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)

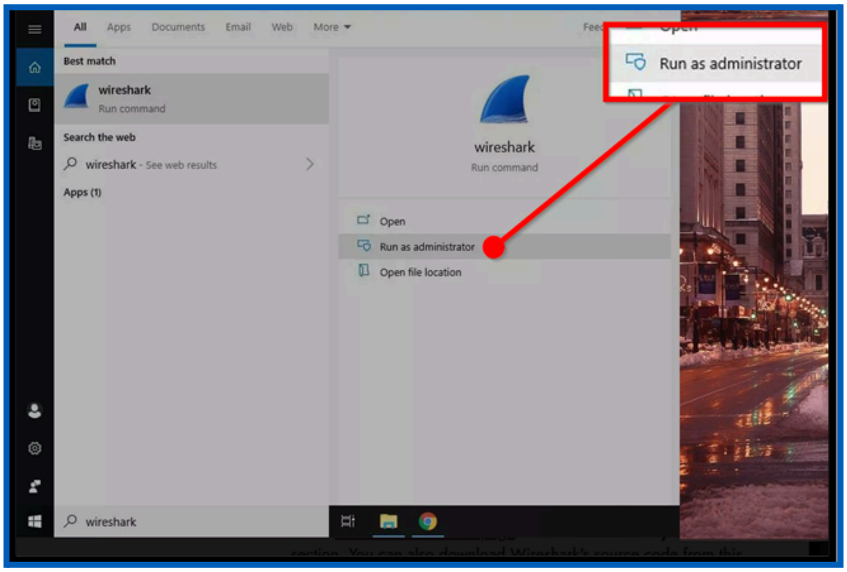
STEPS:

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:

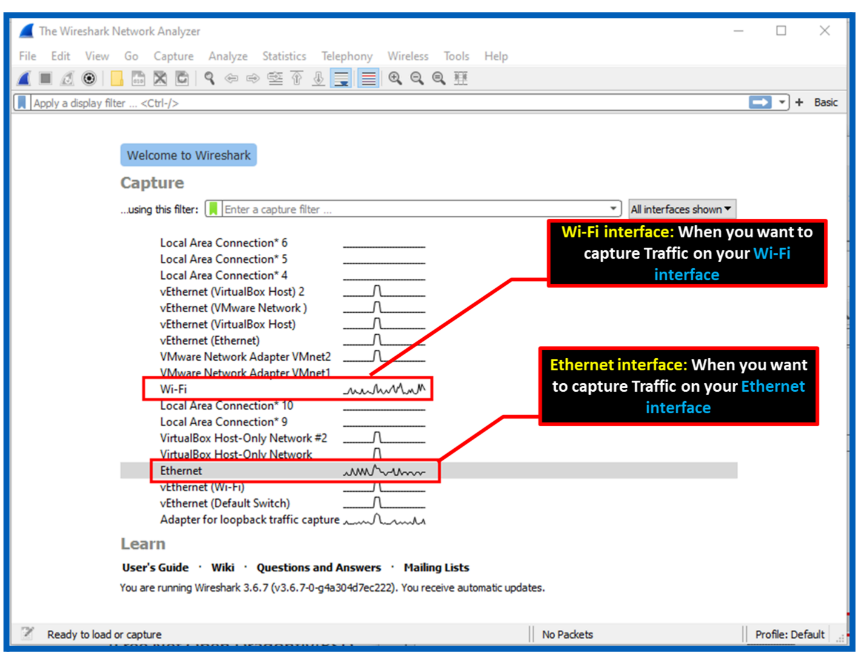
Run Wireshark as administrator as shown in the picture below.



Step 2: Start the Capturing::

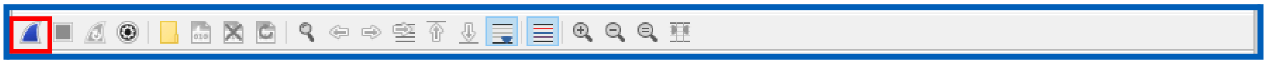
Wireshark can capture traffic from many different network media types, including Ethernet, Wireless LAN, Bluetooth, USB, and more. The specific media types supported may be limited by several factors, including your hardware and operating system (OS).

- 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

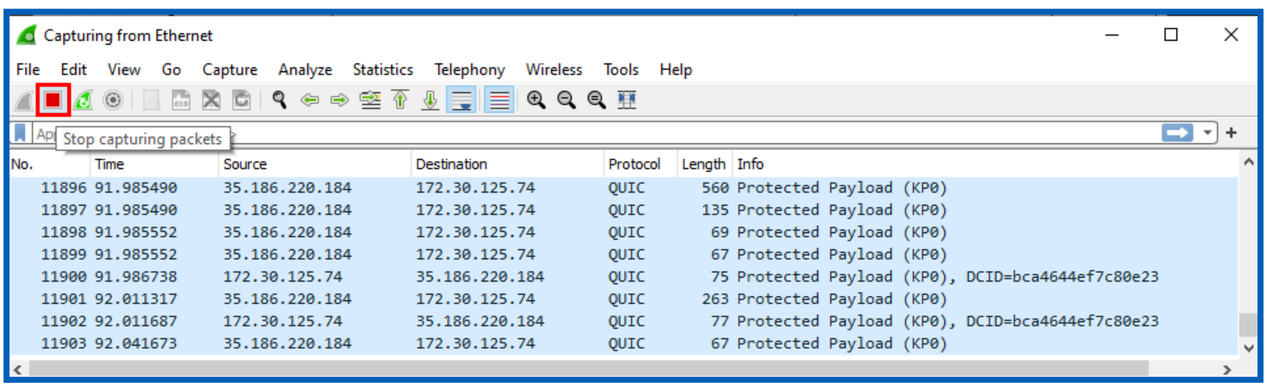
Open CMD and ping:

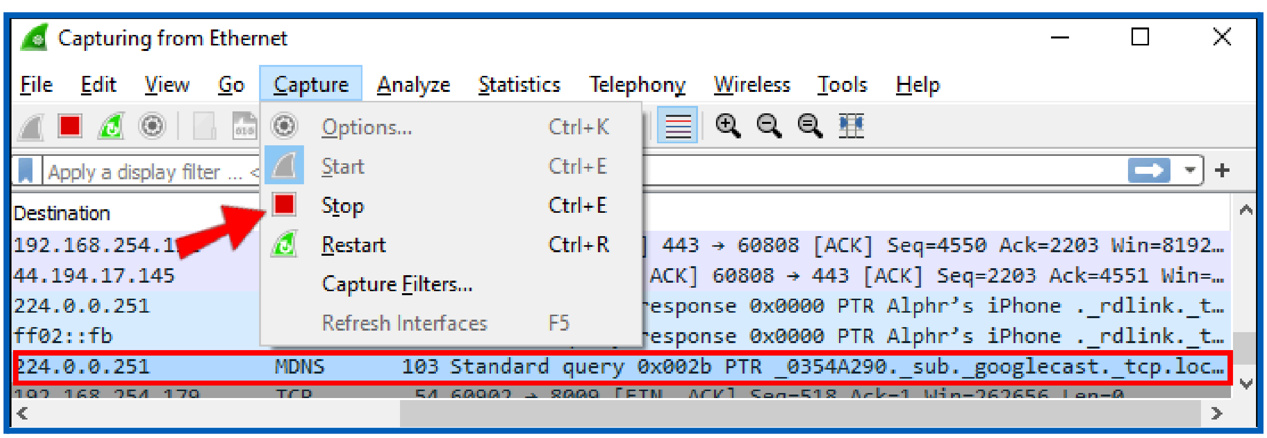
- 8.8.8.8
- Your default gateway
- 127.0.0.1

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**

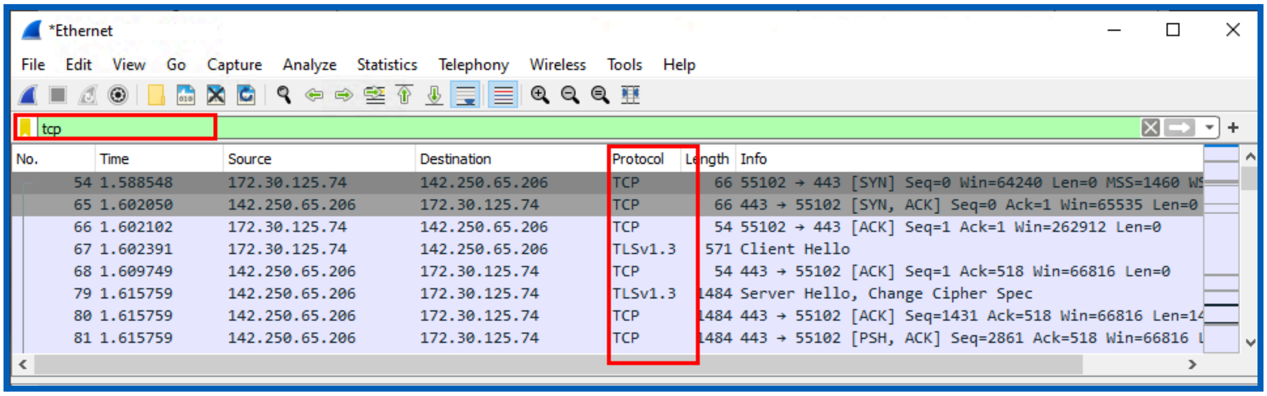




Part 2 - Add Coloring Rule for TCP packet:

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.

