WIRESHARK TOOL

NETWORK ANALYSIS USING WIRESHARK TOOL

INDEX

1. Different protocols that appear in the protocol column in the unfiltered packet-listing window in Wireshark GUI.

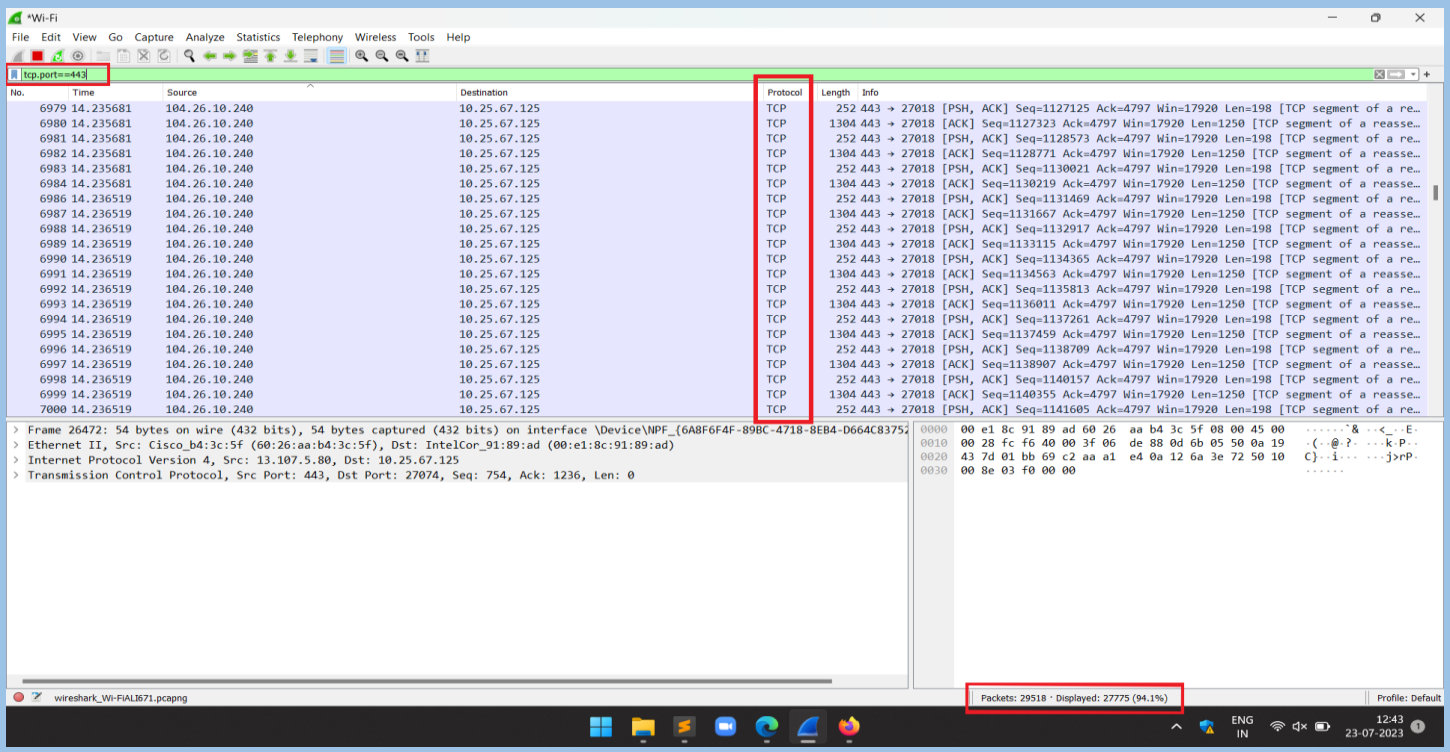
1. Protocols in Networking

- Network protocols are a set of rules outlining how connected devices communicate across a network to exchange information easily and safely.
- Protocols can be Categorized into three categories:
  ➔ Network Communication Protocols: HTTP, TCP, UDP, FTP etc.
  ➔ Network Security Protocols: SFTP, HTTPS, SSL etc.
  ➔ Network Management Protocols: SNMP, ICMP etc.

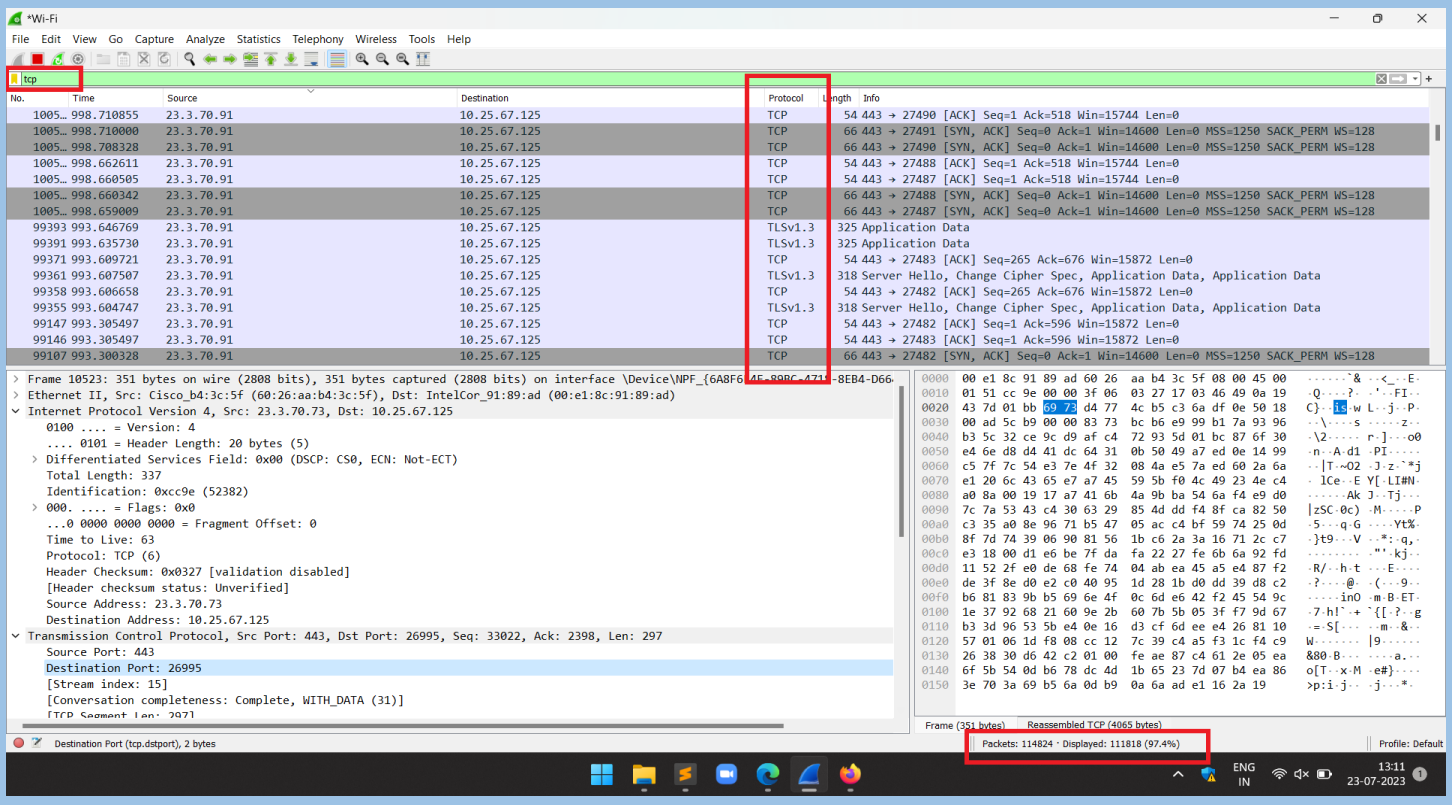
2. Protocols Observed during Network Analysis using Wireshark Tool

A: TCP [Transmission Control Protocol]

- TCP is a connection-oriented protocol that provides reliable, ordered, and error-checked delivery of data. It uses port numbers to identify the source and destination applications or services.
- Common TCP ports include:
  Port 80: HTTP (Hypertext Transfer Protocol)
  Port 443: HTTPS (HTTP Secure)
- Filters used to filter TCP (HTTPS) packets
  1. tcp.port==443 [To get the HTTPS traffic]

