

Wireshark Lab 1.1. Capturing Network Traffic and creating a PCAP file (V1.1)

OVERVIEW

Network traffic, often known as traffic or data traffic, is the amount of data that is traveling through a network at any particular moment.

Network data is made up of packets, which are the smallest, most basic pieces of data sent through a network. Data from network traffic is divided into packets for transmission and reassembled at the destination.

OBJECTIVES:

- 1- Learn how to capture network traffic.
- 2- Learn how to use the wireshark interface and settings.

REQUIREMENTS:

- ☐ Wireshark Application
- ☐ OS (Windows, macOS, or Linux)

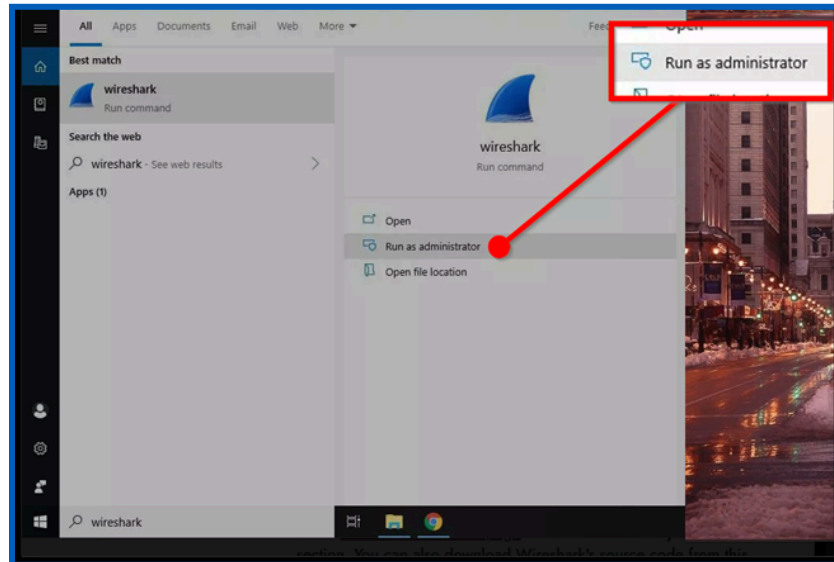
STEPS:

- Part 1 - Capture Traffic on Ethernet NIC /or WiFi Nic.
- Part 2 - Add coloring rule for TCP packet.
- Part 3 - Add Custom Columns (TCP Segment Len).
- Part 4: Save Captured Traffic.

Part 1 - Capture Traffic on Ethernet NIC

Step 1:

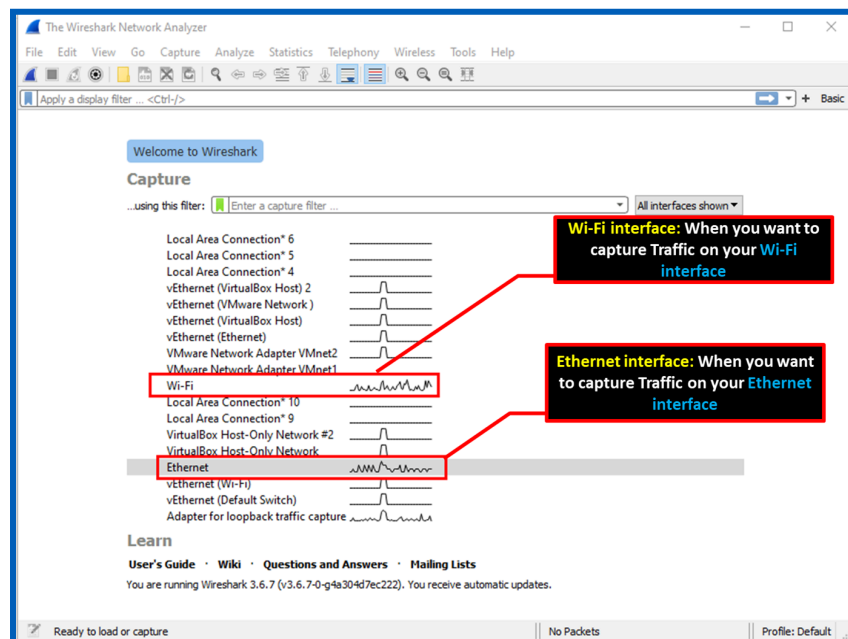
Run Wireshark as administrator as shown in the picture below.



Step 2: Start the Capturing::

Wireshark can capture traffic from many different network media types, including Ethernet, Wireless LAN, Bluetooth, USB, and more. The specific media types supported may be limited by several factors, including your hardware and operating system (OS).

- Before you can see packet data, you need to choose one of the interfaces by clicking on it.
- In our lab, we are going to use the **Ethernet interface**.



Before you start capturing, open **Google Chrome** and the **Command Prompt**.

To **start** capturing the packets, either double-click on the interface or click on as shown in the picture below.



Open **Google Chrome** and navigate to:

- www.perscholas.org
- www.w3schools.com
- www.udemy.com

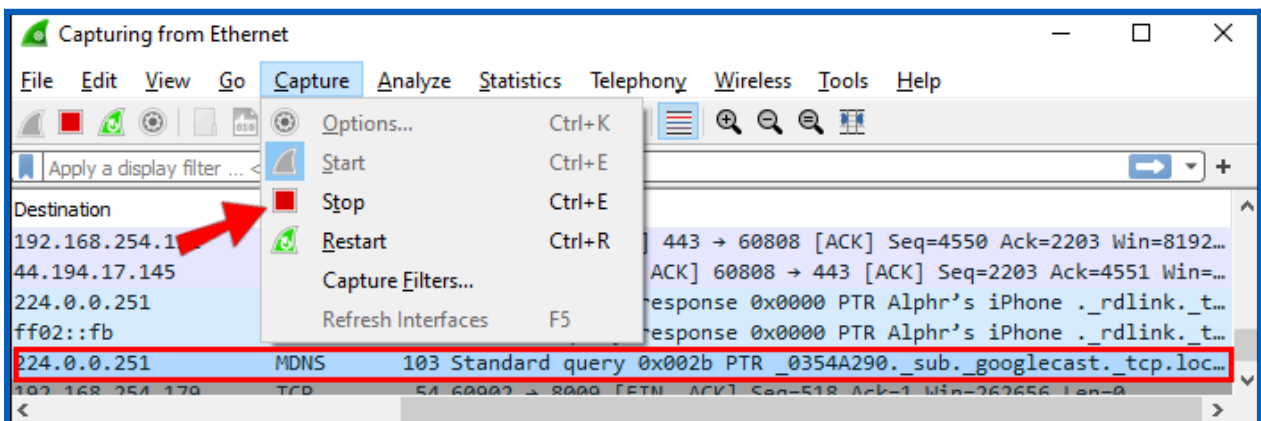
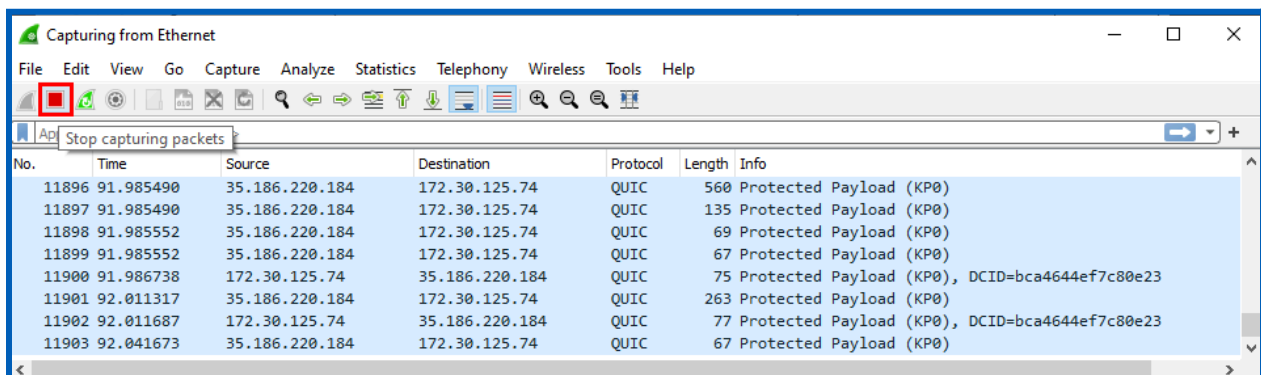
Open **CMD** and ping:

- 8.8.8.8
- Your default gateway
- 127.0.0.1

Step 3: Stop the Running Capture:

A running capture session can be stopped by:

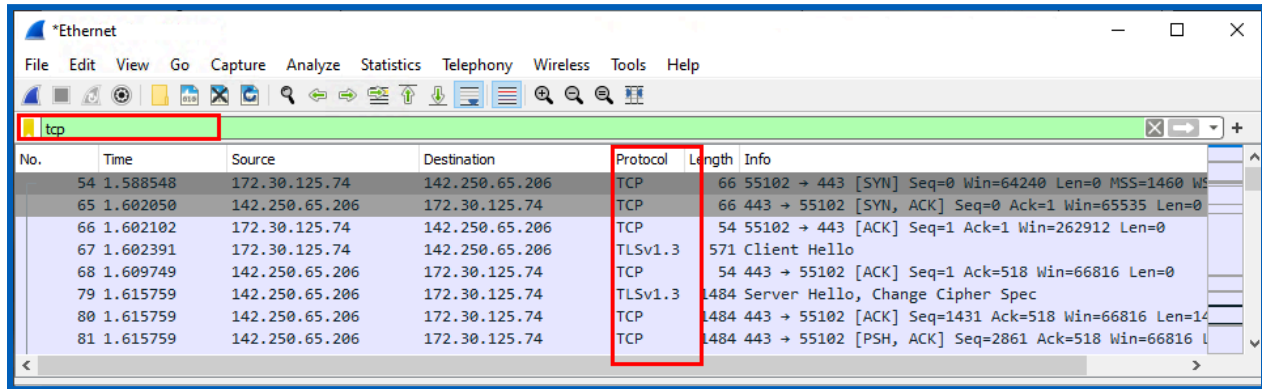
1. The Stop Capture button in the “Capture Information” dialog box.
2. The Capture → Stop menu item.
3. The Stop toolbar button.
4. **Ctrl+E**



Part 2 - Add Coloring Rule for TCP packet:

After you are done with capturing the packets, we want to see the TCP packet.

On the filter bar, type TCP and hit **Enter** as shown in the picture below.

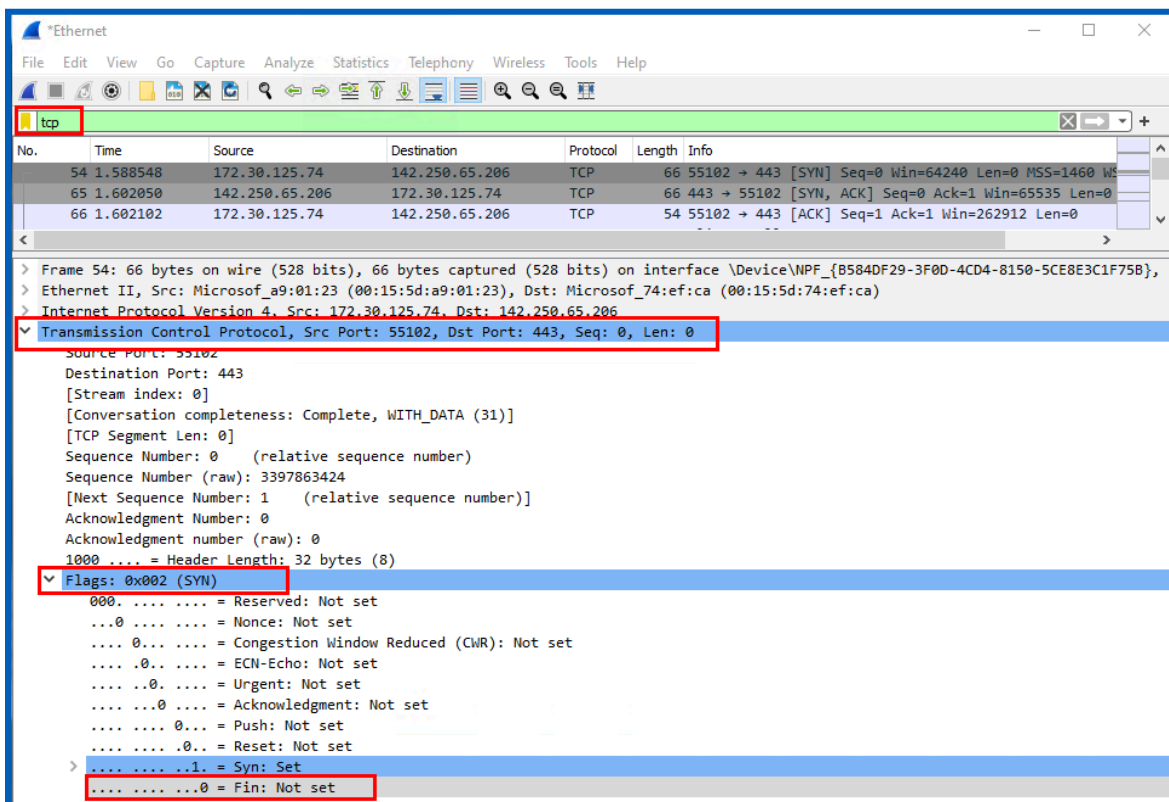


Reminder

The FIN flag indicates the end of data transmission to finish a TCP connection.