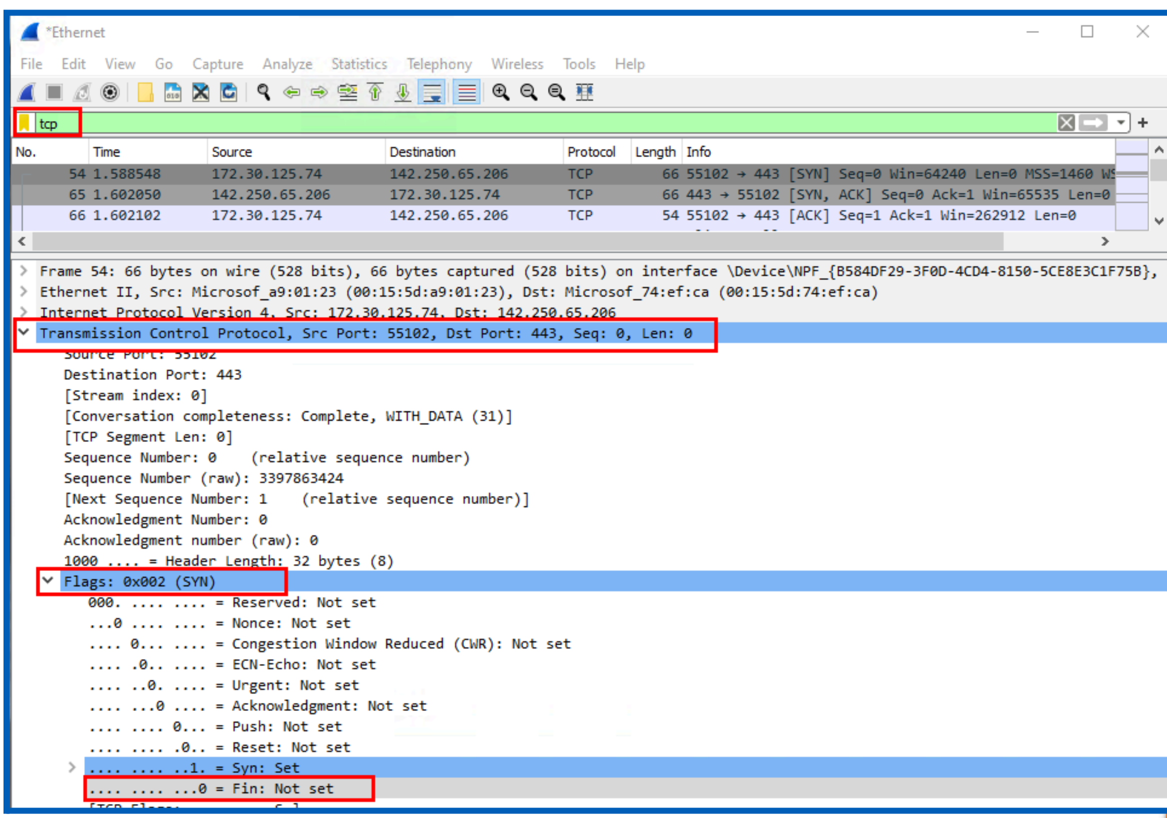


Reminder

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 for **....0 = Fin: Not set.**

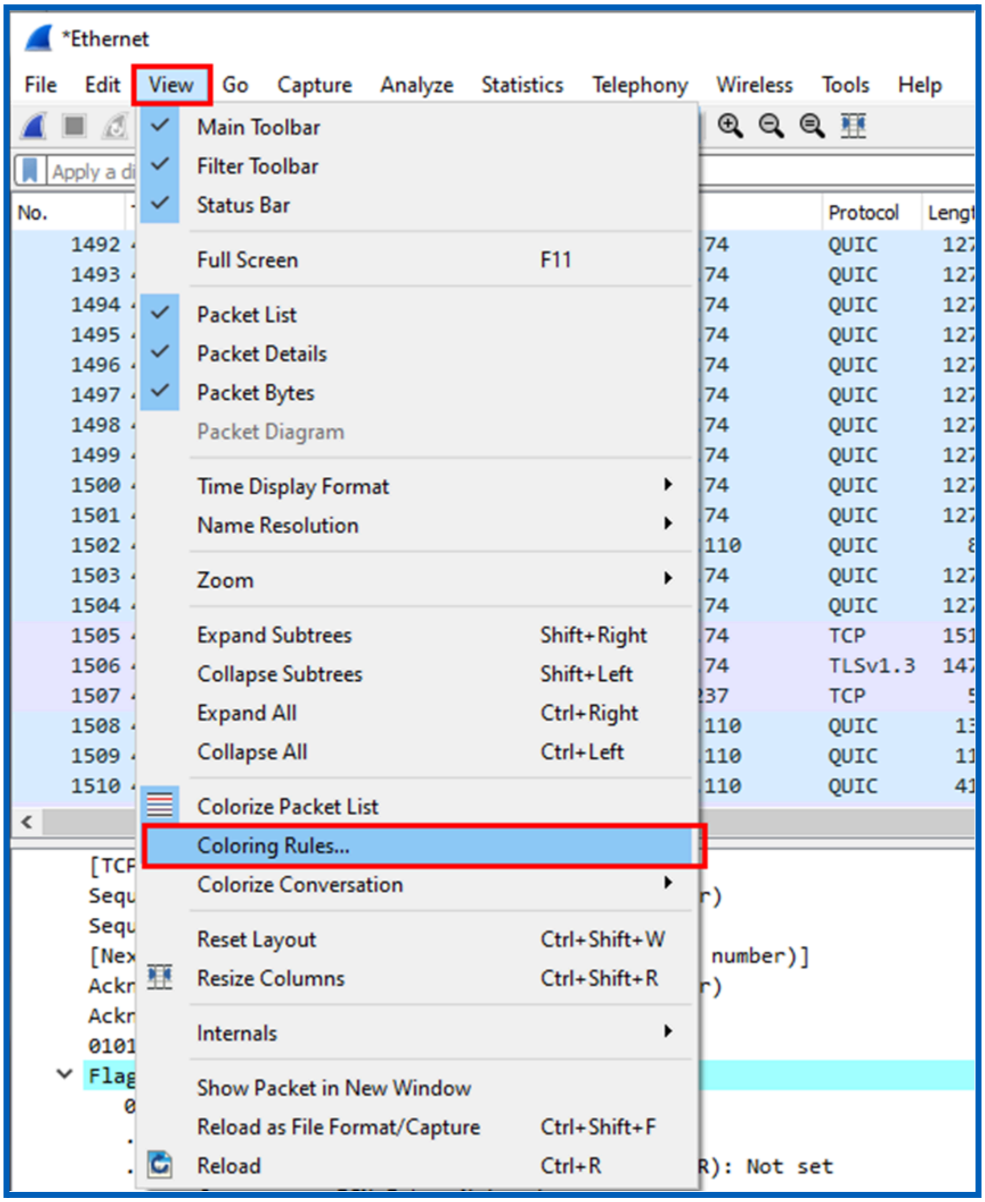


In the previous phase, we started with **TCP**, then **Flags**, and then **fin**, with **fin = 0,** and we want packets with fin = 1; thus, the filter we will apply is: **tcp.flags.fin == 1**