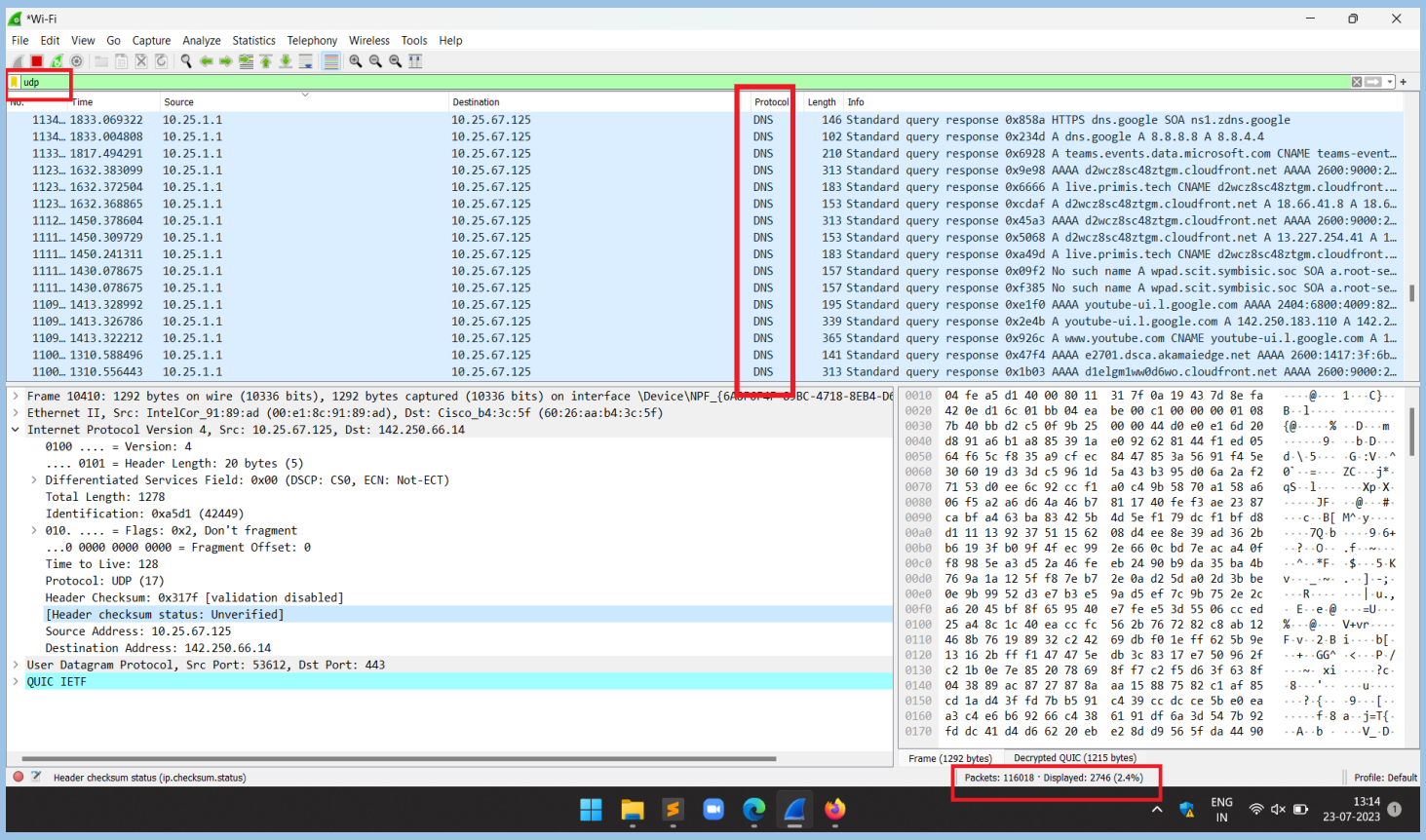


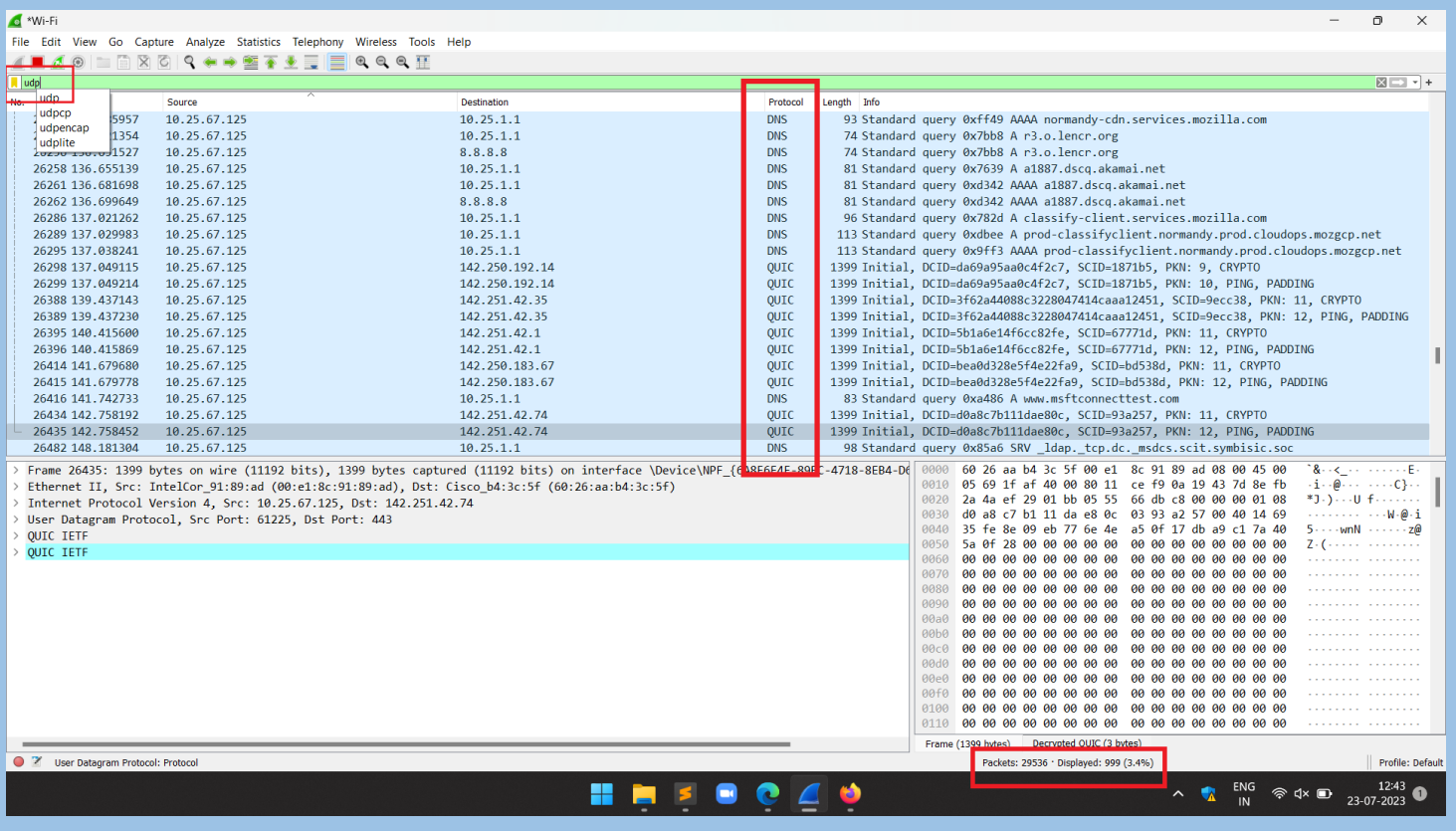
2. tcp [To get the packets using TCP protocol]



B: UDP [User Datagram Protocol]

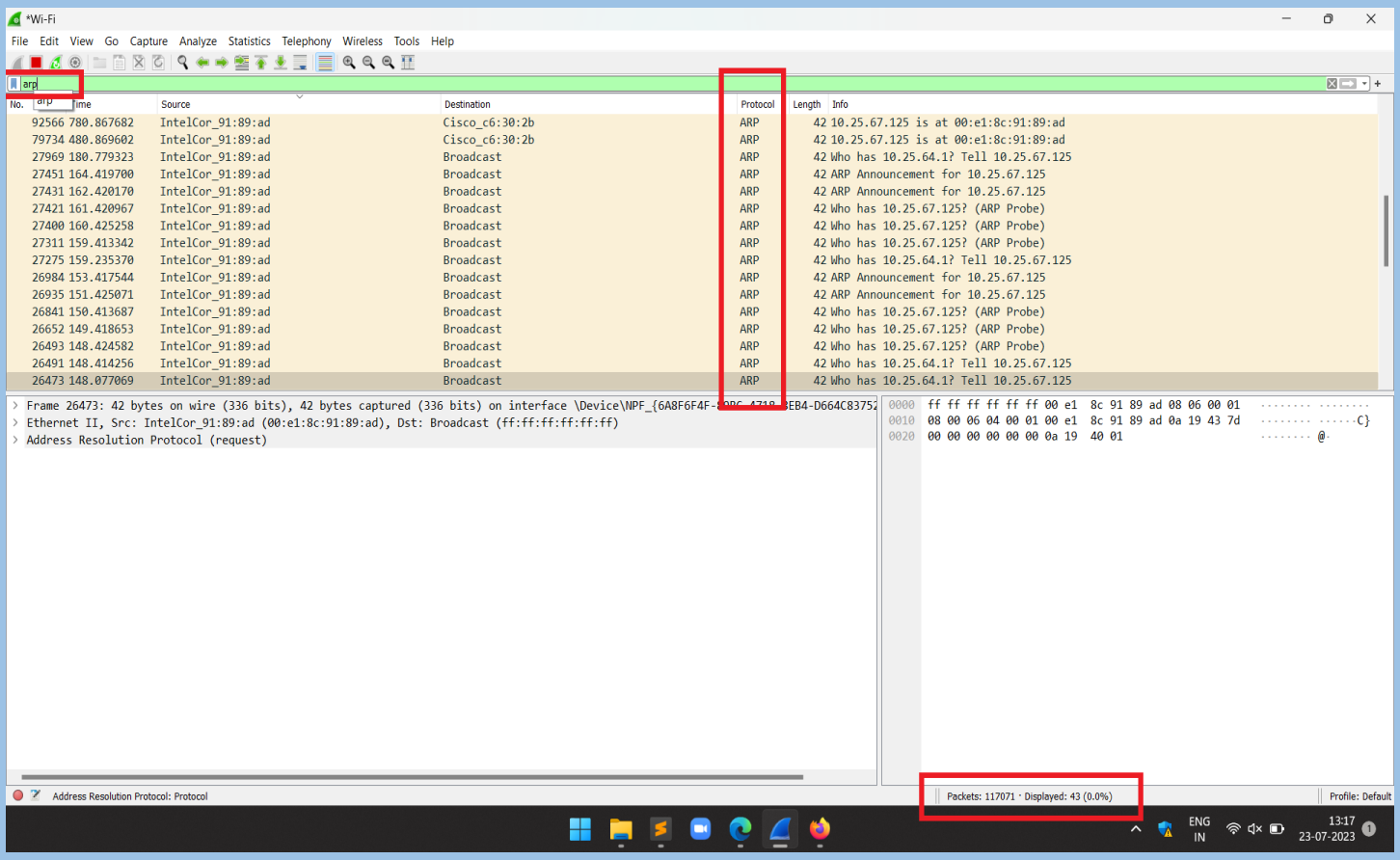
- UDP is a connectionless protocol that does not provide guaranteed delivery or error checking. It is often used for applications that require low latency and do not require reliability. UDP uses port numbers to identify the source and destination applications or services.
- Common UDP ports include:
  Port 53: DNS (Domain Name System)
  Port 67/68: DHCP (Dynamic Host Configuration Protocol)
- Filter used to filter UDP packets: udp


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp
No. Time Source Destination Protocol Length Info
1134… 1833.069322 10.25.1.1 10.25.67.125 DNS 146 Standard query response 0x858a HTTPS dns.google SOA ns1.zdns.google
1134… 1833.004808 10.25.1.1 10.25.67.125 DNS 102 Standard query response 0x234d A dns.google A 8.8.8.8 A 8.8.4.4
1133… 1817.494291 10.25.1.1 10.25.67.125 DNS 210 Standard query response 0x6928 A teams.events.data.microsoft.com CNAME teams-event…
1123… 1632.383099 10.25.1.1 10.25.67.125 DNS 313 Standard query response 0x9e98 AAAA d2wcz8sc48ztgm.cloudfront.net AAAA 2600:9000:2…
1123… 1632.372504 10.25.1.1 10.25.67.125 DNS 183 Standard query response 0x6666 A live.primis.tech CNAME d2wcz8sc48ztgm.cloudfront.…
1123… 1632.368865 10.25.1.1 10.25.67.125 DNS 153 Standard query response 0xcdaf A d2wcz8sc48ztgm.cloudfront.net A 18.66.41.8 A 18.6…
1112… 1450.378604 10.25.1.1 10.25.67.125 DNS 313 Standard query response 0x45a3 AAAA d2wcz8sc48ztgm.cloudfront.net AAAA 2600:9000:2…
1111… 1450.309729 10.25.1.1 10.25.67.125 DNS 153 Standard query response 0x5068 A d2wcz8sc48ztgm.cloudfront.net A 13.227.254.41 A 1…
1111… 1450.241311 10.25.1.1 10.25.67.125 DNS 183 Standard query response 0xa49d A live.primis.tech CNAME d2wcz8sc48ztgm.cloudfront.…
1111… 1430.078675 10.25.1.1 10.25.67.125 DNS 157 Standard query response 0x09f2 No such name A wpad.scit.symbisic.soc SOA a.root-se…
1111… 1430.078675 10.25.1.1 10.25.67.125 DNS 157 Standard query response 0xf385 No such name A wpad.scit.symbisic.soc SOA a.root-se…
1109… 1413.328992 10.25.1.1 10.25.67.125 DNS 195 Standard query response 0xe1f0 AAAA youtube-ui.l.google.com AAAA 2404:6800:4009:82…
1109… 1413.326786 10.25.1.1 10.25.67.125 DNS 339 Standard query response 0x2e4b A youtube-ui.l.google.com A 142.250.183.110 A 142.2…
1109… 1413.322212 10.25.1.1 10.25.67.125 DNS 365 Standard query response 0x926c A www.youtube.com CNAME youtube-ui.l.google.com A 1…
1100… 1310.588496 10.25.1.1 10.25.67.125 DNS 141 Standard query response 0x47f4 AAAA e2701.dsca.akamaiedge.net AAAA 2600:1417:3f:6b…
1100… 1310.556443 10.25.1.1 10.25.67.125 DNS 313 Standard query response 0x1b03 AAAA d1elgm1ww0d6wo.cloudfront.net AAAA 2600:9000:2…
Frame 10410: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface \Device\NPF_{6A8F8F4F-89BC-4718-8EB4-D6
Ethernet II, Src: IntelCor_91:89:ad (00:e1:8c:91:89:ad), Dst: Cisco_b4:3c:5f (60:26:aa:b4:3c:5f)
Internet Protocol Version 4, Src: 10.25.67.125, Dst: 142.250.66.14
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1278
Identification: 0xa5d1 (42449)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 128
Protocol: UDP (17)
Header Checksum: 0x317f [validation disabled]
[Header checksum status: Unverified]
Source Address: 10.25.67.125
Destination Address: 142.250.66.14
User Datagram Protocol, Src Port: 53612, Dst Port: 443
QUIC IETF
0010 04 fe a5 d1 40 00 80 11 31 7f 0a 19 43 7d 8e fa ····@··· 1···C}··
0020 42 0e d1 6c 01 bb 04 ea be 00 c1 00 00 00 01 08 B··l···· ········
0030 7b 40 bb d2 c5 0f 9b 25 00 00 44 d0 e0 e1 6d 20 {@·····% ··D···m
0040 d8 91 a6 b1 a8 85 39 1a e0 92 62 81 44 f1 ed 05 ·····9· ··b·D···
0050 64 f6 5c f8 35 a9 cf ec 84 47 85 3a 56 91 f4 5e d·\·5··· ·G·:V··^
0060 30 60 19 d3 3d c5 96 1d 5a 43 b3 95 d0 6a 2a f2 0`··=··· ZC···j*·
0070 71 53 d0 ee 6c 92 cc f1 a0 c4 9b 58 70 a1 58 a6 qS··l··· ···Xp·X·
0080 06 f5 a2 a6 d6 4a 46 b7 81 17 40 fe f3 ae 23 87 ·····JF· ··@···#·
0090 ca bf a4 63 ba 83 42 5b 4d 5e f1 79 dc f1 bf d8 ···c··B[M^·y····
00a0 d1 11 13 92 37 51 15 62 08 d4 ee 8e 39 ad 36 2b ····7Q·b ····9·6+
00b0 b6 19 3f b0 9f 4f ec 99 2e 66 0c bd 7e ac a4 0f ··?··O·· .f··~···
00c0 f8 98 5e a3 d5 2a 46 fe eb 24 90 b9 da 35 ba 4b ··^··*F· ·$···5·K
00d0 76 9a 1a 12 5f f8 7e b7 2e 0a d2 5d a0 2d 3b be v···_·~· ···]·-;·
00e0 0e 9b 99 52 d3 e7 b3 e5 9a d5 ef 7c 9b 75 2e 2c ···R···· ···|·u.,
00f0 a6 20 45 bf 8f 65 95 40 e7 fe e5 3d 55 06 cc ed · E··e·@ ···=U···
0100 25 a4 8c 1c 40 ea cc fc 56 2b 76 72 82 c8 ab 12 %···@··· V+vr····
0110 46 8b 76 19 89 32 c2 42 69 db f0 1e ff 62 5b 9e F·v··2·B i····b[·
0120 13 16 2b ff f1 47 47 5e db 3c 83 17 e7 50 96 2f ··+··GG^ ·<···P·/
0130 c2 1b 0e 7e 85 20 78 69 8f f7 c2 f5 d6 3f 63 8f ···~· xi ·····?c·
0140 04 38 89 ac 87 27 87 8a aa 15 88 75 82 c1 af 85 ·8···'·· ···u····
0150 cd 1a d4 3f fd 7b b5 91 c4 39 cc dc ce 5b e0 ea ···?·{·· ·9···[··
0160 a3 c4 e6 b6 92 66 c4 38 61 91 df 6a 3d 54 7b 92 ·····f·8 a··j=T{·
0170 fd dc 41 d4 d6 62 20 eb e2 8d d9 56 5f da 44 90 ··A··b · ···V_·D·
Frame (1292 bytes) Decrypted QUIC (1215 bytes)
Header checksum status (ip.checksum.status)
Packets: 116018 · Displayed: 2746 (2.4%)
Profile: Default
ENG IN 13:14 23-07-2023