Adding Filter for TCP FIN Packet

Click on the first TCP packet, and then on the middle panel. Expand the Transmission Control Protocol section, and then expand to the Flags section. Look for0 = Fin: Not set.

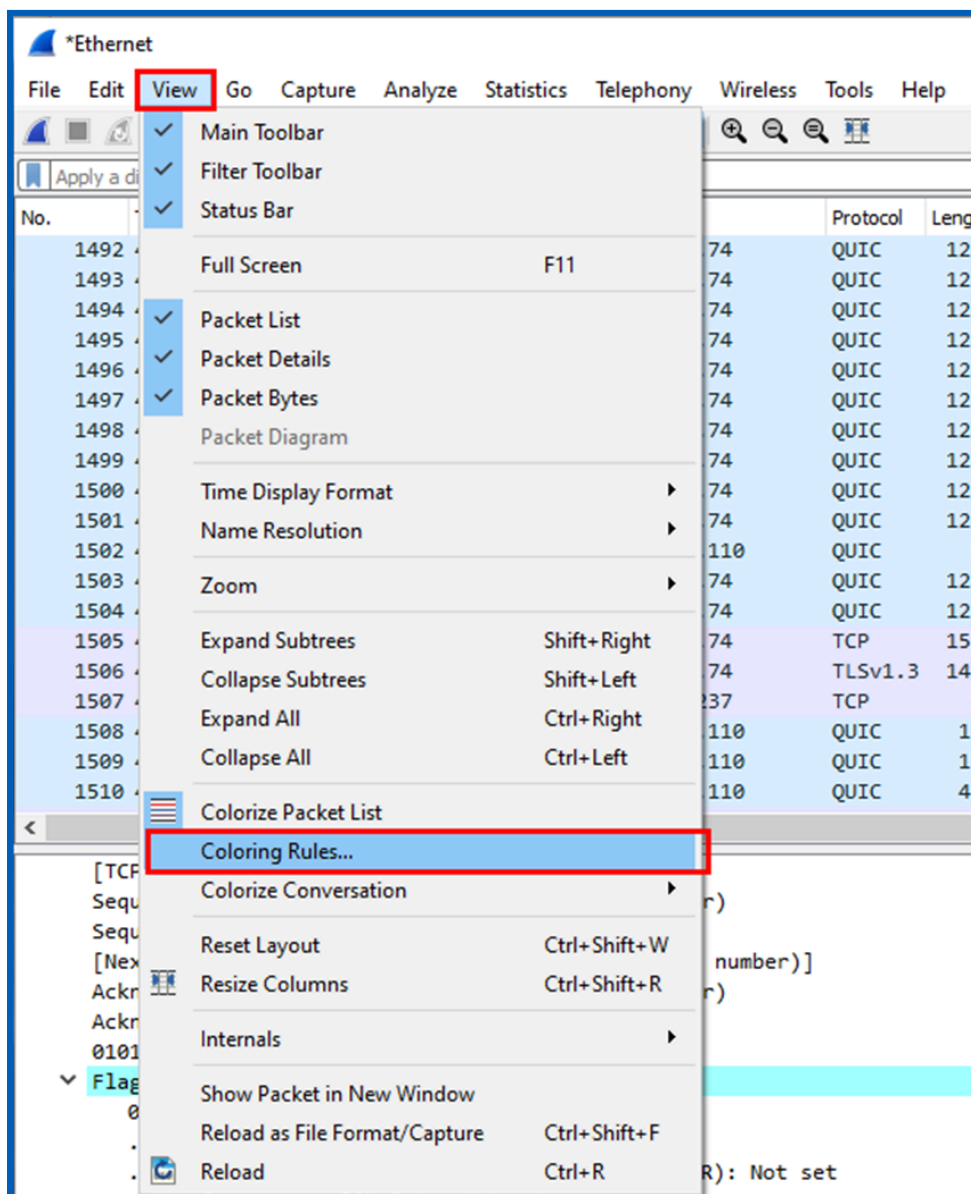


In the previous phase, we started with **TCP**, then **Flags**, and then **fin**, with **fin = 0**, and we want packets with **fin = 1**; thus, the filter we will apply is: **tcp.flags.fin == 1**

