Lab Week 2

The Internet Protocols

Student name: Mona Hamdan Aloufi

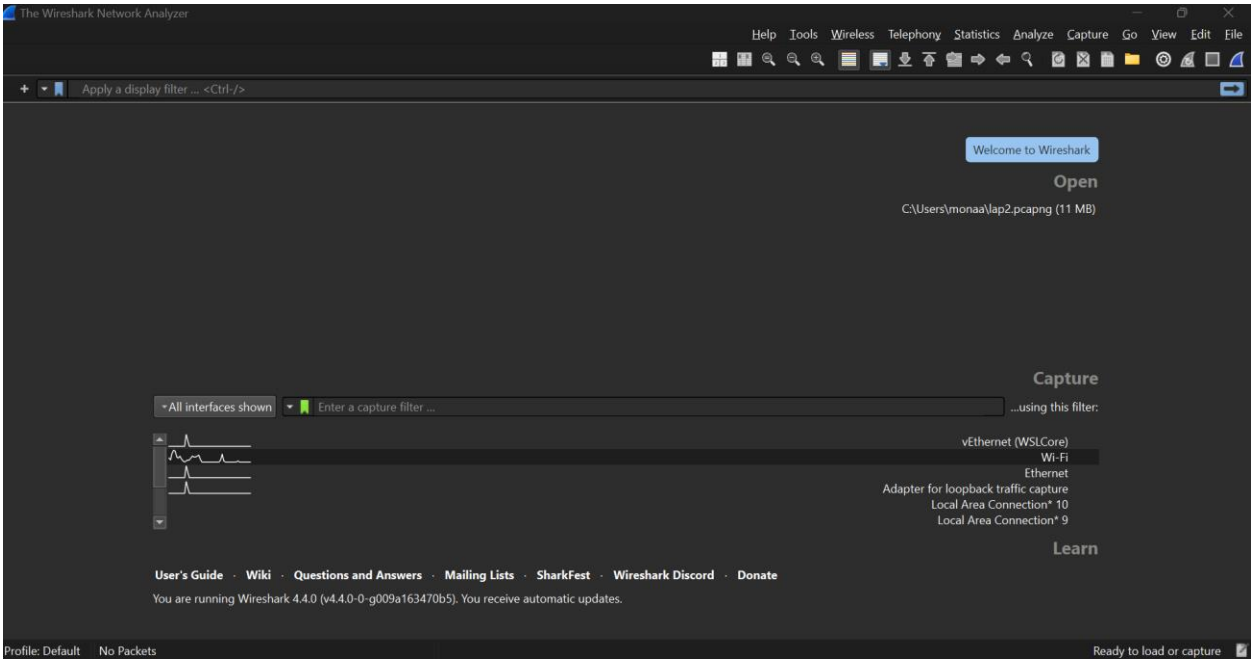
Student ID:421201985

Part 1: Capturing HTTP Traffic.

Task 1: Start Wireshark and capture packets.

Step 1: Open Wireshark.

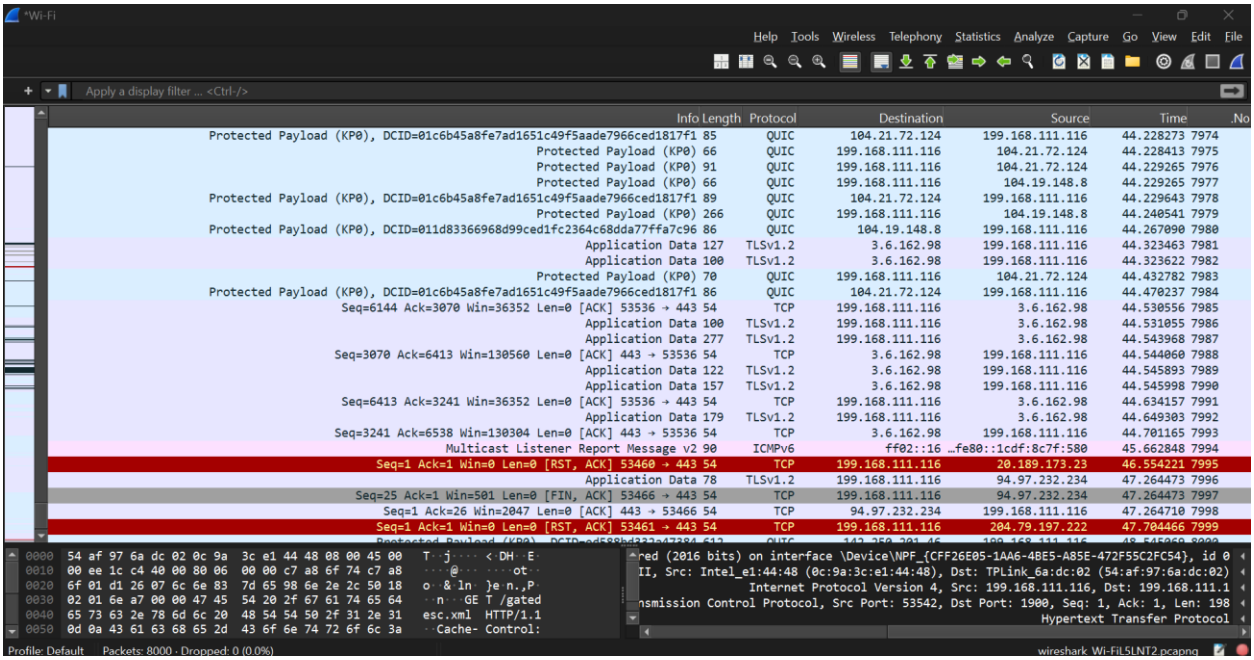
Step 2: Select the network interface connected to the internet (e.g., Ethernet or Wi-Fi)



Step 3: Click the "Start Capturing Packets" button (the shark fin icon).

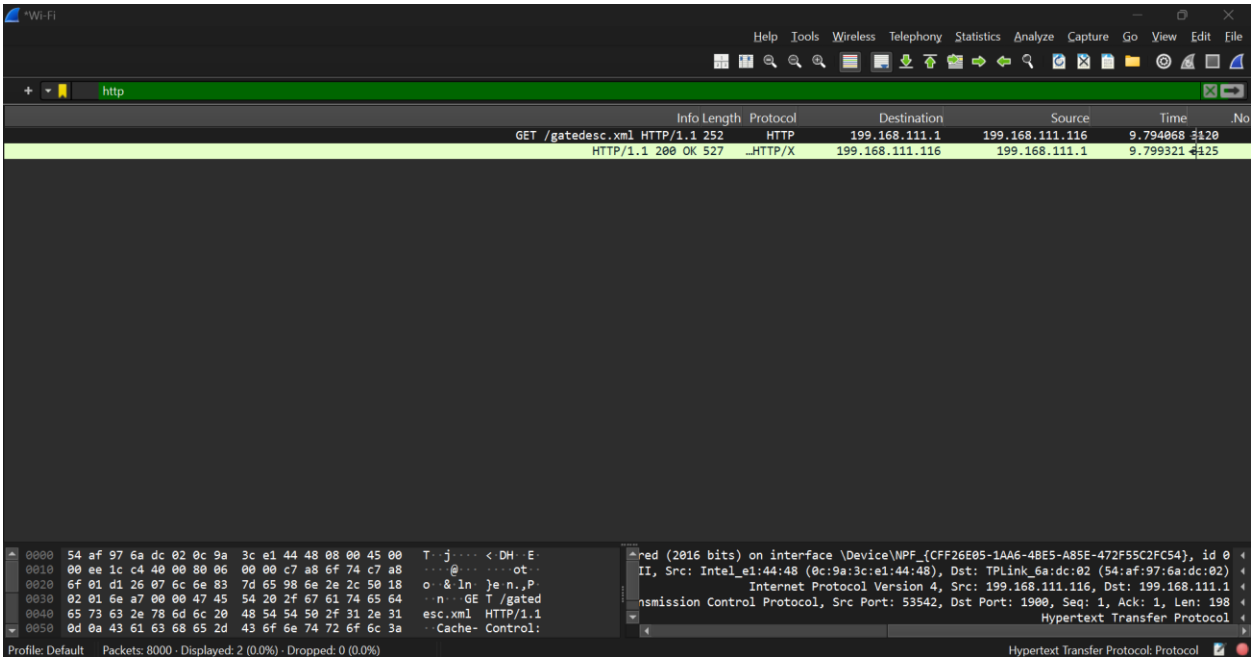
Step 4: Open your favorite web browser and navigate to (https://qu.edu.sa) website.

Step 5: After the website has fully loaded, stop capturing packets by clicking the red stop button in Wireshark.

