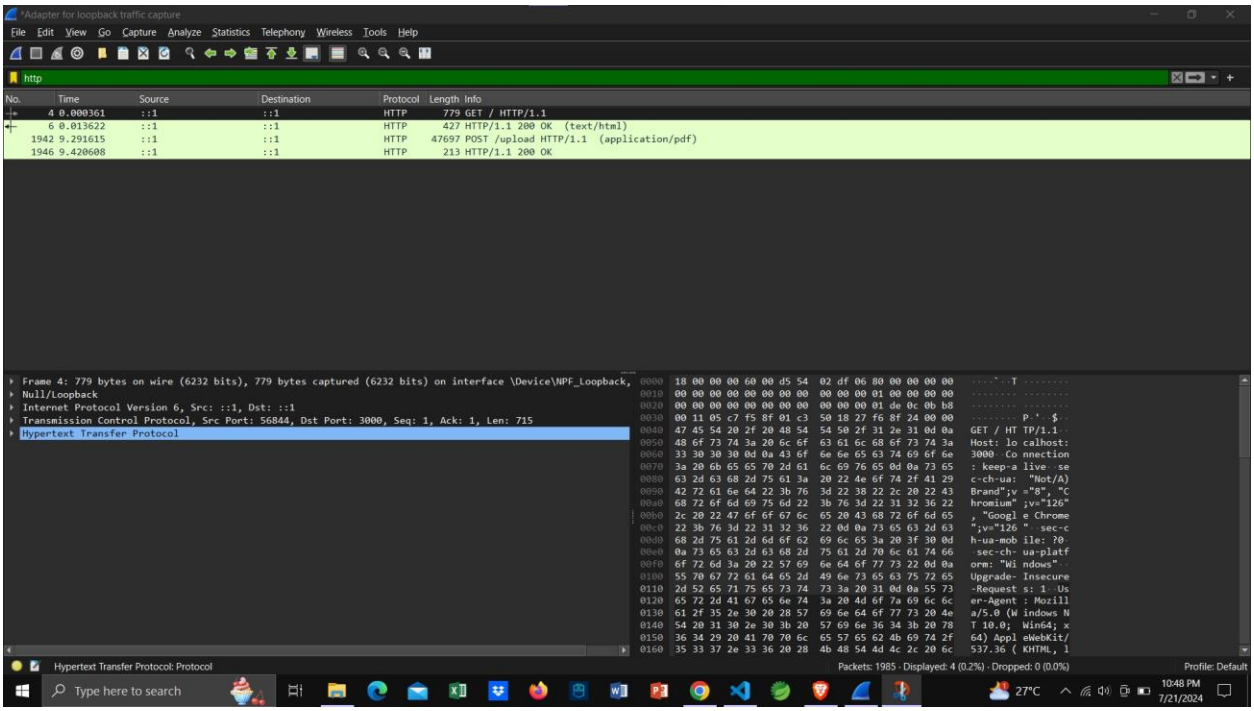
Capturing and Analyzing File Upload with Wireshark

I set up a server and captured the packets during a file upload using Wireshark. Below are the steps and details of the process.

