Using Wireshark for Network Monitoring

What is Wireshark?

Wireshark is an open-source packet capture, analyzer and monitoring tool, which is used for education, software development, communication protocol development, and network troubleshooting.
Here are some reasons people use Wireshark:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Where To Get Wireshark

You can get the latest copy of the Wireshark tool from the following link:
https://www.wireshark.org/download.html.

Choose the installer, as per your operating system, and download it. After downloading install it like any other Windows installation and follow the instructions, as prompted, to complete it. you will find a blue shark tale icon on the desktop. Click it to run.

Network Traffic Analysis using Wireshark

Network Traffic Analysis

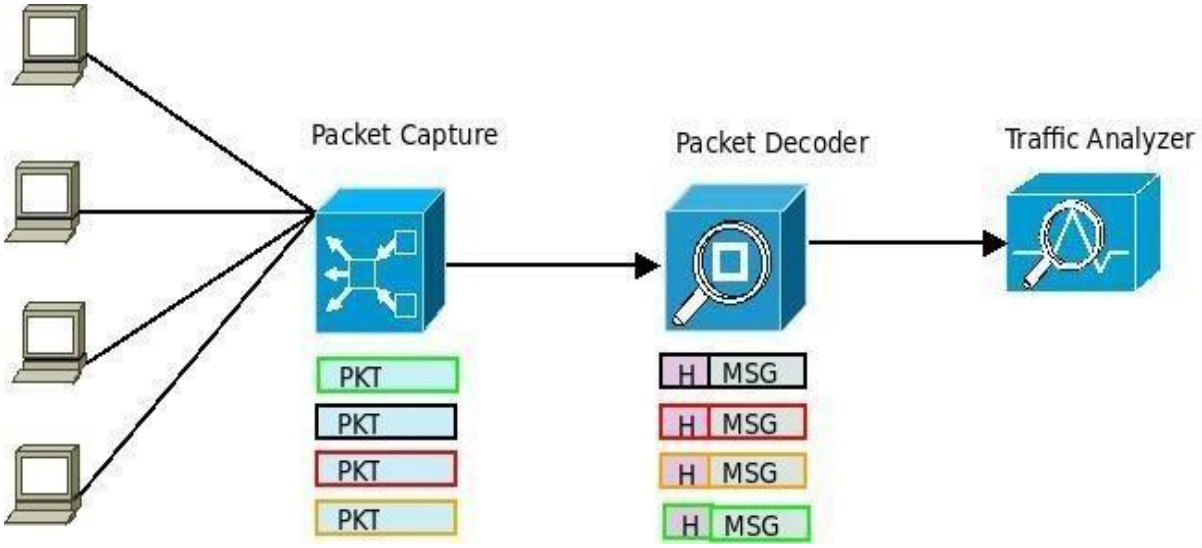
I. Network Traffic analysis is the process of capturing network traffic and inspecting it closely to determine what is happening on the network
II. Provides the details of network activities and their communication pattern in a network

The goal of Traffic Analysis

Network traffic analysis helps to

- Network monitoring
- Accounting and planning
- Performance analysis and improvement
- Security analysis

Wireshark has the following components to perform Traffic Capture, Decode and Analysis described Below.



Packet Capturing:

Wireshark can capture the packets passing through the host machine using this component. the packets can be captured in Normal Mode or Promiscuous Mode.

In "normal mode," a network device accepts only packets addressed to its own MAC Address while in Promiscuous Mode a network device can intercept and read each network packet that arrives from other machines and goes to some other machines also i.e. all the packet passing through its NIC .

Packet Decoder

Packet decoding is the process to

- Extract out the Header and message of the packet
- Identify the parameters for analysis for
    a. Communication pattern
    b. Monitoring
    c. Performance
    d. Protocol
    e. Application Analysis
    f. Host
    g. Location

Traffic Analyzer

It Identifies

- Who communicates with whom and When
- What types of messages
- How long are the messages
- Duration of communication

Input for Traffic Analysis

In Traffic analysis, the pattern of communication is more important than the content.

- Analysis is mainly based on packet header
- Traffic analysis can be done even in encrypted traffic

Most traffic analysis requires minimum information like

- Time and duration of a communication
- Details of the communication stream
- Identities of the communicating parties
- volume of data

Wireshark Features

Basic features of Wireshark –

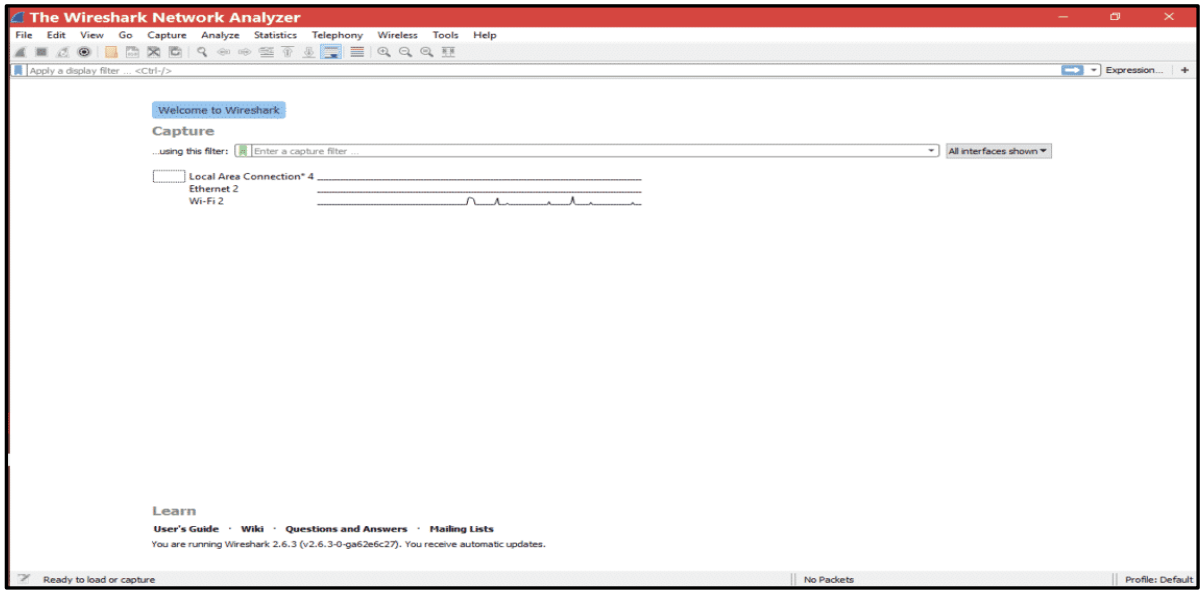
- Can listen on more than two interfaces simultaneously
- Packet coloring: Packets can be displayed in multiple ccolors for better analysis
- Filters
  - Capture filters: Capture packets based on some filter criteria
  - Display filters: Display only packets from the captured packets based on filter criteria.
- Finding packets: apply search to find a set of packets.

Advanced Features of Wireshark

1. Network endpoints and conversation
2. Protocol hierarchy statistics
3. Protocol dissection
4. Graphing

Wireshark Interface

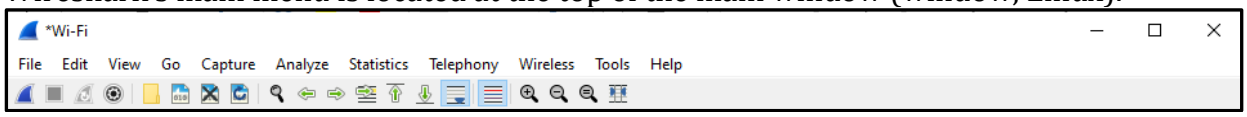
Following is the snapshot of Wireshark running in Windows10.



Let's see the interface components, one by one.

A. The Menu

Wireshark's main menu is located at the top of the main window (window, Linux).



The main menu contains the following Items:

File

This menu contains items to open and merge capture files, save, print, or export capture files in different Formats

Edit

This menu contains items to find a packet, time reference or mark one or more packets, handle configuration profiles, and set your preferences; (cut, copy, and paste are not presently implemented). The Wireshark Edit menu contains the fields as shown in the below image

View

This menu controls the display of the captured data, including colourization of packets, zooming the font, showing a packet in a separate window, expanding and collapsing trees in packet details.

Go

This menu contains items to go to a specific packet.