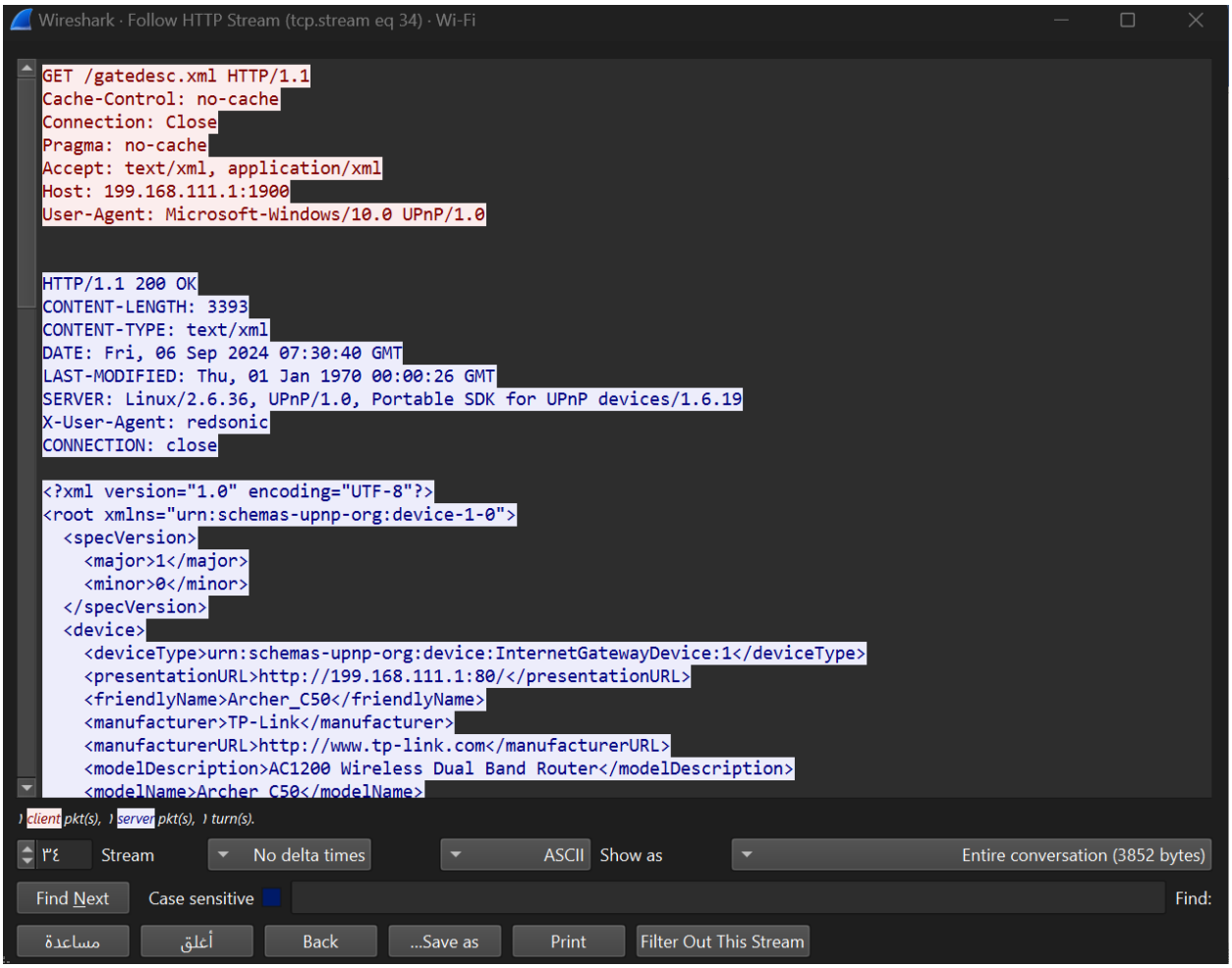


Task 2: Filter HTTP packets and analyze them.

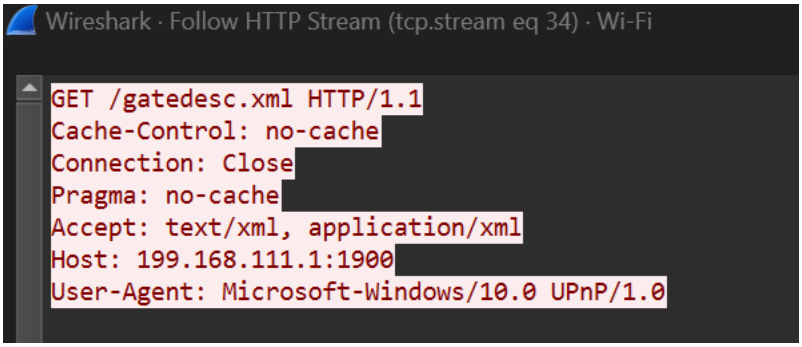
Step 1: In the filter bar, type http and press Enter. This filters out only the HTTP packets from the capture.



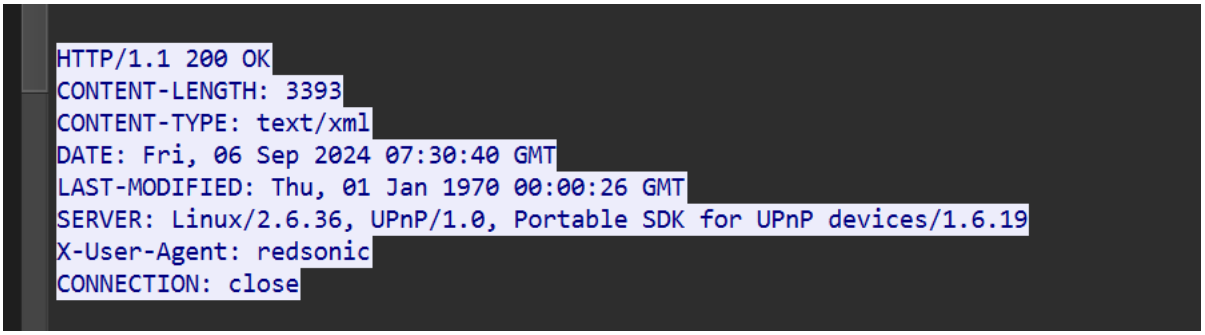
Step 2: Select any HTTP packet to view its details.



First part is HTTP Request:



Second part HTTP Response:



Step 3: Observe the HTTP request and response messages. Note the method (GET, POST), URL, response codes (200 OK, 404 Not Found), etc.

Request Method: GET

This indicates that the client is requesting data from the server.

Request URL: /gatedesc.xml

This is the path on the server for which the client is requesting data.

Response Code: 200 OK

This status code indicates that the request was successful, and the server is returning the requested resource.

Cache Control:

Cache-Control: no-cache

Pragma: no-cache

These headers indicate that the client wants to avoid caching, meaning it requests the most up-to-date version of the resource.

Connection: Close

This header tells the server to close the connection after delivering the response, which can help with managing resource usage on both ends.

Accept: text/xml, application/xml

This specifies the media types the client is willing to receive. The client prefers XML format for the response.

Host: 199.168.111.1:1900

This header specifies the server's IP address and port number to which the request is directed.

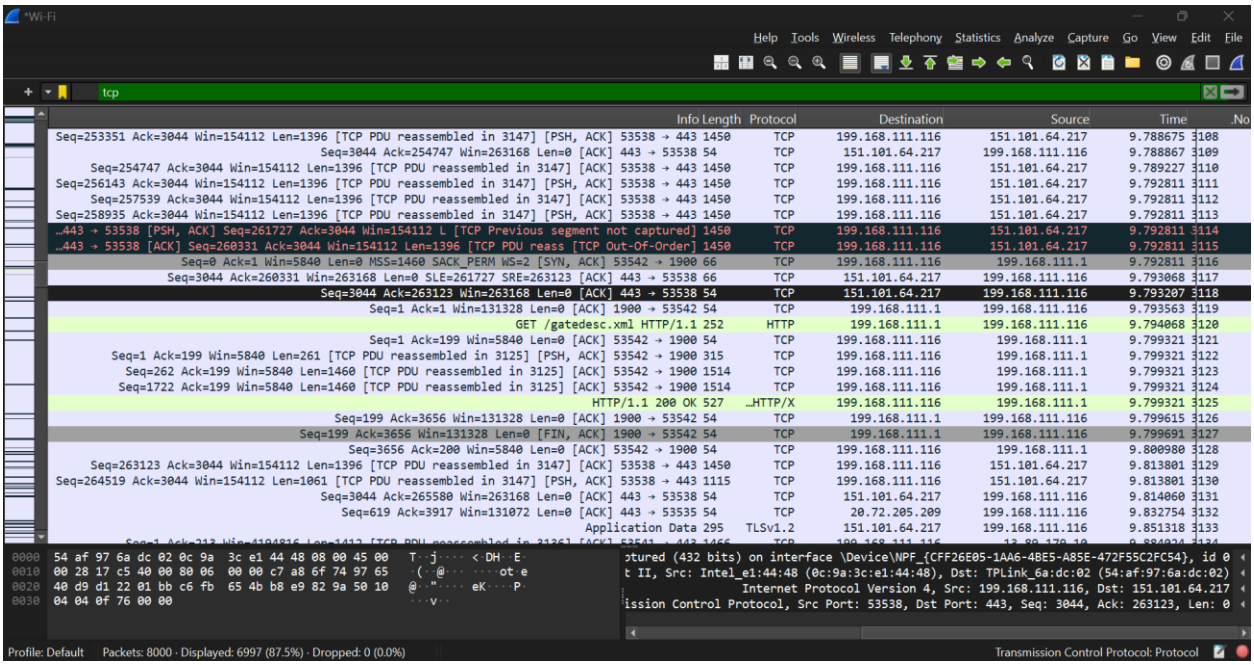
User-Agent: Microsoft-Windows/10.0 UPnP/1.0

This header provides information about the client software making the request. It often includes the OS and application details.