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp
udp
udpcp
udpencap
udplite
No. Source Destination Protocol Length Info
10.25.67.125 10.25.1.1 DNS 93 Standard query 0xff49 AAAA normandy-cdn.services.mozilla.com
10.25.67.125 10.25.1.1 DNS 74 Standard query 0x7bb8 A r3.o.lencr.org
10.25.67.125 8.8.8.8 DNS 74 Standard query 0x7bb8 A r3.o.lencr.org
26258 136.655139 10.25.67.125 10.25.1.1 DNS 81 Standard query 0x7639 A a1887.dscq.akamai.net
26261 136.681698 10.25.67.125 10.25.1.1 DNS 81 Standard query 0xd342 AAAA a1887.dscq.akamai.net
26262 136.699649 10.25.67.125 8.8.8.8 DNS 81 Standard query 0xd342 AAAA a1887.dscq.akamai.net
26286 137.021262 10.25.67.125 10.25.1.1 DNS 96 Standard query 0x782d A classify-client.services.mozilla.com
26289 137.029983 10.25.67.125 10.25.1.1 DNS 113 Standard query 0xdbee A prod-classifyclient.normandy.prod.cloudops.mozgcp.net
26295 137.038241 10.25.67.125 10.25.1.1 DNS 113 Standard query 0x9ff3 AAAA prod-classifyclient.normandy.prod.cloudops.mozgcp.net
26298 137.049115 10.25.67.125 142.250.192.14 QUIC 1399 Initial, DCID=da69a95aa0c4f2c7, SCID=1871b5, PKN: 9, CRYPTO
26299 137.049214 10.25.67.125 142.250.192.14 QUIC 1399 Initial, DCID=da69a95aa0c4f2c7, SCID=1871b5, PKN: 10, PING, PADDING
26388 139.437143 10.25.67.125 142.251.42.35 QUIC 1399 Initial, DCID=3f62a44088c3228047414caaa12451, SCID=9ecc38, PKN: 11, CRYPTO
26389 139.437230 10.25.67.125 142.251.42.35 QUIC 1399 Initial, DCID=3f62a44088c3228047414caaa12451, SCID=9ecc38, PKN: 12, PING, PADDING
26395 140.415600 10.25.67.125 142.251.42.1 QUIC 1399 Initial, DCID=5b1a6e14f6cc82fe, SCID=67771d, PKN: 11, CRYPTO
26396 140.415869 10.25.67.125 142.251.42.1 QUIC 1399 Initial, DCID=5b1a6e14f6cc82fe, SCID=67771d, PKN: 12, PING, PADDING
26414 141.679680 10.25.67.125 142.250.183.67 QUIC 1399 Initial, DCID=bea0d328e5f4e22fa9, SCID=bd538d, PKN: 11, CRYPTO
26415 141.679778 10.25.67.125 142.250.183.67 QUIC 1399 Initial, DCID=bea0d328e5f4e22fa9, SCID=bd538d, PKN: 12, PING, PADDING
26416 141.742733 10.25.67.125 10.25.1.1 DNS 83 Standard query 0xa486 A www.msftconnecttest.com
26434 142.758192 10.25.67.125 142.251.42.74 QUIC 1399 Initial, DCID=d0a8c7b111dae80c, SCID=93a257, PKN: 11, CRYPTO
26435 142.758452 10.25.67.125 142.251.42.74 QUIC 1399 Initial, DCID=d0a8c7b111dae80c, SCID=93a257, PKN: 12, PING, PADDING
26482 148.181304 10.25.67.125 10.25.1.1 DNS 98 Standard query 0x85a6 SRV _ldap._tcp.dc._msdcs.scit.symbisic.soc
Frame 26435: 1399 bytes on wire (11192 bits), 1399 bytes captured (11192 bits) on interface \Device\NPF_{6A8F8F4F-89BC-4718-8EB4-D6
Ethernet II, Src: IntelCor_91:89:ad (00:e1:8c:91:89:ad), Dst: Cisco_b4:3c:5f (60:26:aa:b4:3c:5f)
Internet Protocol Version 4, Src: 10.25.67.125, Dst: 142.251.42.74
User Datagram Protocol, Src Port: 61225, Dst Port: 443
QUIC IETF
QUIC IETF
0000 60 26 aa b4 3c 5f 00 e1 8c 91 89 ad 08 00 45 00 `&··<_·· ····E·
0010 05 69 1f af 40 00 80 11 ce f9 0a 19 43 7d 8e fb ·i··@··· ····C}··
0020 2a 4a ef 29 01 bb 05 55 66 db c8 00 00 00 01 08 *J·)···U f·····
0030 d0 a8 c7 b1 11 da e8 0c 03 93 a2 57 00 40 14 69 ··········· ···W·@·i
0040 35 fe 8e 09 eb 77 6e 4e a5 0f 17 db a9 c1 7a 40 5····wnN ·····z@
0050 5a 0f 28 00 00 00 00 00 00 00 00 00 00 00 00 00 Z·(····· ·····
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ····· ·····
Frame (1399 bytes) Decrypted QUIC (3 bytes)
User Datagram Protocol: Protocol
Packets: 29536 · Displayed: 999 (3.4%)
Profile: Default
ENG IN 12:43 23-07-2023

C: ARP [Address Resolution Protocol]

- Address Resolution Protocol (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address, also known as a media access control (MAC) address, in a local-area network (LAN).
- Filter Used to filter ARP packets: arp



---

2. Applying Different Filters in Wireshark to filter network Packets

Packet Filters:

&& -> Both Filters together

|| -> Any of the provided filter

→ <u>IP-Based Filtering:</u>

- ➔ ip.src==<Source-IP> : Filtering packets on the basis of source IP address.
- ➔ ip.dst==<Destination-IP> : Filtering packets on the basis of Destination IP.
- ➔ ip.addr==<IP-address> : Filtering Packets on the basis of IP address (Source or destination IP)
- ➔ !(ip.addr eq <IP>) : To exclude a IP address Ex: (tcp.port==443)&&!(ip.addr eq <IP>)

→ Port-Based Filtering:

1. udp.port==<port-number>

   - ➔ udp.srcport==<>
   - ➔ udp.dstport==<>

2. tcp.port==<Port-number>

   - ➔ tcp.srcport==<>
   - ➔ tcp.dstport==<>

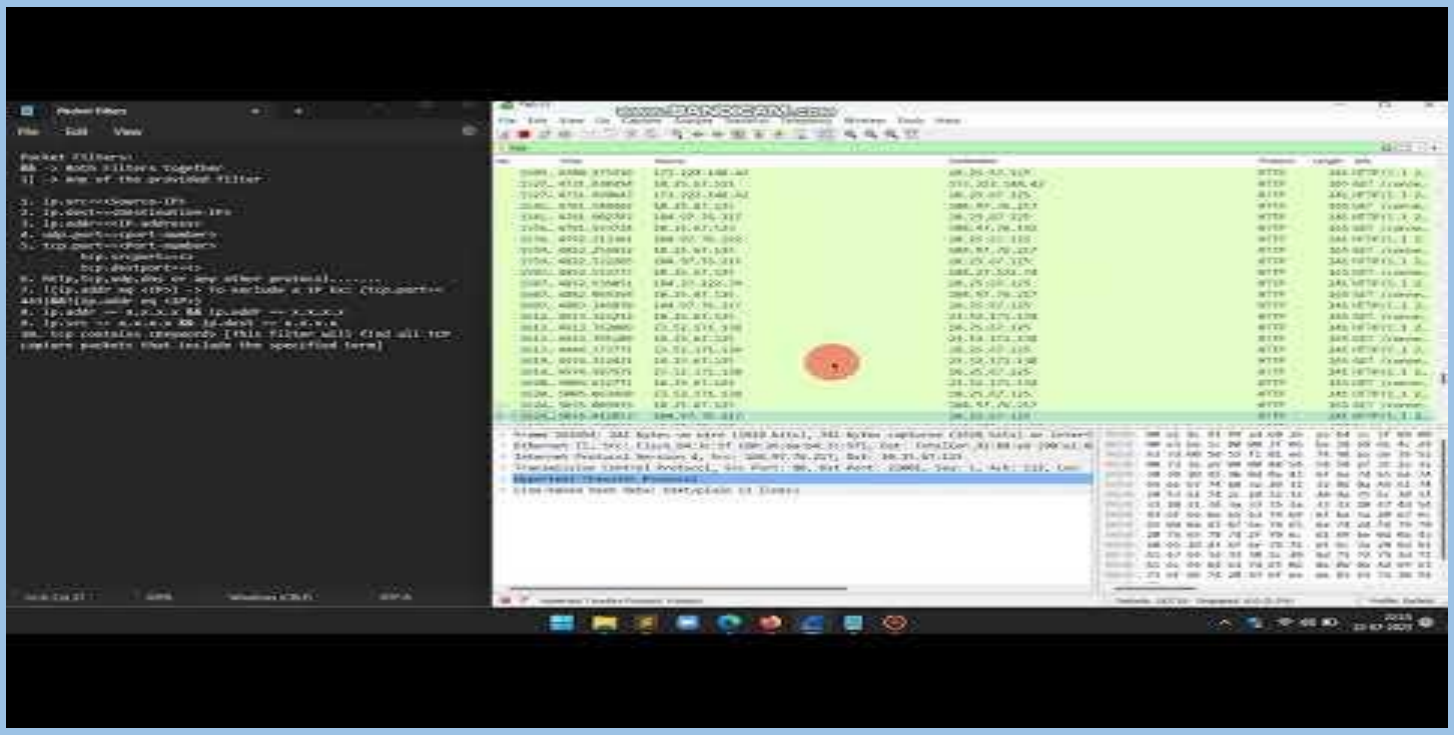
→ Protocol-Based Filtering:

http, tcp, udp, dns, arp or any other protocol.

→ Multiple Filters together:

1. (tcp.port==443)&&!(ip.addr eq <IP>)	: Display HTTPS traffic but exclude the traffic from a particular IP address.
2. ip.addr==<IP>||udp.port==<Port-Number>	: Display Traffic for particular IP address or for Port number which uses UDP protocol (like DHCP, DNS etc..)

Packet-Filtering.mp4 [See video PoC in the attachments provided if Link is Not Working]



https://youtu.be/0BaGbSHNLIE

---

3. Identifying the Internet (IP) address of the URLs visited during the capture and listing the IP address with the site URL.

