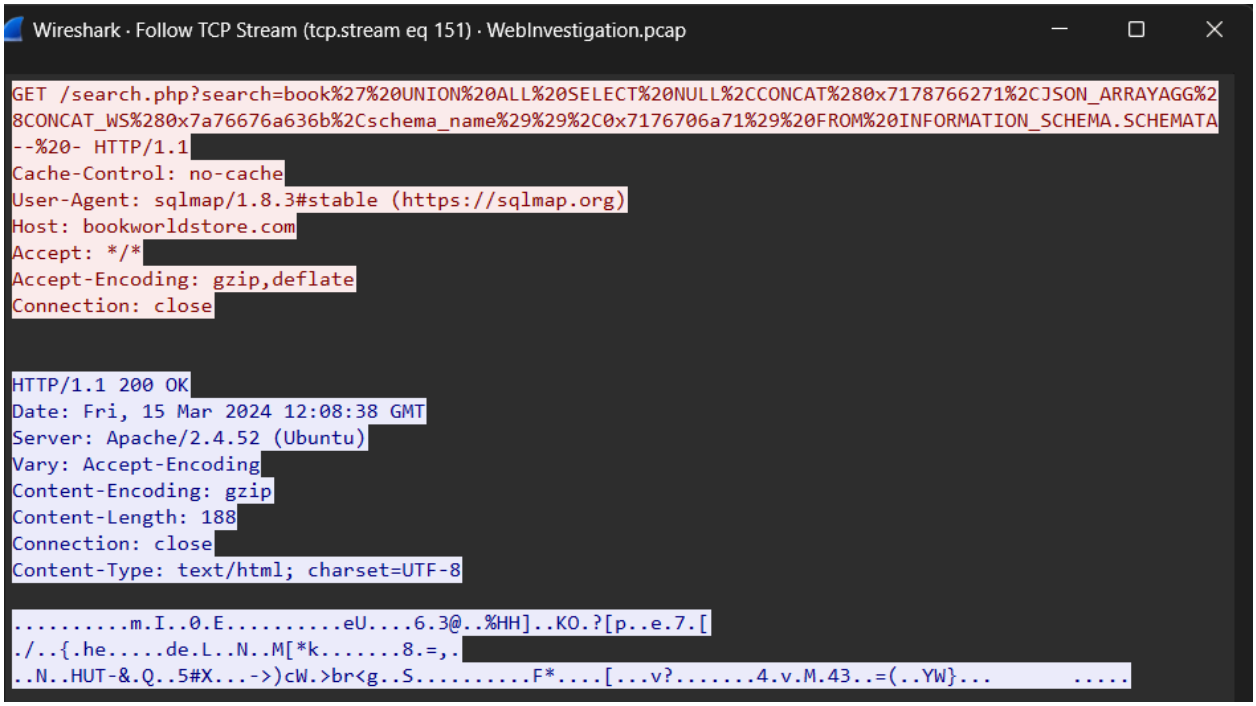
To use this feature:

- *Press **Ctrl + F** .*
- *A search panel will appear where you can enter search terms.*
- *Select **String** as the search type.*
- *Type **mysql** as we are searching for sql injections and click **find.***



This highlights the packet which holds the string mysql.

Let's inspect and see if we get our desired results for the task we have been provided .



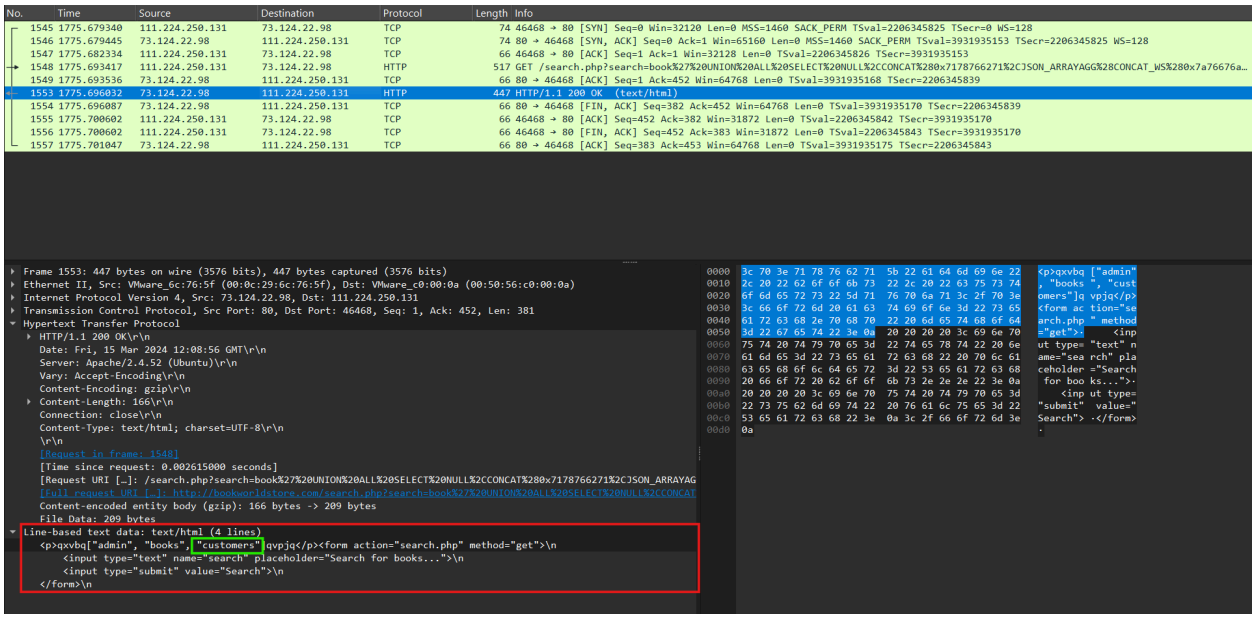
```
Wireshark · Follow TCP Stream (tcp.stream eq 151) · WebInvestigation.pcap

GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%2
8CONCAT_WS%280x7a76676a636b%2Cschema_name%29%29%2C0x7176706a71%29%20FROM%20INFORMATION_SCHEMA.SCHEMATA
--%20- HTTP/1.1
Cache-Control: no-cache
User-Agent: sqlmap/1.8.3#stable (https://sqlmap.org)
Host: bookworldstore.com
Accept: */*
Accept-Encoding: gzip,deflate
Connection: close


HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:08:38 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 188
Connection: close
Content-Type: text/html; charset=UTF-8

..........m.I..0.E..........eU....6.3@..%HH]..KO.?[p..e.7.[
./..{.he.....de.L..N..M[*k.......8.=,.
..N..HUT-&.Q..5#X...->)cW.>br<g..S..........F*....[...v?.......4.v.M.43..=(..YW}...      .....
```

