

Lab Week 2

The Internet Protocols

Student name: Mona Hamdan Aloufi

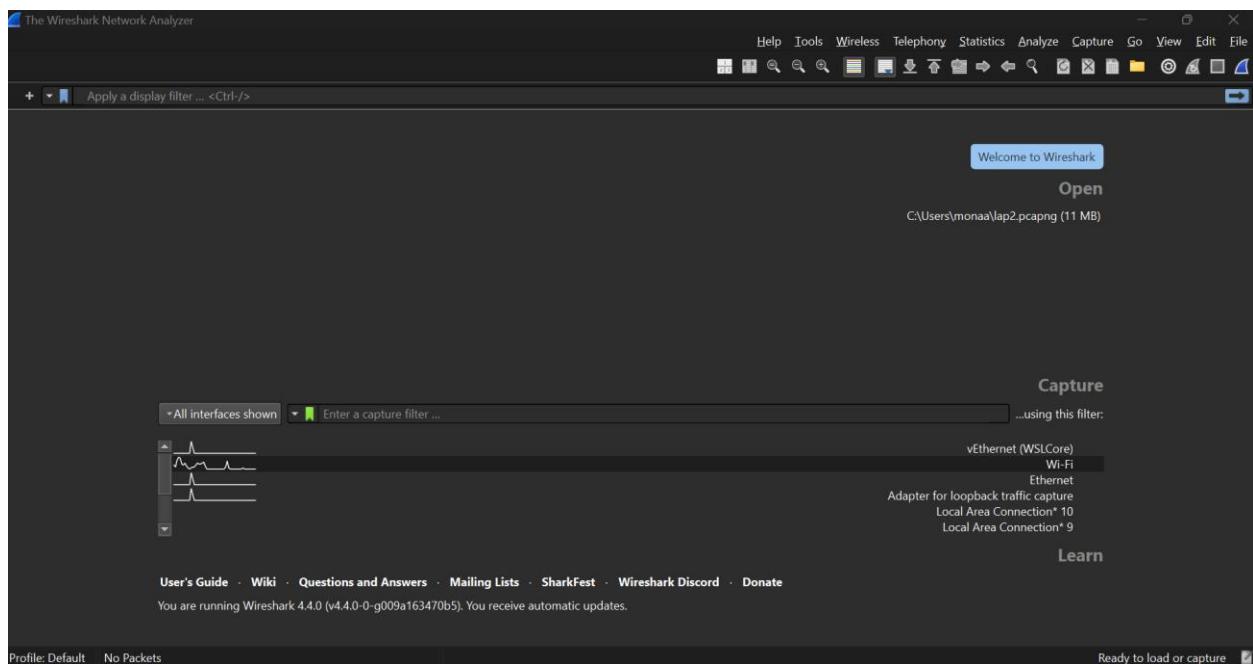
Student ID: 421201985

Part 1: Capturing HTTP Traffic.

Task 1: Start Wireshark and capture packets.

Step 1: Open Wireshark.

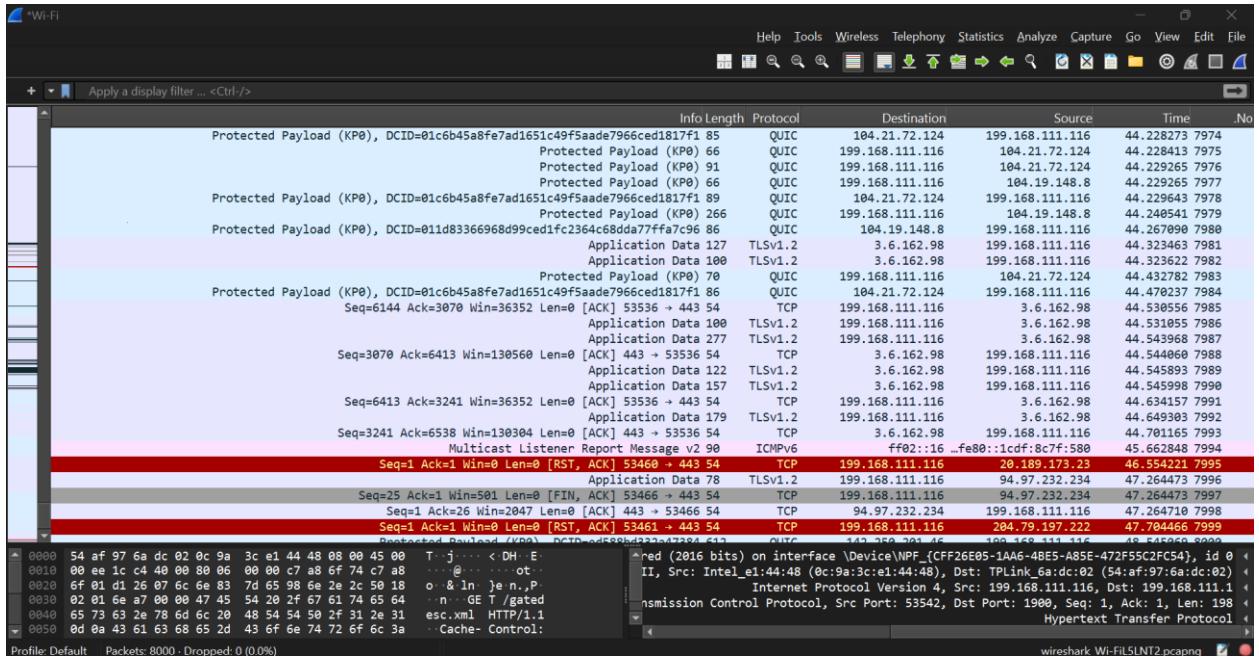
Step 2: Select the network interface connected to the internet (e.g., Ethernet or Wi-Fi)