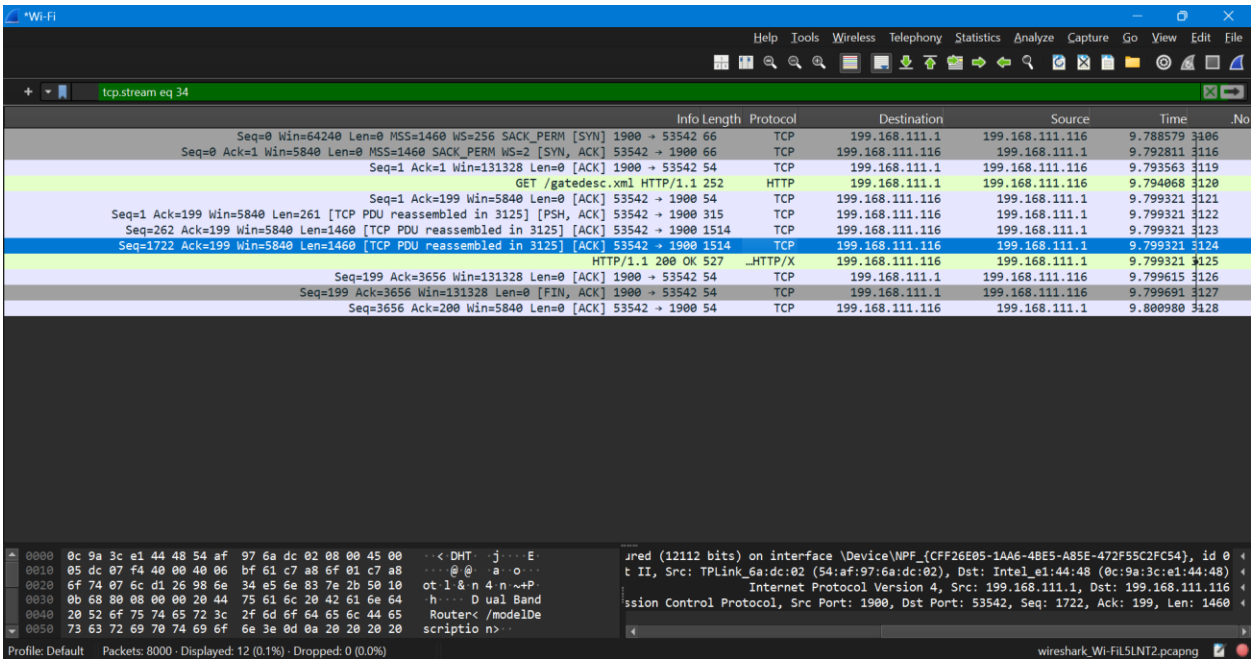
Part 2: Analyzing TCP/IP Traffic.

Task 1: Filter TCP packets

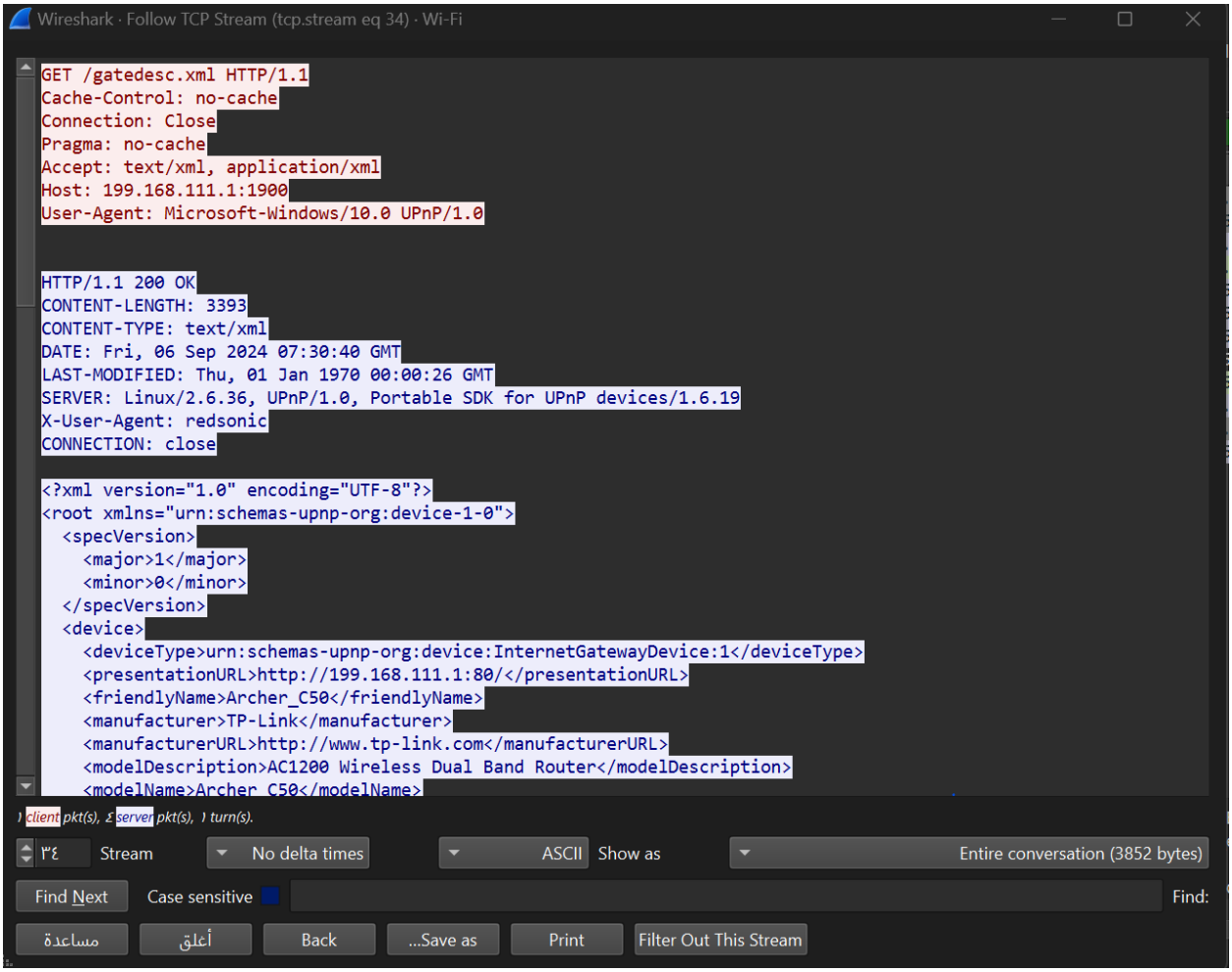
Step 1: Clear the previous filter and type TCP to focus on TCP packets.