Now as we inspected the TCP flow we found the link asked !!

/search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a636b%2Cschema_name%29%29%2C0x7176706a71%29%20FROM%20INFORMATION_SCHEMA.SCHEMATA--%20-

Q6) Assessing the impact of the breach and data access is crucial, including the potential harm to the organization's reputation. What's the table name containing the website users data?

Here we went the manual way checking each packet which was very time consuming and reached at a packet and got the following information.



So we found this information below and the name of the table is ***customer .***

Q7) The website directories hidden from the public could serve as an unauthorized access point or contain sensitive functionalities not intended for public access. Can you provide the name of the directory discovered by the attacker?

Hidden website directories can pose security risks, as they may contain sensitive functionalities or provide unauthorized access points. In this case, our goal is to identify the directory that was accessed by the attacker.

Since the suspicious IP ***111.224.250.131*** *attempted to exploit the web server, we need to analyze its interactions with the server. One way to do this is by examining* ***POST*** *requests, as these may indicate attempts to access or manipulate sensitive resources.*

So here i used the the filter:

ip.src == 111.224.250.131 && http.request.method==POST

ip.src == 111.224.250.131 && http.request.method==POST

No.	Time	Source	Destination	Protocol	Length	Info
88664	2023.126774	111.224.250.131	73.124.22.98	HTTP	655	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
88677	2069.847495	111.224.250.131	73.124.22.98	HTTP	658	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
88681	2073.888784	111.224.250.131	73.124.22.98	HTTP	659	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
88699	2294.305526	111.224.250.131	73.124.22.98	HTTP	661	POST /admin/login.php HTTP/1.1 (application/x-www-form-urlencoded)
88757	2697.157173	111.224.250.131	73.124.22.98	HTTP	1122	POST /admin/index.php HTTP/1.1 (application/x-php)

We manually checked for the packets and found the following result from the packet tcp follow request.

Wireshark · Follow TCP Stream (tcp.stream eq 646) · WebInvestigation.pcap

```
POST /admin/login.php HTTP/1.1
Host: bookworldstore.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Origin: http://bookworldstore.com
Connection: keep-alive
Referer: http://bookworldstore.com/admin/login.php
Cookie: PHPSESSID=ae7mvmmf2krhir4kngnmio680a
Upgrade-Insecure-Requests: 1

username=admin&password=changeme
HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:13:50 GMT
Server: Apache/2.4.52 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 291
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

..........mQ.N.0...+..I....#!....=...N.mVJ.`o...8/."|.fw=..}....@..[. 8...ap.l.y.......@.M.'r......Z
1.P.<.xd.......J.w.........       ..y.P:.h..@.+e....     \.Yb.u.J].....TL\c.h.....de6....bd..i....G#.
.....%..J...+g.h]......T.@.m;.>....w.?.Y.....#.&..?Sl.0.P...4g.?.....
E.qu]...5.lt6.1..-O... .......
POST /admin/login.php HTTP/1.1
Host: bookworldstore.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
Origin: http://bookworldstore.com
Connection: keep-alive
Referer: http://bookworldstore.com/admin/login.php
Cookie: PHPSESSID=ae7mvmmf2krhir4kngnmio680a
Upgrade-Insecure-Requests: 1

username=default&password=default
HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:13:54 GMT
```