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

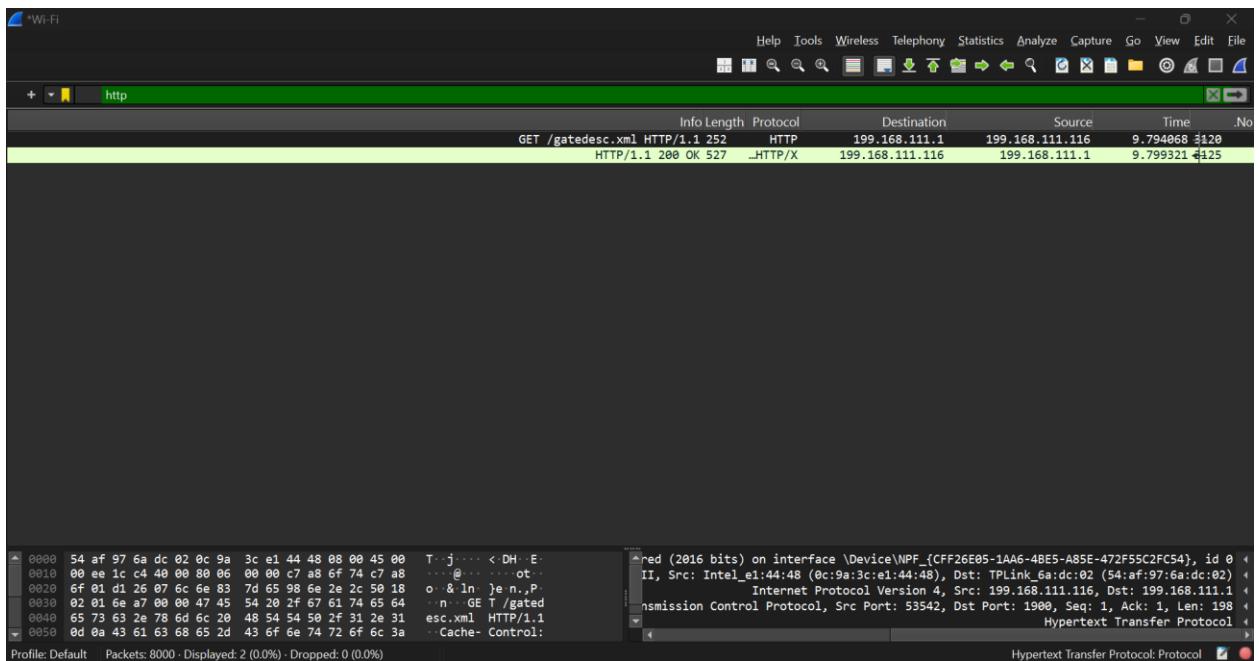
Step 4: Open your favorite web browser and navigate to (<https://qu.edu.sa>) website.

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.

```
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>
  </device>
</root>
```

client pkt(s), server pkt(s), 1 turn(s).

Stream No delta times ASCII Show as Entire conversation (3852 bytes)

Find Next Case sensitive Find:

مساعدة أعلق Back ...Save as Print Filter Out This Stream

First part is HTTP Request:

```
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
```

Second part HTTP Response:

```
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
```

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.

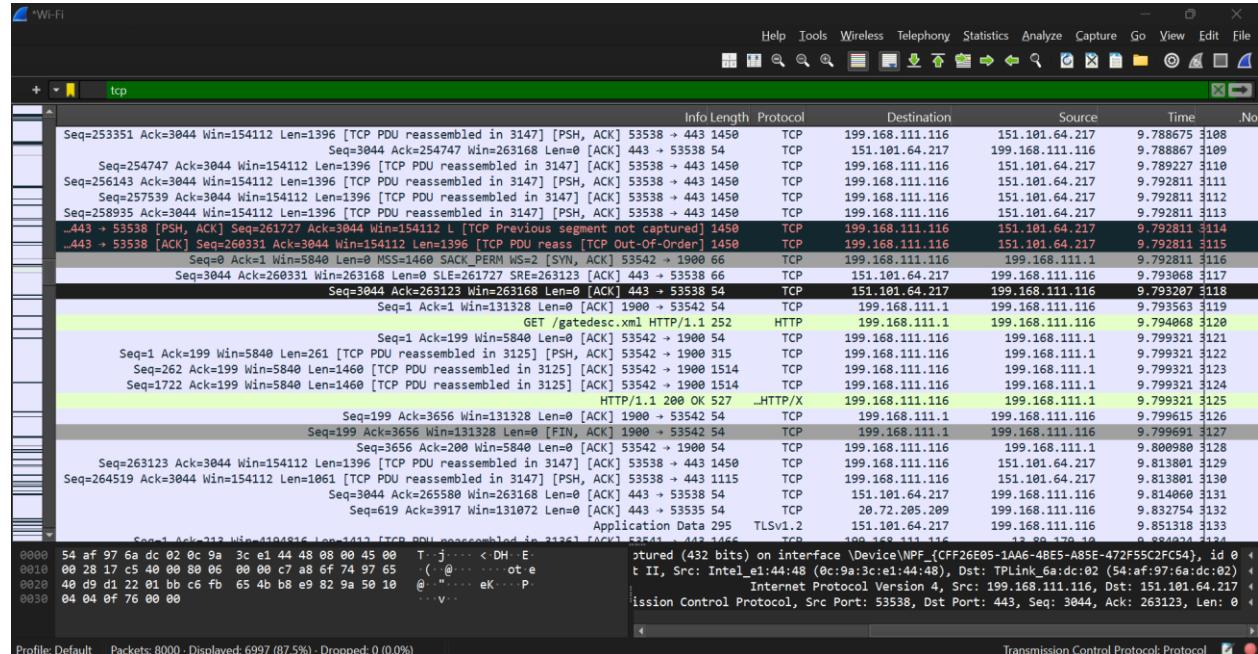
User-Agent: Microsoft-Windows/10.0 UPnP/1.0