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:

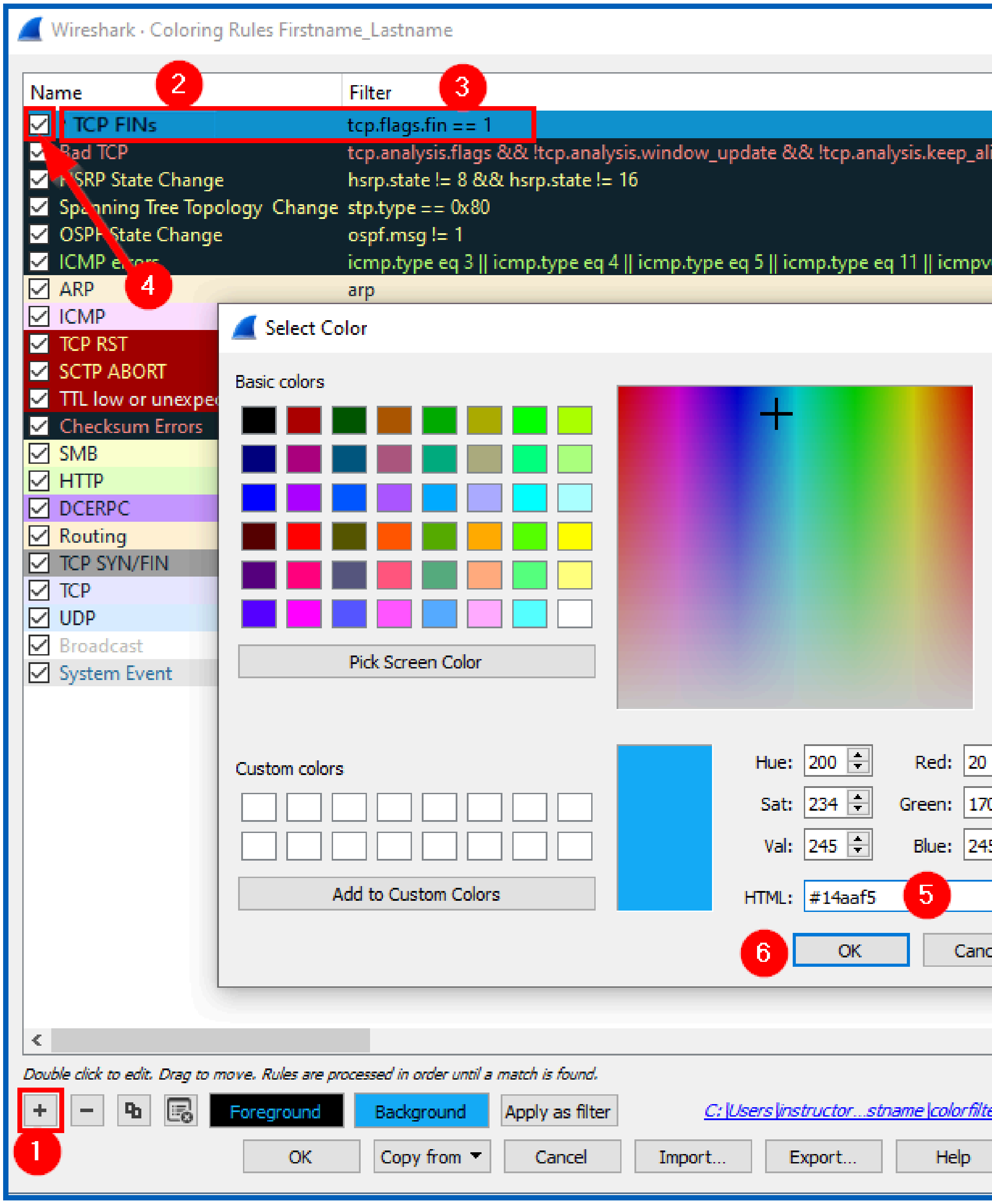
1) Copy the filter: tcp.flags.fin == 1.
2) Click on the View section on the menu tab, then Coloring Rules.



