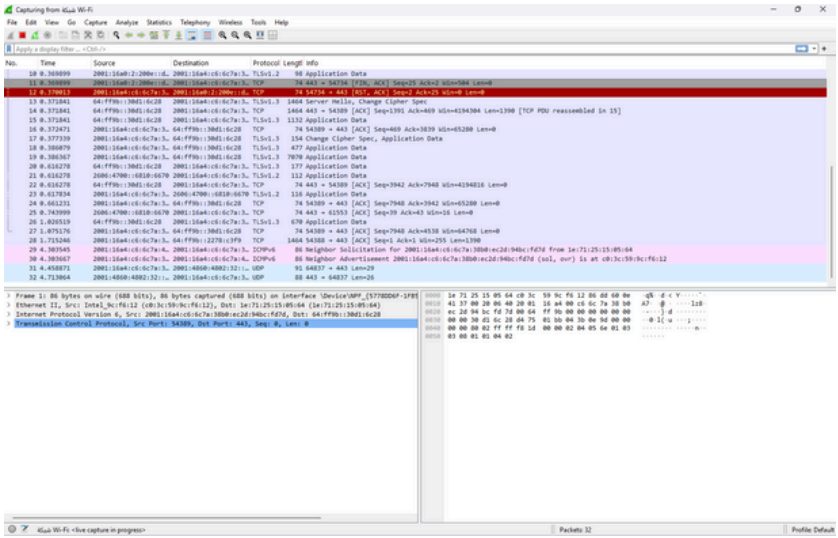
Lab 2: solution

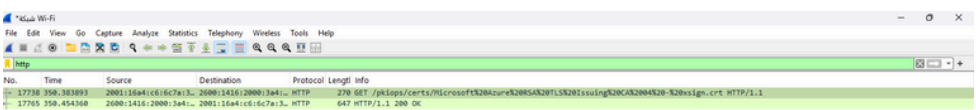
Part 1:

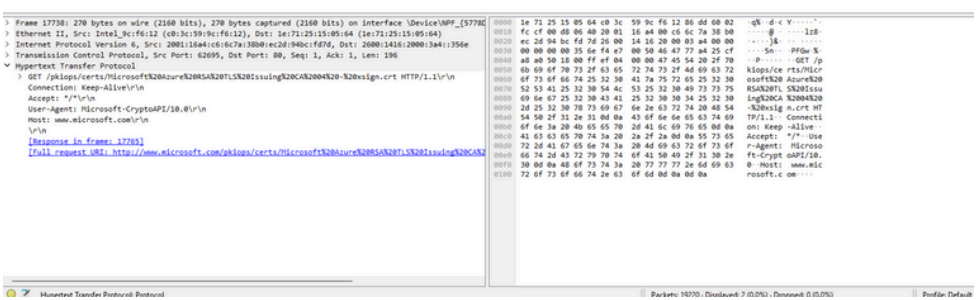
Task 1: Start Wireshark and capture packets:

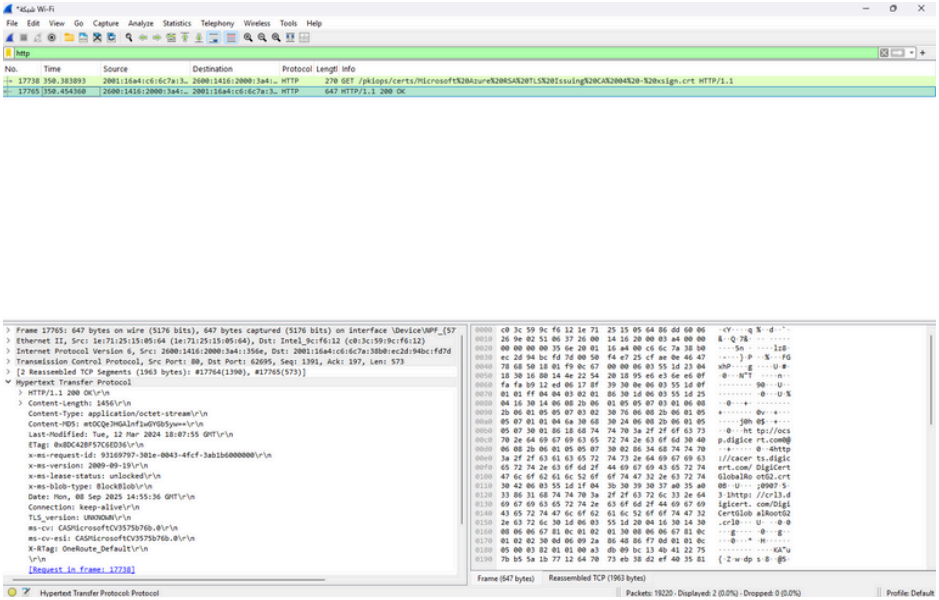


Task 2: Filter HTTP packets:

I type http in the filter bar and result is the request (get) and the response (200 0k)

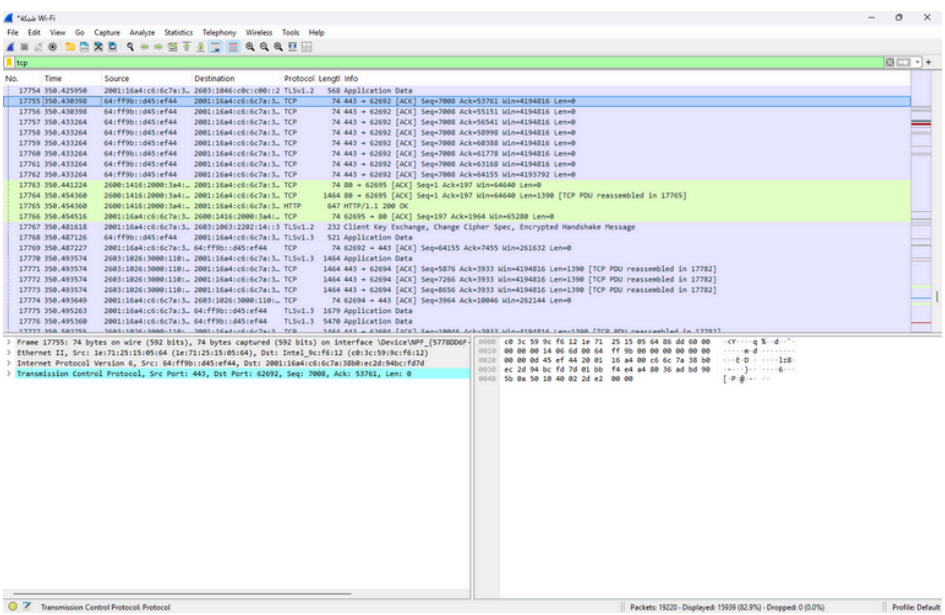




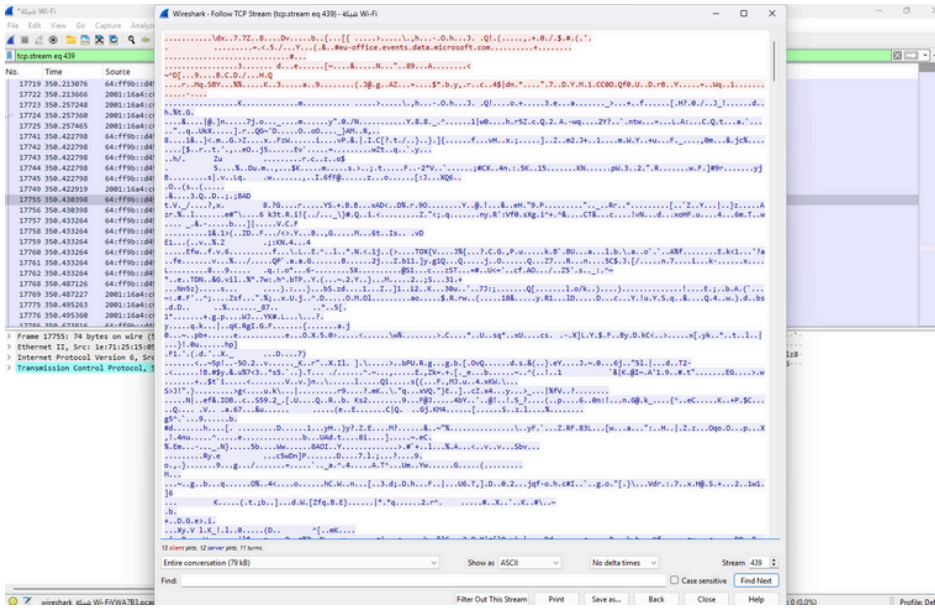


Part 2:

Task 1: Filter TCP packets:

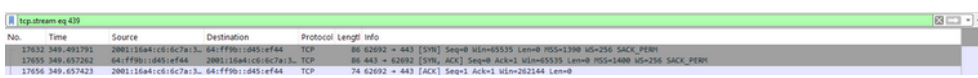


TCP Stream of one of the packets:



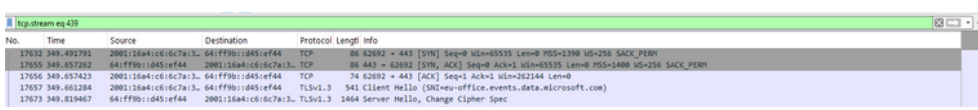
Task 2: Analyze TCP handshake

Select the packets of the 3 - way handshake:



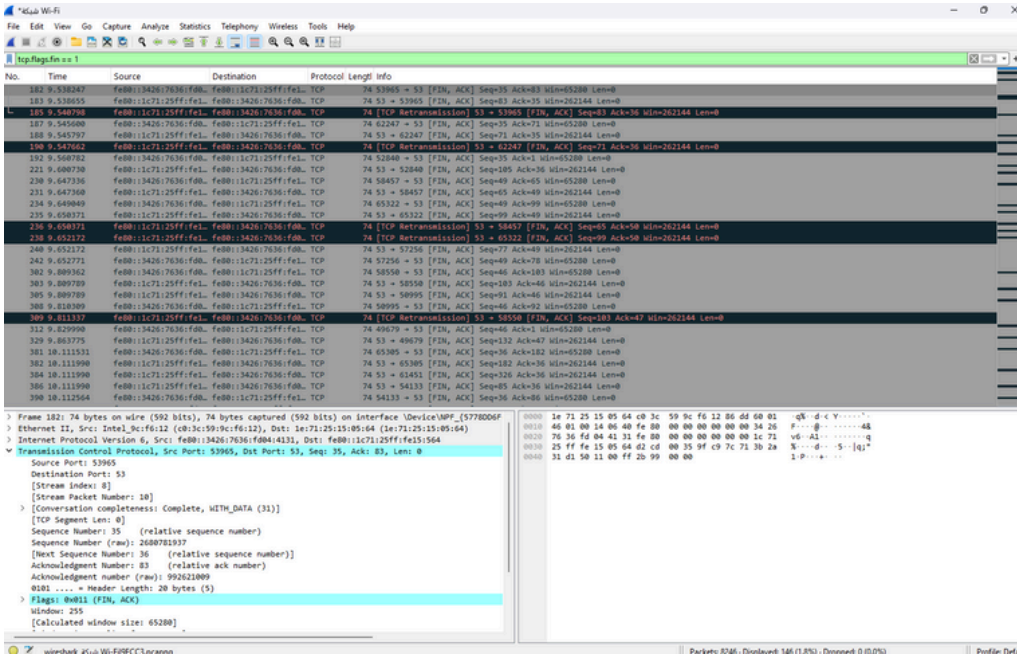
No.	Time	Source	Destination	Protocol	Length	Info
17632	349.491791	2001:16a4:c6:6c7a:3…	64:ff9b::d45:ef44	TCP	86	62692 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1390 WS=256 SACK_PERM
17655	349.657262	64:ff9b::d45:ef44	2001:16a4:c6:6c7a:3…	TCP	86	443 → 62692 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 WS=256 SACK_PERM
17656	349.657423	2001:16a4:c6:6c7a:3…	64:ff9b::d45:ef44	TCP	74	62692 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0

Data packets exchanged between client and server



No.	Time	Source	Destination	Protocol	Length	Info
17632	349.491791	2001:16a4:c6:6c7a:3…	64:ff9b::d45:ef44	TCP	86	62692 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1390 WS=256 SACK_PERM
17655	349.657262	64:ff9b::d45:ef44	2001:16a4:c6:6c7a:3…	TCP	86	443 → 62692 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 WS=256 SACK_PERM
17656	349.657423	2001:16a4:c6:6c7a:3…	64:ff9b::d45:ef44	TCP	74	62692 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
17657	349.661284	2001:16a4:c6:6c7a:3…	64:ff9b::d45:ef44	TLSv1.3	541	Client Hello (SNI=eu-office.events.data.microsoft.com)
17673	349.819467	64:ff9b::d45:ef44	2001:16a4:c6:6c7a:3…	TLSv1.3	1464	Server Hello, Change Cipher Spec