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

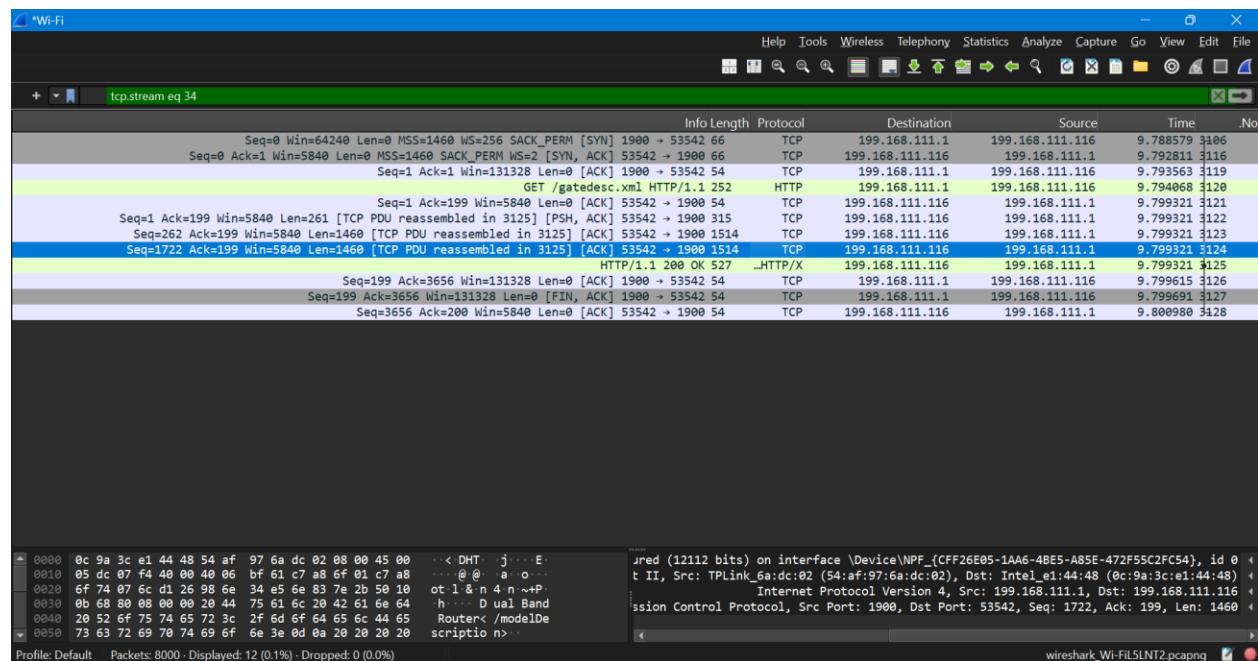
Part 2: Analyzing TCP/IP Traffic.

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>
  </device>
</root>

client pkt(s), server pkt(s), 1 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>
  </device>
</root>

client pkt(s), server pkt(s), 1 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

```

<modelNumber>6.0</modelNumber>
<modelURL>http://199.168.111.1:80/</modelURL>
<serialNumber>1.0</serialNumber>
<UDN>uuid:f9f865b3-f5da-4ad5-85b7-7404637fdf37</UDN>
<serviceList>
  <service>
    <serviceType>urn:schemas-upnp-org:service:Layer3Forwarding:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:L3Forwarding1</serviceId>
    <controlURL>/upnp/control/dummy</controlURL>
    <eventSubURL>/upnp/control/dummy</eventSubURL>
    <SCPDURL>/dummy.xml</SCPDURL>
  </service>
</serviceList>
<device>
  <deviceType>urn:schemas-upnp-org:device:WANDevice:1</deviceType>
  <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>
  <modelNumber>6.0</modelNumber>
  <modelURL>http://199.168.111.1:80/</modelURL>
  <serialNumber>1.0</serialNumber>
  <UDN>uuid:f9f865b3-f5da-4ad5-85b7-7404637fdf38</UDN>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:WANCommonIFC1</serviceId>
      <controlURL>/upnp/control/WANCommonIFC1</controlURL>
      <eventSubURL>/upnp/control/WANCommonIFC1</eventSubURL>
      <SCPDURL>/gateicfgSCPD.xml</SCPDURL>
    </service>
  </serviceList>

```

client pkt(s), 2 server pkt(s), 1 turn(s).

Stream Find Next Case sensitive Back ...Save as Print Filter Out This Stream

Find: Entire conversation (3852 bytes)

Wireshark - Follow TCP Stream (tcp.stream eq 34) - Wi-Fi

```

<SCPDURL>/gateicfgSCPD.xml</SCPDURL>
</service>
</serviceList>
<deviceList>
<device>
  <deviceType>urn:schemas-upnp-org:device:WANConnectionDevice:1</deviceType>
  <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>
  <modelNumber>6.0</modelNumber>
  <modelURL>http://199.168.111.1:80/</modelURL>
  <serialNumber>1.0</serialNumber>
  <UDN>uuid:9f0f865b3-f5da-4ad5-85b7-7404637fdf39</UDN>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-org:service:WANIPConnection:1</serviceType>
      <serviceId>urn:upnp-org:serviceId:WANIPConn1</serviceId>
      <controlURL>/upnp/control/WANIPConn1</controlURL>
      <eventSubURL>/upnp/control/WANIPConn1</eventSubURL>
      <SCPDURL>/gateconnSCPD.xml</SCPDURL>
    </service>
  </serviceList>
</device>
</deviceList>
</device>
</deviceList>
</root>

```

client pkt(s), 2 server pkt(s), 1 turn(s).

Stream Find Next Case sensitive Back ...Save as Print Filter Out This Stream

Find: Entire conversation (3852 bytes)

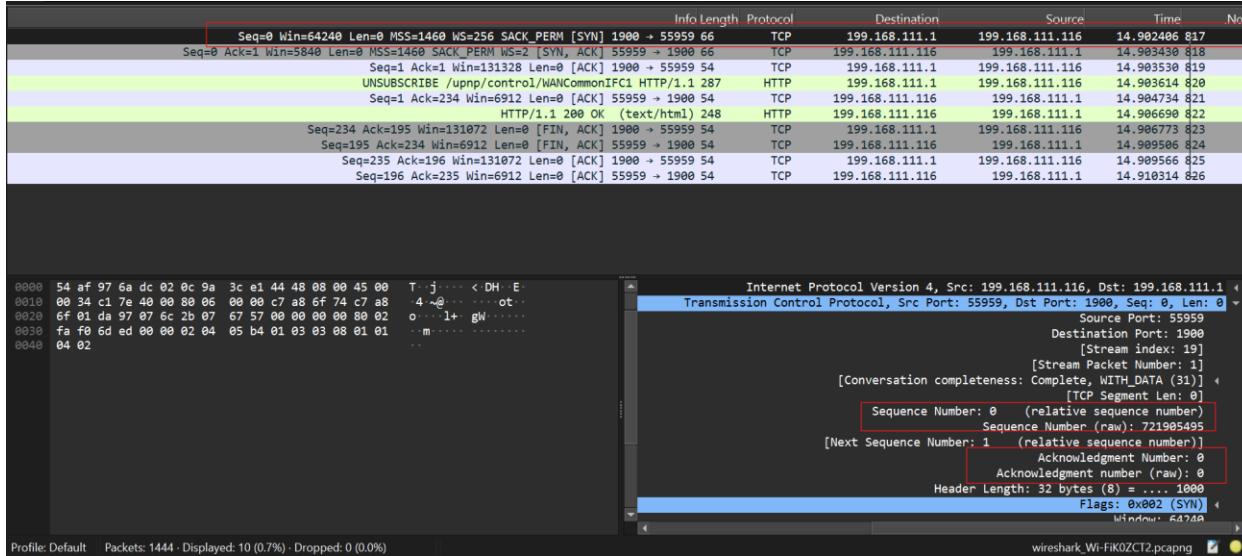
Step 4: This shows the entire conversation between the client and server.