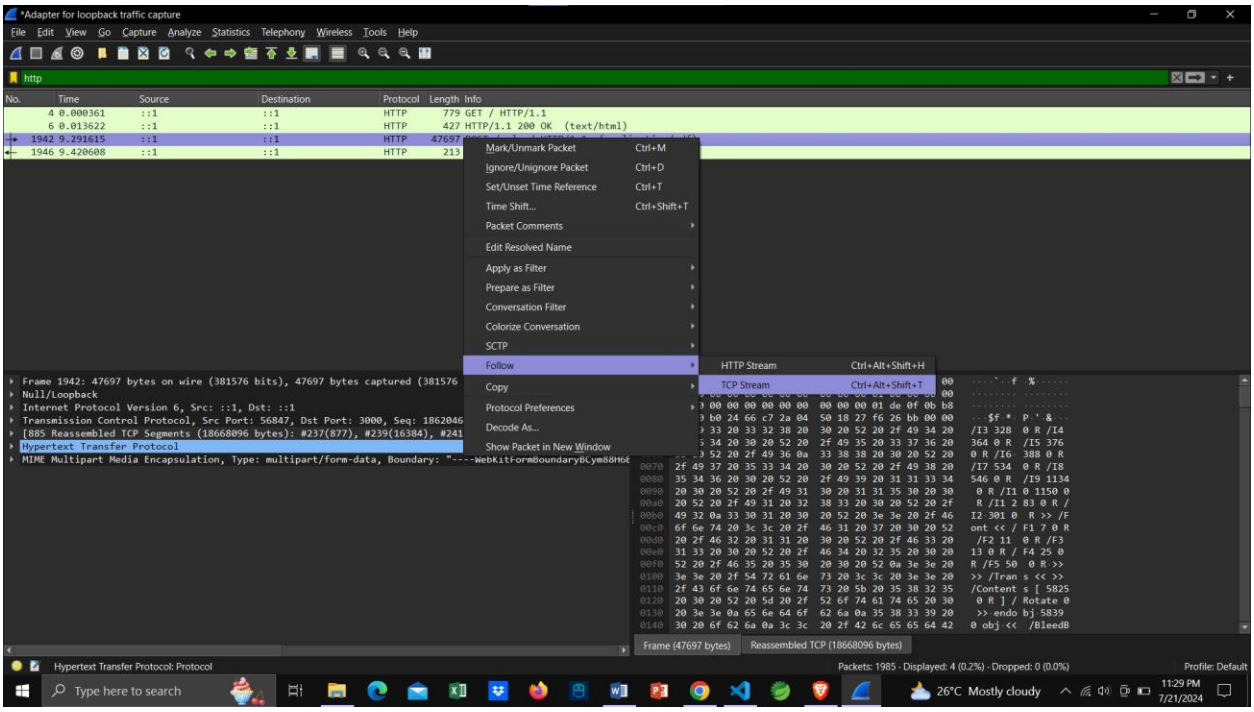
Step-by-Step Process

- **Running Wireshark**:
    - Wireshark application was run, and the loopback interface was chosen for packet capturing.
- **Creating a File Upload Web Application**:
    - A file uploading web application was created using Node.js and HTML, which enabled file upload and storage in a chosen directory.
- **Starting the Capture**:
    - Capturing was started in Wireshark.
- **Filtering for HTTP Packets**:
    - Initially, the http filter was selected in Wireshark to show only HTTP packets.