URL: http://testphp.vulnweb.com/

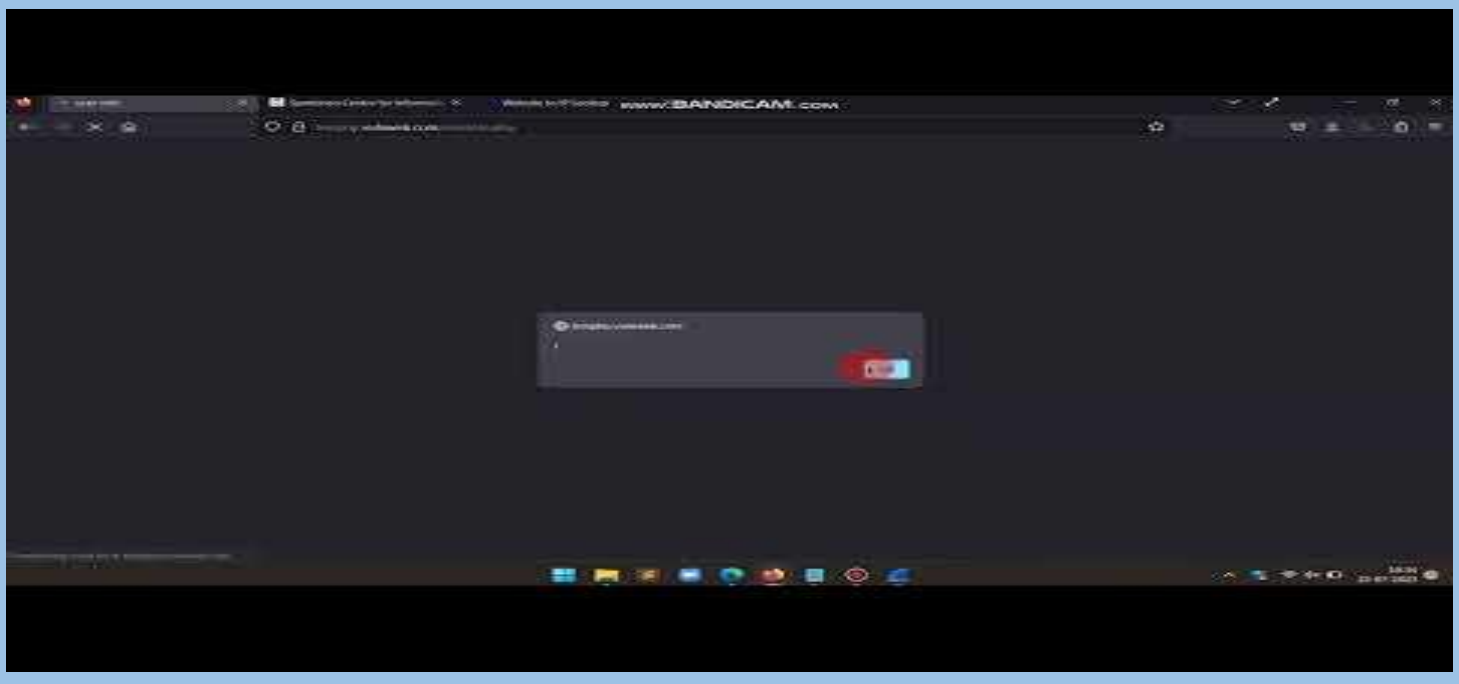
Filter Used: http contains "http://testphp.vulnweb.com/"

IP Address: Source-IP (10.25.67.125) and Destination-IP (44.228.249.3)

Additional: Capturing HTTP Traffic: [Credentials are transmitted in plain text]

- ➔ Open Wireshark and start capturing packets by clicking on the blue icon.
- ➔ In the filter tab, add filter "http". [It will sort out and show only HTTP traffic in the Wireshark interface]
- ➔ Go to the website which uses HTTP Instead of HTTPS. [In this case, we are using "http://testphp.vulnweb.com/login.php"]
- ➔ Provide login credentials and observe the traffic in Wireshark.
- ➔ Look for Endpoint "/userinfo.php" and the HTTP Method will be POST [Because a form is submitted]
- ➔ In that Request, navigate to HTTP Header and look for "cookies". In those cookies, login credentials are transmitted in clear text.
- ➔ Also navigate to HTML Header, under that you can see the login credentials are transmitted in plain text.

HTTP-Traffic.mp4 [See video PoC in the attachments provided if Link is Not Working]



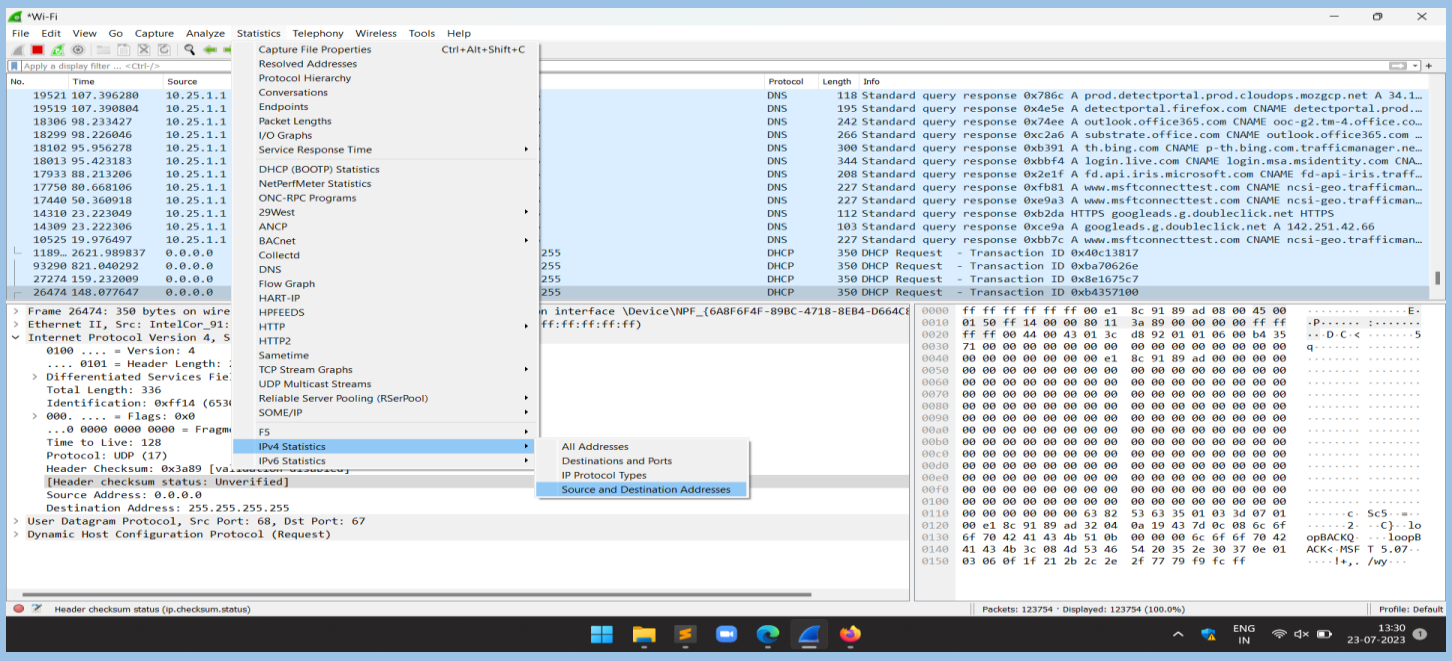
https://youtu.be/NcnhLK3pnu8

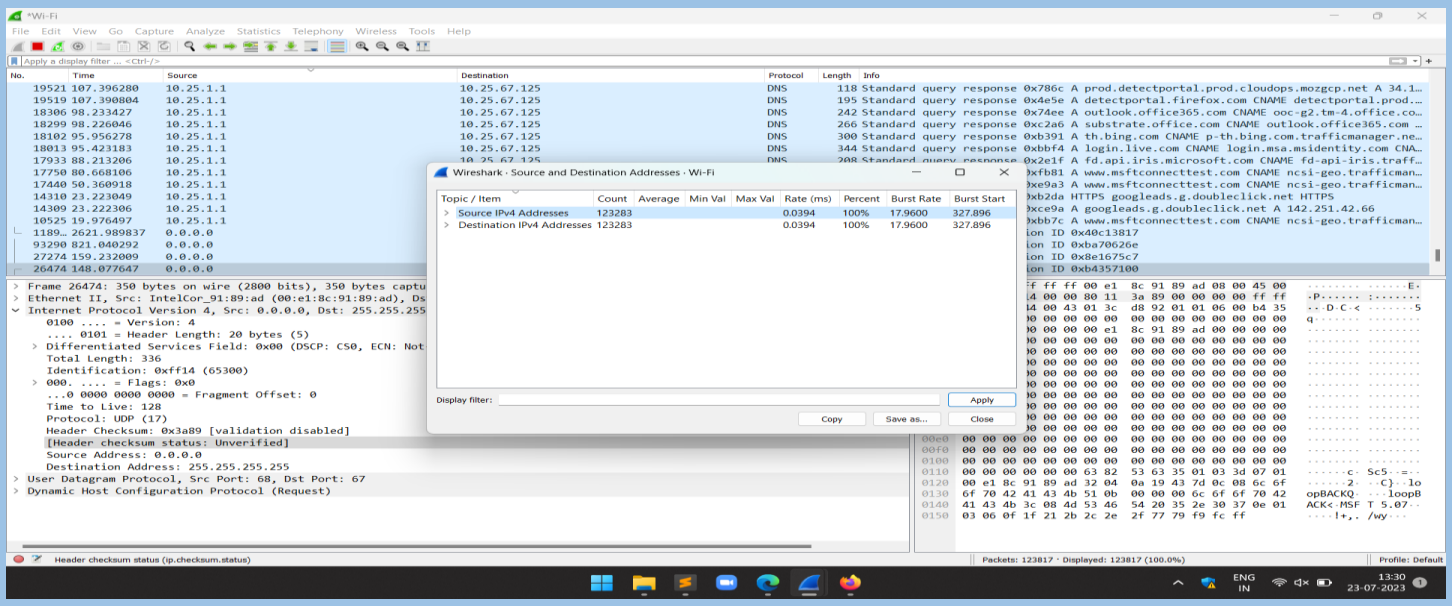
4. Calculate total number of captured packets for each protocol. [See Point-1 Screenshots]

- TCP: 111818 packets captured out of total packets captured which were 114284. [TCP packets were 97.4% of total packets captured]
- UDP: 2746 Packets captured out of total packets captured which were 116018. [UDP packets were 2.4% of total packets captured]
- ARP: 43 Packets captured out of total packets captured which were 117071.
- HTTP: 179 Packets captured out of total packets captured which were 117837. [HTTP packets were 0.2% of total packets captured]
- DHCP: 6 Packets captured out of total packets captured which were 118863.