*So we found the hidden directory with **/admin/***

Q8) Knowing which credentials were used allows us to determine the extent of account compromise. What are the credentials used by the attacker for logging in?

```
POST /admin/login.php HTTP/1.1
Host: bookworldstore.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Origin: http://bookworldstore.com
Connection: keep-alive
Referer: http://bookworldstore.com/admin/login.php
Cookie: PHPSESSID=ae7mvmmf2krhir4kngnmio680a
Upgrade-Insecure-Requests: 1

username=admin&password=admin123%21
HTTP/1.1 302 Found
Date: Fri, 15 Mar 2024 12:17:34 GMT
Server: Apache/2.4.52 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
location: index.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

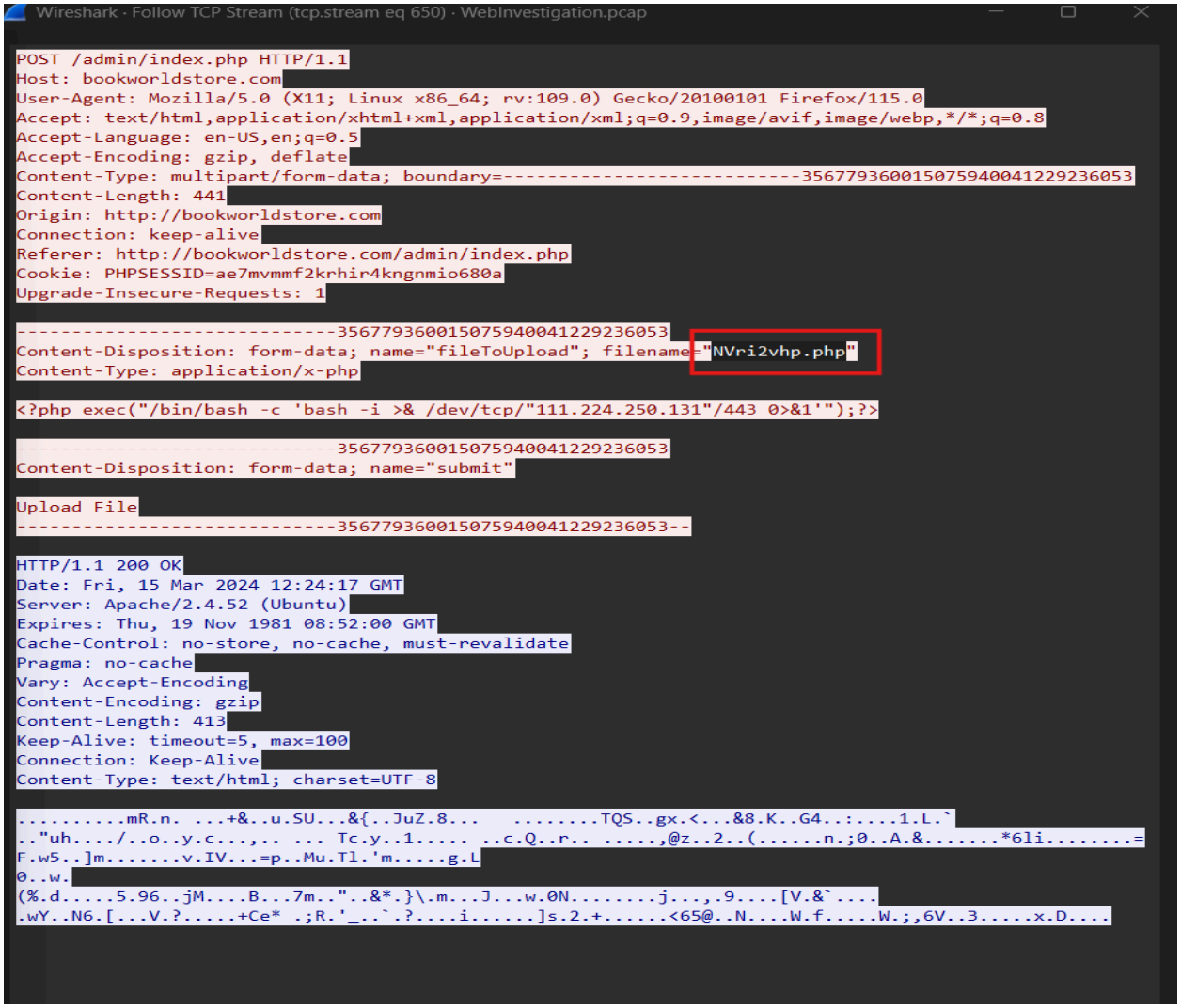
The encoded password *admin123%21* *appears to use URL encoding.*

In URL encoding: %21 represents ! (exclamation mark). So, decoding admin123%21 gives: ***admin123!***

Our required result is ***admin:admin123!***

Q9) We need to determine if the attacker gained further access or control of our web server. What's the name of the malicious script uploaded by the attacker?

I found that the attacker attempted to upload a malicious script and succeeded in doing so our last file with the most size was investigated.



So our desired answer is ***"NVri2vhp.php"*** **.**