Capture

This menu allows you to start and stop captures and edit capture filters. Some of the important filters that make our capture more efficient are described below.

Analyze

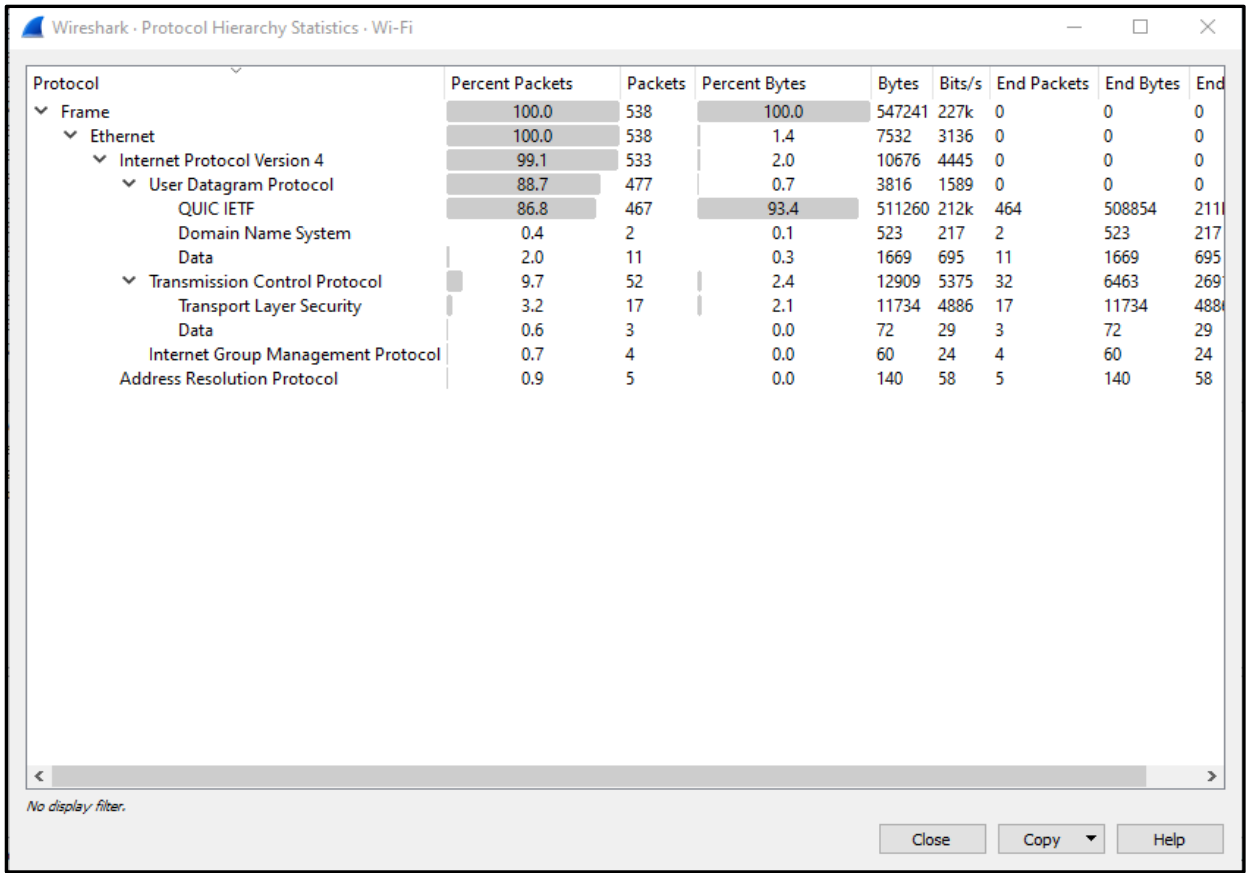
This menu contains items to manipulate display filters, enable or disable the dissection of protocols, configure user-specified decodes and follow a TCP stream.

Statistics

This menu contains items to display various statistic windows, including a summary of the packets that have been captured, a display protocol hierarchy statistic and much more. Some of the important filters that make our Trace analysis more efficient are described below.

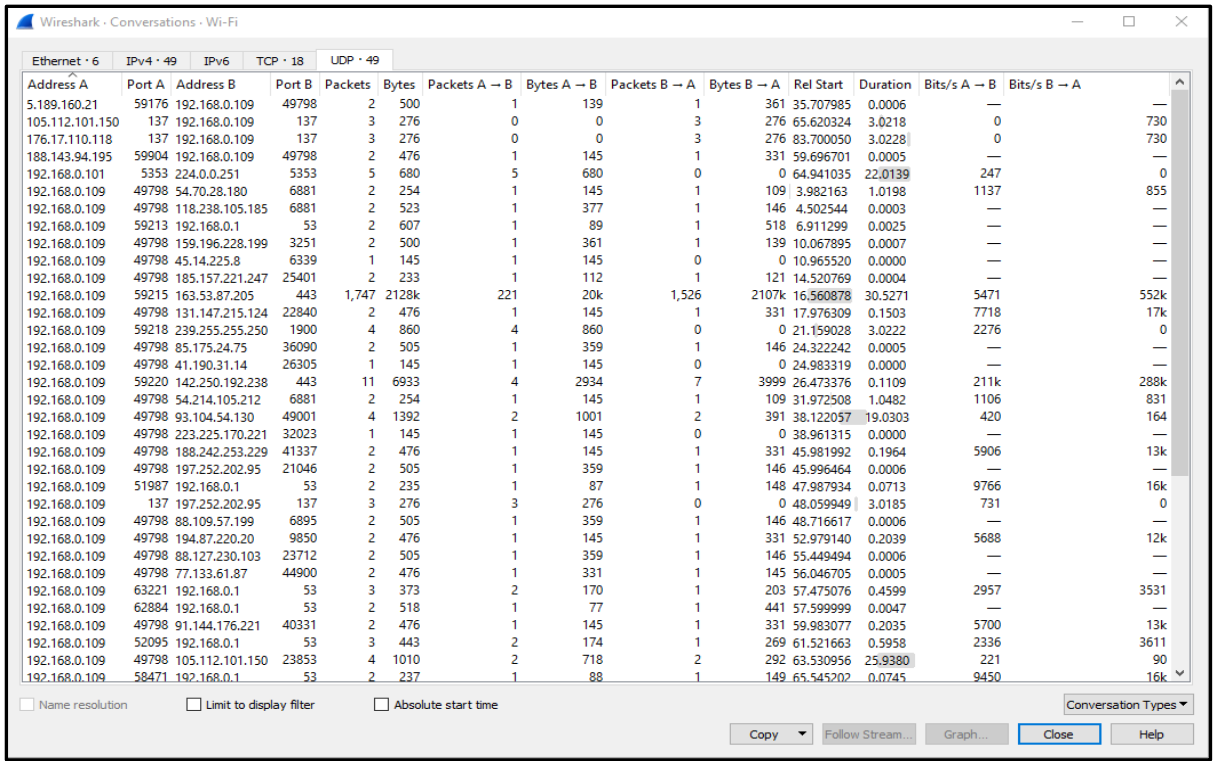
Statistics -> Protocol Hierarchy

- Presents descriptive statistics per protocol.
- Useful for determining the types, amounts, and relative proportions of protocols within a trace