Wireshark

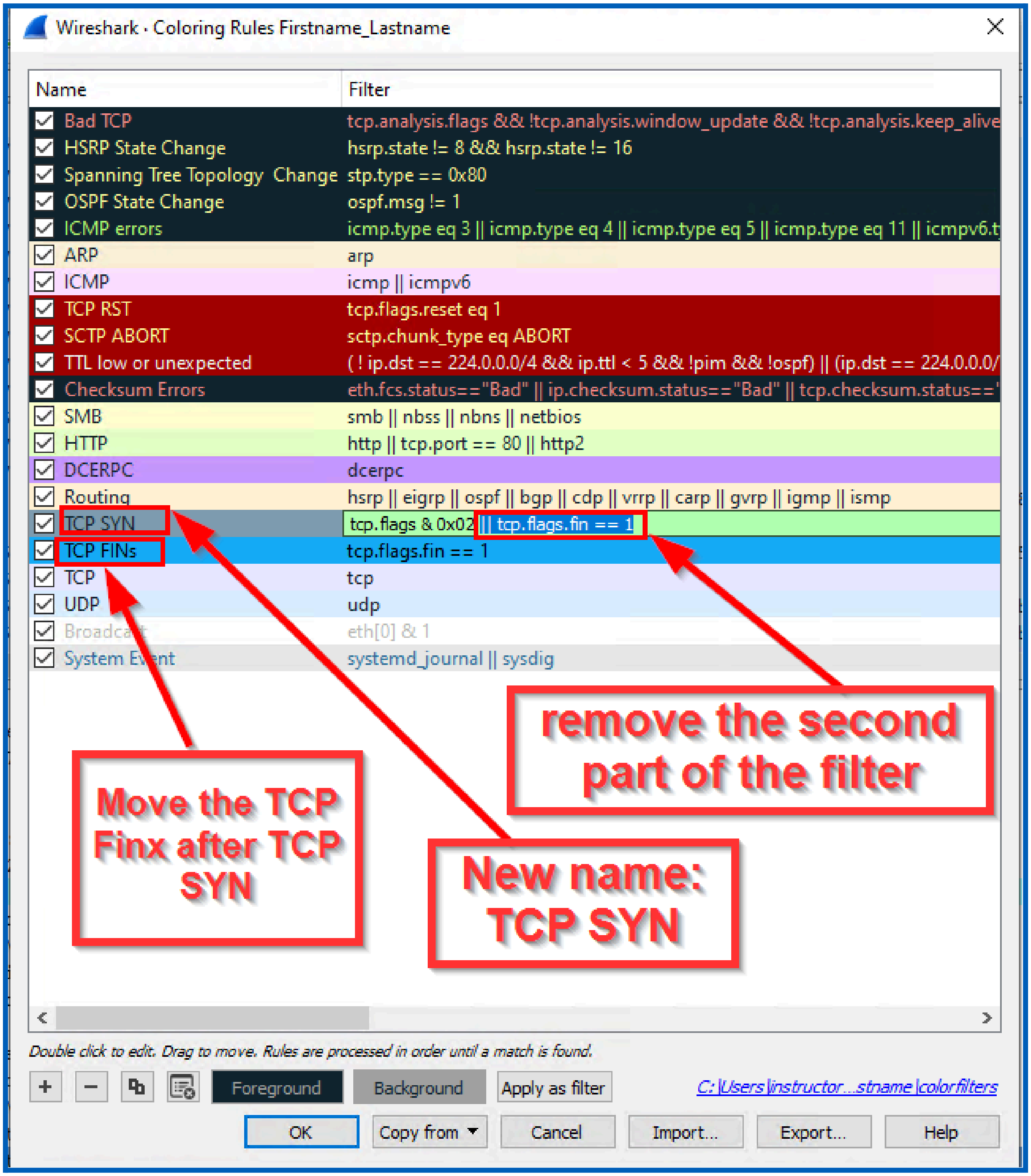
PER SCHOLAS ACADEMY

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).

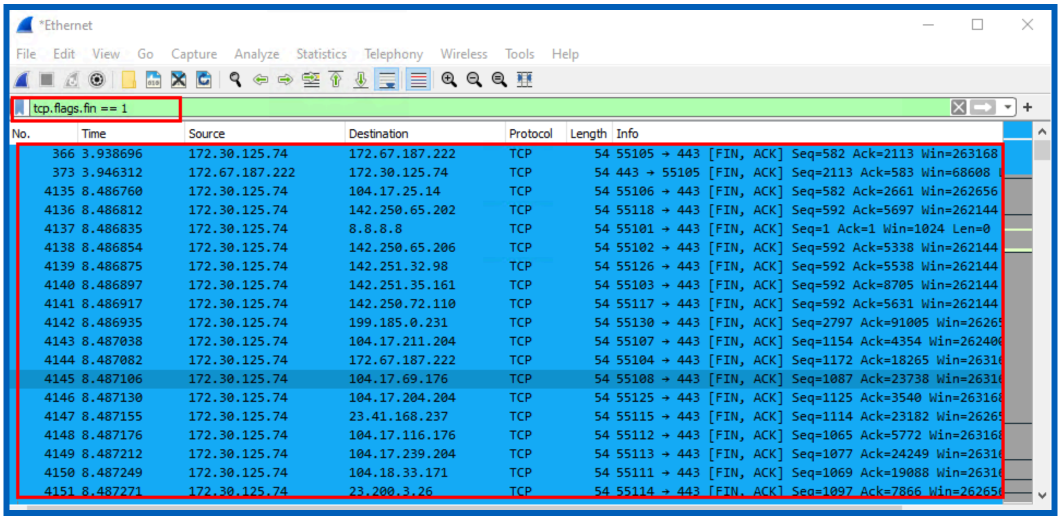


6) Click: **OK**

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.