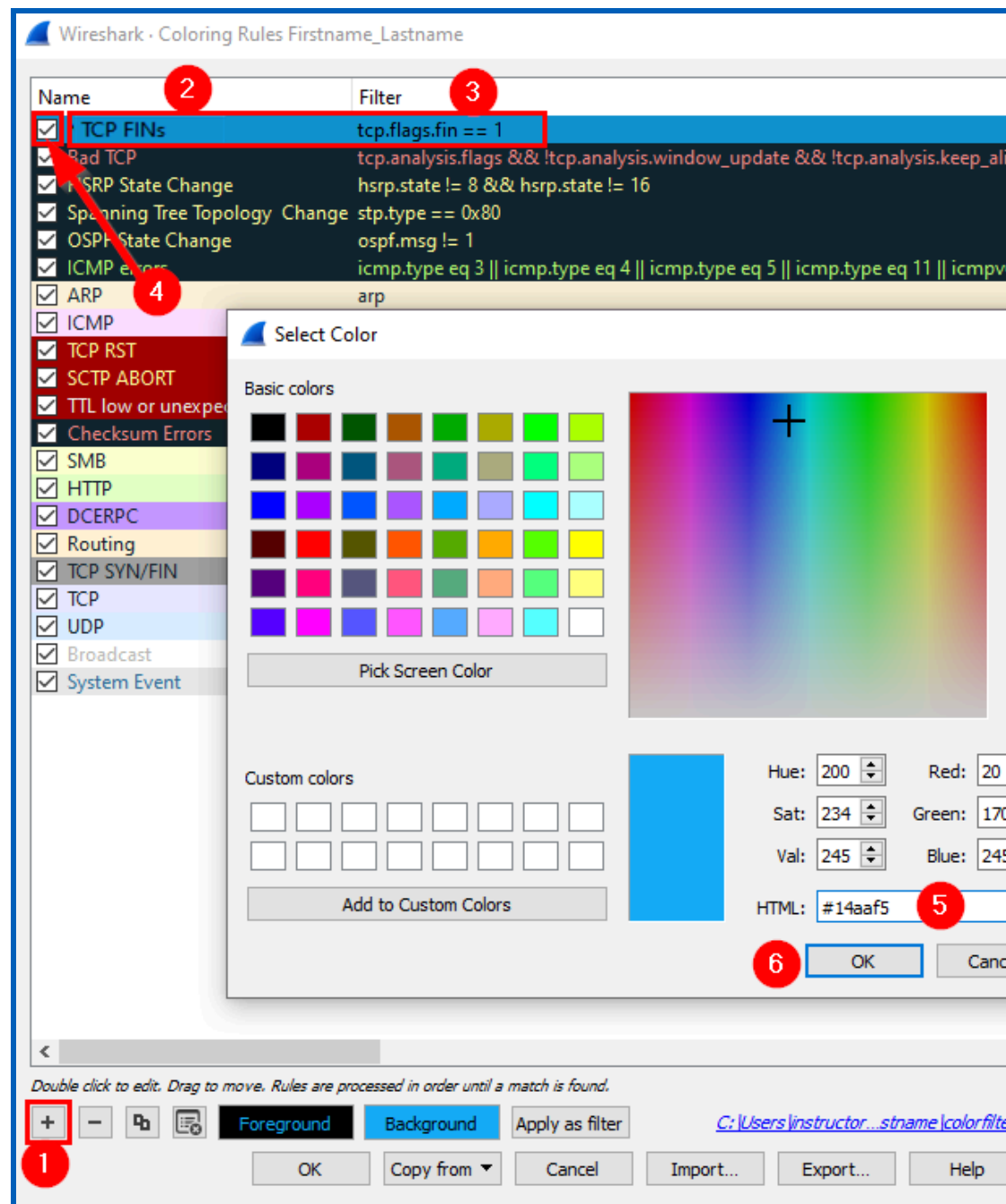
We do not want the fin packets to have the default color, so we are going to set a color code for the fin = 1 packet.

To do that, we need to:

- 1) Copy the filter: **tcp.flags.fin == 1**.
- 2) Click on the View section on the menu tab, then Coloring Rules.



- 3) Click on the plus sign (+) to add a new coloring rule.
 - a) Name: TCP FINs
 - b) Filter: `tcp.flags.fin == 1`
- 4) Activate the rule by checking the box next to the name.
- 5) Click on the rule, and then at the bottom, click on Background and select a color (color hex code : #14AAF5).



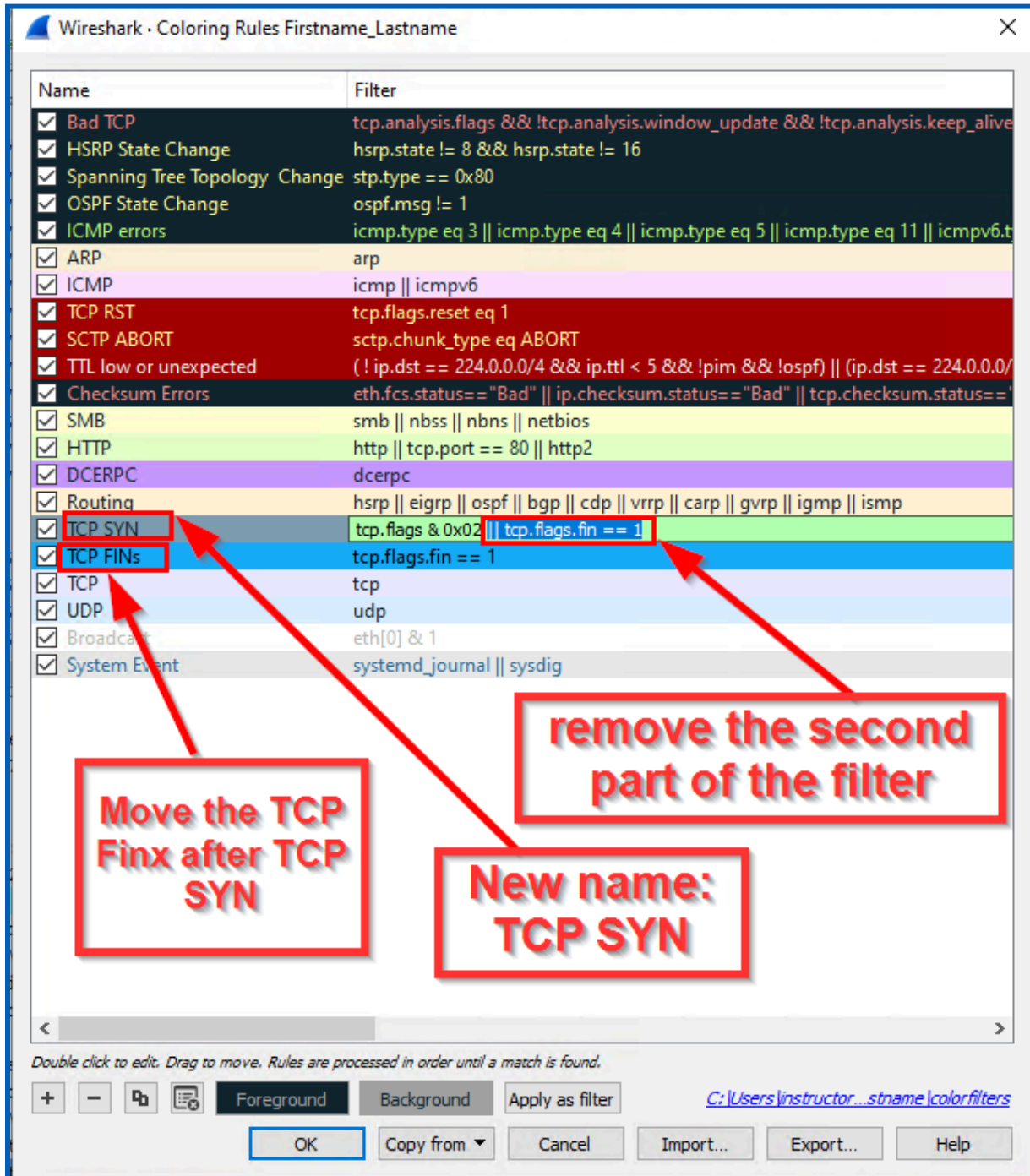
6) Click: OK

7) Rename **TCP SYN / FIN** to **TCP SYN**.

8) Remove the second part of the filter in **TCP SYN** Line.