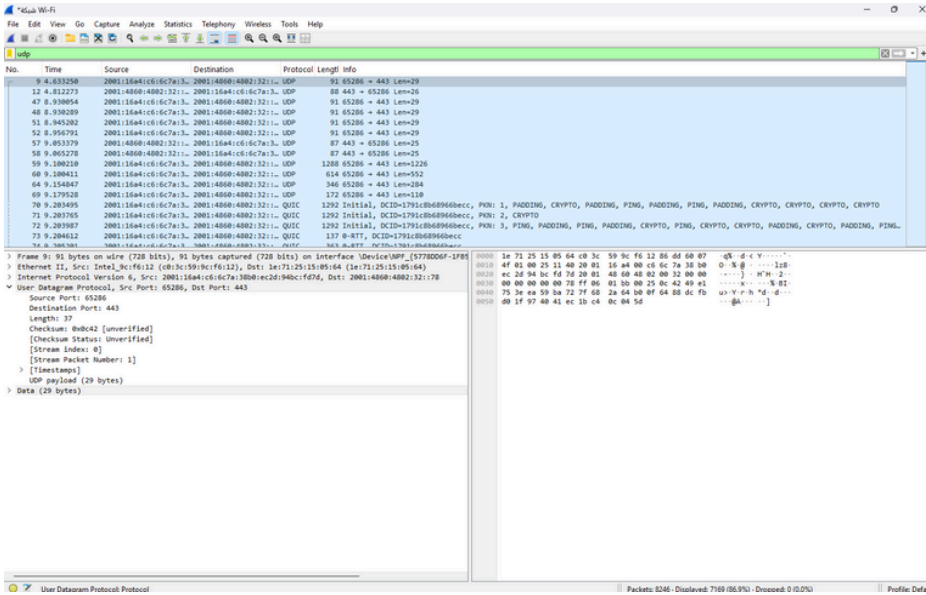
TCP termination process (FIN/ACK packets):



```
Transmission Control Protocol, Src Port: 53965, Dst Port: 53, Seq: 35, Ack: 83, Len: 0
    Source Port: 53965
    Destination Port: 53
    [Stream index: 8]
    [Stream Packet Number: 10]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 35    (relative sequence number)
    Sequence Number (raw): 2680781937
    [Next Sequence Number: 36    (relative sequence number)]
    Acknowledgment Number: 83    (relative ack number)
    Acknowledgment number (raw): 992621009
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window: 255
    [Calculated window size: 65280]
```

Part 3:

Task 1&2: Capture and analyzing UDP:



```
> Frame 9: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{5778DD6F-1FB5
> Ethernet II, Src: Intel_9c:f6:12 (c0:3c:59:9c:f6:12), Dst: 1e:71:25:15:05:64 (1e:71:25:15:05:64)
> Internet Protocol Version 6, Src: 2001:16a4:c6:6c7a:38b0:ec2d:94bc:fd7d, Dst: 2001:4860:4802:32::78
v User Datagram Protocol, Src Port: 65286, Dst Port: 443
    Source Port: 65286
    Destination Port: 443
    Length: 37
    Checksum: 0x0c42 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
    [Stream Packet Number: 1]
  > [Timestamps]
    UDP payload (29 bytes)
> Data (29 bytes)
```

Compare the simplicity of UDP headers with TCP headers:

UDP Header Simplicity

- Header size: 8 bytes (fixed).
- Fields (only 4): Source Port, Destination Port, Length, Checksum

TCP Header Complexity

- Header size: 20 bytes (minimum, can go up to 60 bytes with options).
- Fields (many more than UDP): Source Port, Destination Port, Sequence Number Acknowledgment Number, Header Length, Flags, Window Size, Checksum, Urgent Pointer, Options

UDP headers are simple (8 bytes, 4 fields) → lightweight, fast.
TCP headers are complex (20–60 bytes, many fields) → reliable, feature-rich, but with more overhead.

Part 4:

Comparing TCP and UDP:

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

	TCP or UDP	Reasons
Reliability and Connection Establishment	TCP	TCP is connection-oriented ensures reliable data transfer with acknowledgments and retransmissions.
Data Integrity and Ordering	TCP	TCP guarantees data arrives in order and without duplication using sequence numbers and error checking.

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

	TCP	UDP
Use cases	Web browsing (HTTP/HTTPS), email (SMTP, IMAP, POP3), file transfer (FTP), remote login	Streaming (video/audio), online gaming, VoIP, DNS queries, live broadcasts.
Performance	Slower due to overhead (connection setup, error checking, ordering, acknowledgments).	Faster, lightweight (no handshake, no guaranteed delivery or ordering).