---

5. Find out the IP addresses of the client and server using statistics tool of Wireshark.



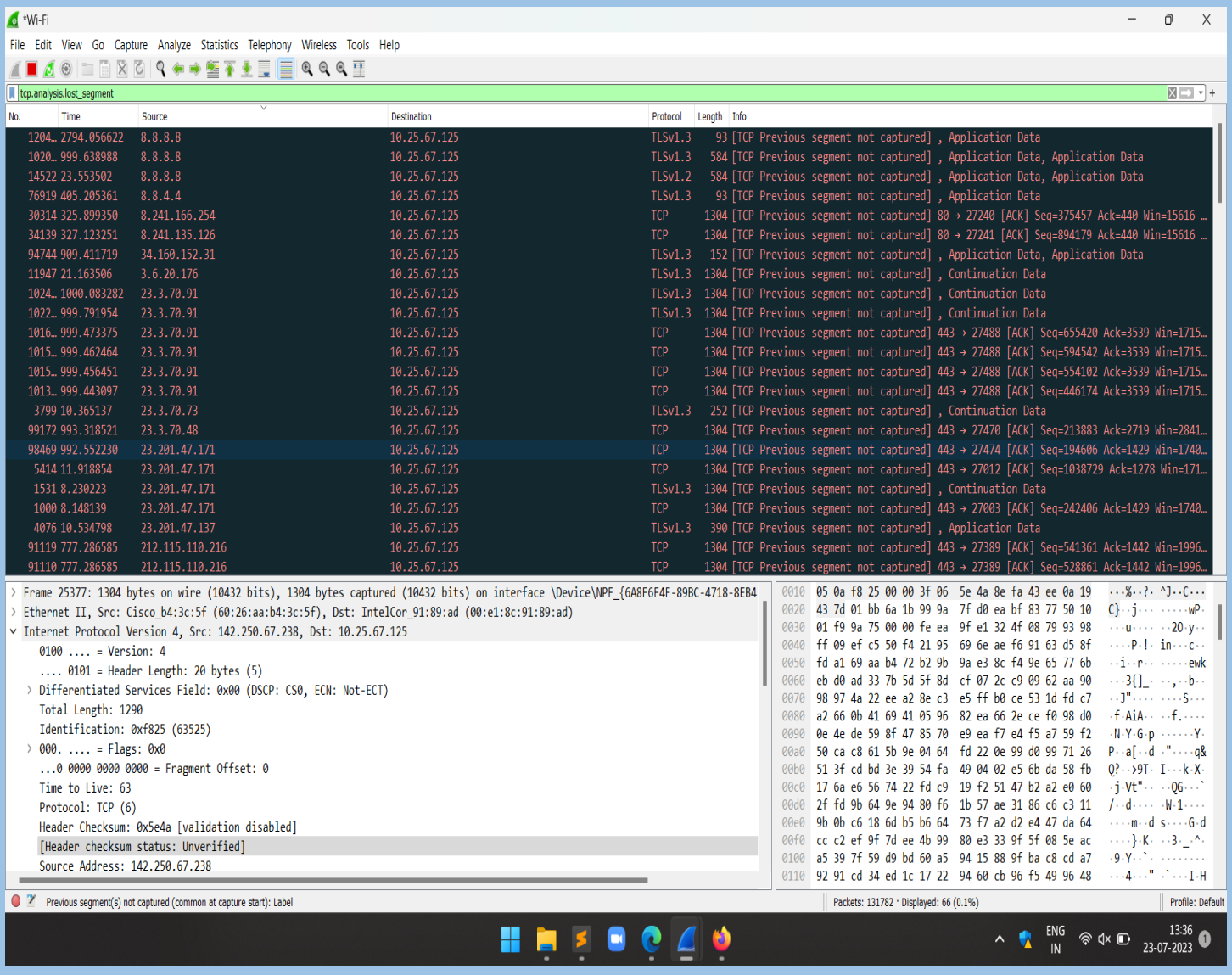


6. Evaluating the total Number of lost packets using Wireshark.

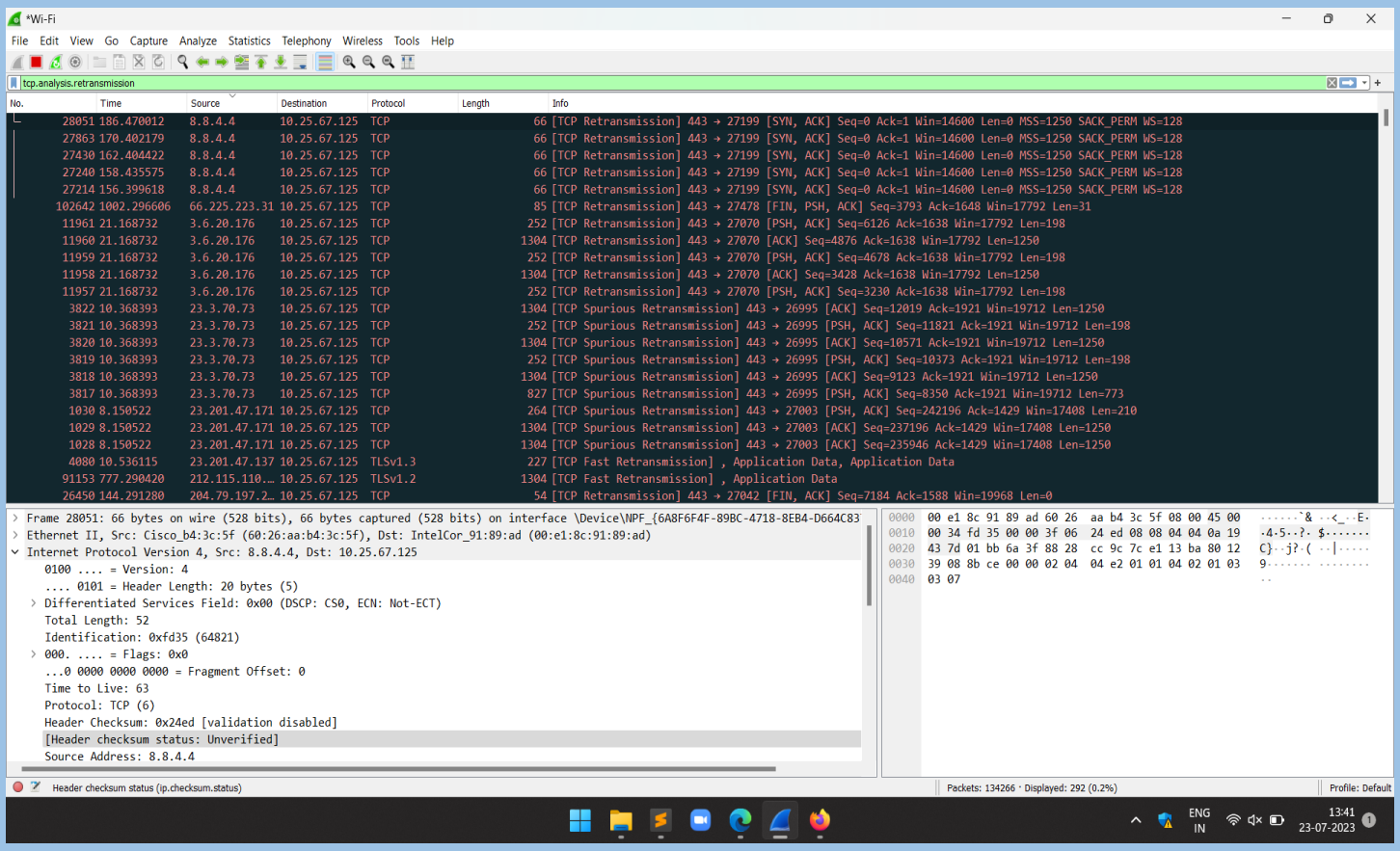
Number of Packets Lost-segment **(While capture is live):** 66

Filter-Used:

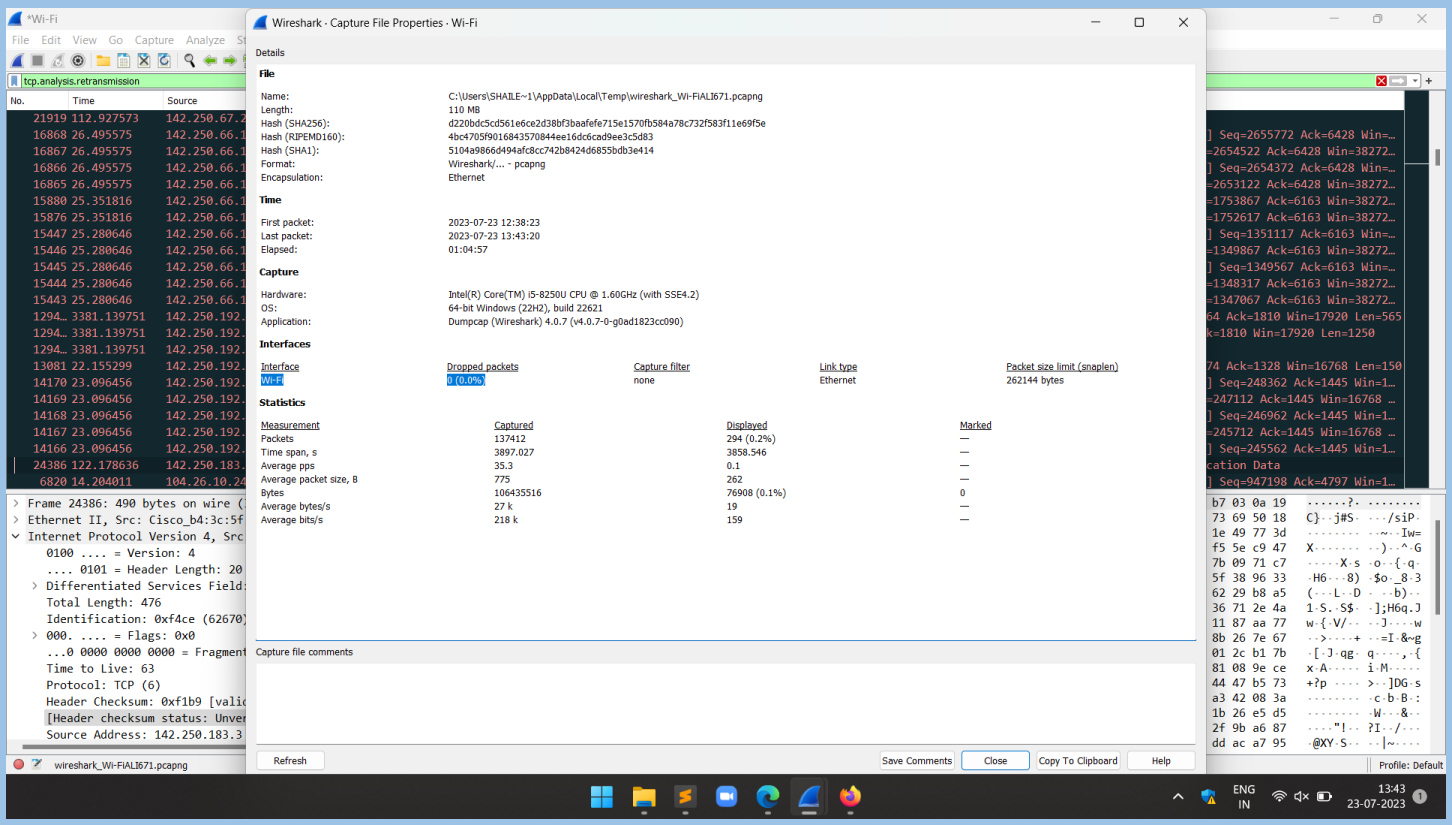
1. tcp.analysis.lost_segment: Indicates we've seen a gap in sequence numbers in the capture. Packet loss can lead to duplicate ACKs, which leads to retransmissions.



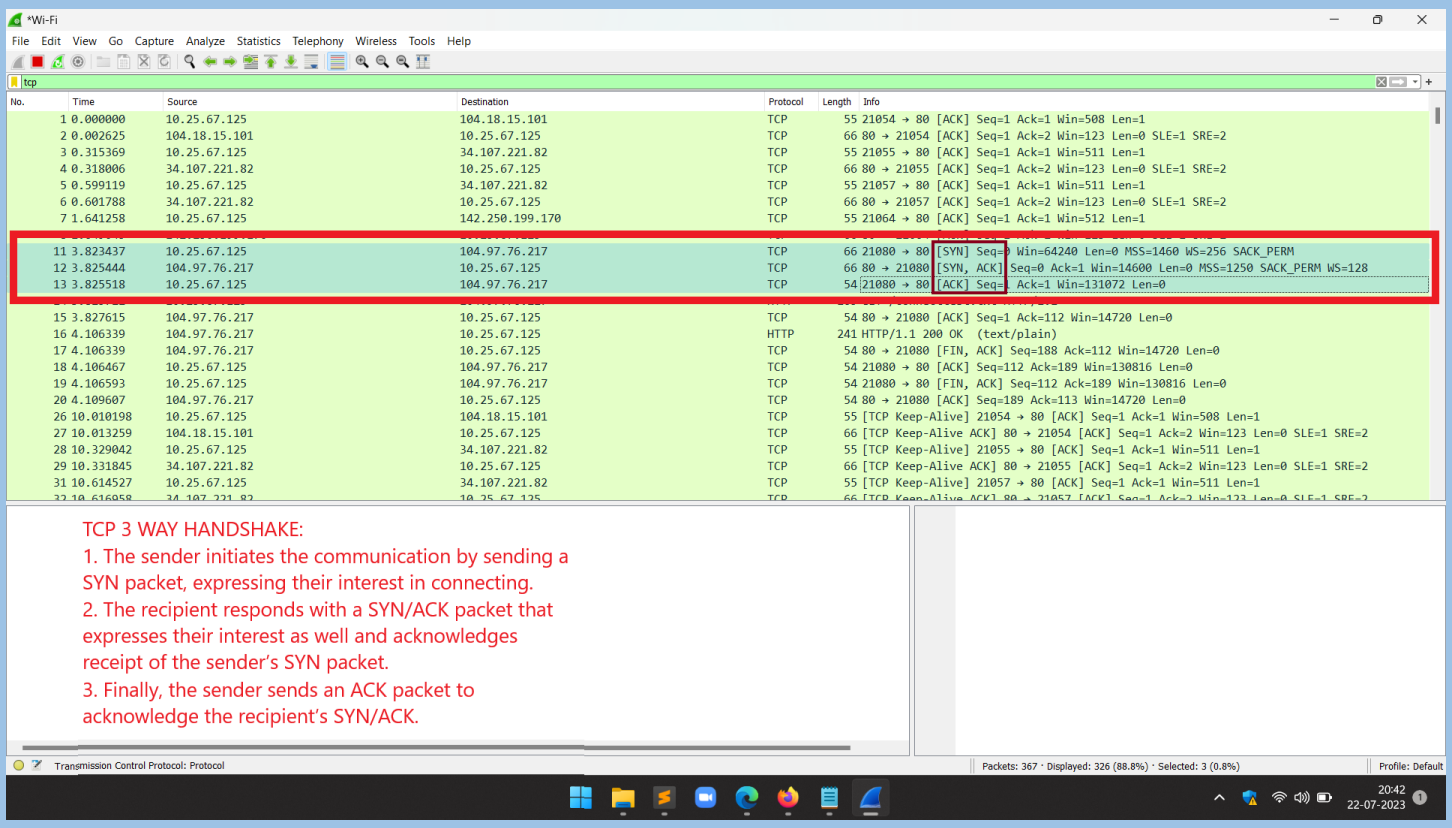
2. tcp.analysis.retransmission: Displays all retransmissions in the capture.