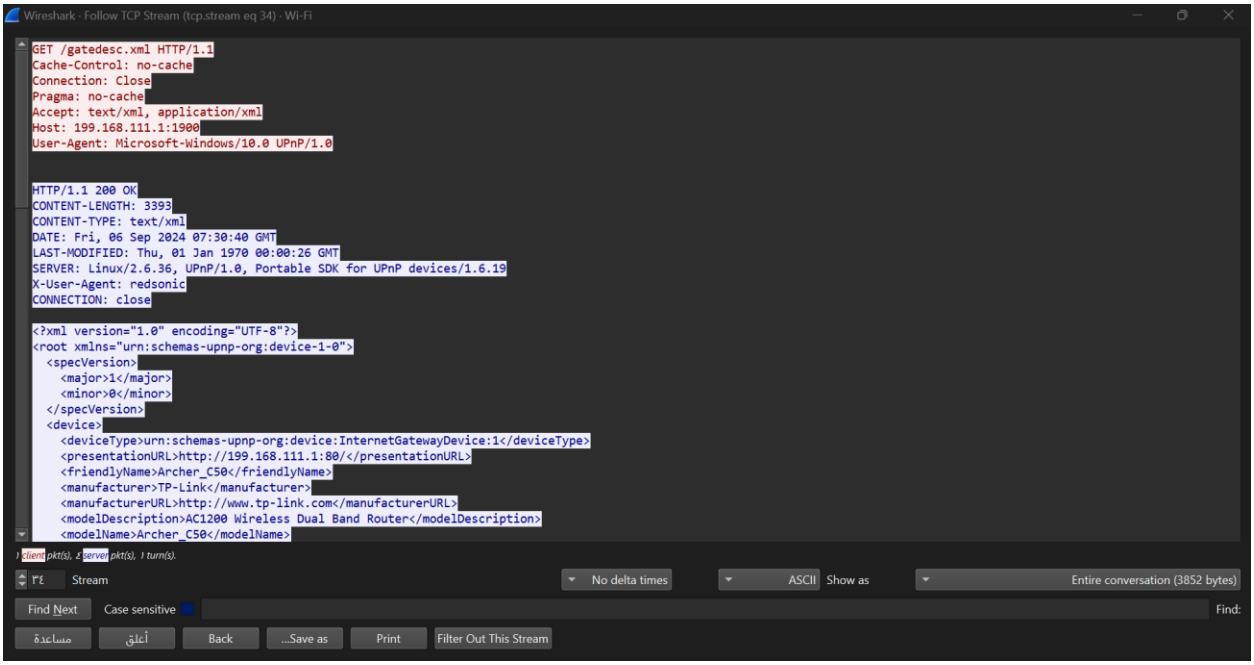


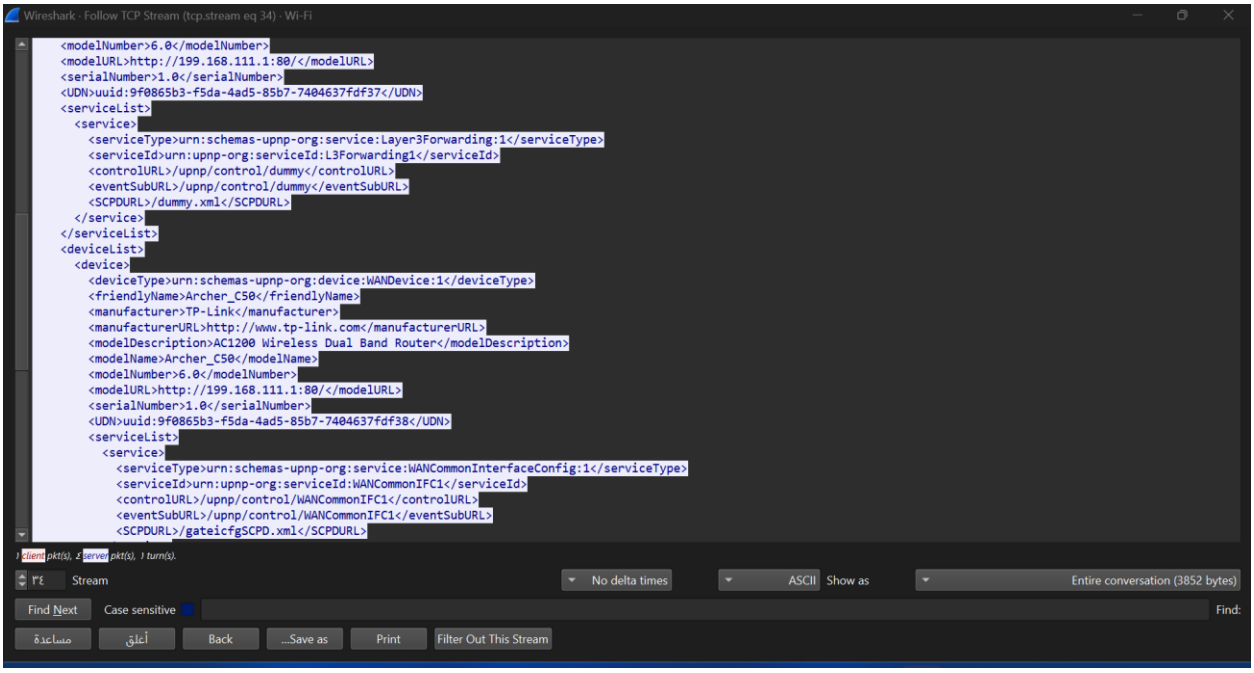
Step 2: Select a TCP packet related to your HTTP request/response.

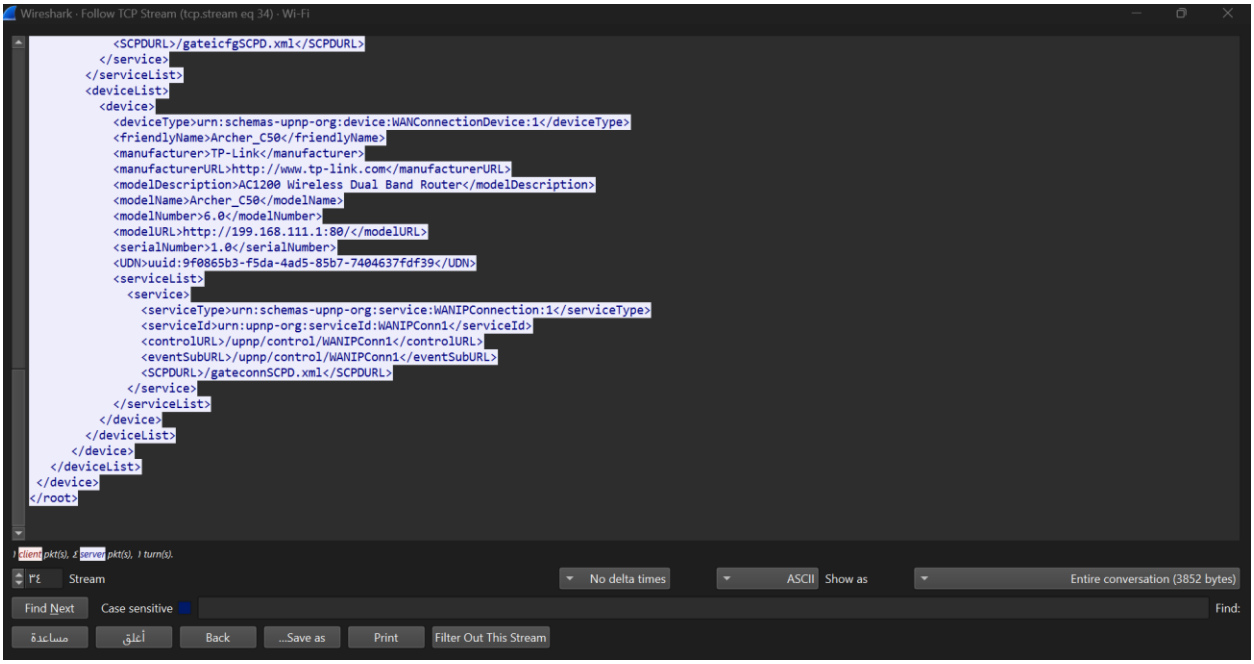


Step 3: Right-click on the packet and select "Follow" -> "TCP Stream".


Wireshark · Follow TCP Stream (tcp.stream eq 34) · Wi-Fi
GET /gatedesc.xml HTTP/1.1
Cache-Control: no-cache
Connection: Close
Pragma: no-cache
Accept: text/xml, application/xml
Host: 199.168.111.1:1900
User-Agent: Microsoft-Windows/10.0 UPnP/1.0
HTTP/1.1 200 OK
CONTENT-LENGTH: 3393
CONTENT-TYPE: text/xml
DATE: Fri, 06 Sep 2024 07:30:40 GMT
LAST-MODIFIED: Thu, 01 Jan 1970 00:00:26 GMT
SERVER: Linux/2.6.36, UPnP/1.0, Portable SDK for UPnP devices/1.6.19
X-User-Agent: redsonic
CONNECTION: close
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
<device>
<deviceType>urn:schemas-upnp-org:device:InternetGatewayDevice:1</deviceType>
<presentationURL>http://199.168.111.1:80/</presentationURL>
<friendlyName>Archer_C50</friendlyName>
<manufacturer>TP-Link</manufacturer>
<manufacturerURL>http://www.tp-link.com</manufacturerURL>
<modelDescription>AC1200 Wireless Dual Band Router</modelDescription>
<modelName>Archer_C50</modelName>
client pkt(s), server pkt(s), turn(s).
Stream
No delta times
ASCII
Show as
Entire conversation (3852 bytes)
Find Next
Case sensitive
Find:
مساعدة
أغلق
Back
...Save as
Print
Filter Out This Stream


Wireshark · Follow TCP Stream (tcp.stream eq 34) · Wi-Fi
GET /gatedesc.xml HTTP/1.1
Cache-Control: no-cache
Connection: Close
Pragma: no-cache
Accept: text/xml, application/xml
Host: 199.168.111.1:1900
User-Agent: Microsoft-Windows/10.0 UPnP/1.0
HTTP/1.1 200 OK
CONTENT-LENGTH: 3393
CONTENT-TYPE: text/xml
DATE: Fri, 06 Sep 2024 07:30:40 GMT
LAST-MODIFIED: Thu, 01 Jan 1970 00:00:26 GMT
SERVER: Linux/2.6.36, UPnP/1.0, Portable SDK for UPnP devices/1.6.19
X-User-Agent: redsonic
CONNECTION: close
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion>
<major>1</major>
<minor>0</minor>
</specVersion>
<device>
<deviceType>urn:schemas-upnp-org:device:InternetGatewayDevice:1</deviceType>
<presentationURL>http://199.168.111.1:80/</presentationURL>
<friendlyName>Archer_C50</friendlyName>
<manufacturer>TP-Link</manufacturer>
<manufacturerURL>http://www.tp-link.com</manufacturerURL>
<modelDescription>AC1200 Wireless Dual Band Router</modelDescription>
<modelName>Archer_C50</modelName>
client pkt(s), server pkt(s), turn(s).
Stream
No delta times
ASCII
Show as
Entire conversation (3852 bytes)
Find Next
Case sensitive
Find:
مساعدة
أغلق
Back
...Save as
Print
Filter Out This Stream





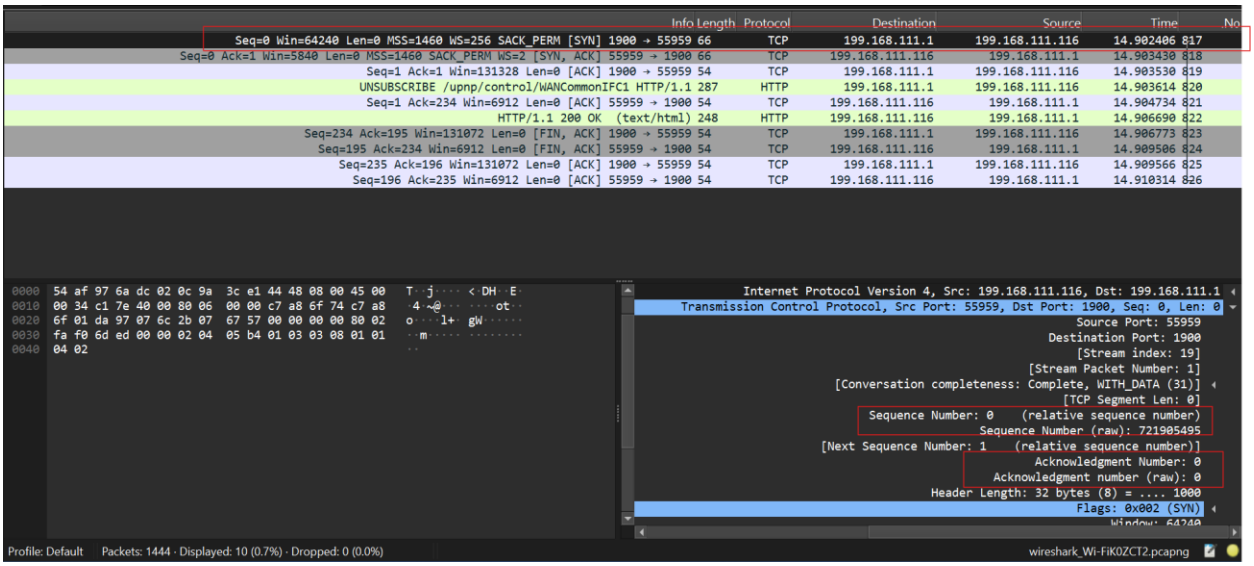
Step 4: This shows the entire conversation between the client and server.