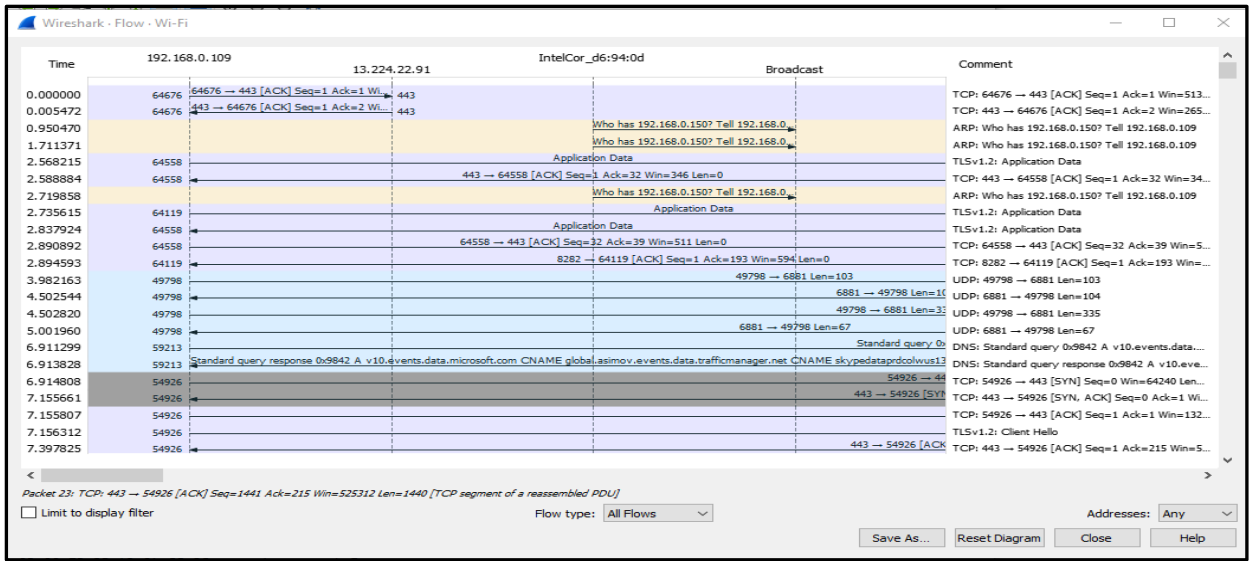
Statistics -> Conversations

- Generates descriptive statistics about each conversation for each protocol in the trace.



Statistics -> Flow Graph

- Generates a sequence graph for the selected traffic.
- Useful for understanding seq. and ack. calculations.



Telephony

This menu contains items to display various telephony related statistic windows, including a media analysis, flow diagrams, display protocol hierarchy statistics and much more.

Wireless

This menu contains items to display Bluetooth and IEEE 802.11 wireless statistics.

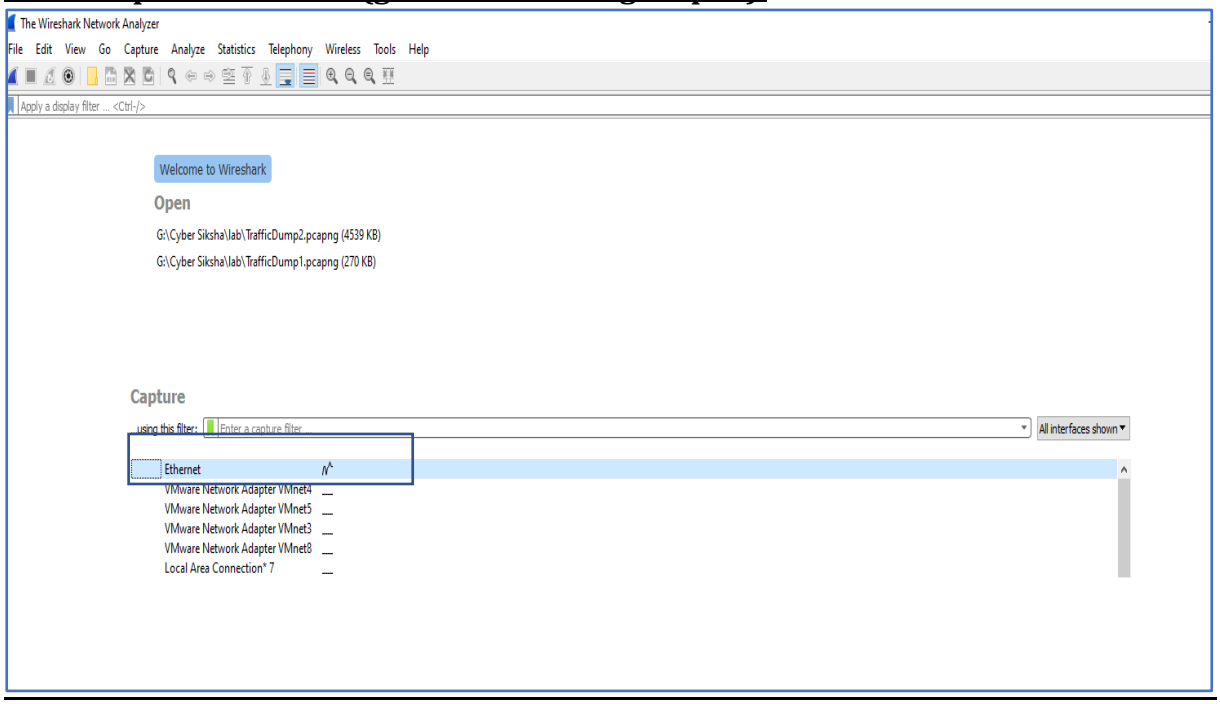
Tools

This menu contains various tools available in Wireshark, such as creating Firewall ACL Rules.

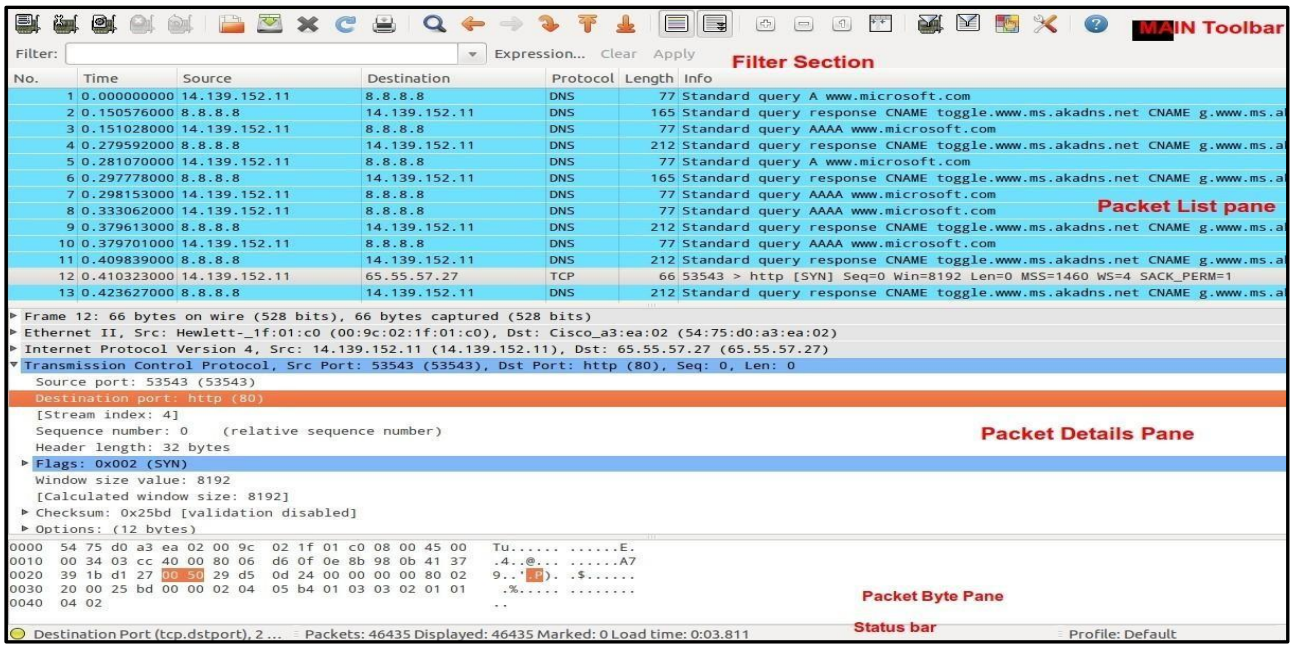
Help

This menu contains items to help the user, e.g. access to some basic help, manual pages of the various command-line tools, online access to some of the webpages, and the usual dialogue.

<u>When you click on the interface, to capture packets from, like Ethernet here. the traffic capture interface (given below will get open).</u>



The interface with a zigzag traffic pattern shows the active interface.