*Finally, with **<u>Wireshark capture Stop (after you have captured all you need/want)</u>**, then go to Statistics -> Capture File Properties. There you can see the Dropped packets (Under Interface Heading) which is 0.0% in our case.*



7. Capturing one TCP 3-way handshake and explaining the process.



- TCP (Transmission Control Protocol) uses a three-way handshake to establish a reliable connection between two devices.
- The 3-way handshake process:
  Client Server State of Communication
  SYN_SENT --> SYN_RECEIVED (State: LISTEN)

  <-- SYN_RECEIVED (State: SYN_RECEIVED)
  [SYN_ACK_SENT]

  ESTABLISHED --> ESTABLISHED (State: ESTABLISHED)
  {Sends ACK packet}

1. <u>SYN_SENT:</u>

- The client sends a SYN packet to the server, indicating that it wants to establish a connection.
- The SYN packet includes a random sequence number that is used to identify packets in the connection.
- The client enters the SYN_SENT state, waiting for a response from the server.

2. <u>SYN_RECEIVED:</u>

- The server receives the SYN packet from the client and responds with a SYN-ACK packet.
- The SYN-ACK packet includes a random sequence number and an acknowledgment number that is equal to the client's sequence number plus one.
- The server enters the SYN_RECEIVED state, waiting for the final ACK packet from the client.

3. ESTABLISHED:

- The client receives the SYN-ACK packet from the server and sends an ACK packet back to the server.
- The ACK packet includes the acknowledgment number that was sent in the server's SYN-ACK packet.

Once the server receives the ACK packet, the connection is established, and both devices enter the ESTABLISHED state. They are now ready to exchange data over the connection.

---

8. TCP packet Analysis and exploring the features in the packet header window. [TCP header and IP header details for the selected packet.]

1. TCP Header

-> Source and Destination Ports

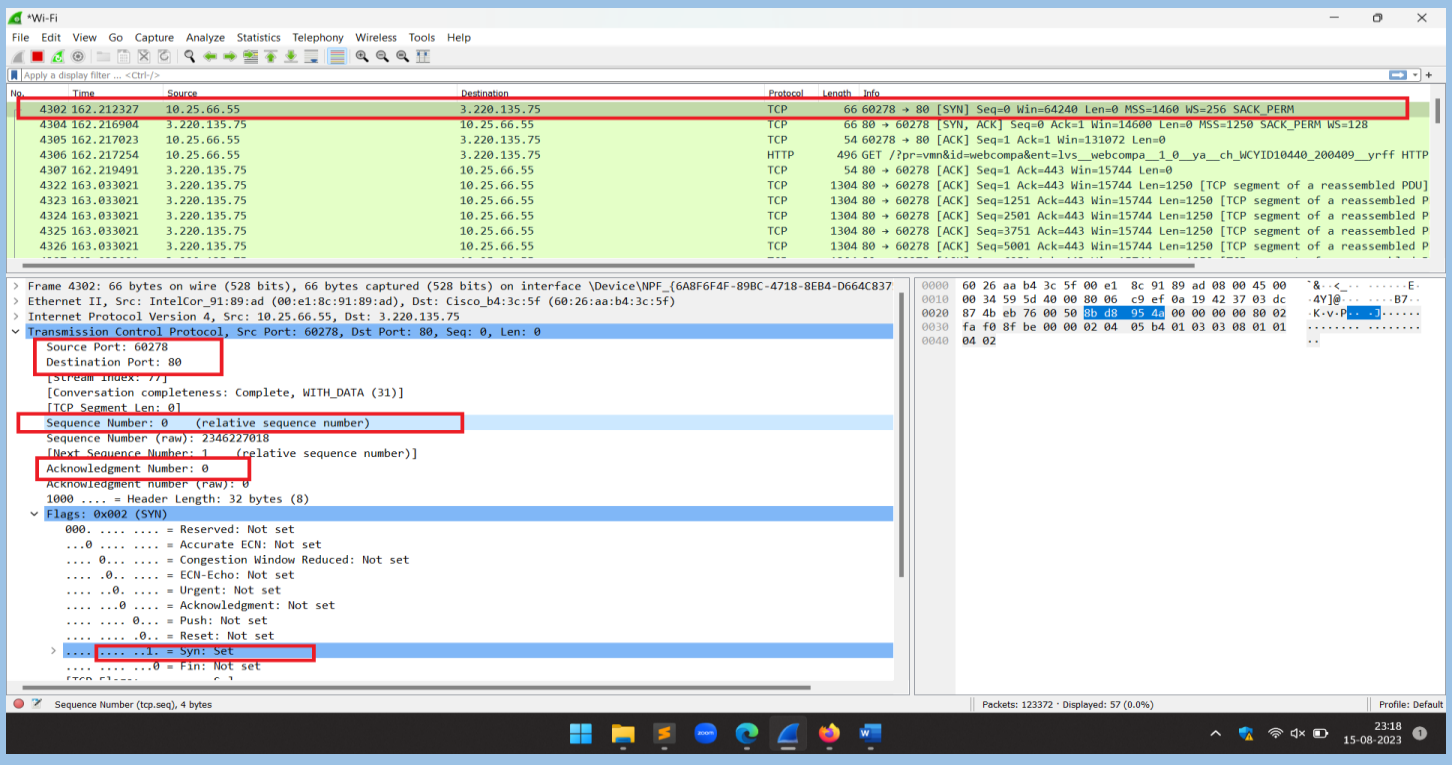
Source port: this is a 16-bit field that specifies the port number of the sender.

Destination port: this is a 16-bit field that specifies the port number of the receiver.

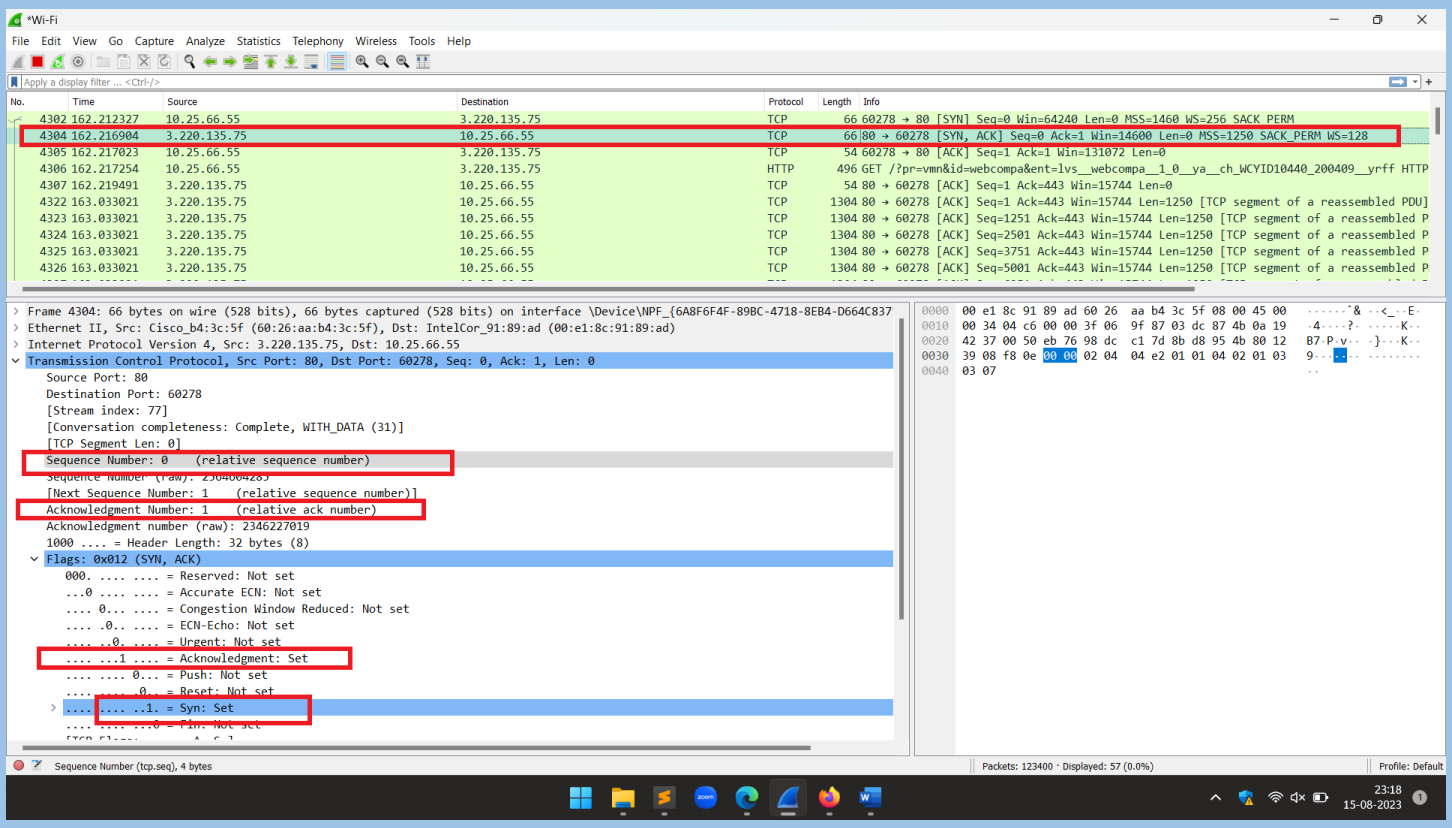
-> Understanding TCP 3 Way handshake by analyzing TCP header of the TCP packets.

TCP utilizes a number of flags, in its header to control the state of a connection.

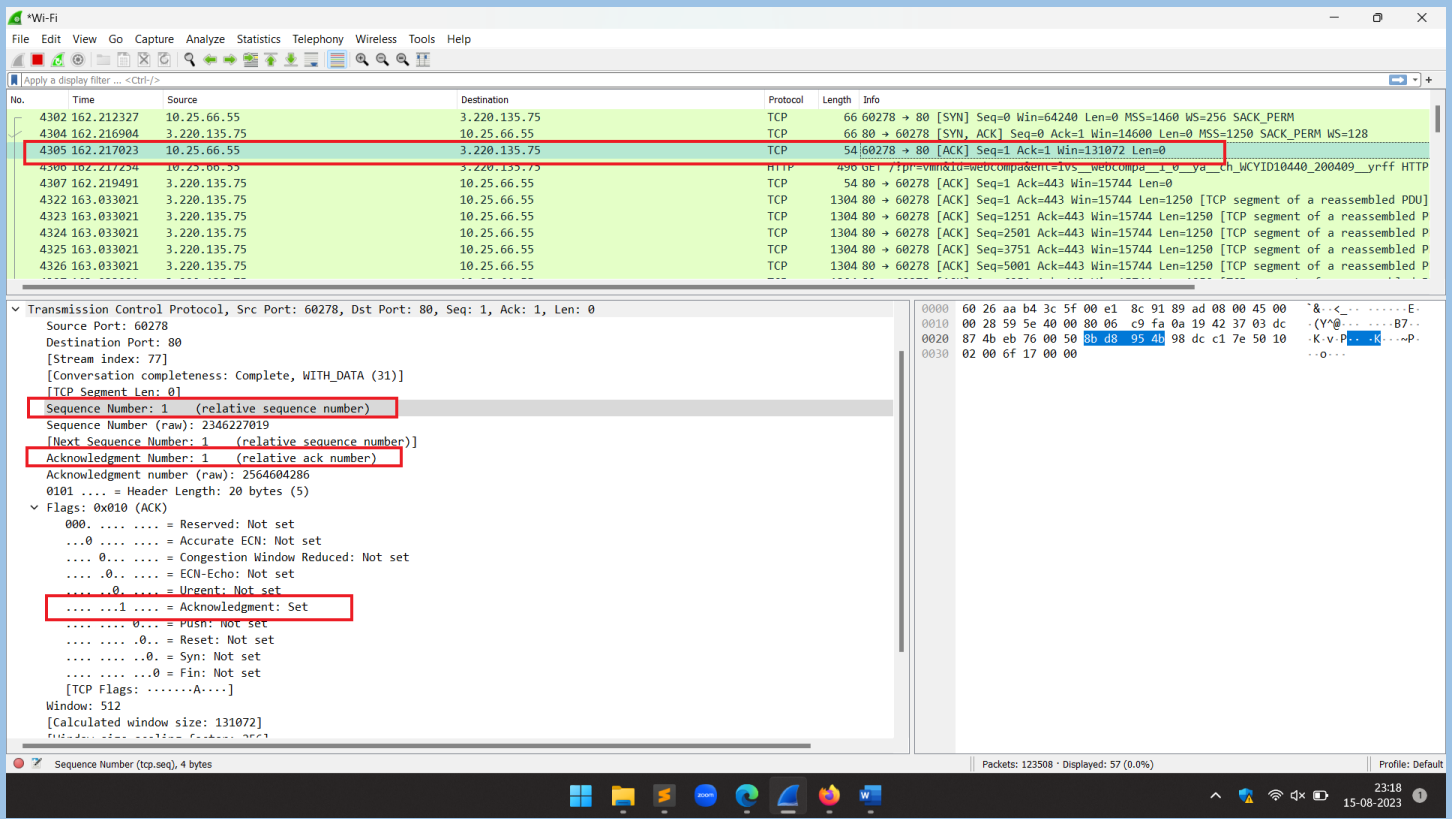
1. Select packet [SYN] in Wireshark and expand the TCP layer analysis in the middle pane, and further expand the "Flags" field within the TCP header. Here we can see all of the TCP flags broken down. Note that the SYN flag is on (set to 1).