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

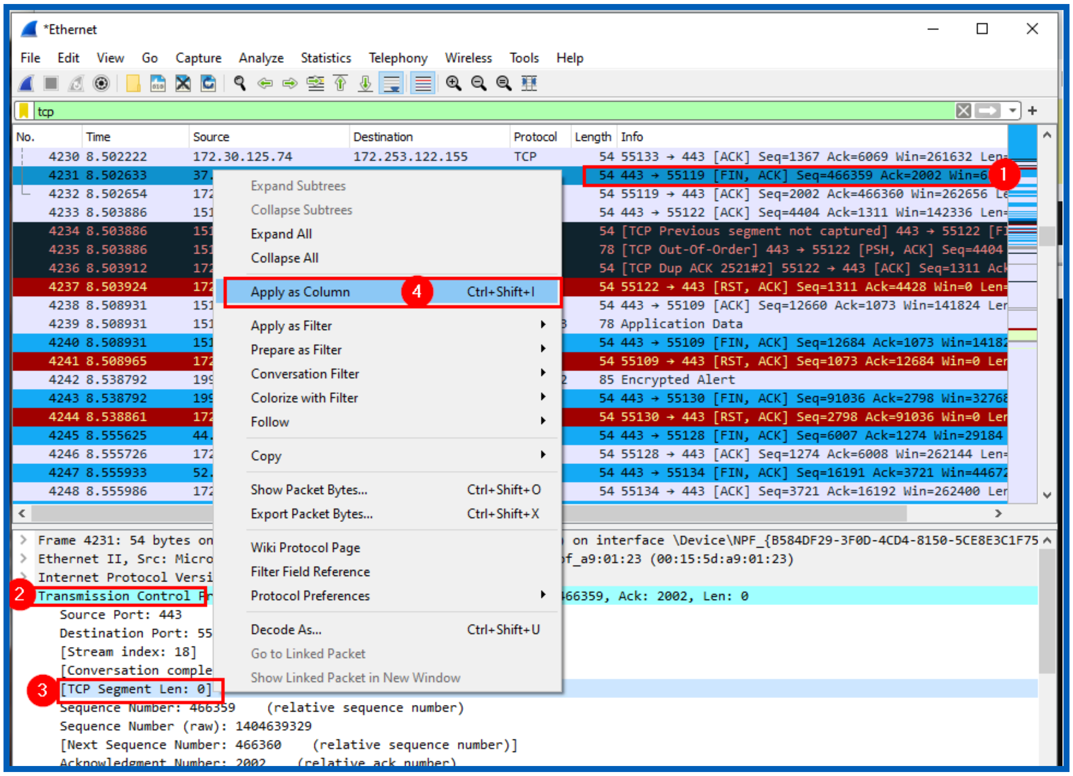


Part 3 - adding Custom Columns (TCP Segment Len):

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.