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

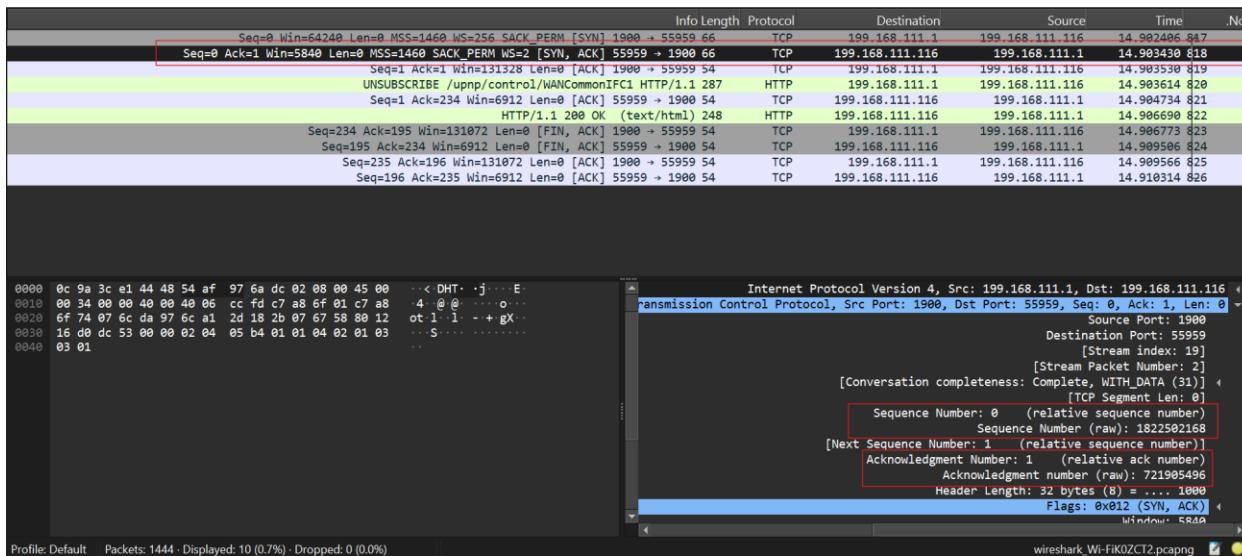
Step 1: Find and select packets related to the TCP three-way handshake:

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	199.168.111.116	199.168.111.1	14.903530 819

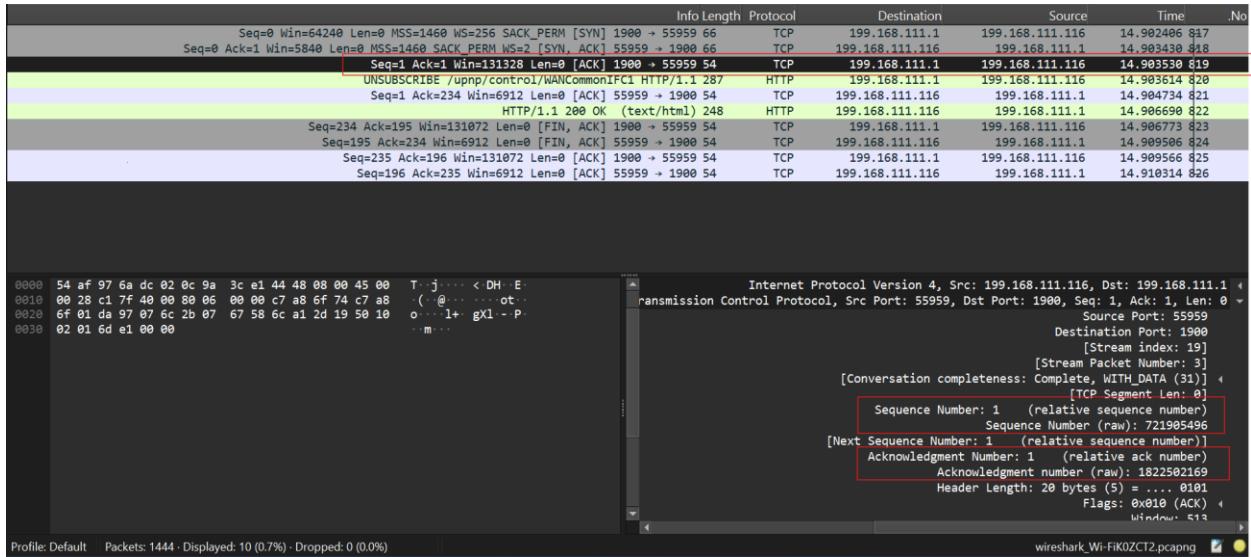
o SYN: Initiates a connection.