This interface can be broken into following parts:

A. Packet List pane

The packet list pane displays all the packets in the current capture file.
Each line in thepacket list corresponds to one packet in the capture file

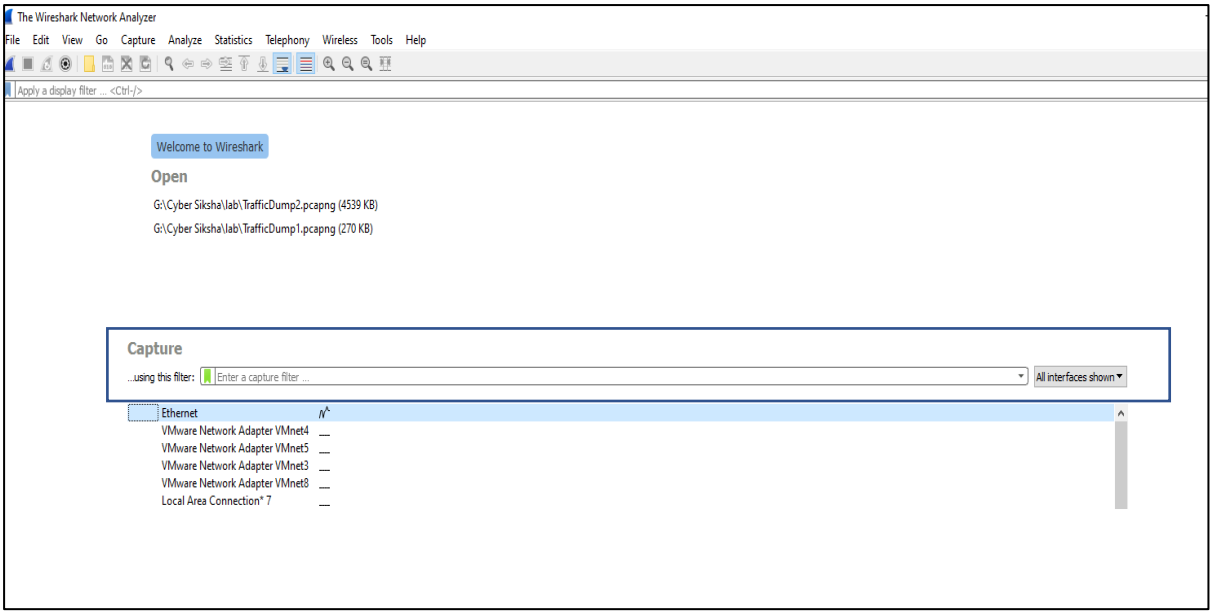
No	The number of the packet in the capture file
Timestamp	Timestamp of the packets
Source	Source address of the packet
Destination	Destination address of the Packet.
Info	Additional Information about the packet

B. Packet Details pane

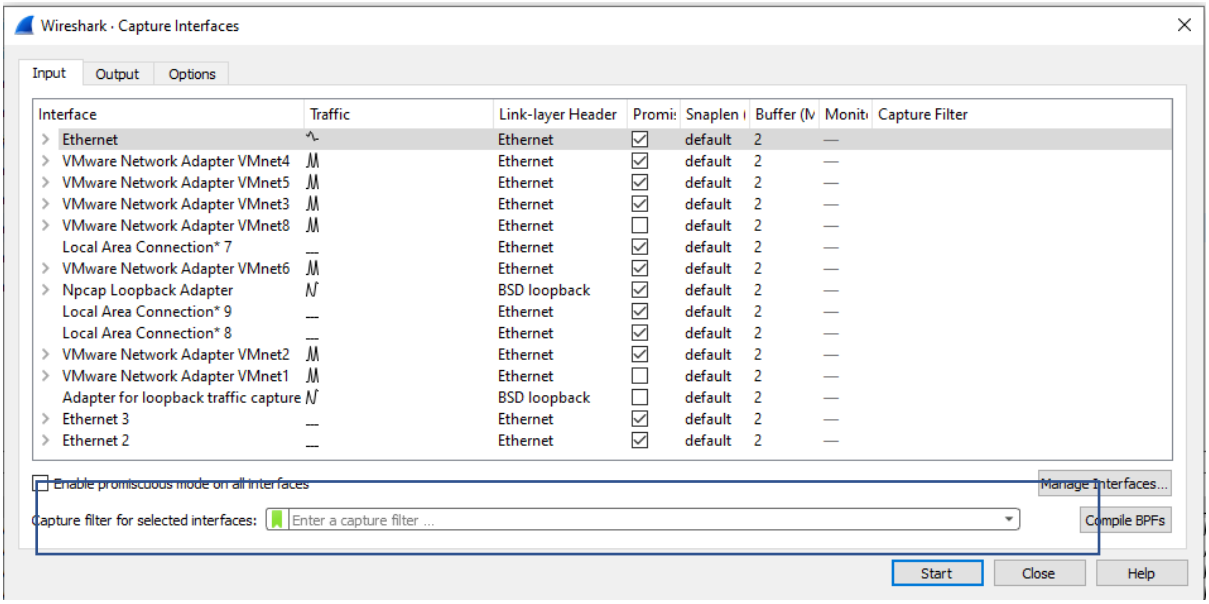
- Packet details pane shows the current packet (selected in the “Packet List” pane) in a more detailed form.
- This pane shows the protocols and protocol fields of the packet selected in the “Packet List” pane.
- The protocols and fields of the packet are displayed using a tree structure.

C. Capture Filter

A capture filter is used to select which packets should be saved to disk while capturing.
Capture filters works on BPF (Berkeley Packet Filtering) format.
You can set the capture filter either on the Wireshark home page or in the Capture-> Options page.



Home Page



The page after clicking on Capture Menu-> Options.

Capture Filter – Example

1. **host :** to capture from/to a particular host/IP

host 192.168.1.2
host www.google.com

2. **net :** to capture from/to a range of IPs

net 192.168.0.0/24

3. **src :** to capture traffic from a particular host/range of IPs

src host 192.168.1.1
src net 192.168.0.0/24
Examples:

4. **dst:** to capture traffic to a particular host/range ofIps dsthost 192.168.1.1

dst net 192.168.0.0/24

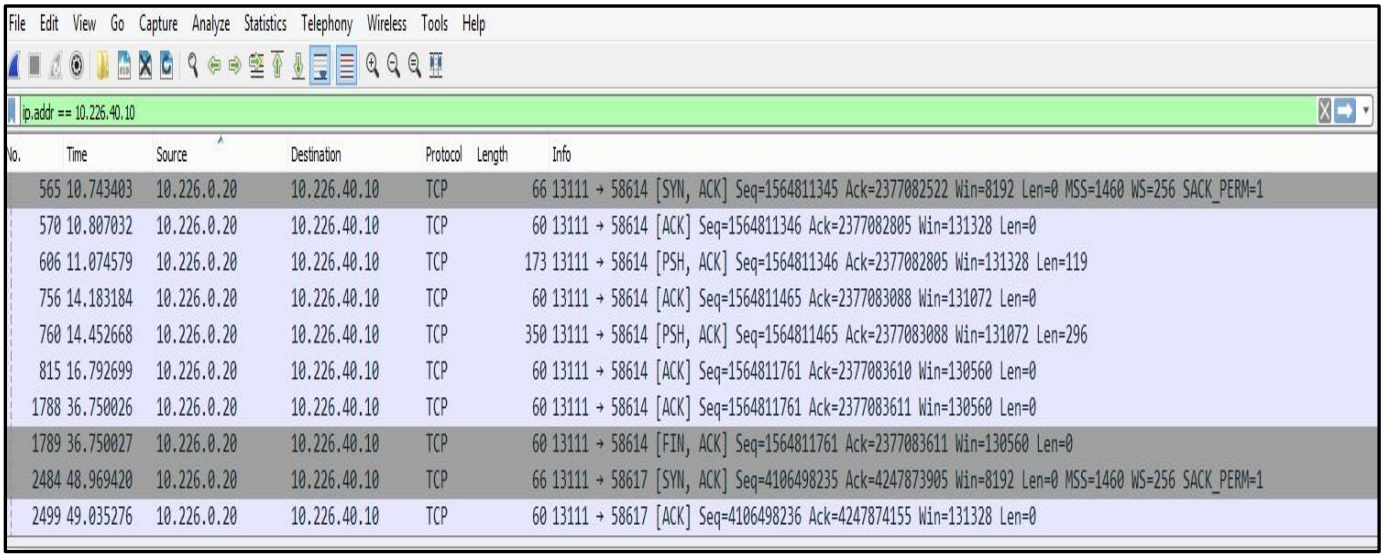
5. **port:** to capture traffic of a port
   port 22 , tcp port 443, udp port 53
6. **port range:** to capture traffic from a range of TCP ports TCP
   portrange 1500-2400
7. **ethernet:** to capture traffic from ethernet device