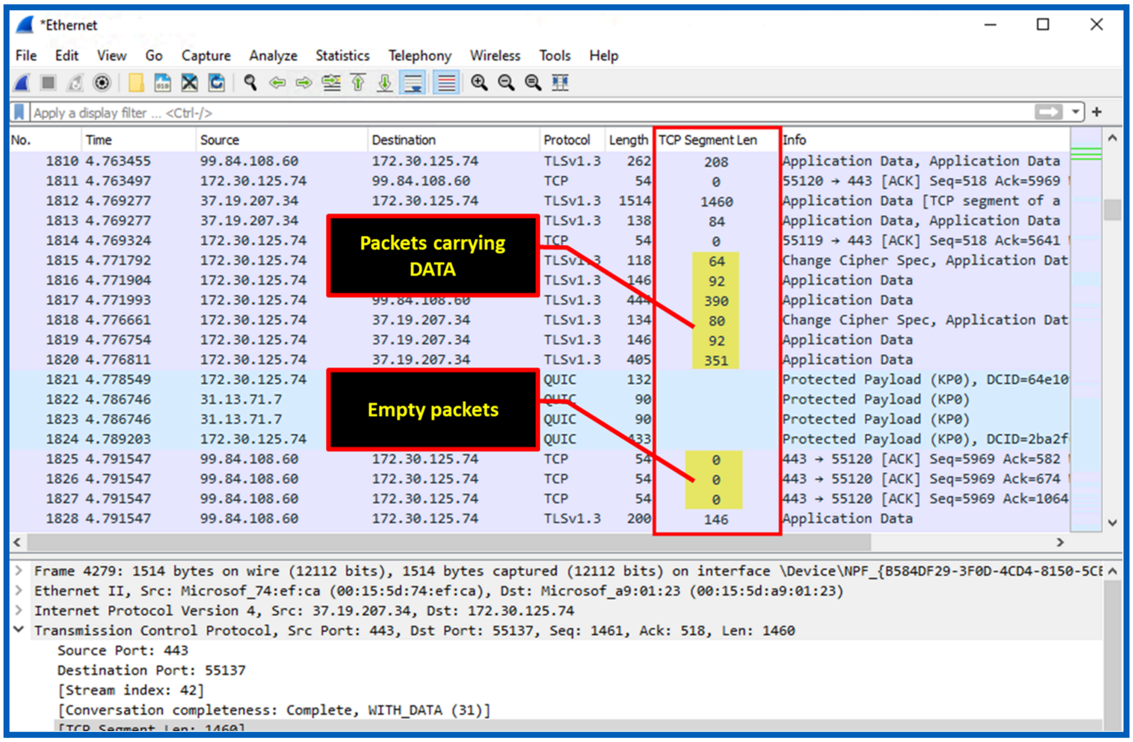
To add the **TCP Segment Len:**

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



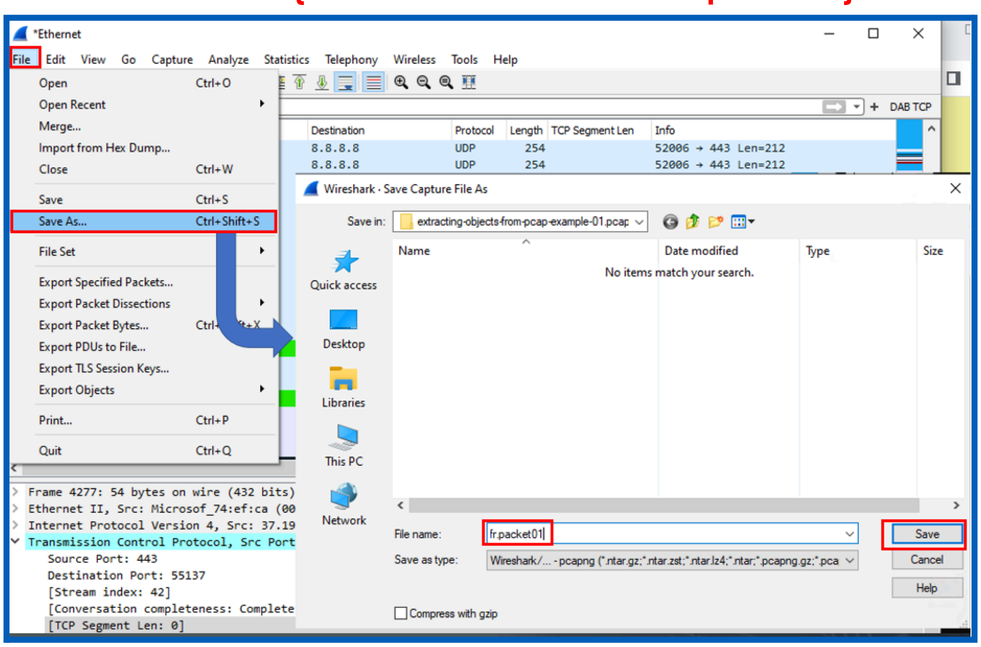
If a **TCP Segment Len = 0,** it means that the packet is empty.

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



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?*

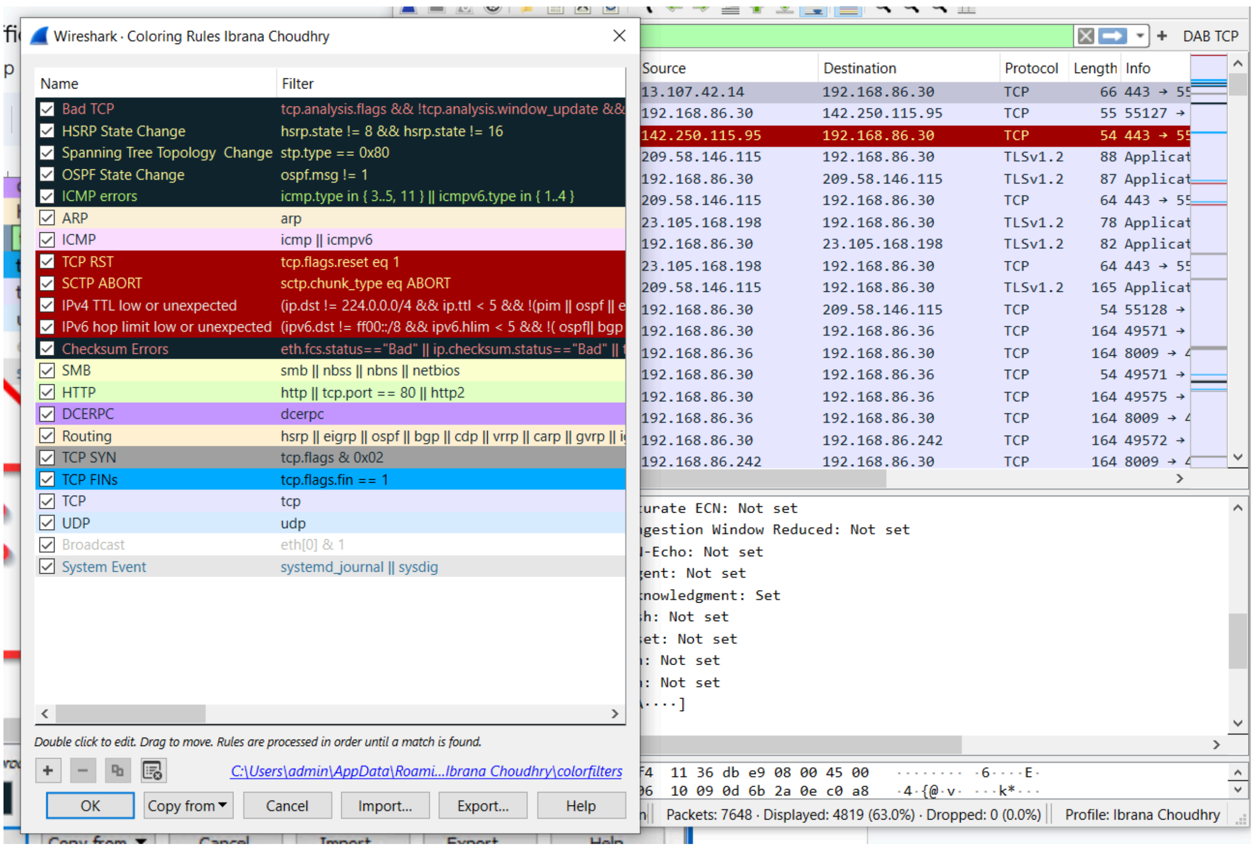
Capture filters only keep copies of packets that match the filter. Display filters are used when you've captured everything, but need to cut through the noise to analyze specific packets or flows. Capture filters and display filters are created using different syntaxes.