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



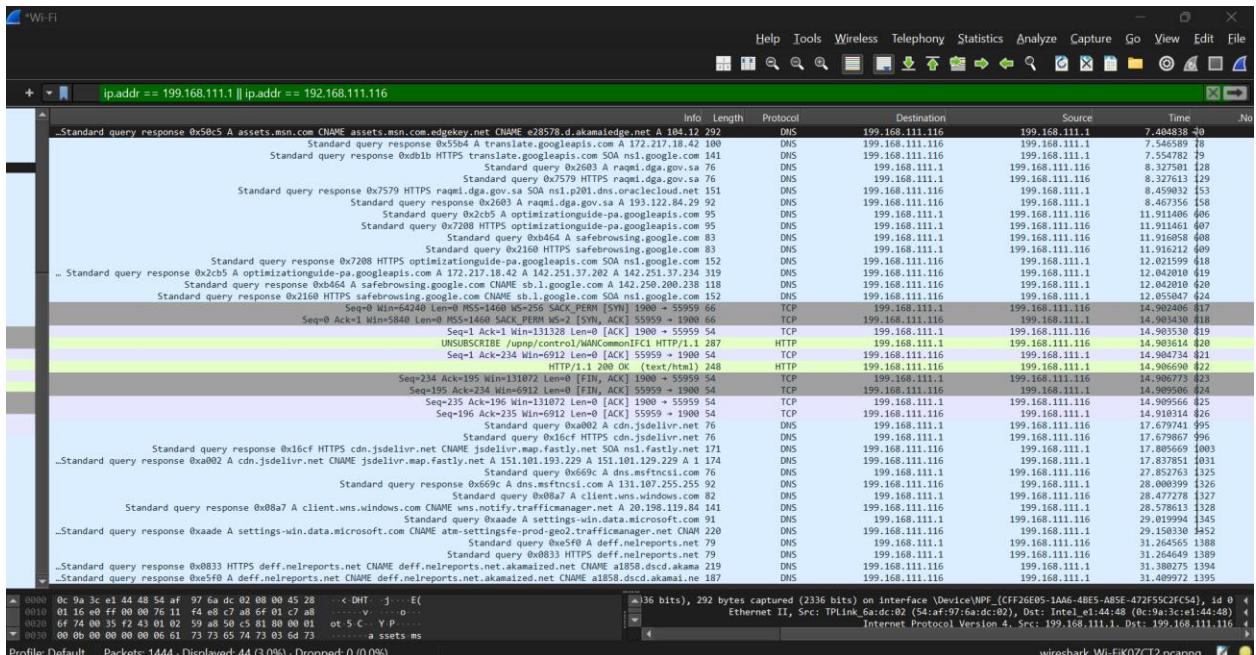
o ACK: Acknowledges the SYN-ACK and establishes the connection.



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.

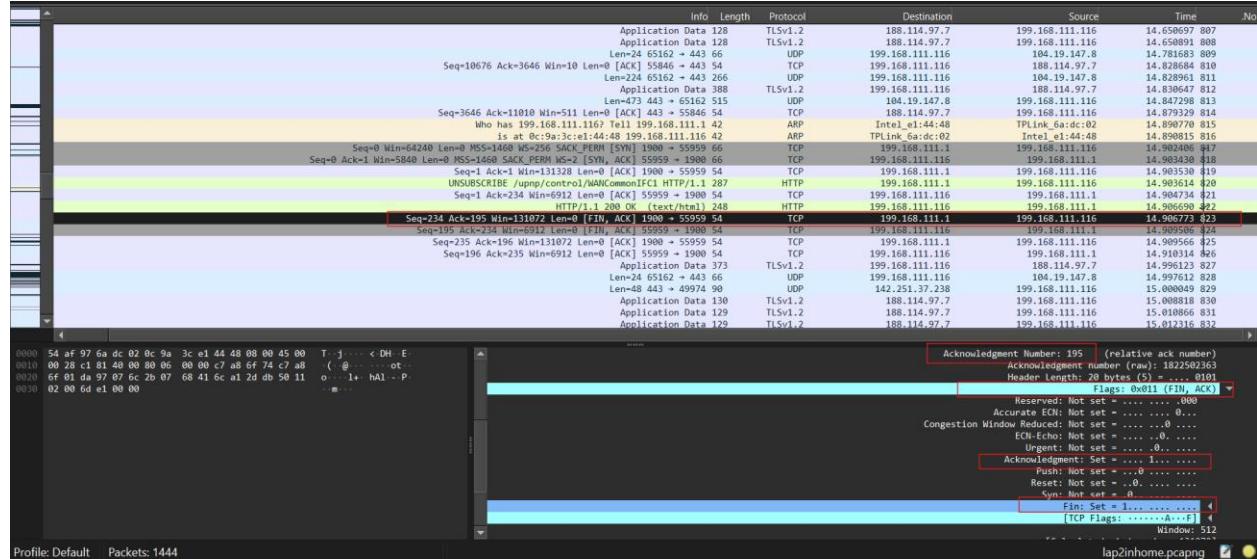


Step 4: Look at the TCP termination process (FIN, ACK packets).

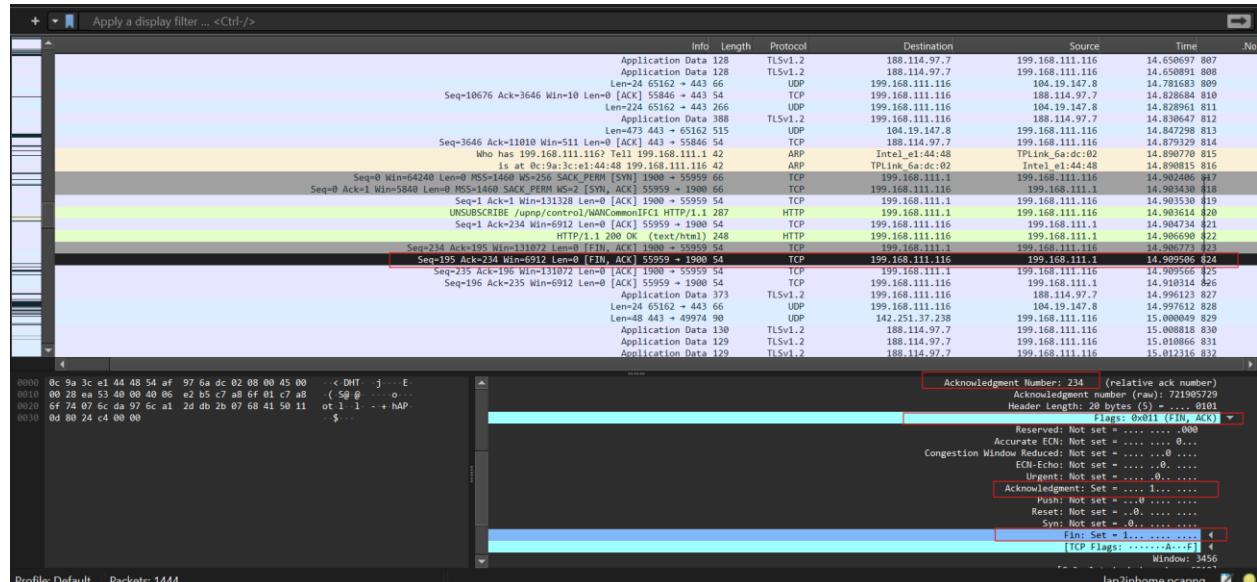
Four packets for TCP termination process