Task 2: Analyze TCP handshake and investigate Data Transfer and Termination

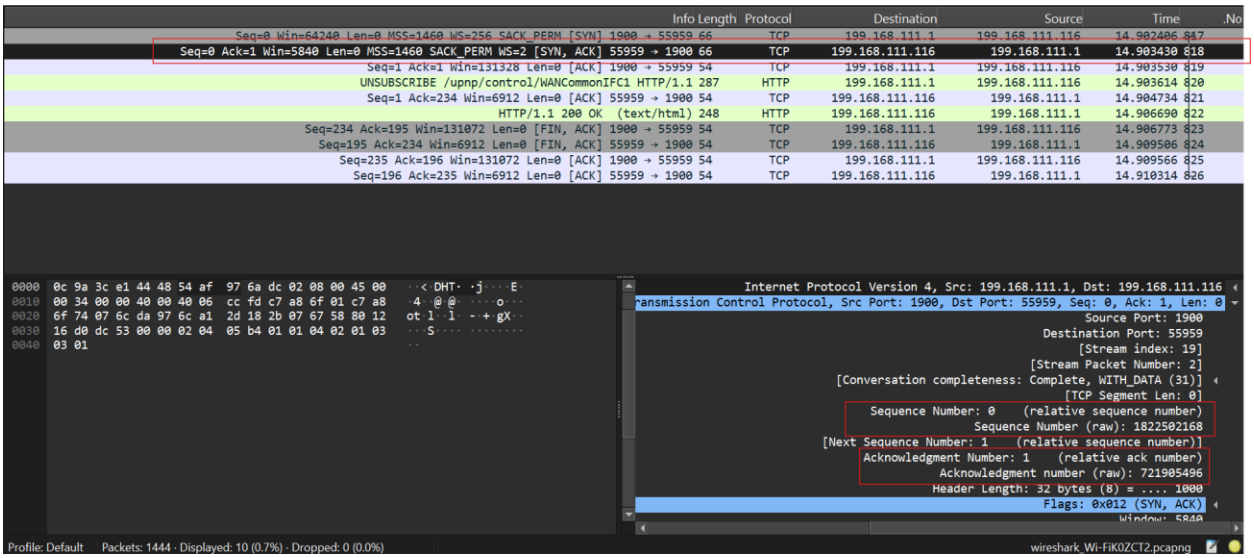
Step 1: Find and select packets related to the TCP three-way handshake:

Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM [SYN] 1900 → 55959 66	TCP	199.168.111.1	199.168.111.116	14.902406 817
Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM WS=2 [SYN, ACK] 55959 → 1900 66	TCP	199.168.111.116	199.168.111.1	14.903430 818
Seq=1 Ack=1 Win=131328 Len=0 [ACK] 1900 → 55959 54	TCP	199.168.111.1	199.168.111.116	14.903530 819

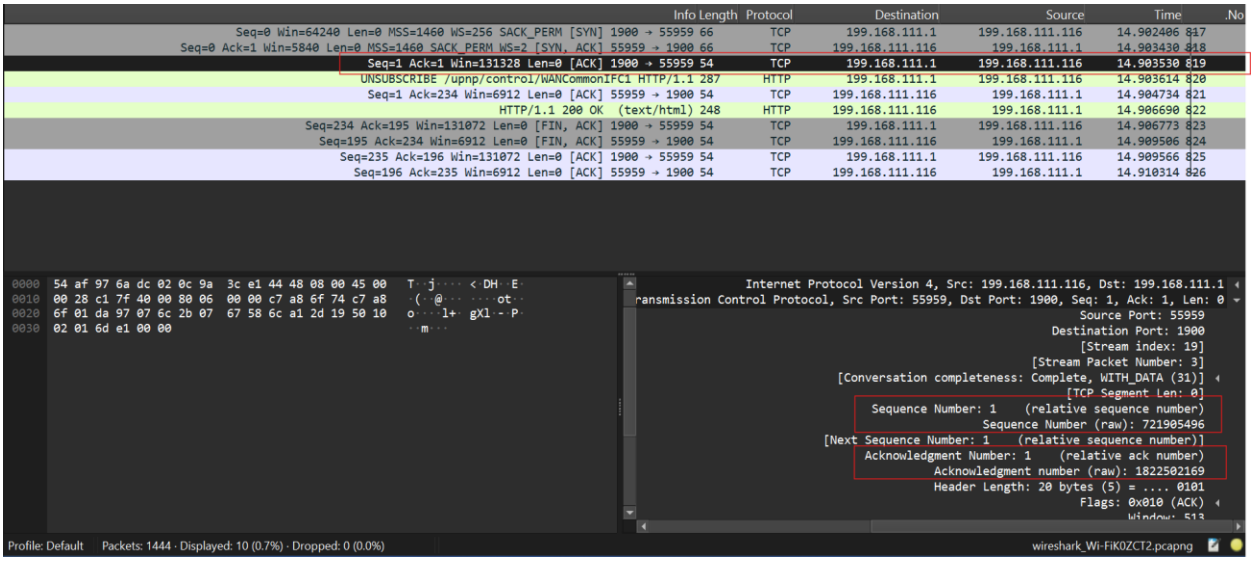
o SYN: Initiates a connection.



o SYN-ACK: Acknowledges and responds to the SYN.



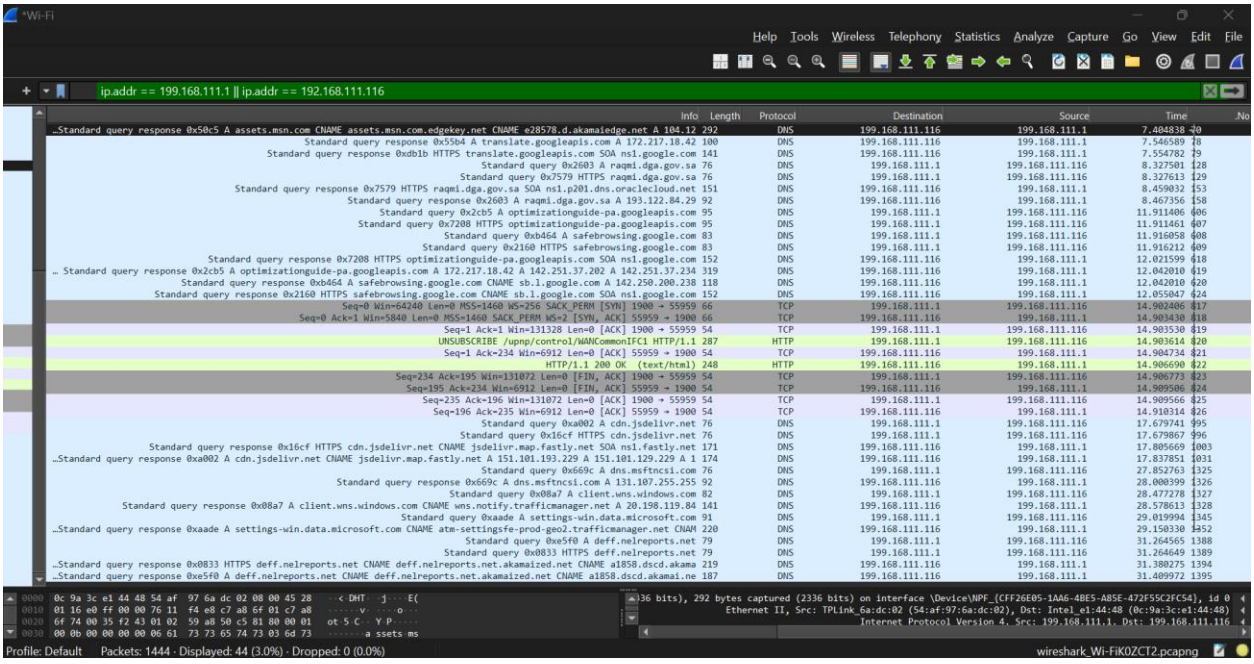
o ACK: Acknowledges the SYN-ACK and establishes the connection.



Step 2: Note the sequence and acknowledgment numbers. Screenshot and upload your image to your online git repository.