9) Move the **TCP FINs** after the **TCP SYN**.

10) Click OK.



Wireshark · Coloring Rules Firstname_Lastname

Name	Filter
<input checked="" type="checkbox"/> Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update && !tcp.analysis.keep_alive
<input checked="" type="checkbox"/> HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/> Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/> OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/> ICMP errors	icmp.type eq 3 icmp.type eq 4 icmp.type eq 5 icmp.type eq 11 icmpv6.t
<input checked="" type="checkbox"/> ARP	arp
<input checked="" type="checkbox"/> ICMP	icmp icmpv6
<input checked="" type="checkbox"/> TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/> SCTP ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/> TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) (ip.dst == 224.0.0.0/
<input checked="" type="checkbox"/> Checksum Errors	eth.fcs.status=="Bad" ip.checksum.status=="Bad" tcp.checksum.status=="
<input checked="" type="checkbox"/> SMB	smb nbss nbns netbios
<input checked="" type="checkbox"/> HTTP	http tcp.port == 80 http2
<input checked="" type="checkbox"/> DCE RPC	dcerpc
<input checked="" type="checkbox"/> Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ismp
<input checked="" type="checkbox"/> TCP SYN	tcp.flags & 0x02 tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP FINs	tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP	tcp
<input checked="" type="checkbox"/> UDP	udp
<input checked="" type="checkbox"/> Broadcast	eth[0] & 1
<input checked="" type="checkbox"/> System Event	systemd_journal sysdig

Double click to edit. Drag to move. Rules are processed in order until a match is found.

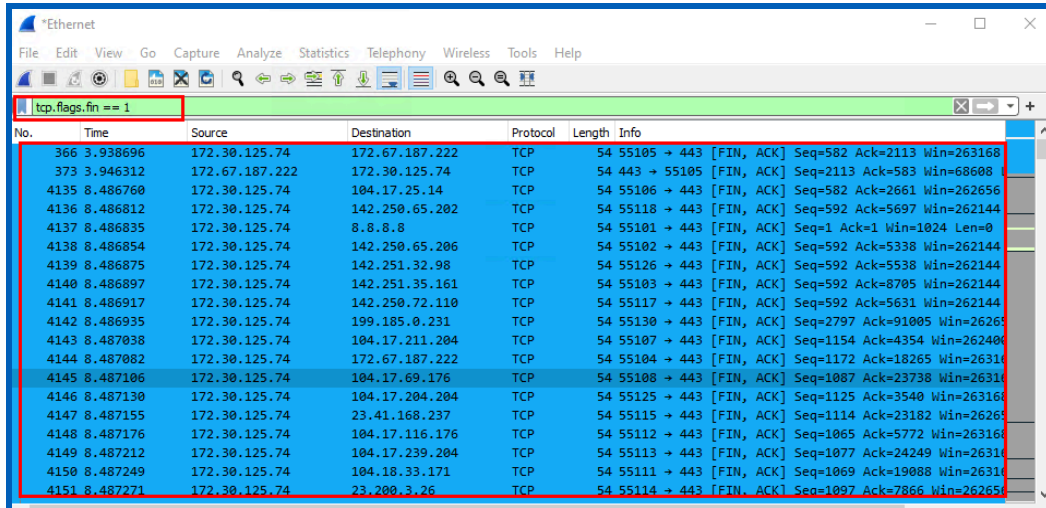
<C:\Users\instructor...sname\colorfilters>

Move the TCP Finx after TCP SYN

remove the second part of the filter

New name: TCP SYN

Now we have the blue color code for TCP FIN packets.



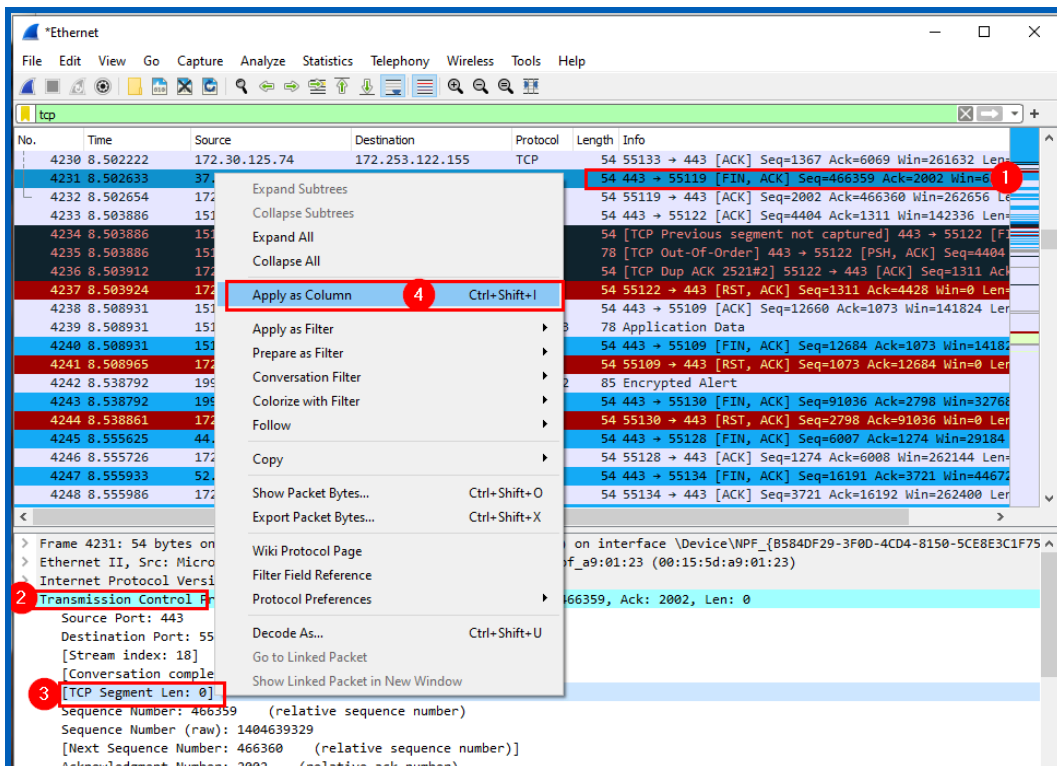
No.	Time	Source	Destination	Protocol	Length	Info
366	3.938696	172.30.125.74	172.67.187.222	TCP	54	55105 → 443 [FIN, ACK] Seq=582 Ack=2113 Win=263168
373	3.946312	172.67.187.222	172.30.125.74	TCP	54	443 → 55105 [FIN, ACK] Seq=2113 Ack=583 Win=68608
4135	8.486760	172.30.125.74	104.17.25.14	TCP	54	55106 → 443 [FIN, ACK] Seq=582 Ack=2661 Win=262656
4136	8.486812	172.30.125.74	142.250.65.202	TCP	54	55118 → 443 [FIN, ACK] Seq=592 Ack=5697 Win=262144
4137	8.486835	172.30.125.74	8.8.8.8	TCP	54	55101 → 443 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
4138	8.486854	172.30.125.74	142.250.65.206	TCP	54	55102 → 443 [FIN, ACK] Seq=592 Ack=5338 Win=262144
4139	8.486875	172.30.125.74	142.251.32.98	TCP	54	55126 → 443 [FIN, ACK] Seq=592 Ack=5538 Win=262144
4140	8.486897	172.30.125.74	142.251.35.161	TCP	54	55103 → 443 [FIN, ACK] Seq=592 Ack=8705 Win=262144
4141	8.486917	172.30.125.74	142.250.72.110	TCP	54	55117 → 443 [FIN, ACK] Seq=592 Ack=5631 Win=262144
4142	8.486935	172.30.125.74	199.185.0.231	TCP	54	55130 → 443 [FIN, ACK] Seq=2797 Ack=91005 Win=262656
4143	8.487038	172.30.125.74	104.17.211.204	TCP	54	55107 → 443 [FIN, ACK] Seq=1154 Ack=4354 Win=262400
4144	8.487082	172.30.125.74	172.67.187.222	TCP	54	55104 → 443 [FIN, ACK] Seq=1172 Ack=18265 Win=263168
4145	8.487106	172.30.125.74	104.17.69.176	TCP	54	55108 → 443 [FIN, ACK] Seq=1087 Ack=23738 Win=263168
4146	8.487130	172.30.125.74	104.17.204.204	TCP	54	55125 → 443 [FIN, ACK] Seq=1125 Ack=3540 Win=263168
4147	8.487155	172.30.125.74	23.41.168.237	TCP	54	55115 → 443 [FIN, ACK] Seq=1114 Ack=23182 Win=262656
4148	8.487176	172.30.125.74	104.17.116.176	TCP	54	55112 → 443 [FIN, ACK] Seq=1065 Ack=5772 Win=263168
4149	8.487212	172.30.125.74	104.17.239.204	TCP	54	55113 → 443 [FIN, ACK] Seq=1077 Ack=24249 Win=263168
4150	8.487249	172.30.125.74	104.18.33.171	TCP	54	55111 → 443 [FIN, ACK] Seq=1069 Ack=19088 Win=263168
4151	8.487271	172.30.125.74	23.208.3.26	TCP	54	55114 → 443 [FIN, ACK] Seq=1097 Ack=7866 Win=262656