	Info	Length	Protocol	Destination	Source	Time	No
Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM [SYN] 1900 ~ 55959 66	TCP	199.168.111.110	199.168.111.1	14.902406 417			
Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM WS=2 [SYN, ACK] 55959 ~ 1900 54	TCP	199.168.111.116	199.168.111.1	14.903438 418			
Seq=1 Ack=1 Win=113128 Len=0 [ACK] 1900 ~ 55959 54	TCP	199.168.111.11	199.168.111.116	14.903538 419			
UNSUBSCRIBE /urnp/control/WANCommonIFC1 HTTP/1.1 287	HTTP	199.168.111.1	199.168.111.116	14.903614 420			
Seq=1 Ack=234 Win=6912 Len=0 [ACK] 55959 ~ 1900 54	TCP	199.168.111.116	199.168.111.1	14.904734 421			
HTTP/1.1 200 OK [text/html] 248	HTTP	199.168.111.116	199.168.111.1	14.906598 422			
Seq=234 Ack=195 Win=131072 Len=0 [ACK] 1900 ~ 55959 54	TCP	199.168.111.116	199.168.111.1	14.906773 423			
Seq=235 Ack=196 Win=131072 Len=0 [ACK] 1900 ~ 55959 54	TCP	199.168.111.11	199.168.111.116	14.909566 425			
Seq=196 Ack=235 Win=6912 Len=0 [ACK] 55959 ~ 1900 54	TCP	199.168.111.116	199.168.111.1	14.910314 426			
Standard query 0xa002 A cdn.jsdelivr.net 76	DNS	199.168.111.1	199.168.111.116	17.679741 995			

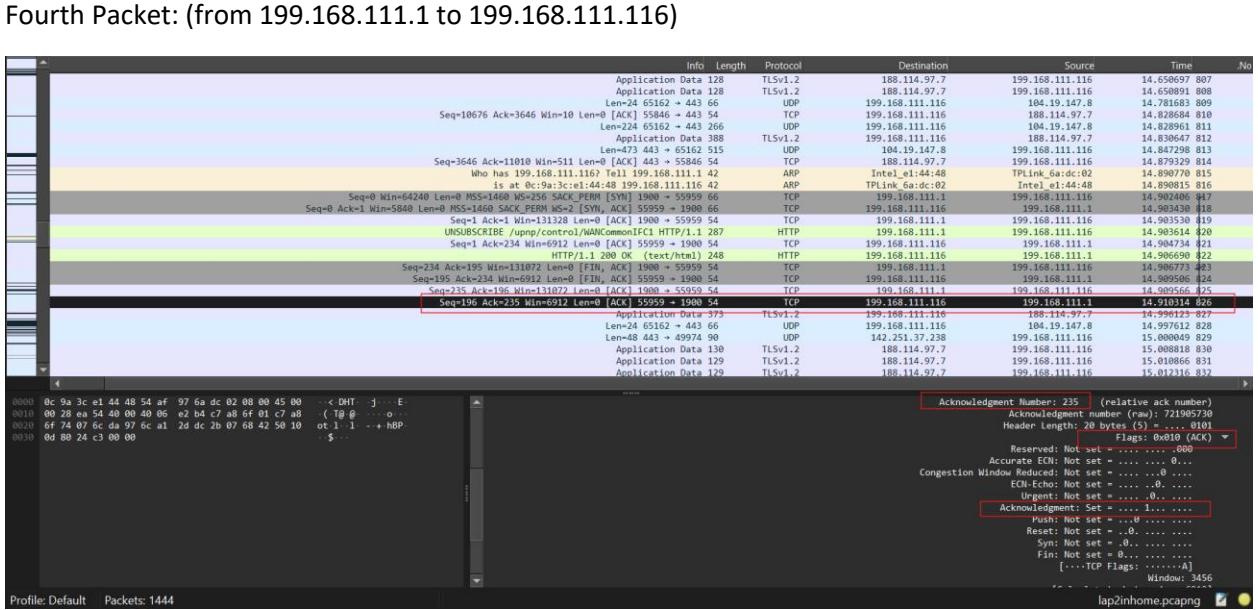
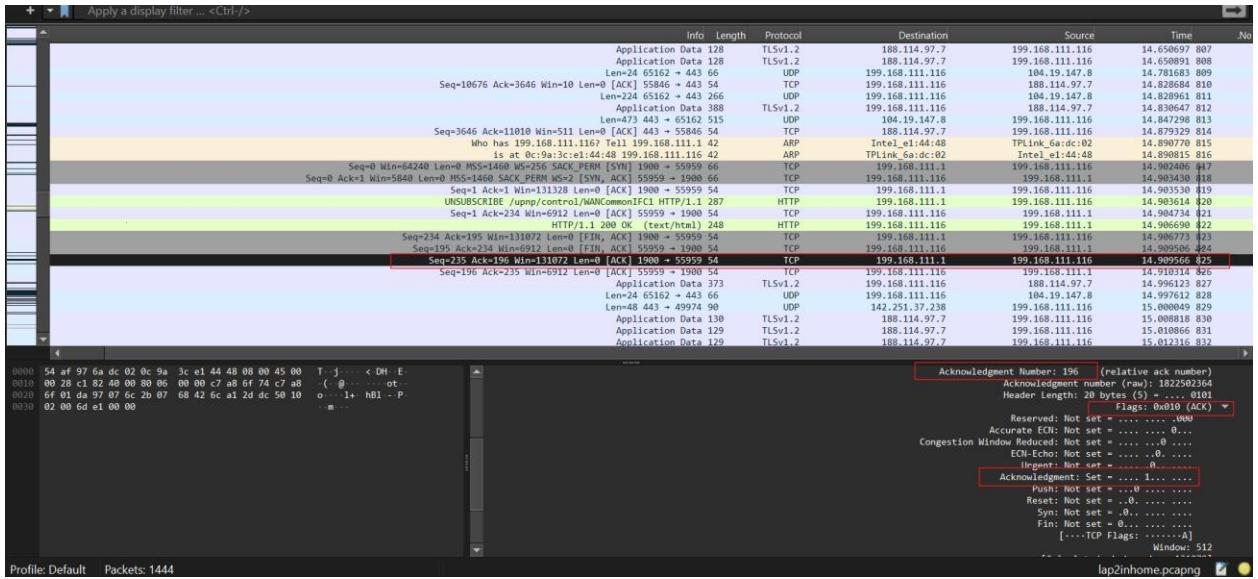
First packet: (from 199.168.111.116 to 199.168.111.1)



Second packet: (from 199.168.111.1 to 199.168.111.116)



Third Packet: (from 199.168.111.116 to 199.168.111.1)