ether host 00:08:15:00:08:15

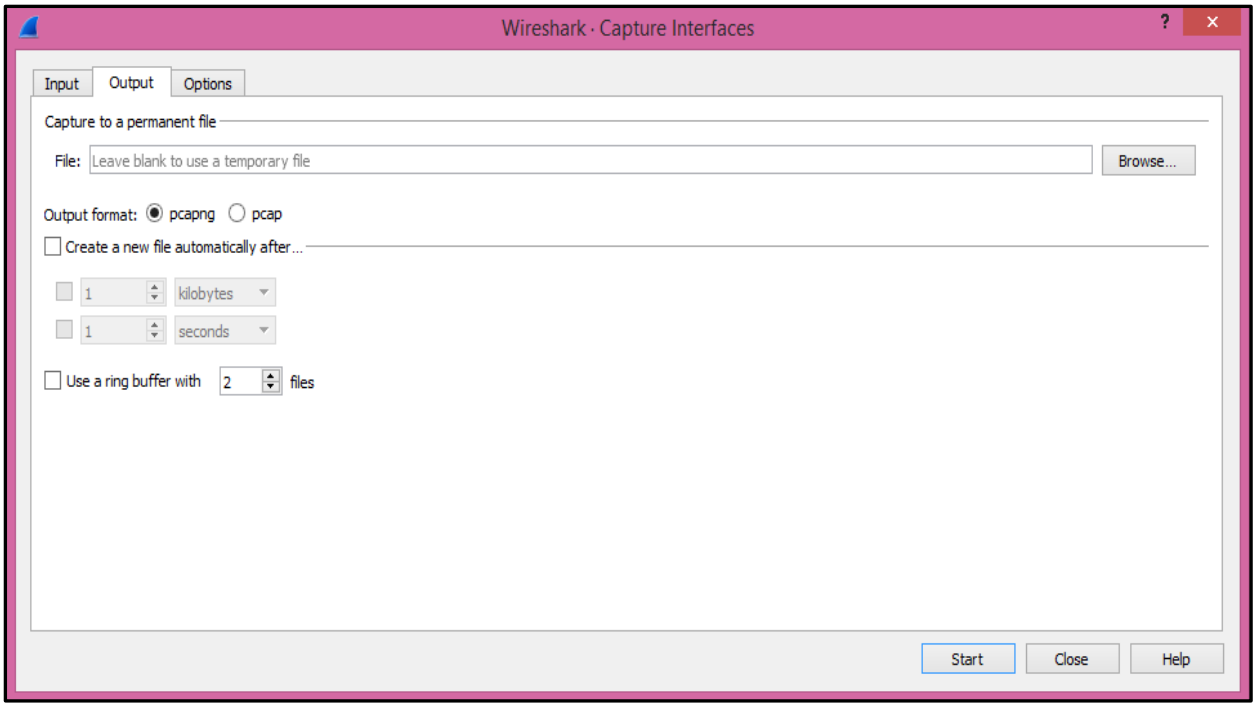
8. **IP :** to capture only IP traffic
   ip
9. **ARP:** to capture only ARP traffic
   arp
10. **not broadcast :**to filter all broadcast
11. **not multicast**: to filter all multicast

Here are some examples of using Wireshark

a. List down the network interfaces connected to your host

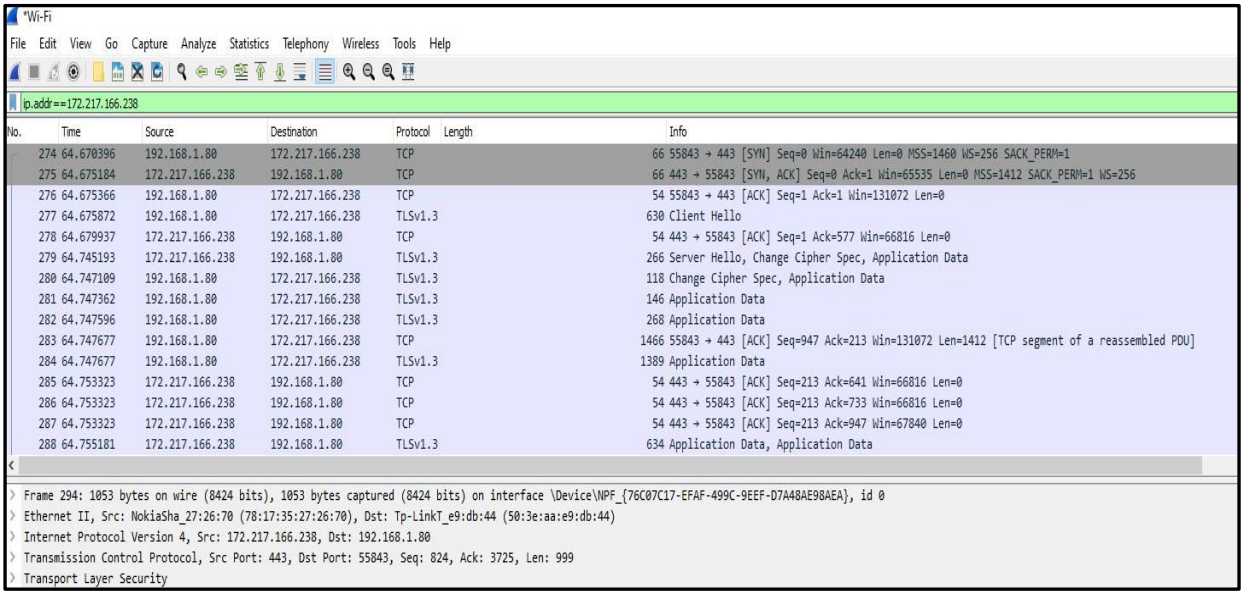


b. Configure the capture stop option of the wireshark



c. Capture live traffic from a site(for example google.com)

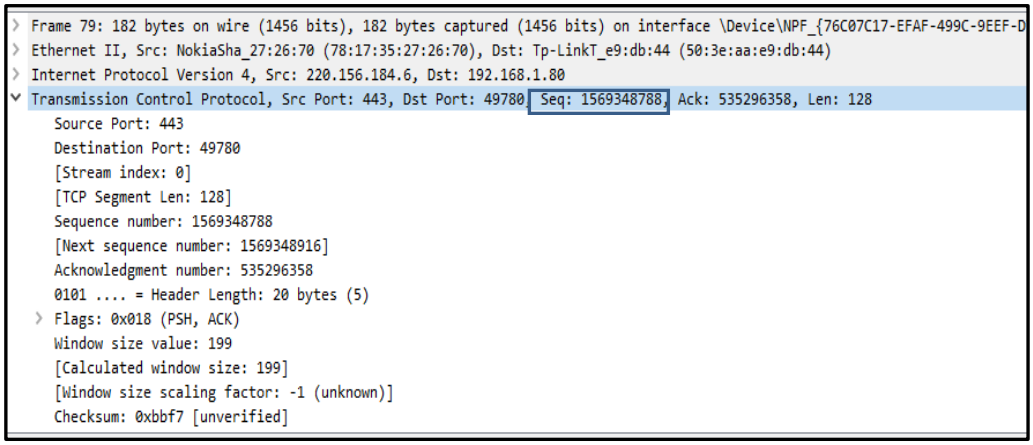
- Open browser clear cookies and history.
- Browse Google .com
- Ping google.com and get ip address.
- Set display filter ip.addr== ip of google.com