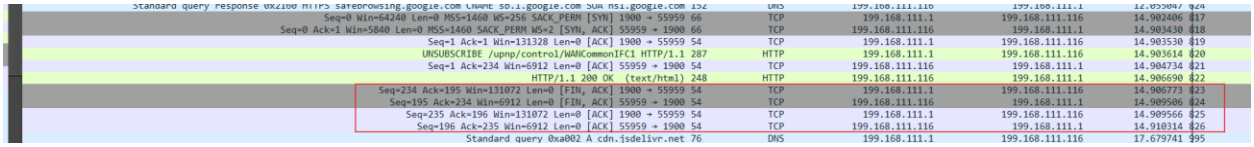
In the three approved images, I used red rectangles to highlight the sequence and acknowledgment numbers for each packet.

Step 3: Observe the data packets exchanged between the client and server.Take a screenshot and upload it to your online git repo.

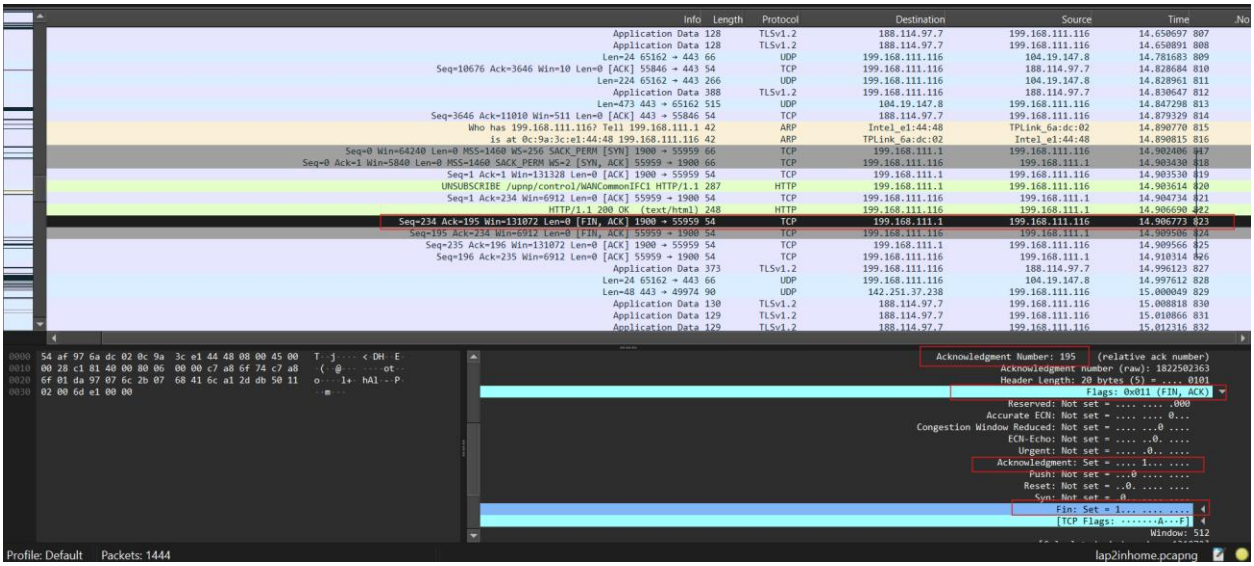


Step 4: Look at the TCP termination process (FIN, ACK packets).

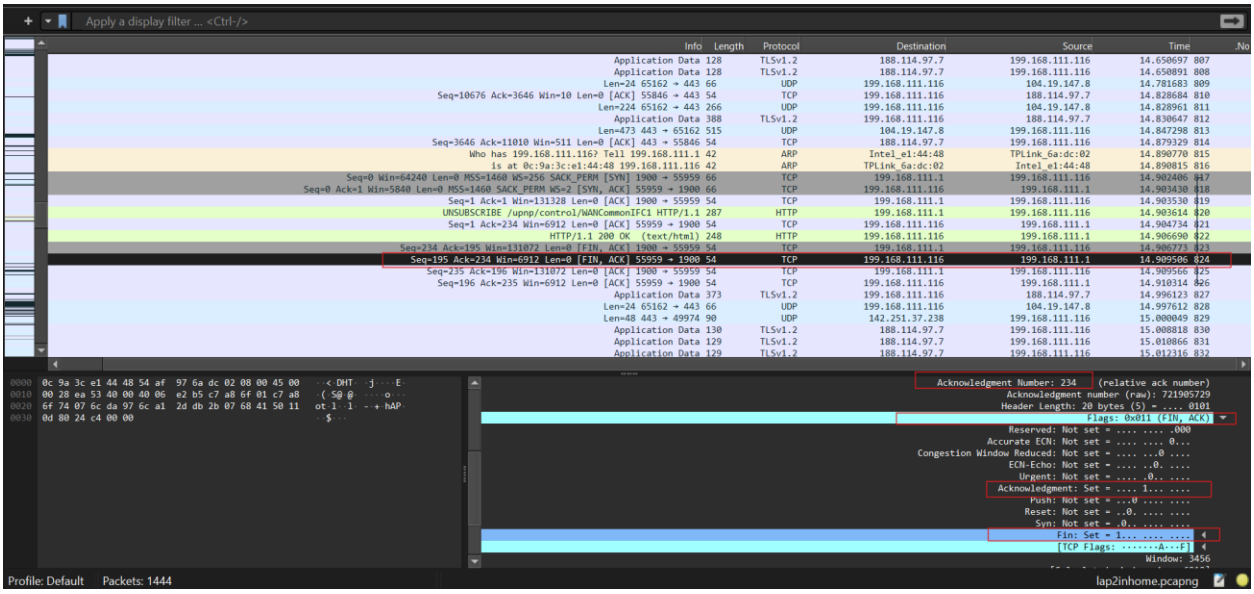
Four packets for TCP termination process



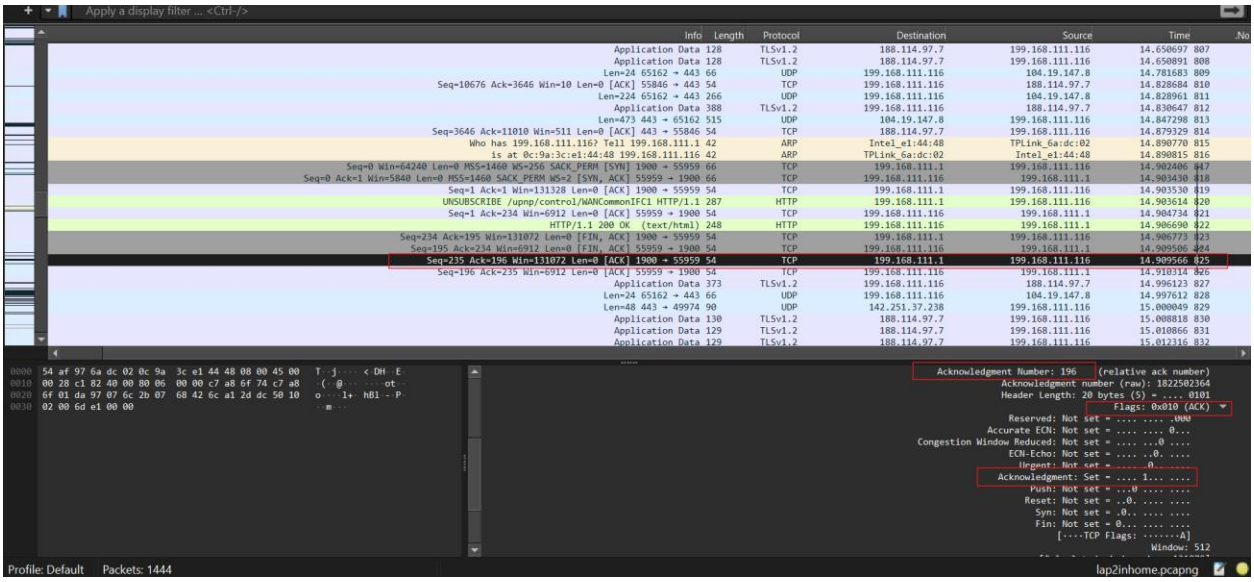
First packet: (from 199.168.111.116 to 199.168.111.1)



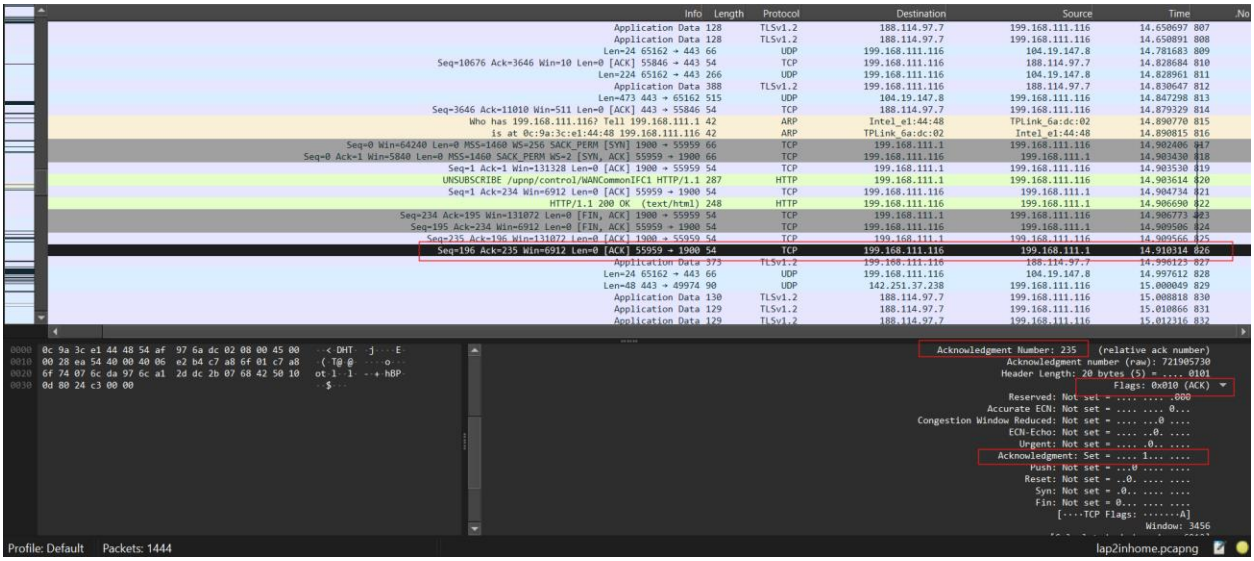
Second packet: (from 199.168.111.1 to 199.168.111.116)