Part 3 - adding Custom Columns (TCP Segment Len):

We need to know how much data is actually carried in a TCP packet. This is why we are going to add the **TCP segment length** column.

To add the **TCP Segment Len**:

- 1) Click on any TCP packet.
- 2) Expand the TCP section.
- 3) Right-click On [TCP Segment Len: 0].
- 4) Apply as Column.



The screenshot shows the Wireshark interface with a packet list and packet details pane. The filter 'tcp' is applied. The packet list shows a list of TCP packets. The packet details pane shows the expanded TCP section for packet 4231. The context menu is open over the 'TCP Segment Len: 0' field, and the 'Apply as Column' option is selected. The packet details pane also shows the 'TCP Segment Len: 0' field.

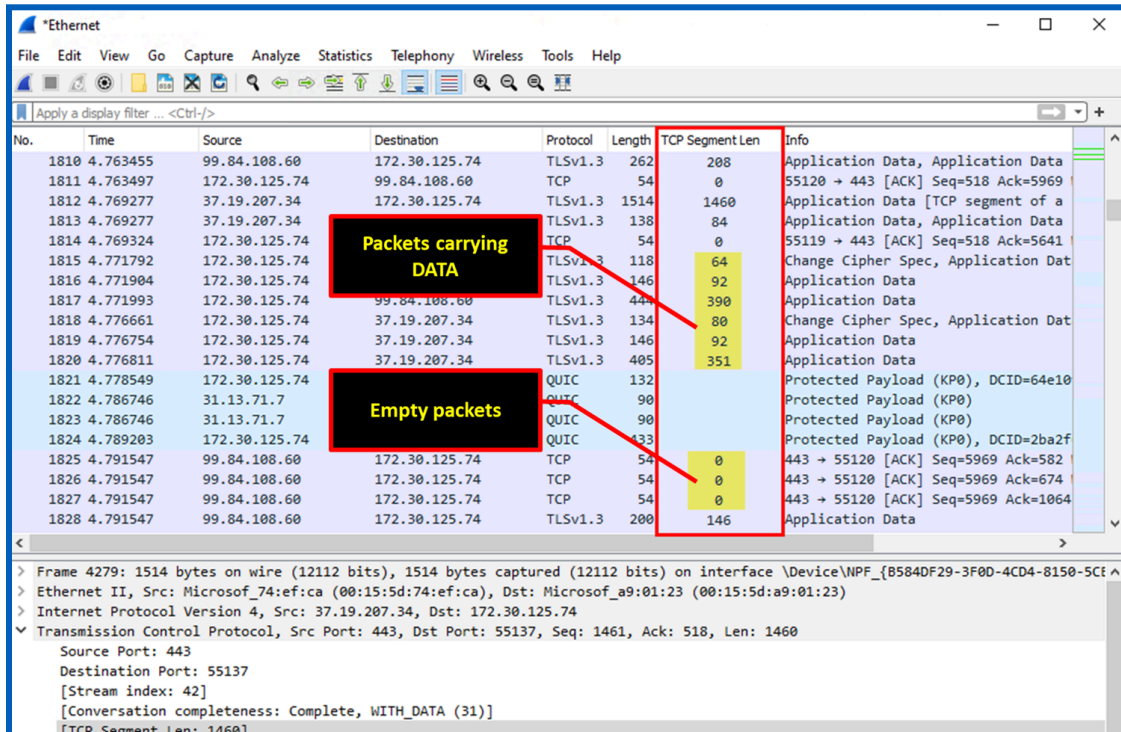
No.	Time	Source	Destination	Protocol	Length	Info
4230	8.502222	172.30.125.74	172.253.122.155	TCP	54	55133 → 443 [ACK] Seq=1367 Ack=6069 Win=261632 Len=0
4231	8.502633	172.30.125.74	172.253.122.155	TCP	54	443 → 55119 [FIN, ACK] Seq=466359 Ack=2002 Win=0 Len=0
4232	8.502654	172.30.125.74	172.253.122.155	TCP	54	55119 → 443 [ACK] Seq=2002 Ack=466360 Win=262656 Len=0
4233	8.503886	172.30.125.74	172.253.122.155	TCP	54	443 → 55122 [ACK] Seq=4404 Ack=1311 Win=142336 Len=0
4234	8.503886	172.30.125.74	172.253.122.155	TCP	54	[TCP Previous segment not captured] 443 → 55122 [FIN, ACK] Seq=4404 Ack=1311 Win=142336 Len=0
4235	8.503886	172.30.125.74	172.253.122.155	TCP	78	[TCP Out-Of-Order] 443 → 55122 [PSH, ACK] Seq=4404 Ack=1311 Win=142336 Len=0
4236	8.503912	172.30.125.74	172.253.122.155	TCP	54	[TCP Dup ACK 2521#2] 55122 → 443 [ACK] Seq=1311 Ack=466359 Win=0 Len=0
4237	8.503924	172.30.125.74	172.253.122.155	TCP	54	55122 → 443 [RST, ACK] Seq=1311 Ack=4428 Win=0 Len=0
4238	8.508931	172.30.125.74	172.253.122.155	TCP	54	443 → 55109 [ACK] Seq=12660 Ack=1073 Win=141824 Len=0
4239	8.508931	172.30.125.74	172.253.122.155	TCP	78	Application Data
4240	8.508931	172.30.125.74	172.253.122.155	TCP	54	443 → 55109 [FIN, ACK] Seq=12684 Ack=1073 Win=141824 Len=0
4241	8.508965	172.30.125.74	172.253.122.155	TCP	54	55109 → 443 [RST, ACK] Seq=1073 Ack=12684 Win=0 Len=0
4242	8.538792	199.185.0.231	172.30.125.74	TCP	85	Encrypted Alert
4243	8.538792	199.185.0.231	172.30.125.74	TCP	54	443 → 55130 [FIN, ACK] Seq=91036 Ack=2798 Win=32768 Len=0
4244	8.538861	172.30.125.74	172.253.122.155	TCP	54	55130 → 443 [RST, ACK] Seq=2798 Ack=91036 Win=0 Len=0
4245	8.555625	172.30.125.74	172.253.122.155	TCP	54	443 → 55128 [FIN, ACK] Seq=6007 Ack=1274 Win=29184 Len=0
4246	8.555726	172.30.125.74	172.253.122.155	TCP	54	55128 → 443 [ACK] Seq=1274 Ack=6008 Win=262144 Len=0
4247	8.555933	172.30.125.74	172.253.122.155	TCP	54	443 → 55134 [FIN, ACK] Seq=16191 Ack=3721 Win=44672 Len=0
4248	8.555986	172.30.125.74	172.253.122.155	TCP	54	55134 → 443 [ACK] Seq=3721 Ack=16192 Win=262400 Len=0