2. Now do the same for packet [SYN, ACK]. Notice that it has two flags set: ACK to acknowledge the receipt of the client's SYN packet, and SYN to indicate that the server also wishes to establish a TCP connection.



3. Packet [ACK], from the client, has only the ACK flag set. These three packets complete the initial TCP three-way handshake.



-> Sequence and Acknowledgment Numbers

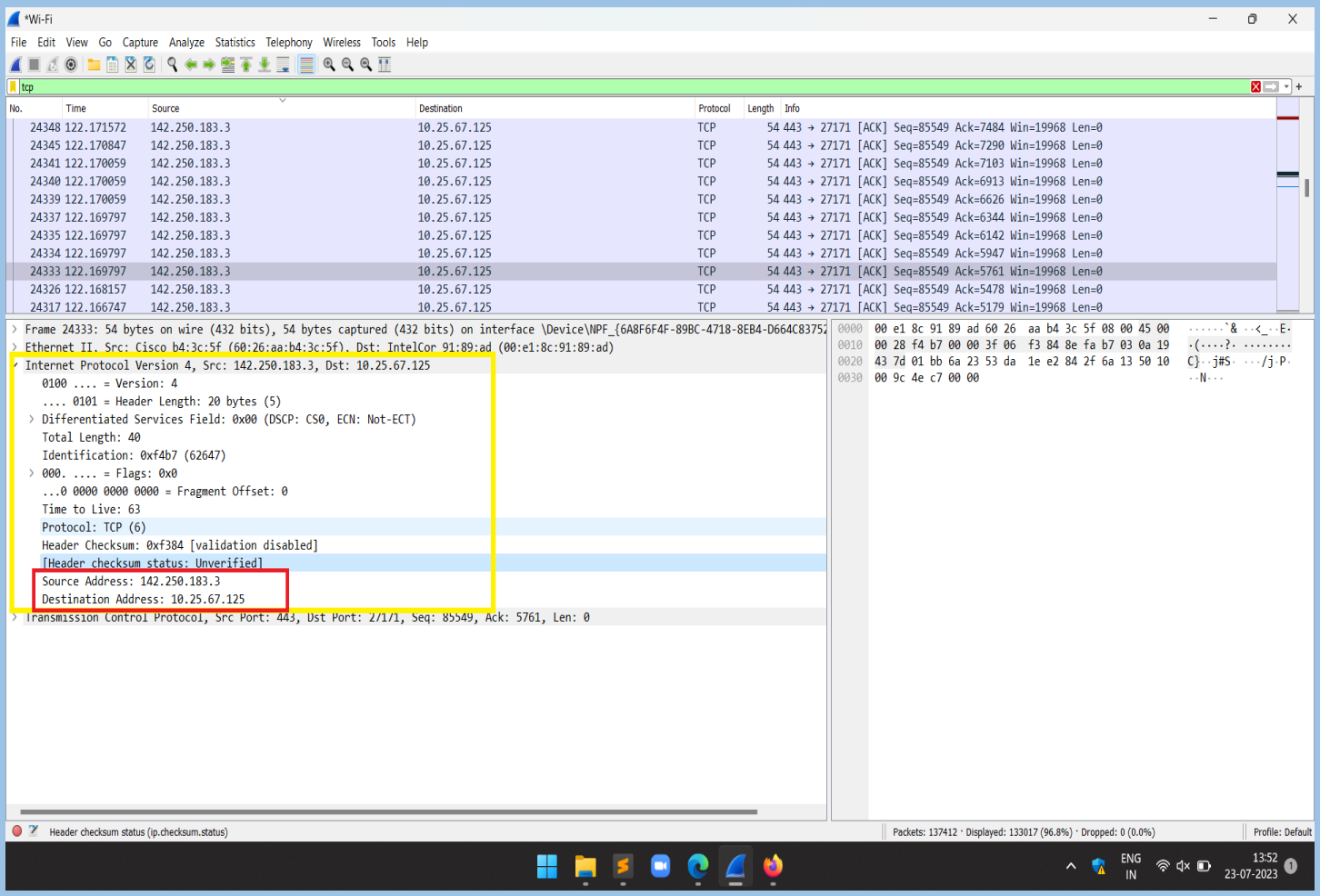
The client on either side of a TCP session maintains a 32-bit sequence number it uses to keep track of how much data it has sent. This sequence number is included on each transmitted packet, and acknowledged by the opposite host as an acknowledgement number to inform the sending host that the transmitted data was received successfully.

When a host initiates a TCP session, its initial sequence number is effectively random; it may be any value between 0 and 4,294,967,295, inclusive. However, protocol analyzers like Wireshark will typically display relative sequence and acknowledgement numbers in place of the actual values. These numbers are relative to the initial sequence number of that stream. This is handy, as it is much easier to keep track of relatively small, predictable numbers rather than the actual numbers sent on the wire.

For example, the initial relative sequence number shown in packet [SYN] is 0.

2. IP Header

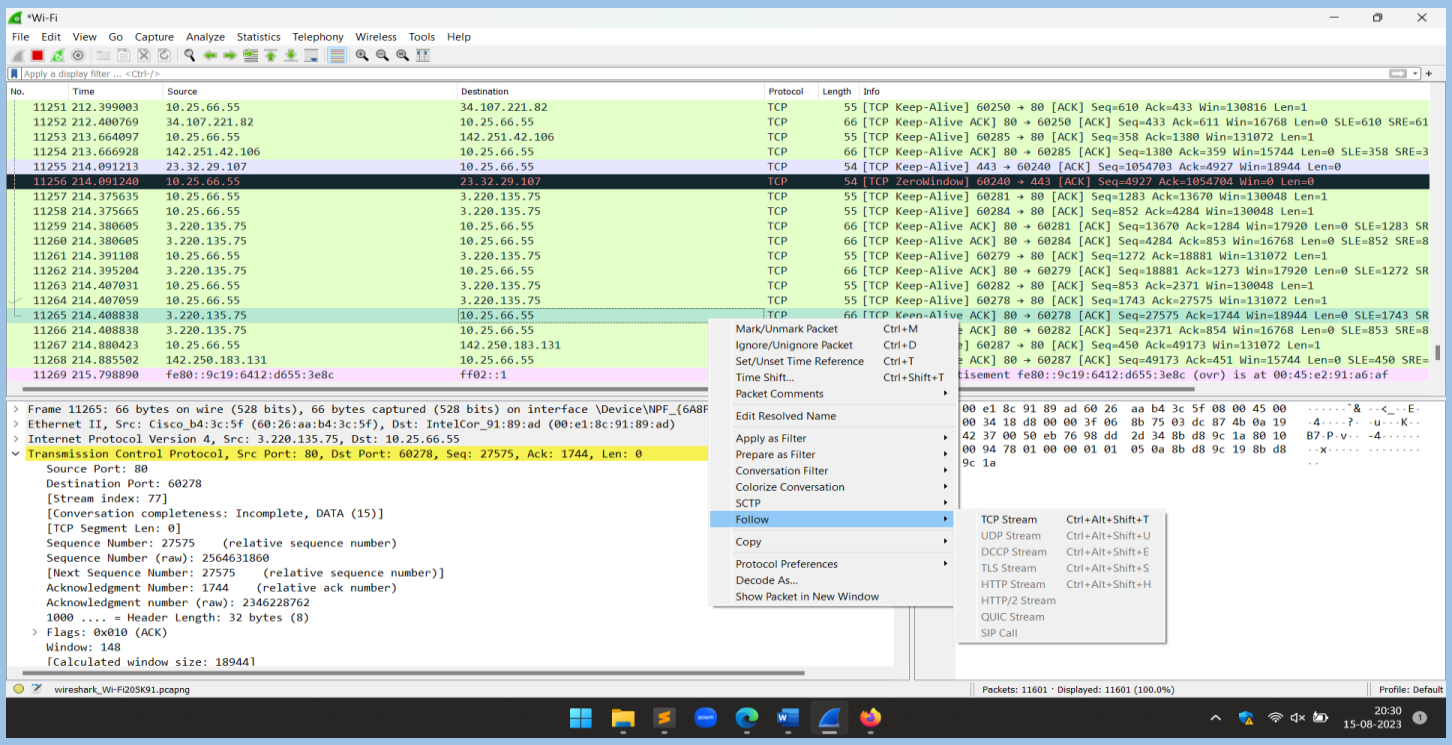
- Source Address (the IP address of the original sender of the packet)
- Destination Address (the IP address of the final destination of the packet)

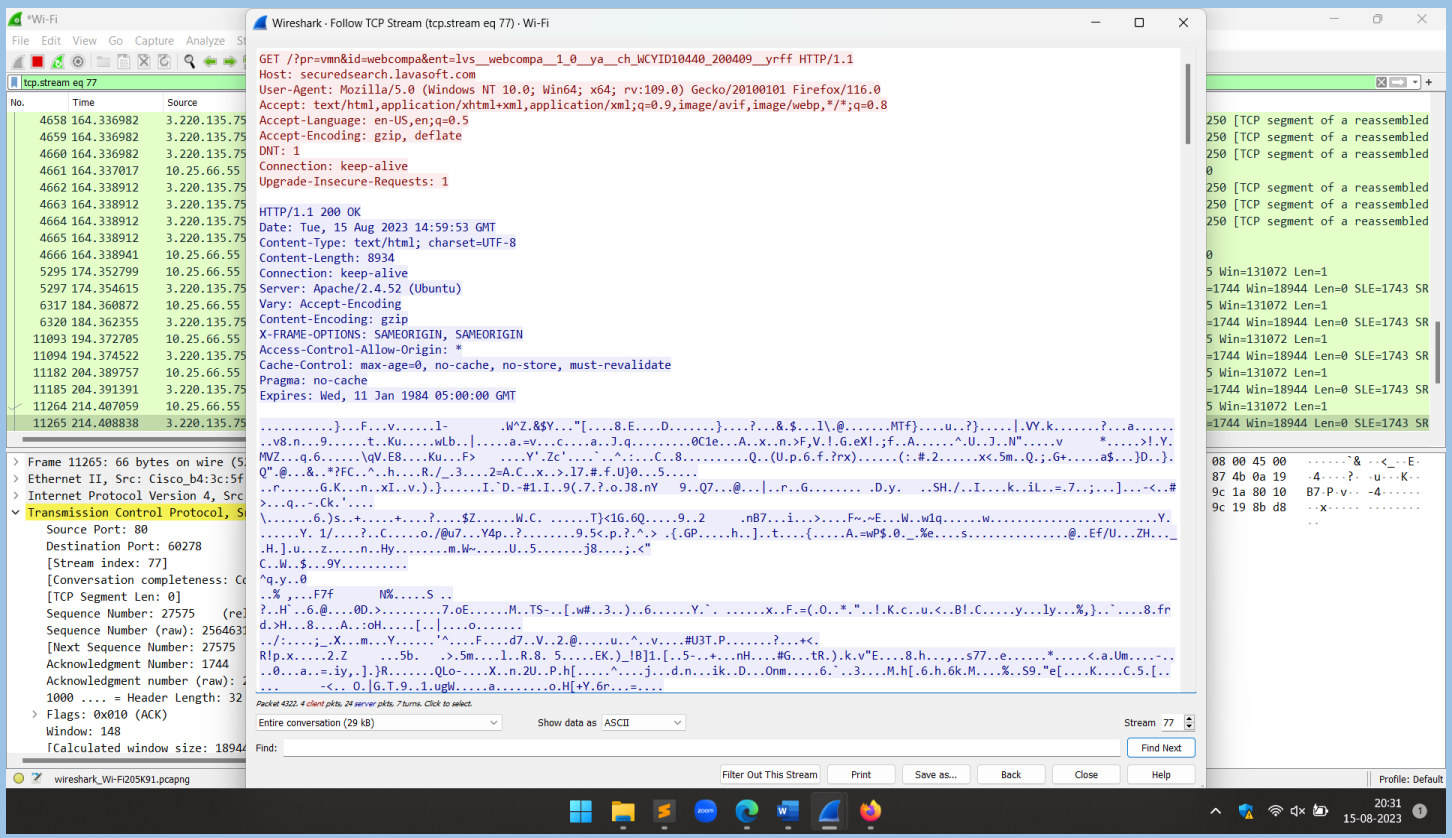


9. Exploring the Follow TCP stream feature in Wireshark

TCP-STREAM: [To see the data from a TCP stream in the way that the application layer sees it]

Simply select a TCP packet in the packet list of the stream/connection you are interested in and then select the Follow TCP Stream menu item from the Wireshark Tools menu (or use the context menu in the packet list). Wireshark will set an appropriate display filter and pop up a dialog box with all the data from the TCP stream laid out in order





Key-Points:

1. The stream content is displayed in the same sequence as it appeared on the network. HTTP Request and then HTTP Response.