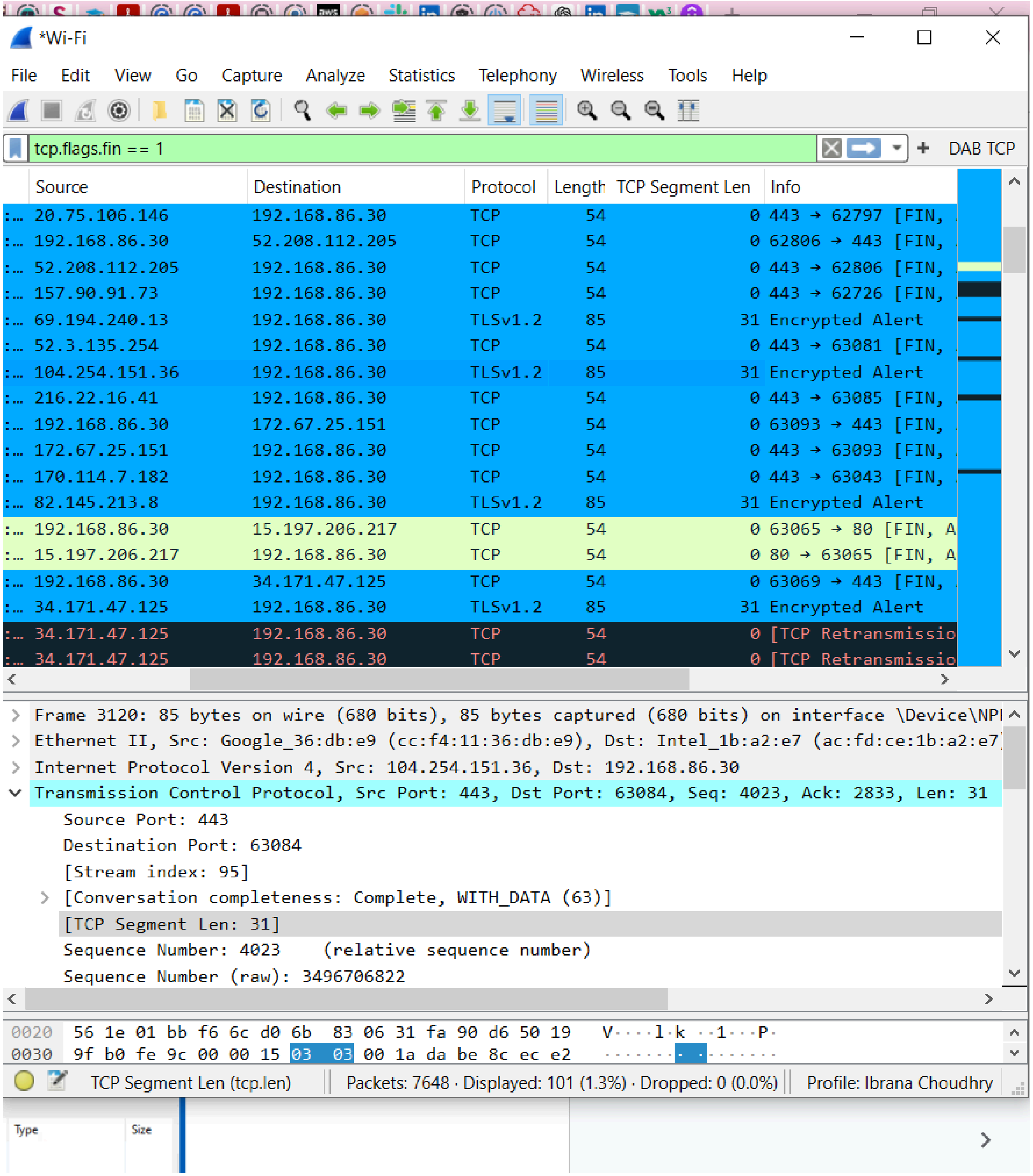
4. *Which tools are commonly used in packet capture and analysis?*

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.


Wireshark · Coloring Rules Ibrana Choudhry
Name
Filter
Bad TCP
tcp.analysis.flags && !tcp.analysis.window_update &&
HSRP State Change
hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Change
stp.type == 0x80
OSPF State Change
ospf.msg != 1
ICMP errors
icmp.type in { 3..5, 11 } || icmpv6.type in { 1..4 }
ARP
arp
ICMP
icmp || icmpv6
TCP RST
tcp.flags.reset eq 1
SCTP ABORT
sctp.chunk_type eq ABORT
IPv4 TTL low or unexpected
(ip.dst != 224.0.0.0/4 && ip.ttl < 5 && !(pim || ospf || e
IPv6 hop limit low or unexpected
(ipv6.dst != ff00::/8 && ipv6.hlim < 5 && !(ospf|| bgp
Checksum Errors
eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || t
SMB
smb || nbss || nbns || netbios
HTTP
http || tcp.port == 80 || http2
DCERPC
dcerpc
Routing
hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || i
TCP SYN
tcp.flags & 0x02
TCP FINs
tcp.flags.fin == 1
TCP
tcp
UDP
udp
Broadcast
eth[0] & 1
System Event
systemd_journal || sysdig
Double click to edit. Drag to move. Rules are processed in order until a match is found.
C:\Users\admin\AppData\Roami...Ibrana Choudhry\colorfilters
OK
Copy from
Cancel
Import...
Export...
Help
DAB TCP
Source
Destination
Protocol
Length
Info
Packets: 7648 · Displayed: 4819 (63.0%) · Dropped: 0 (0.0%)
Profile: Ibrana Choudhry



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.flags.fin == 1
DAB TCP
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\NPI
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