Third Packet: (from 199.168.111.116 to 199.168.111.1)



Fourth Packet: (from 199.168.111.1 to 199.168.111.116)



Part 3: Capturing and Analyzing UDP Traffic

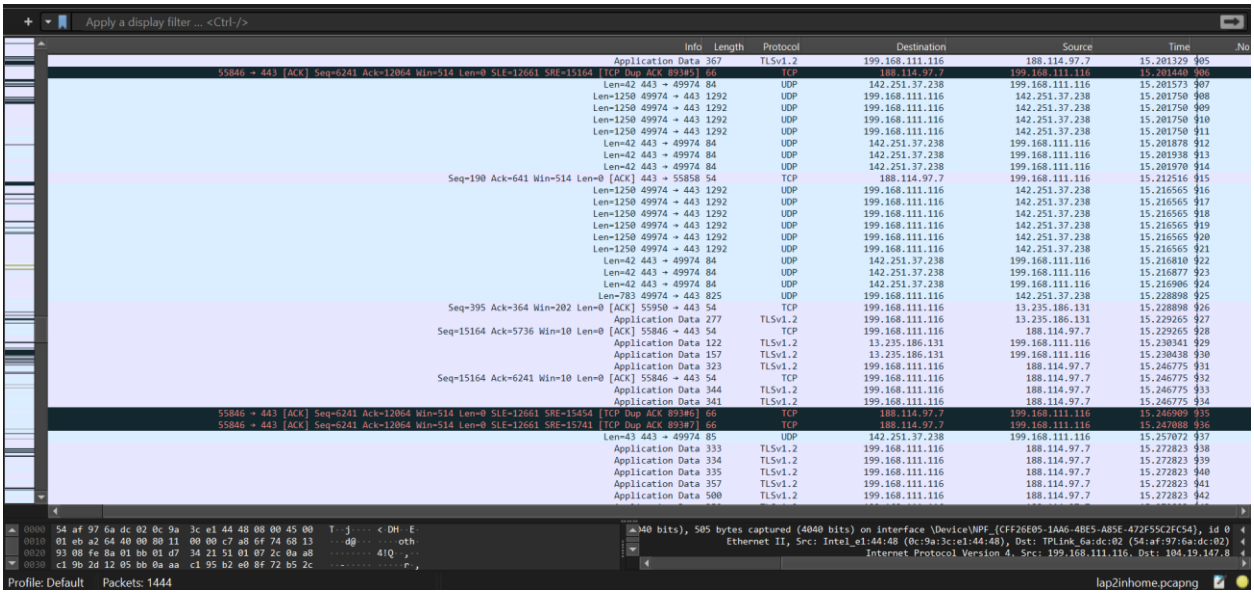
Task 1: Generate UDP traffic and capture packets

Step 1: Open a network application that uses UDP (e.g., streaming video, VoIP software, or custom script).

Step 2: Start the application to generate UDP traffic.

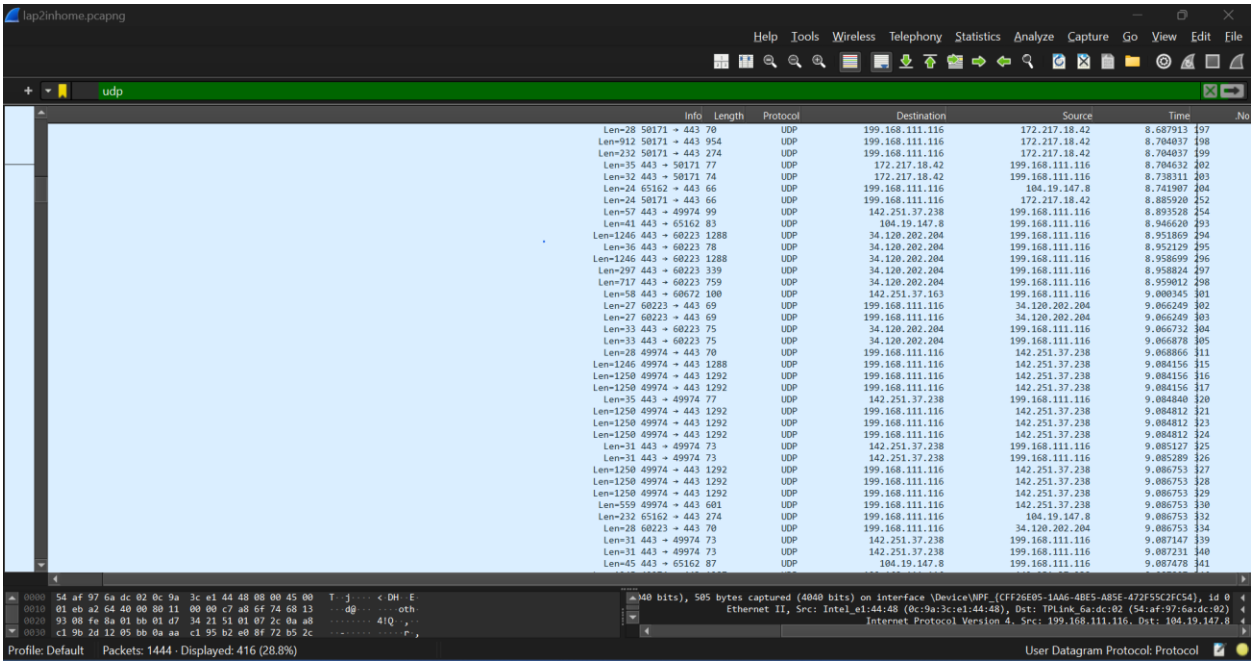
Step 3: Start capturing packets in Wireshark while the UDP application is running.

Step 4: After sufficient traffic is generated, stop capturing packets.



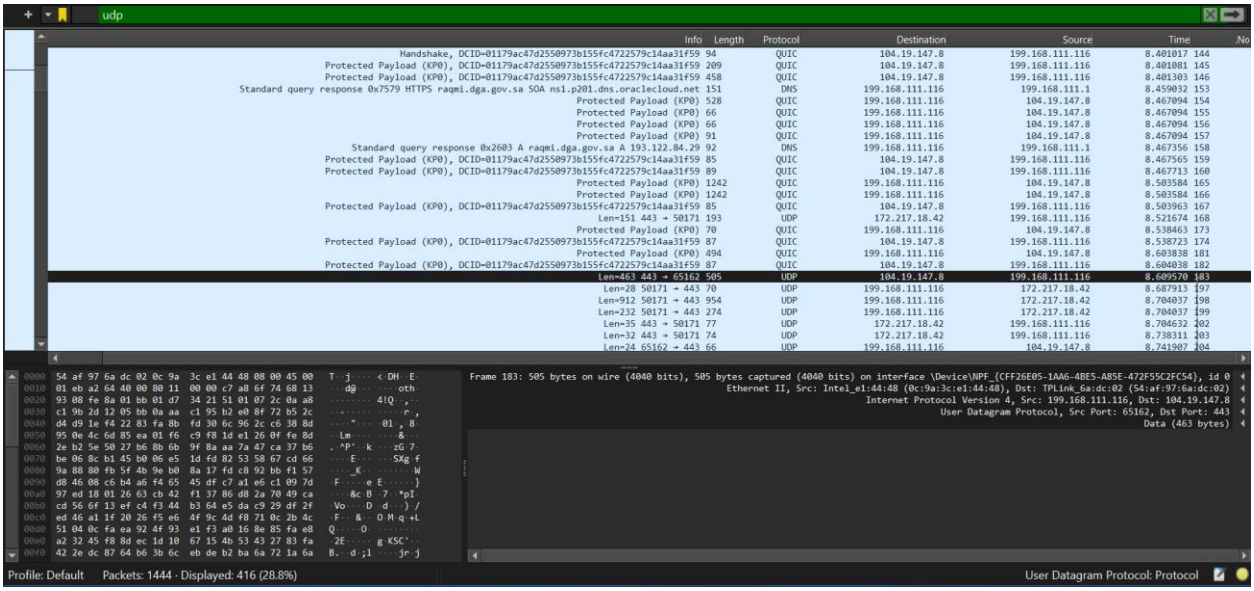
Task 2: Filter and analysis UDP Packets

Step 1: In the filter bar, type UDP and press Enter.