2. Traffic from the client to the server is colored red, while traffic from the server to the client is colored blue. [These colors can be changed by opening Edit → Preferences and under Appearance → Font and Colors, selecting different colors for the Sample "Follow Stream" client text and Sample "Follow Stream" server text options.]

---

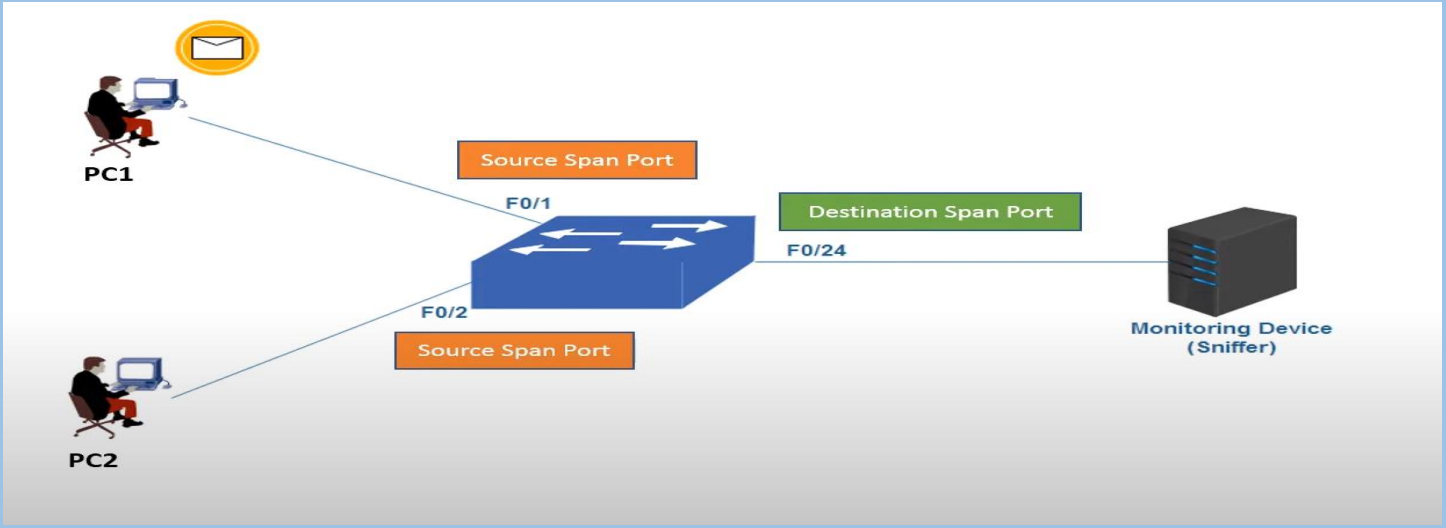
10. Port Mirroring [SPAN Port] and Network Analysis using Wireshark

→ Port Mirroring:

- When port mirroring is enabled, the TOR [Top-of-Rack] switch sends a copy of the network packet from the mirrored ports to the monitor port. This feature is typically used for monitoring and intrusion detection.
- Port mirroring is used on a network switch to send a copy of network packets seen on one switch port (or an entire VLAN) to a network monitoring connection on another switch port.

→ Switch with SPAN [Switched Port Analyzer] Port:

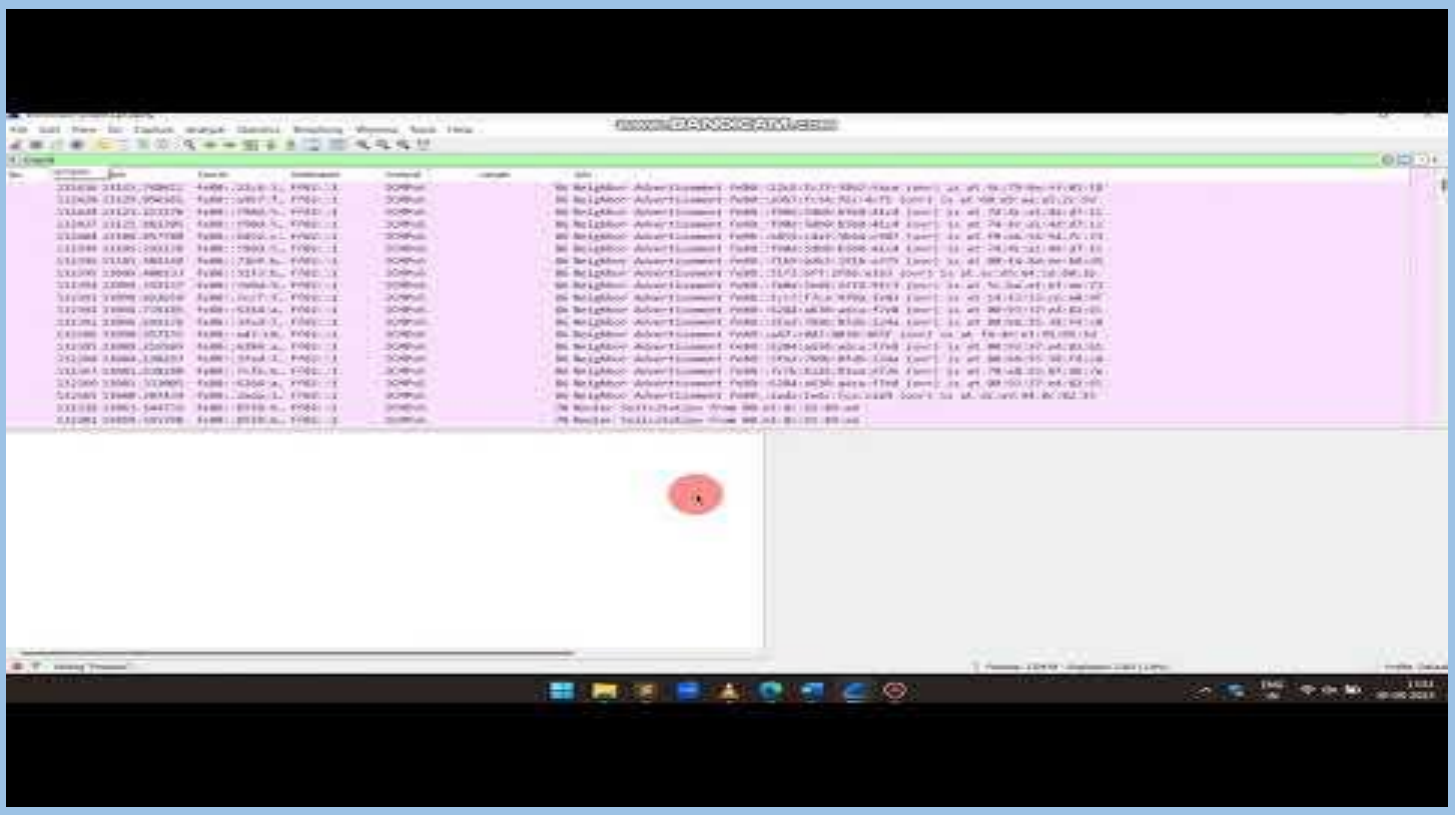
- SPAN works by copying the traffic from one or more source ports. The copy is then sent out a SPAN destination port. The destination port will often be connected to a host running packet analyzing software, such as Wireshark. Because SPAN only makes a copy of traffic, the source traffic is never affected.
- So, Source Port [the device for which packets are to be captured] traffic is captured and sent to SPAN destination Port [On which Wireshark is running].



→ **PoC's:**

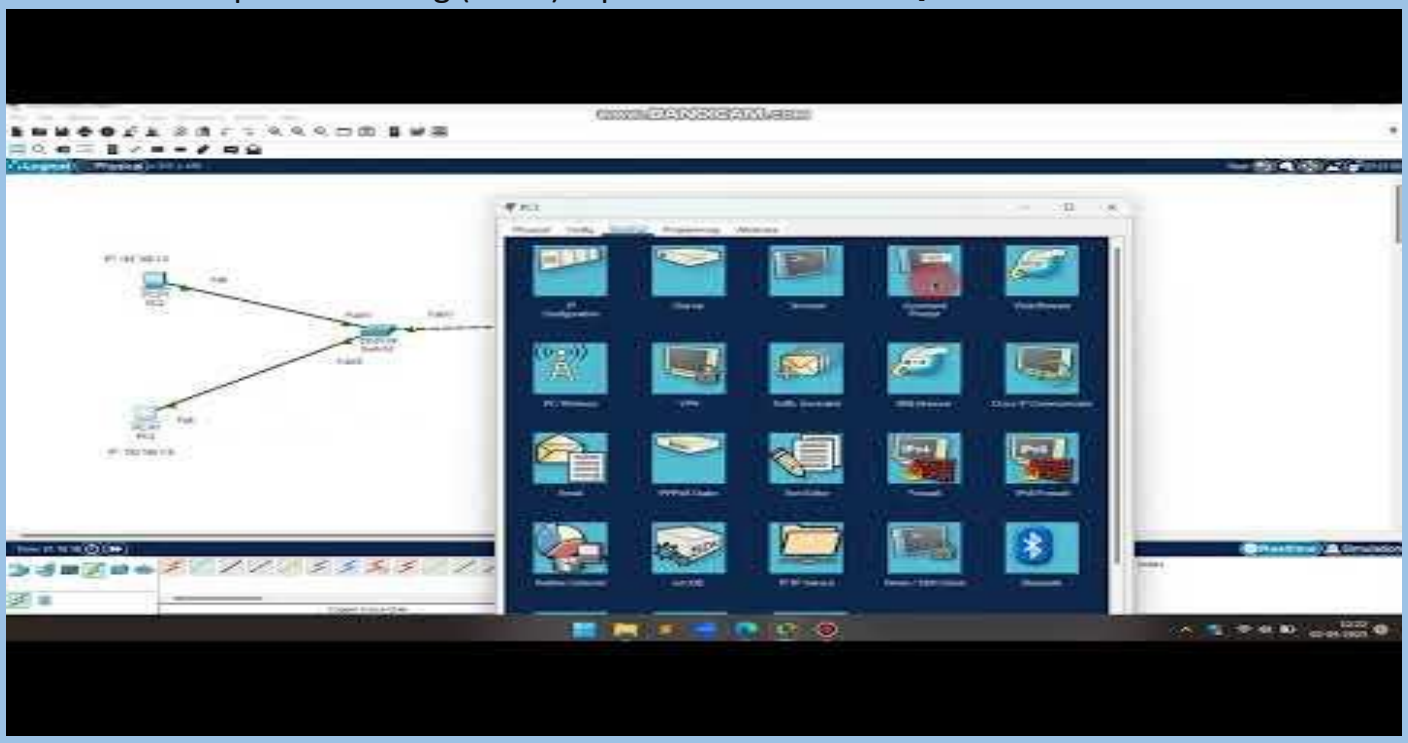
1	3 Way Handshake	Refer File "3-way-handshake-SPAN-SWITCH.pcapng" and Point-7 For 3-way handshake explanation.
2	TCP packet Analysis and exploring the features in the packet header window. [TCP header and IP header details for the selected packet.]	Refer Point-8 Explanation above. Also Refer the File "TCP-Header-Analysis.pcapng" to see the captured packets.

3. **Other parts such as different protocols, different Filters, total number of captured packets for each protocol and exploring the Follow TCP stream feature in Wireshark. [Refer "Wireshark-Project-PoC.mp4" in Video-PoC Folder]**



https://youtu.be/LUOQi2nAcn0

4. Port Mirroring simulation using Cisco-Packet Tracer [Refer “Packet-Sniffing-Simulation-via-port-mirroring (SPAN).mp4” in Video-PoC Folder]



https://youtu.be/1FgHt_Oi3Ms

→ **Team Members:**

1.	SHAILENDRA SINGH SACHAN	23030241186
2.	ADITI PANDEY	23030241148
3.	MOHIT RATHOUR	23030241174
4.	RIYA JAIN	23030241087
5.	HRUSHIKESH THATE	23030241047
6.	ASHITA GADE	23030241150