Packet Details for 4231: 54 bytes on wire (432 bits) captured on interface \Device\NPF_{B584DF29-3F0D-4CD4-8150-5CE8E3C1F75} (00:15:5d:a9:01:23) (00:15:5d:a9:01:23)

- Ethernet II, Src: Micro...
- Internet Protocol Version 4, Src: 172.30.125.74, Destination: 172.253.122.155
- Transmission Control Protocol, Src Port: 443, Destination Port: 55119, [Stream index: 18], [Conversation complete]
- TCP Segment Len: 0
 - Sequence Number: 466359 (relative sequence number)
 - Sequence Number (raw): 1404639329
 - [Next Sequence Number: 466360 (relative sequence number)]
 - Acknowledgment Number: 2002 (relative ack number)

If a **TCP Segment Len = 0**, it means that the packet is empty.

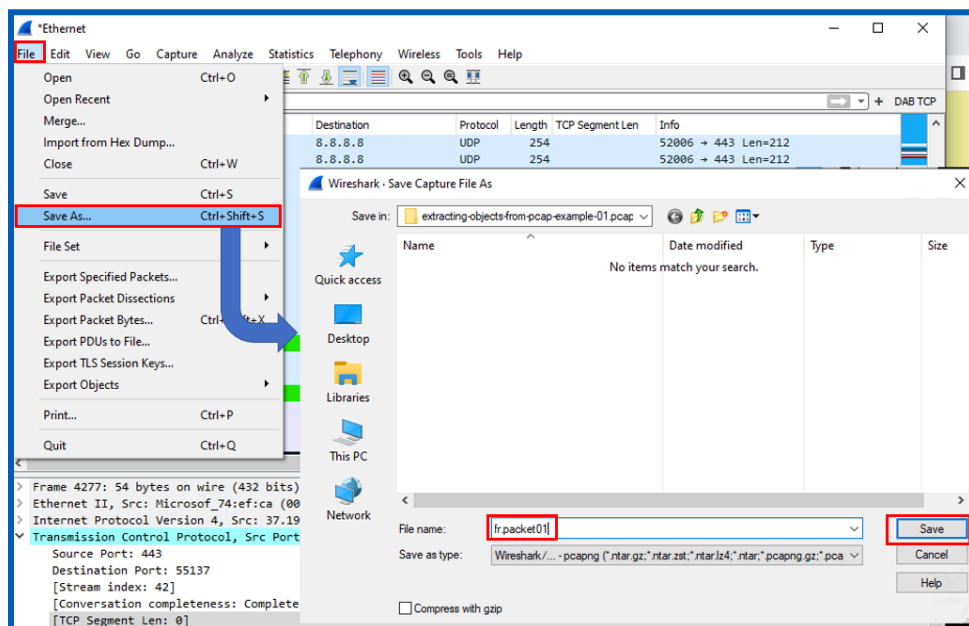
Remove any filters and scroll down. You will see empty packets and others with data, as shown in the picture below.