d. Finding Absolute packet number of a captured Packet

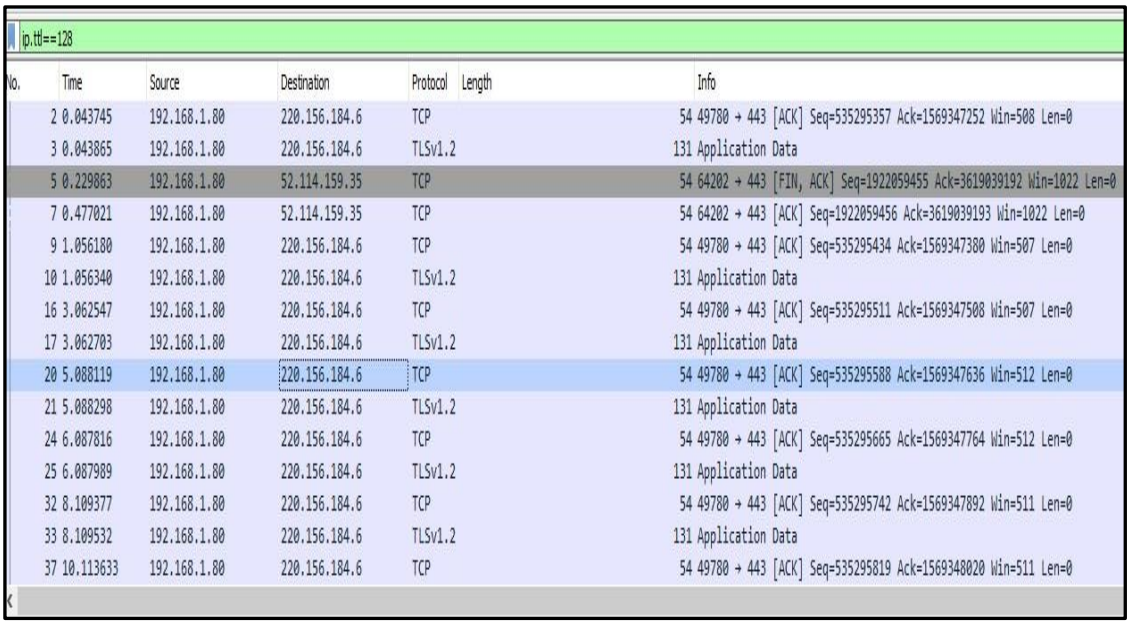
- Go to edit->preferences
- Expand protocol tab.
- Select tcp
- Uncheck “Relative sequence number” checkbox.

- The absolute sequence number will be seen in the packet detail.



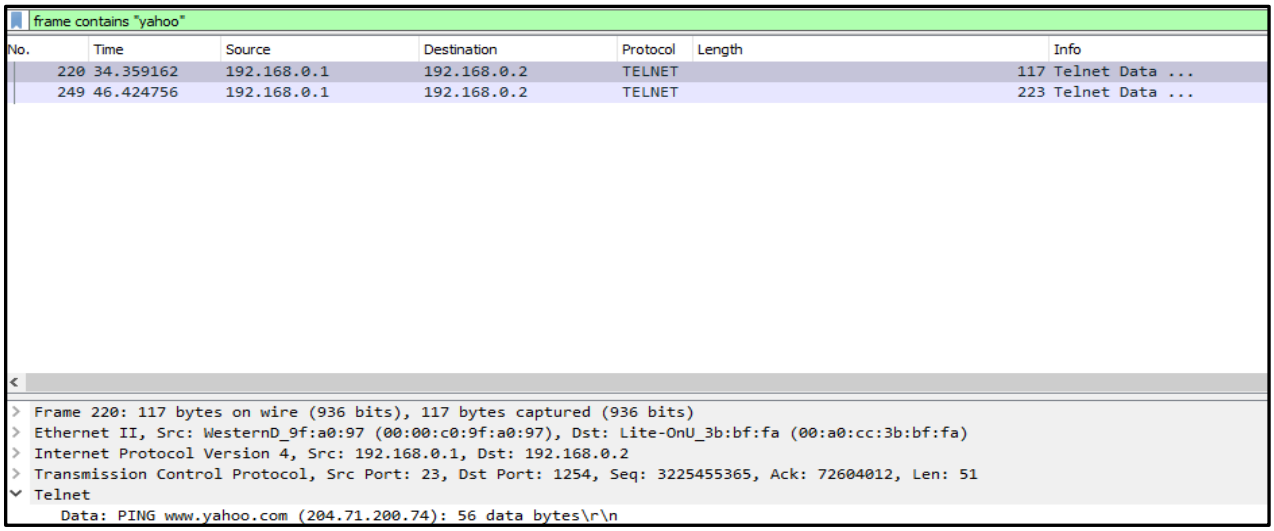
e. Finding packets with a particular ttl value

Set display filter **ip. ttl == 128**



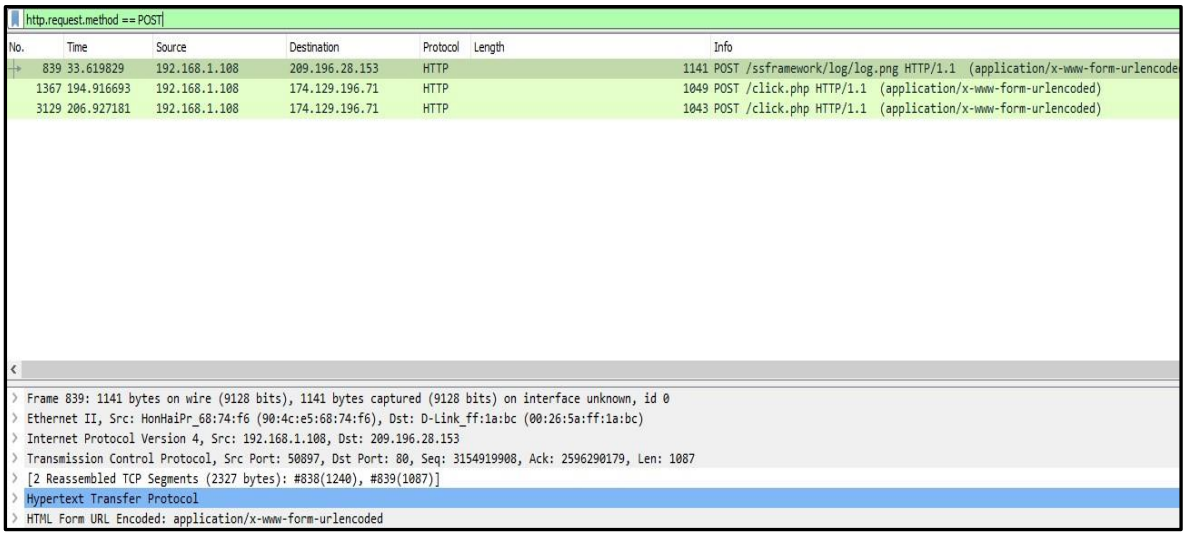
f. Finding a string in the capture

Use display filter frame containing “string”



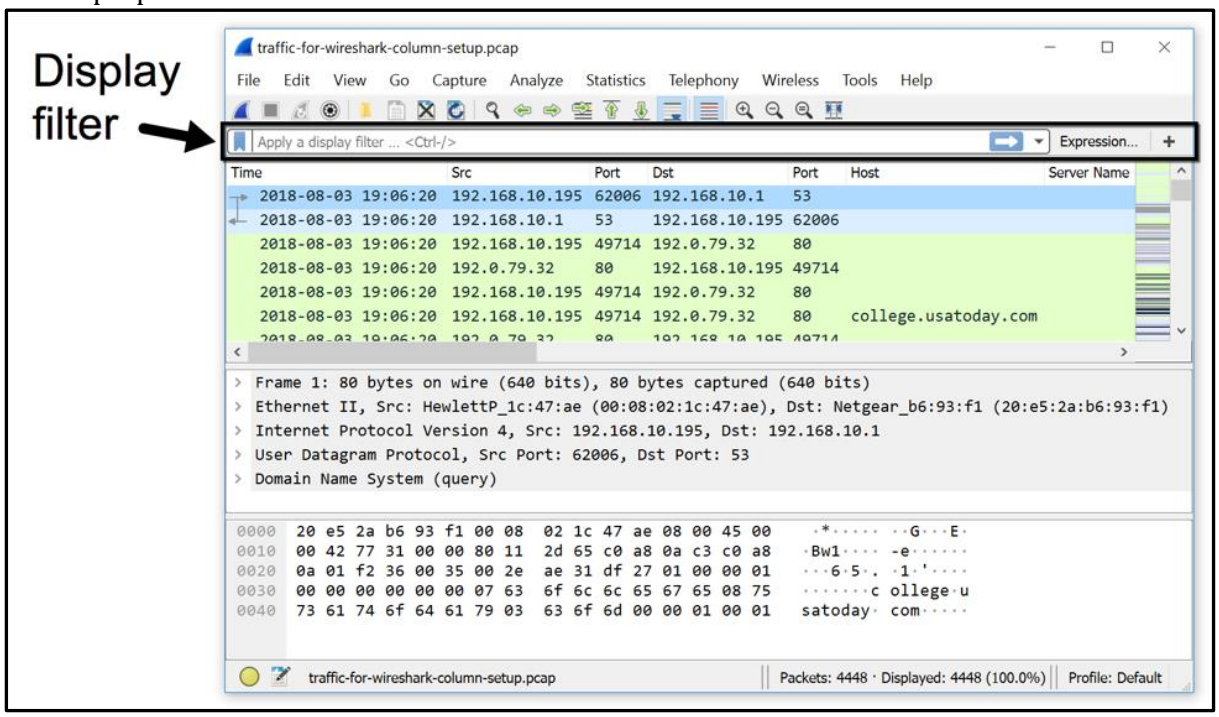
Searching for a http request method in the capture

Set display filter http. request. Method==POST



B. The Wireshark Display Filter

Wireshark's display filter is a bar located right above the column display section. This is where you type expressions to filter the frames, IP packets, or TCP segments that Wireshark displays from a pcap file.