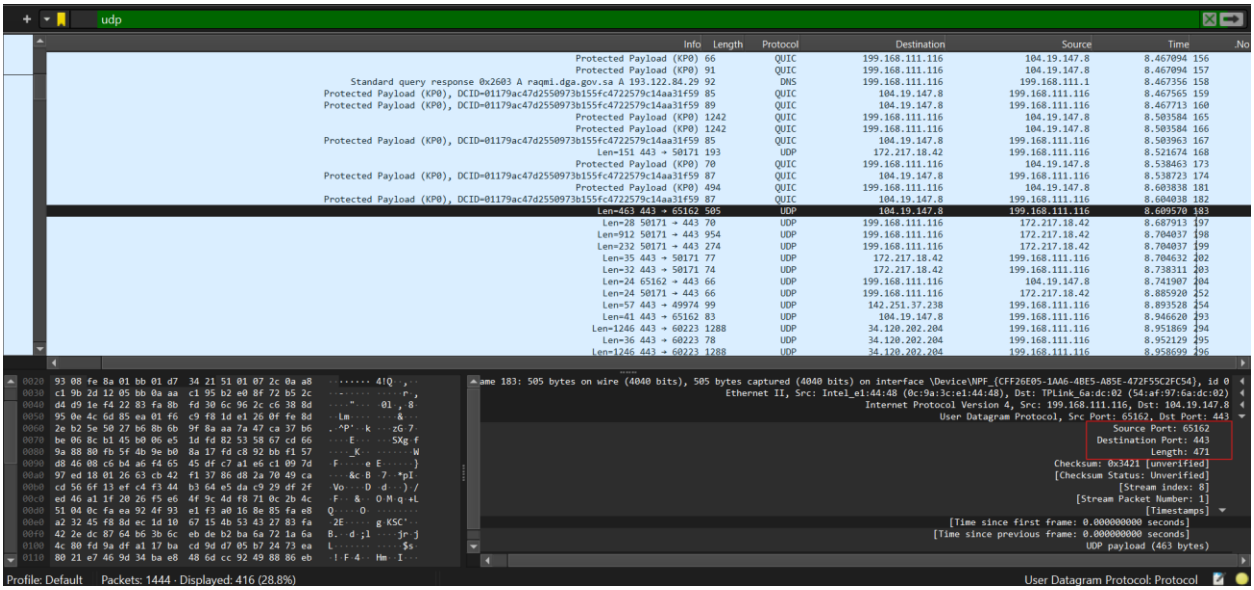


Step 2: This filters out only the UDP packets from the capture.

Step 3: Select any UDP packet to view its details.

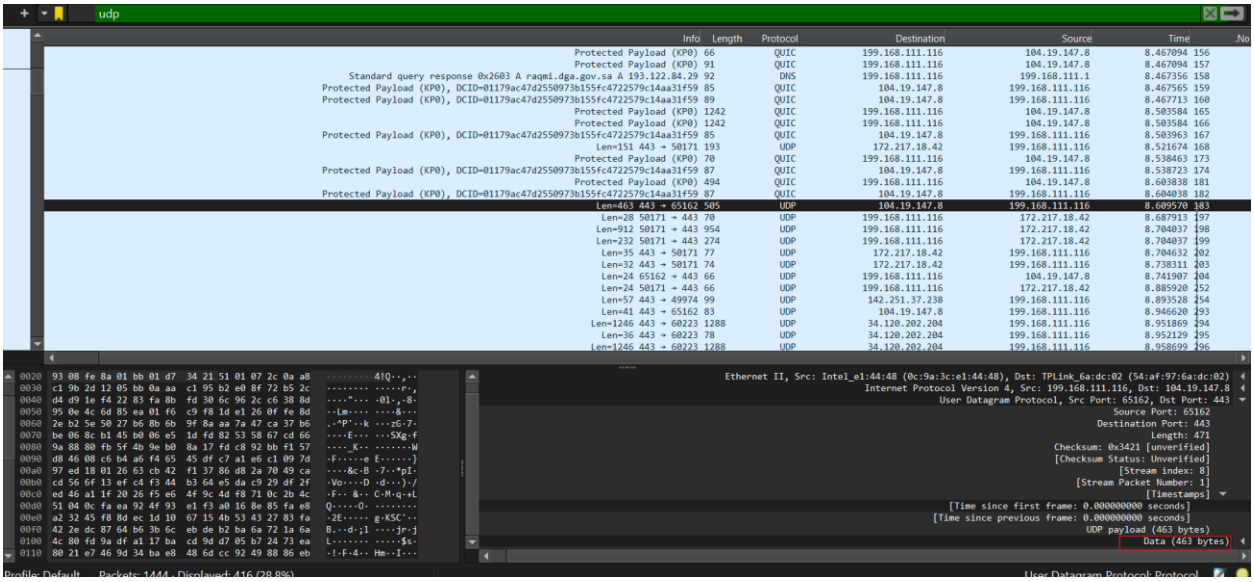


Step 4: Observe the source and destination ports, length, and data.



Source port (16bit):65162

Destination port (16bit):443



Length (471 bytes): Total size of the UDP packet (header + data).

Data (463 bytes): Actual data size, excluding the UDP header.

Header Size = Total Length - Data Size

Header Size = 471 bytes - 463 bytes = 8 bytes

Step 5: Compare the simplicity of UDP headers with TCP headers.

Header Size:

UDP: The UDP header is fixed at 8 bytes. Its simplicity provides a streamlined approach to packet delivery.

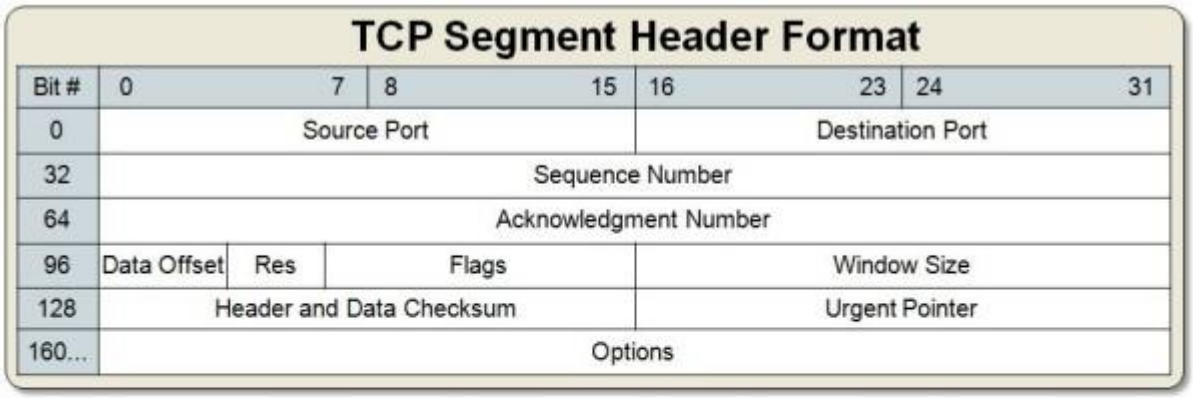
TCP: The TCP header ranges from a minimum of 20 bytes to a maximum of 60 bytes or more, depending on the options and padding used. This variable length supports TCP's advanced features and ensures reliable communication.

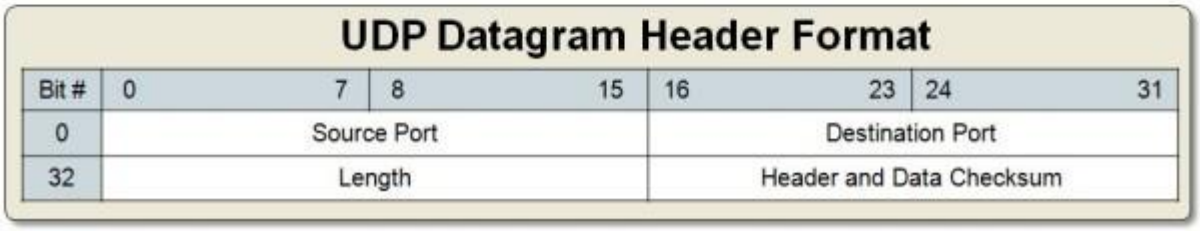
Simplicity:

The UDP header is simpler with only 4 fields, making it faster and more efficient for applications that do not require reliable delivery.

Complexity:

The TCP header is more complex with additional fields for ensuring reliable, ordered, and error-checked delivery of data.





Part 4: Comparing TCP and UDP by filling in the following tables. Save your work (e.g., in an MS Word document), and upload it to your online git repo.

Task 1: Fill in the following table and provide reasons.

	TCP or UDP	Reasons
Reliability and Connection Establishment	TCP	TCP provides reliable data transfer with acknowledgments, retransmissions, and error recovery. As a connection-oriented protocol, it requires a handshake to establish a connection before data transfer begins.
Data Integrity and Ordering	TCP	TCP ensures data integrity with checksums, which detect errors and prompt retransmission if needed. It maintains data ordering by using sequence numbers, which help reassemble packets in the correct order even if they arrive out of sequence.in the same sequence they were sent.

Task 2: Identify the use Cases and Performance of TCP and UDP

	TCP	UDP
Use cases	-Web Browsing: Hypertext Transfer Protocol (HTTP) / Hypertext Transfer Protocol Secure (HTTPS) -File Transfers: File Transfer Protocol (FTP) -Email: Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP), Post Office Protocol (POP3) -Secure Communications: Secure Sockets Layer (SSL) / Transport Layer Security (TLS)	-Live Streaming -Voice over IP (VoIP) -Domain Name System (DNS) Queries -Broadcasting -Online Gaming
Performance	- High reliability with error detection, acknowledgment, and retransmissions - Higher latency due to connection setup and error recovery - Suitable for high-throughput scenarios	- Lower reliability; no guarantees for delivery, ordering, or integrity - Lower latency due to lack of connection setup and acknowledgments - Higher throughput potential but can suffer from packet loss and ordering issues