No.	Time	Source	Destination	Protocol	Length	TCP Segment Len	Info
1810	4.763455	99.84.108.60	172.30.125.74	TLSv1.3	262	208	Application Data, Application Data
1811	4.763497	172.30.125.74	99.84.108.60	TCP	54	0	55120 → 443 [ACK] Seq=518 Ack=5969
1812	4.769277	37.19.207.34	172.30.125.74	TLSv1.3	1514	1460	Application Data [TCP segment of a
1813	4.769277	37.19.207.34	172.30.125.74	TLSv1.3	138	84	Application Data, Application Data
1814	4.769324	172.30.125.74	37.19.207.34	TCP	54	0	55119 → 443 [ACK] Seq=518 Ack=5641
1815	4.771792	172.30.125.74	37.19.207.34	TLSv1.3	118	64	Change Cipher Spec, Application Data
1816	4.771904	172.30.125.74	37.19.207.34	TLSv1.3	146	92	Application Data
1817	4.771993	172.30.125.74	37.19.207.34	TLSv1.3	444	390	Application Data
1818	4.776661	172.30.125.74	37.19.207.34	TLSv1.3	134	80	Change Cipher Spec, Application Data
1819	4.776754	172.30.125.74	37.19.207.34	TLSv1.3	146	92	Application Data
1820	4.776811	172.30.125.74	37.19.207.34	TLSv1.3	405	351	Application Data
1821	4.778549	172.30.125.74	37.19.207.34	QUIC	132	0	Protected Payload (KP0), DCID=64e10
1822	4.786746	31.13.71.7	172.30.125.74	QUIC	90	0	Protected Payload (KP0)
1823	4.786746	31.13.71.7	172.30.125.74	QUIC	90	0	Protected Payload (KP0)
1824	4.789203	172.30.125.74	37.19.207.34	QUIC	433	0	Protected Payload (KP0), DCID=2ba2f
1825	4.791547	99.84.108.60	172.30.125.74	TCP	54	0	443 → 55120 [ACK] Seq=5969 Ack=582
1826	4.791547	99.84.108.60	172.30.125.74	TCP	54	0	443 → 55120 [ACK] Seq=5969 Ack=674
1827	4.791547	99.84.108.60	172.30.125.74	TCP	54	0	443 → 55120 [ACK] Seq=5969 Ack=1064
1828	4.791547	99.84.108.60	172.30.125.74	TLSv1.3	200	146	Application Data

Part 4: Saving captured Traffic:

- ☐ Once you have collected enough data, press **“Command + E”** to stop capturing data.
- ☐ Select **“Save As”** from the top menu to save the captured data.
- ☐ Save the file name **{first and last name initials.packet01}**



This concludes this lab.

Please discuss the following questions with your instructor.

LAB SUBMISSION REQUIREMENTS

Please submit a pdf with the following:

1. A screenshot of the snapshot taken once the lab is completed.
2. One to three screenshots demonstrating the configurations that you made during this lab.
3. Discussion questions with your answers.

DISCUSSION QUESTIONS:

1. *How Wireshark can be used for traffic capture and analysis?*

Wireshark is a network packet analyzer that can capture and analyze network traffic data. It can help with troubleshooting by allowing users to examine captured packets. Here are some ways to use Wireshark for traffic capture and analysis:

Capture network traffic

Analyze captured packets

Use filters

Decode and analyze protocols

Reconstruct network packets

View statistics and visualizations

2. *What kind of traffic does Wireshark capture?*

Wireshark is a network protocol analyzer that captures network traffic packets from a connection, such as between your computer and the internet. It can capture traffic

from many different network media types, including: Ethernet, Wireless LAN (IEEE.802.11), Bluetooth, USB, and Token ring.

3. *What is the difference between a capture filter and a display filter?*

Capture filters only keep copies of packets that match the filter. Display filters are used when you've captured everything, but need to cut through the noise to analyze specific packets or flows. Capture filters and display filters are created using different syntaxes.

4. *Which tools are commonly used in packet capture and analysis?*

Two of the most useful and quick-to-use packet capture tools are tcpdump and Wireshark. Tcpdump is a command line tool that allows the capture and display of packets on the network. Wireshark provides a graphical interface for capturing and analyzing packet data.

5. *Can Wireshark see other computers?*

Wireshark itself cannot directly "see" or discover other computers on a network. However, it can capture and analyze network traffic on the network interface it is monitoring. This means that if other computers are communicating over the network, Wireshark can capture and display that traffic, allowing you to see details about those communications.

*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.fin == 1

Source	Destination	Protocol	Length	TCP Segment Len	Info
20.75.106.146	192.168.86.30	TCP	54	0 443 → 62797 [FIN,	
192.168.86.30	52.208.112.205	TCP	54	0 62806 → 443 [FIN,	
52.208.112.205	192.168.86.30	TCP	54	0 443 → 62806 [FIN,	
157.90.91.73	192.168.86.30	TCP	54	0 443 → 62726 [FIN,	
69.194.240.13	192.168.86.30	TLSv1.2	85	31 Encrypted Alert	
52.3.135.254	192.168.86.30	TCP	54	0 443 → 63081 [FIN,	
104.254.151.36	192.168.86.30	TLSv1.2	85	31 Encrypted Alert	
216.22.16.41	192.168.86.30	TCP	54	0 443 → 63085 [FIN,	
192.168.86.30	172.67.25.151	TCP	54	0 63093 → 443 [FIN,	
172.67.25.151	192.168.86.30	TCP	54	0 443 → 63093 [FIN,	
170.114.7.182	192.168.86.30	TCP	54	0 443 → 63043 [FIN,	
82.145.213.8	192.168.86.30	TLSv1.2	85	31 Encrypted Alert	
192.168.86.30	15.197.206.217	TCP	54	0 63065 → 80 [FIN, A	
15.197.206.217	192.168.86.30	TCP	54	0 80 → 63065 [FIN, A	
192.168.86.30	34.171.47.125	TCP	54	0 63069 → 443 [FIN,	
34.171.47.125	192.168.86.30	TLSv1.2	85	31 Encrypted Alert	
34.171.47.125	192.168.86.30	TCP	54	0 [TCP Retransmissio	
34.171.47.125	192.168.86.30	TCP	54	0 [TCP Retransmissio	

> Frame 3120: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface \Device\NPF{...}

> Ethernet II, Src: Google_36:db:e9 (cc:f4:11:36:db:e9), Dst: Intel_1b:a2:e7 (ac:fd:ce:1b:a2:e7)

> Internet Protocol Version 4, Src: 104.254.151.36, Dst: 192.168.86.30

✓ Transmission Control Protocol, Src Port: 443, Dst Port: 63084, Seq: 4023, Ack: 2833, Len: 31

Source Port: 443

Destination Port: 63084

[Stream index: 95]

> [Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 31]

Sequence Number: 4023 (relative sequence number)

Sequence Number (raw): 3496706822

0020 56 1e 01 bb f6 6c d0 6b 83 06 31 fa 90 d6 50 19 V...l.k..1...P.

0030 9f b0 fe 9c 00 00 15 03 03 00 1a da be 8c ec e2

TCP Segment Len (tcp.len) | Packets: 7648 · Displayed: 101 (1.3%) · Dropped: 0 (0.0%) | Profile: Ibrana Choudhry

Type	Size
------	------