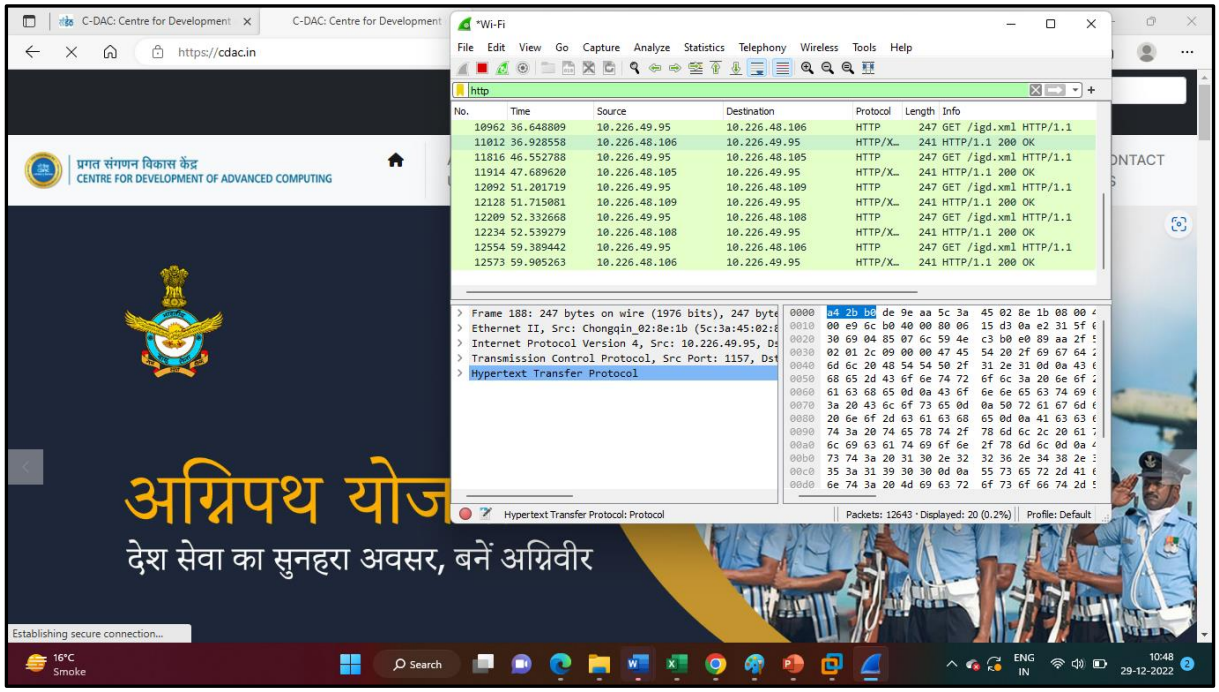
Below are some examples given for display filters.

a. Filter by Protocol

Its very easy to apply a filter for a particular protocol. Just write the name of that protocol in the filter tab and hit enter. In the example below we tried to filter the results for http protocol using this filter:

Launch your browser. In this instance, "Chrome" is open. Any web browser is an option.

The traffic will appear and the packet exchange will begin as soon as we open the browser and type any website address. This is depicted in the following image:



Using OR Condition in Filter

This filter helps filtering the packets that match either one or the other condition.
Suppose, there may arise a requirement to see packets that either have protocol 'http' or 'arp'. In that case one cannot apply separate filters. So there exists the '||' filter expression that ORs two conditions to display packets matching any or both the conditions. In the example below, we tried to filter the http or arp packets using this filter:

display filter : arp || http

i. IPv4 Protocol Filtering

Shows IP traffic (this includes TCP, UDP, as well as application layer protocols DNS, HTTP - that is, almost everything except the data link layer protocols that do not use IP addresses for data transmission (in local Ethernet networks they use MAC addresses)):

(i) Show traffic associated with a specific IP address (enter it instead of x.x.x.x). Packets will be shown in which this IP address is the source of the data OR the recipient:

ip.addr == x.x.x.x

(ii) Show traffic associated with these two IP addresses. According to the only possible situation, one of these addresses is the source, and the second is the destination address.

ip.addr == x.x.x.x && ip.addr == y.y.y.y

(iii) Shows traffic originated from the host with the IP address 138.201.81.199:

ip.src == 138.201.81.199

(iv) Shows traffic whose destination is the host with the IP address 138.201.81.199:

ip.dst == 138.201.81.199

Filter subnets and IP ranges in Wireshark

(v) You can specify a subnet instead of a single IP address:

ip.addr == 192.168.1.0/24

(vi) If you need to filter out traffic whose source is the subnet, then use a filter of the form:

ip.src == 192.168.1.0/24

(vii) If you need to filter traffic whose destination is a subnet, then use a filter of the form:

ip.dst == 192.168.1.0/24

To see only TCP traffic:

TCP

S.No.	Objective	Filter
1.	Show traffic whose source or destination port is a specific port, for example, 8080:	tcp.port==8080
2.	Show traffic originating from port 80:	tcp.srcport == 80
3.	Show the traffic that is sent to the service listening on port 80:	tcp.dstport == 80
4.	Show TCP packets with the SYN flag enabled:	tcp.flags.syn==1
5.	Show TCP packets with the SYN flag enabled and the ACK flag disabled:	tcp.flags.syn==1 && tcp.flags.ack==0
6.	Similarly for other flags:	
7.	ACK	tcp.flags.ack==1
8.	RST	tcp.flags.reset==1
9.	FIN	tcp.flags.fin==1
10.	CWR	tcp.flags.cwr==1
11.	ECE	tcp.flags.ecn==1
12.	URG	tcp.flags.urg==1
13.	PSH	tcp.flags.push==1
14.	NS	tcp.flags.ns==1

Application layer traffic

For the application protocols of HTTP, DNS, SSH, FTP, SMTP, RDP, SNMP, RTSP, GQUIC, CDP, LLMNR, SSDP there are filters that are called like the protocols themselves, but are written in small letters.

For example, to see HTTP traffic:

1http

To see the traffic of the new HTTP/2 protocol:

1http2

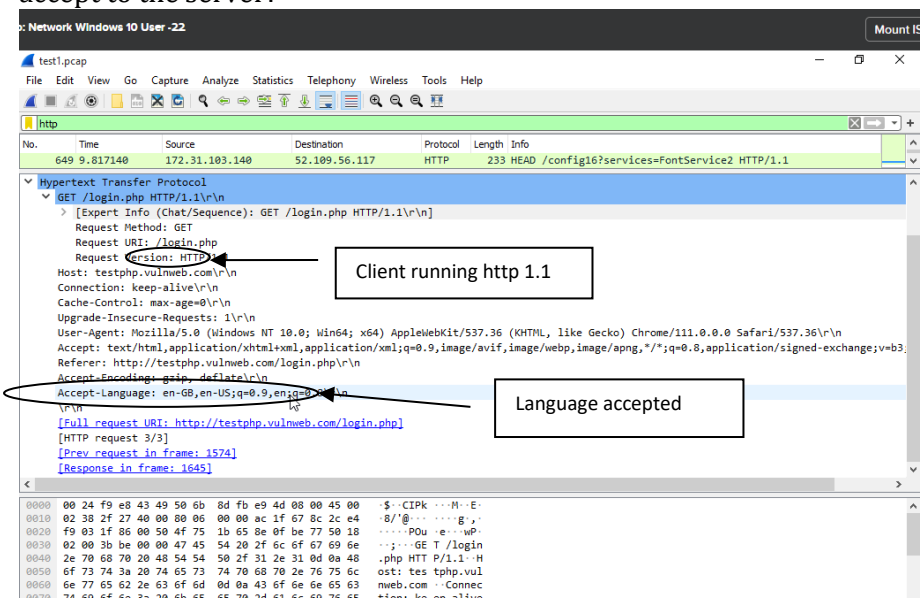
Remember that when deciding which protocol the transmitted data belongs to, the program considers the used port number. If a non-standard port is used, the program will not be able to find the necessary data. For example, if you connect to SSH on port 1234, the **ssh** filter will not find SSH traffic.