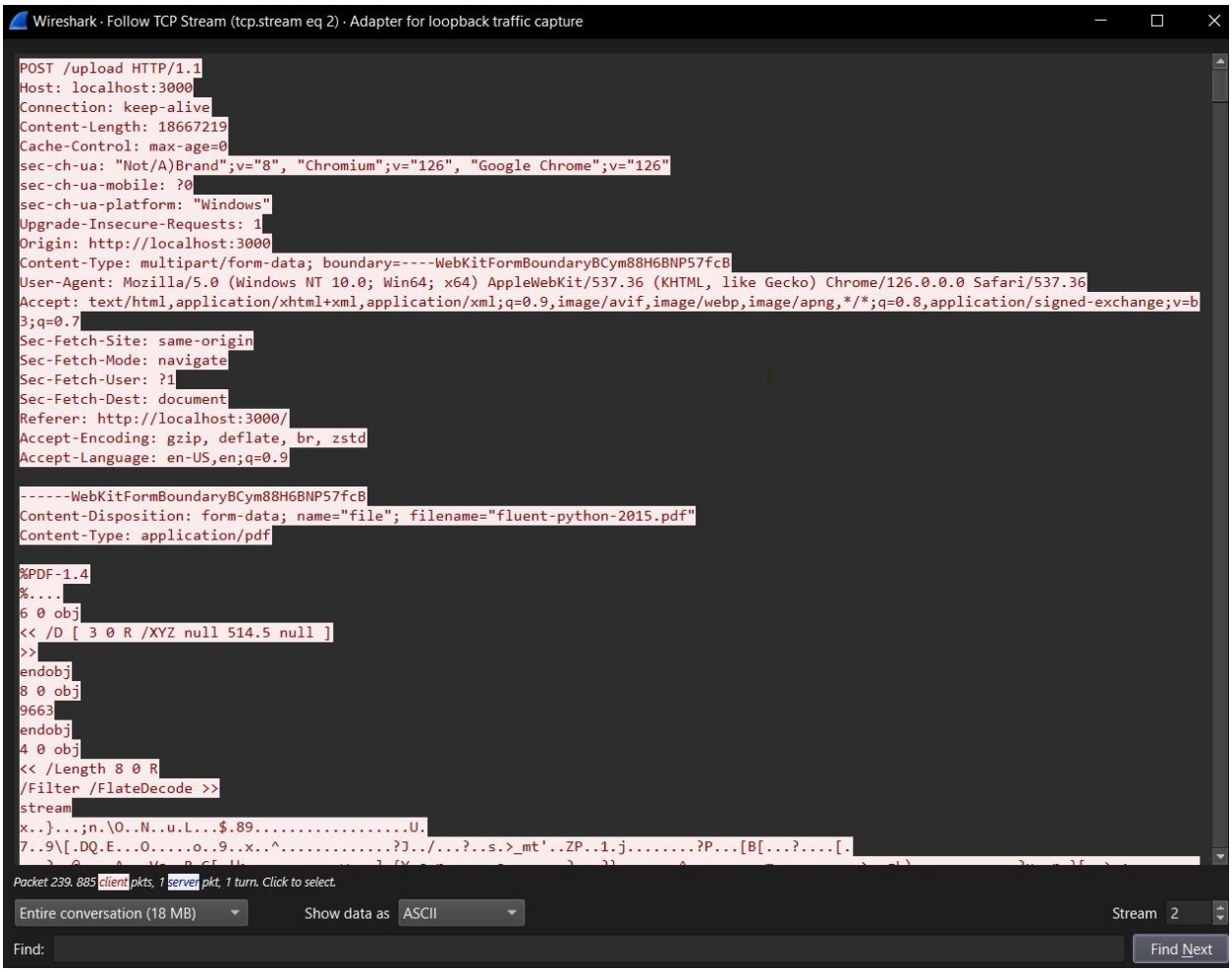


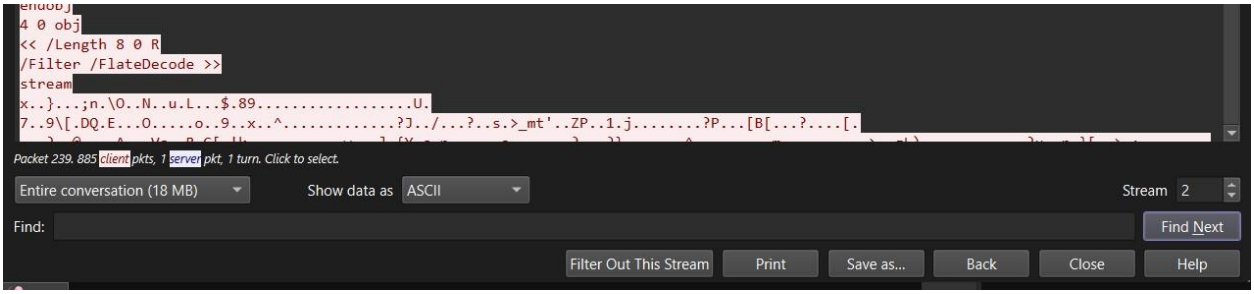
- **Locating the File Upload Request**:

- I identified the HTTP POST request that initiates the file upload. The POST request had a large payload as I was uploading a file.