Part 3: Capturing and Analyzing UDP Traffic

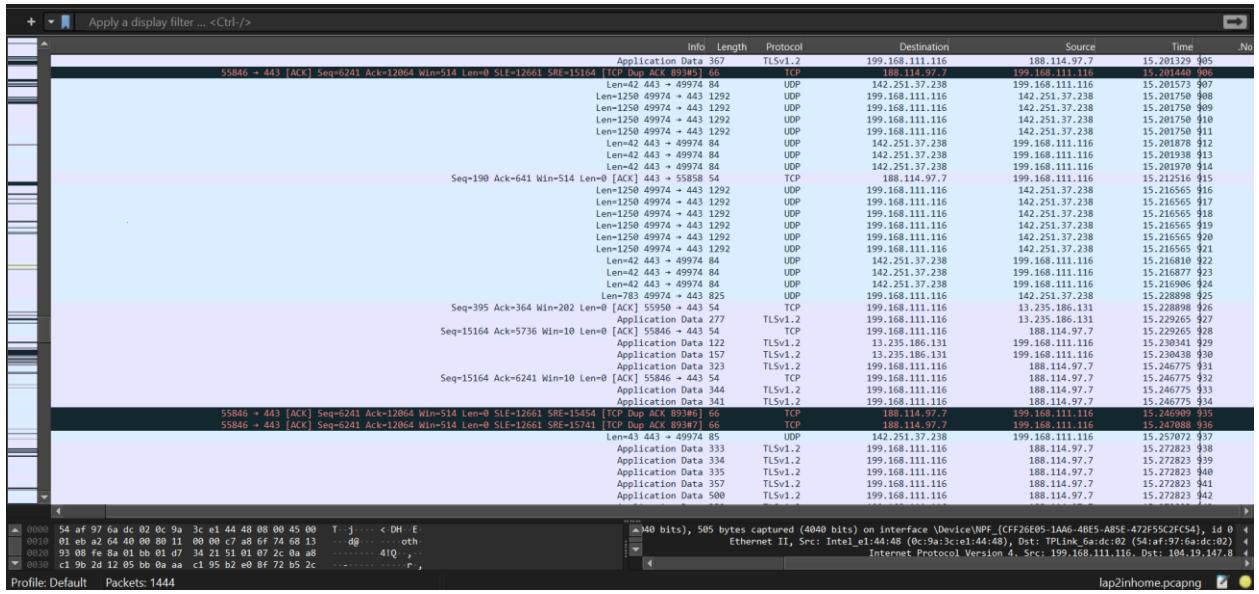
Task 1: Generate UDP traffic and capture packets

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.

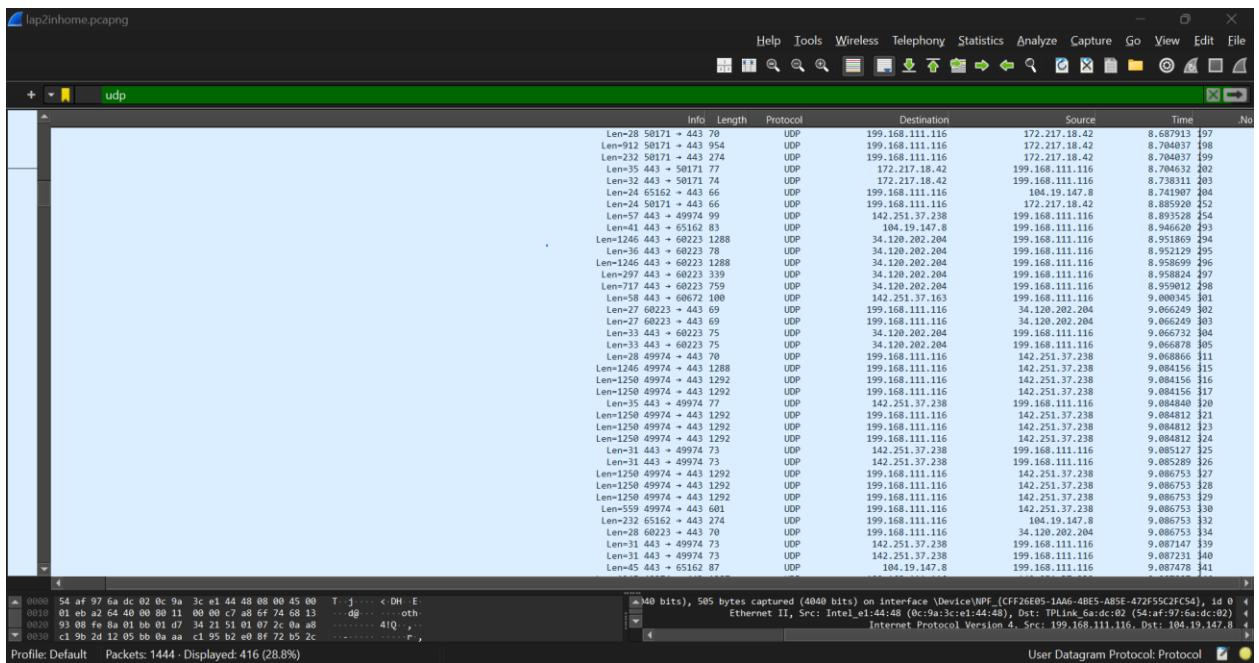
Step 3: Start capturing packets in Wireshark while the UDP application is running.

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.

The screenshot shows the NetworkMiner interface with the following details:

- Left Panel (Timeline):** Shows a list of 14444 captured packets. The first few are highlighted in yellow, indicating they are selected for analysis.
- Middle Panel (Selected Packet Details):**
 - Frame 183:** Contains 505 bytes of data over 4040 bits on the wire and 4040 bits on the interface.
 - Protocol:** User Datagram Protocol (UDP).
 - Source:** 199.168.111.116 (Local Machine).
 - Destination:** 199.168.111.116 (Local Machine).
 - Time:** 8:48:19.17 144.
 - No:** 8.
- Right Panel (Hex Dump):** Displays the raw data of the selected UDP packet, showing ASCII and hex values.