S.No.	Objective	Filter
1.	A filter that shows only the data sent by the POST method:	http.request.method == "POST"
	A filter that shows only the data transmitted by the GET method:	http.request.method == "GET"
	Search for requests to a specific site (host):	http.host == "<URL>"
	Search requests to a specific site by part of the name:	http.host contains "here.particle.name"
	Filter for outputting HTTP requests in which cookies were transmitted:	http.cookie
	Requests in which the server has set cookies in the user's browser.	http.set_cookie
	To search for any transferred images:	http.content_type contains "image"
	To search for certain types of images:	1. http.content_type contains "gif" 2. http.content_type contains "jpeg" 3. http.content_type contains "png"
	To search for files of a specific type:	1. http.content_type contains "text" 2. http.content_type contains "xml" 3. http.content_type contains "html" 4. http.content_type contains "json" 5. http.content_type contains "javascript" 6. http.content_type contains "x-www-form-urlencode"

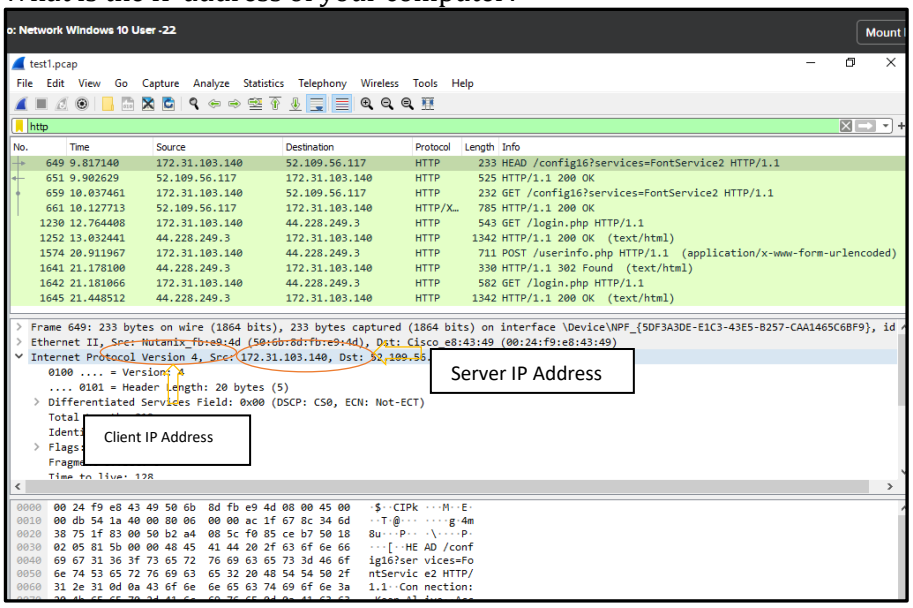
		7. http.content_type contains "compressed" 8. http.content_type contains "application"
	Search for requests for files of a certain type. For example, to search for transferred ZIP archives:	http.request.uri contains "zip"

Lab Assignment

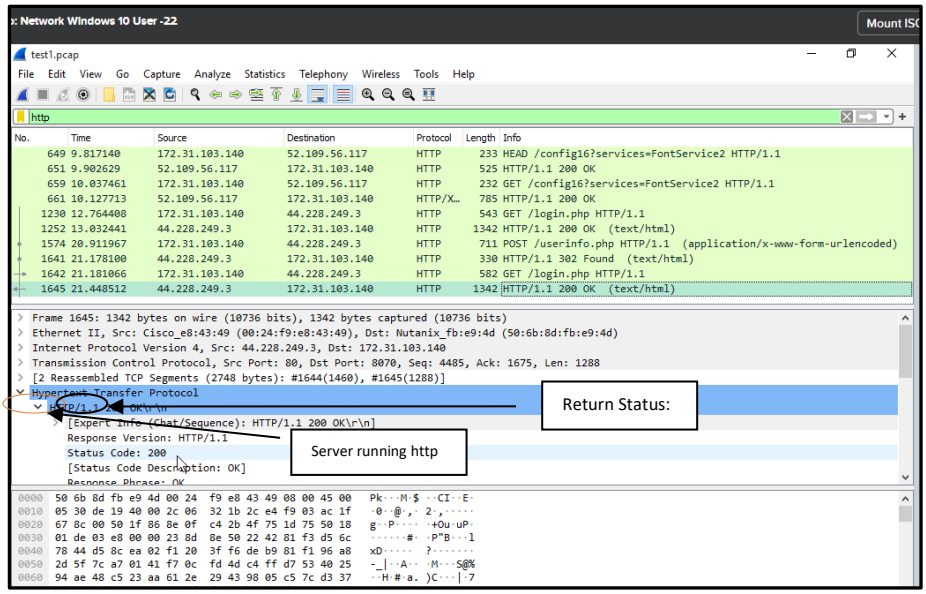
1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running? What languages (if any) does your browser indicate that it can accept to the server?



2. What is the IP address of your computer?



3. What is the status code returned from the server to your browser?


Network Windows 10 User -22
Mount ISO
test1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
649 9.817140 172.31.103.140 52.109.56.117 HTTP 233 HEAD /config16?services=FontService2 HTTP/1.1
651 9.902629 52.109.56.117 172.31.103.140 HTTP 525 HTTP/1.1 200 OK
659 10.037461 172.31.103.140 52.109.56.117 HTTP 232 GET /config16?services=FontService2 HTTP/1.1
661 10.127713 52.109.56.117 172.31.103.140 HTTP/X... 785 HTTP/1.1 200 OK
1230 12.764408 172.31.103.140 44.228.249.3 HTTP 543 GET /login.php HTTP/1.1
1252 13.032441 44.228.249.3 172.31.103.140 HTTP 1342 HTTP/1.1 200 OK (text/html)
1574 20.911967 172.31.103.140 44.228.249.3 HTTP 711 POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
1641 21.178100 44.228.249.3 172.31.103.140 HTTP 330 HTTP/1.1 302 Found (text/html)
1642 21.181066 172.31.103.140 44.228.249.3 HTTP 582 GET /login.php HTTP/1.1
1645 21.448512 44.228.249.3 172.31.103.140 HTTP 1342 HTTP/1.1 200 OK (text/html)
Frame 1645: 1342 bytes on wire (10736 bits), 1342 bytes captured (10736 bits)
Ethernet II, Src: Cisco_e8:43:49 (00:24:f9:e8:43:49), Dst: Nutanix_fb:e9:4d (50:6b:8d:fb:e9:4d)
Internet Protocol Version 4, Src: 44.228.249.3, Dst: 172.31.103.140
Transmission Control Protocol, Src Port: 80, Dst Port: 8070, Seq: 4485, Ack: 1675, Len: 1288
[2 Reassembled TCP Segments (2748 bytes): #1644(1460), #1645(1288)]
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Return Status:
Server running http
0000 50 6b 8d fb e9 4d 00 24 f9 e8 43 49 08 00 45 00 Pk···M·$ ··CI··E·
0010 05 30 de 19 40 00 2c 06 32 1b 2c e4 f9 03 ac 1f ·0··@·,· 2·,·····
0020 67 8c 00 50 1f 86 8e 0f c4 2b 4f 75 1d 75 50 18 g··P···· ·+Ou·uP·
0030 01 de 03 e8 00 00 23 8d 8e 50 22 42 81 f3 d5 6c ·····#· ·P"B···l
0040 78 44 d5 8c ea 02 f1 20 3f f6 de b9 81 f1 96 a8 xD····· ?·····
0050 2d 5f 7c a7 01 41 f7 0c fd 4d c4 ff d7 53 40 25 -_|··A·· ·M···S@%
0060 94 ae 48 c5 23 aa 61 2e 29 43 98 05 c5 7c d3 37 ··H·#·a.)C···|·7