- **Following the TCP Stream**:
  - I right-clicked on the POST request packet.
  - Selected "Follow" > "TCP Stream".

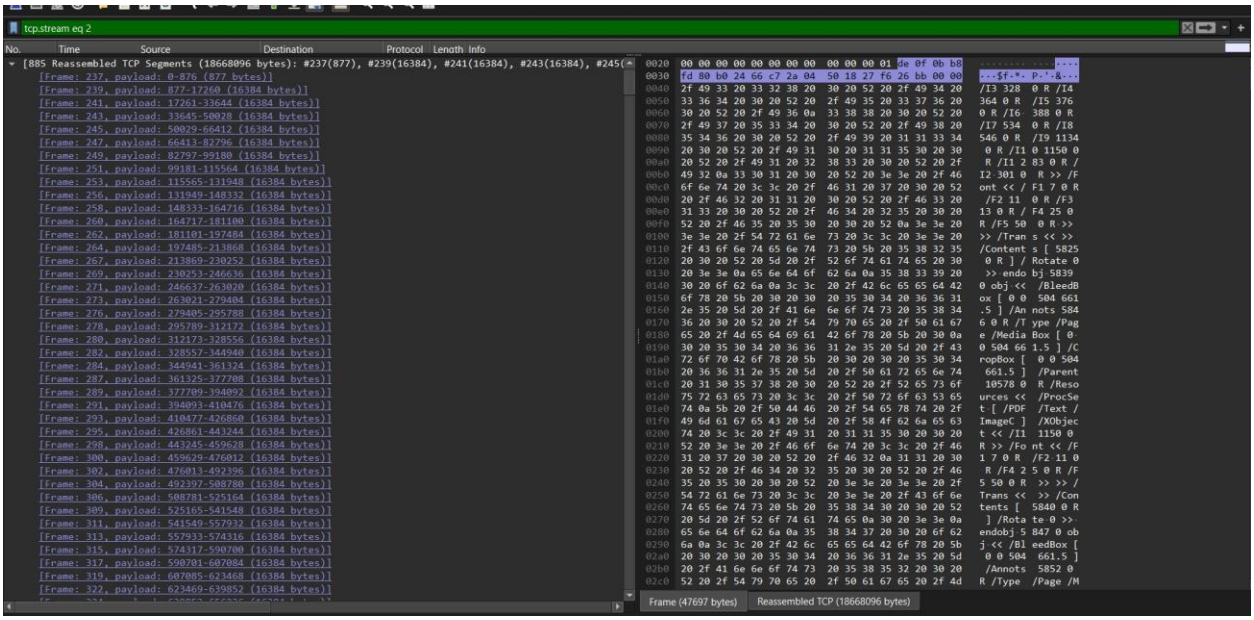




- This displayed all the packets exchanged between the client and the server during the file upload.
- In the TCP stream view, I observed there were altogether 885 client packets and 1 server packet.

- **Analyzing the TCP Segments**:

- In the TCP stream view, I could see all the TCP segments exchanged.
- I viewed different TCP segment lengths and starting and ending bytes transferred from the TCP layer in the window pane.
- I observed the sequence numbers and the length of each segment to understand how the file was divided.



- After closing the TCP stream view, I returned to the main Wireshark window.
- I used the filter tcp.stream eq <stream_number> (replacing <stream_number> with the actual stream number found in the TCP stream view) to isolate the specific TCP stream for the file upload.
- I counted the number of segments (packets) in this stream.

Conclusion

By following these steps, I was able to capture and analyze the file upload process using Wireshark. This included capturing the packets, filtering for HTTP traffic, and analyzing the TCP segments involved in the upload. The analysis showed that there were 885 client packets and 1 server packet exchanged during the file upload. This method provided a detailed understanding of the packet exchange between the client and server during the file upload.