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

Protected Payload (KPO) 66 QUIC 199.168.111.116 104.19.147.8 8.467904 156

Protected Payload (KPO) 91 QUIC 199.168.111.116 104.19.147.8 8.467904 157

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 85 QUIC 199.168.111.116 104.19.147.8 8.467565 159

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 89 QUIC 199.168.111.116 104.19.147.8 8.467713 160

Protected Payload (KPO) 1242 QUIC 199.168.111.116 104.19.147.8 8.503584 165

Protected Payload (KPO) 1242 QUIC 199.168.111.116 104.19.147.8 8.503584 166

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 85 QUIC 199.168.111.116 104.19.147.8 8.503583 167

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 89 QUIC 199.168.111.116 104.19.147.8 8.503583 168

Protected Payload (KPO) 1242 QUIC 199.168.111.116 104.19.147.8 8.503584 169

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 85 QUIC 199.168.111.116 104.19.147.8 8.503584 170

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 89 QUIC 199.168.111.116 104.19.147.8 8.503584 171

Protected Payload (KPO) 1242 QUIC 199.168.111.116 104.19.147.8 8.503584 172

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 85 QUIC 199.168.111.116 104.19.147.8 8.503584 173

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 89 QUIC 199.168.111.116 104.19.147.8 8.503838 181

Protected Payload (KPO), DCID=01179ac47d2550973b155fc4722579c14aa31f59 87 QUIC 199.168.111.116 104.19.147.8 8.604038 182

Len=463 443 + 6162 595 UDP 104.19.147.8 199.168.111.116 8.609578 183

Len=912 50171 + 443 954 UDP 199.168.111.116 172.217.18.42 8.704037 198

Len=232 50171 + 443 274 UDP 199.168.111.116 172.217.18.42 8.704037 199

Len=35 443 + 50171 77 UDP 172.217.18.42 199.168.111.116 8.704632 202

Len=32 443 + 50171 74 UDP 172.217.18.42 199.168.111.116 8.738311 203

Len=24 65162 + 443 66 UDP 199.168.111.116 104.19.147.8 8.741907 204

Len=171 65162 + 443 66 UDP 199.168.111.116 172.217.18.42 8.805682 205

Len=57 443 + 489 99 UDP 142.168.251.37 104.19.147.8 8.859528 206

Len=41 443 + 65162 83 UDP 104.19.147.8 199.168.111.116 8.946620 203

Len=1246 443 + 60223 1288 UDP 34.120.202.204 199.168.111.116 8.951869 294

Len=36 443 + 60223 78 UDP 34.120.202.204 199.168.111.116 8.952199 295

Len=1246 443 + 60223 1288 UDP 34.120.202.204 199.168.111.116 8.958699 296

Source port (16bit):65162

Destination port (16bit):443

Length	Protocol	Destination	Source	Time	No
60	Protected Payload (KPO)	199.168.111.116	104.19.147.8	8.467694 156	
91	Protected Payload (KPO)	199.168.111.116	104.19.147.8	8.467694 157	
92	DNS	199.168.111.116	199.168.111.11	8.467356 158	
85	Protected Payload (KPO)	104.19.147.8	199.168.111.116	8.467565 159	
89	Protected Payload (KPO)	104.19.147.8	199.168.111.116	8.467713 160	
1242	Protected Payload (KPO)	199.168.111.116	104.19.147.8	8.493293 165	
85	Protected Payload (KPO)	104.19.147.8	199.168.111.116	8.509384 166	
193	Len=159 443 + 50171	172.217.18.42	199.168.111.116	8.512674 168	
70	Protected Payload (KPO)	199.168.111.116	104.19.147.8	8.538463 173	
87	Protected Payload (KPO)	104.19.147.8	199.168.111.116	8.538723 174	
83	Protected Payload (KPO)	104.19.147.8	104.19.147.8	8.540211 175	
193	Len=159 443 + 65162	104.19.147.8	199.168.111.116	8.603039 182	
505	UDP	199.168.111.116	104.19.147.8	8.609570 183	
70	UDP	199.168.111.116	172.217.18.42	8.687913 197	
954	UDP	199.168.111.116	172.217.18.42	8.704037 198	
274	UDP	199.168.111.116	172.217.18.42	8.704037 199	
7	UDP	172.217.18.42	199.168.111.116	8.704632 202	
74	UDP	172.217.18.42	199.168.111.116	8.713733 203	
66	UDP	199.168.111.116	104.19.147.8	8.714987 204	
66	UDP	199.168.111.116	172.217.18.42	8.889520 252	
99	UDP	142.251.37.238	199.168.111.116	8.893528 254	
83	UDP	104.19.147.8	199.168.111.116	8.946620 293	
78	UDP	34.120.202.204	199.168.111.116	8.951869 294	
1788	UDP	34.120.202.204	199.168.111.116	8.952129 295	
443	UDP	34.120.202.204	199.168.111.116	8.958699 296	

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.

Comparison of UDP and 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.

Complexity:

UDP: UDP headers are designed to be simple and minimalistic. They include only the essential fields required for packet delivery, making UDP efficient but less versatile.

TCP: TCP headers are complex and feature-rich. They incorporate mechanisms for reliable data transfer, flow control, connection management, and error recovery, reflecting TCP's more sophisticated capabilities.

Functionality:

UDP: As a connectionless protocol, UDP does not establish or tear down connections. It does not guarantee delivery, ordering, or data integrity. This makes UDP suitable for applications where speed is critical, and some data loss is acceptable.

TCP: TCP is connection-oriented and ensures reliable data delivery. It guarantees that data is received in order and supports error recovery and flow control. These features make TCP ideal for applications requiring reliable and accurate data transmission.

Use Cases:

UDP: Best suited for scenarios where high speed is essential and where the application can handle error recovery. Common use cases include live streaming, online gaming, and VoIP (Voice over IP).

TCP: Preferred for applications where reliability and data integrity are crucial. Typical examples include web browsing, file transfers, and email.

TCP Segment Header Format										
Bit #	0	7	8	15	16	23	24			
0	Source Port				Destination Port					
32	Sequence Number									
64	Acknowledgment Number									
96	Data Offset	Res	Flags		Window Size					
128	Header and Data Checksum				Urgent Pointer					
160...	Options									

UDP Datagram Header Format								
Bit #	0	7	8	15	16	23	31	
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

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	<ul style="list-style-type: none"> - Web browsing (HTTP/HTTPS) - File transfers (FTP) - Email (SMTP, IMAP, POP3) - Database access - Secure communications (SSL/TLS) 	<ul style="list-style-type: none"> - Live streaming - VoIP - DNS queries - Broadcasting - Online gaming
Performance	<ul style="list-style-type: none"> - High reliability with error detection, acknowledgment, and retransmissions - Higher latency due to connection setup and error recovery - Suitable for high-throughput scenarios 	<ul style="list-style-type: none"> - 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