

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

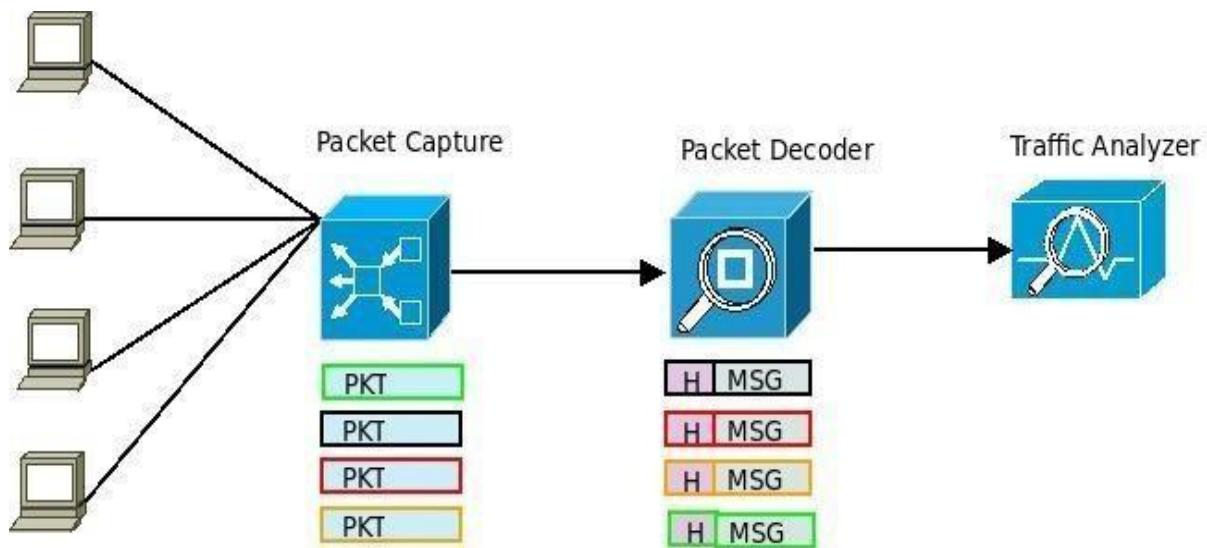
- I. Network Traffic analysis is the process of capturing network traffic and inspecting it closely to determine what is happening on the network
- II. Provides the details of network activities and their communication pattern in a network

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 –

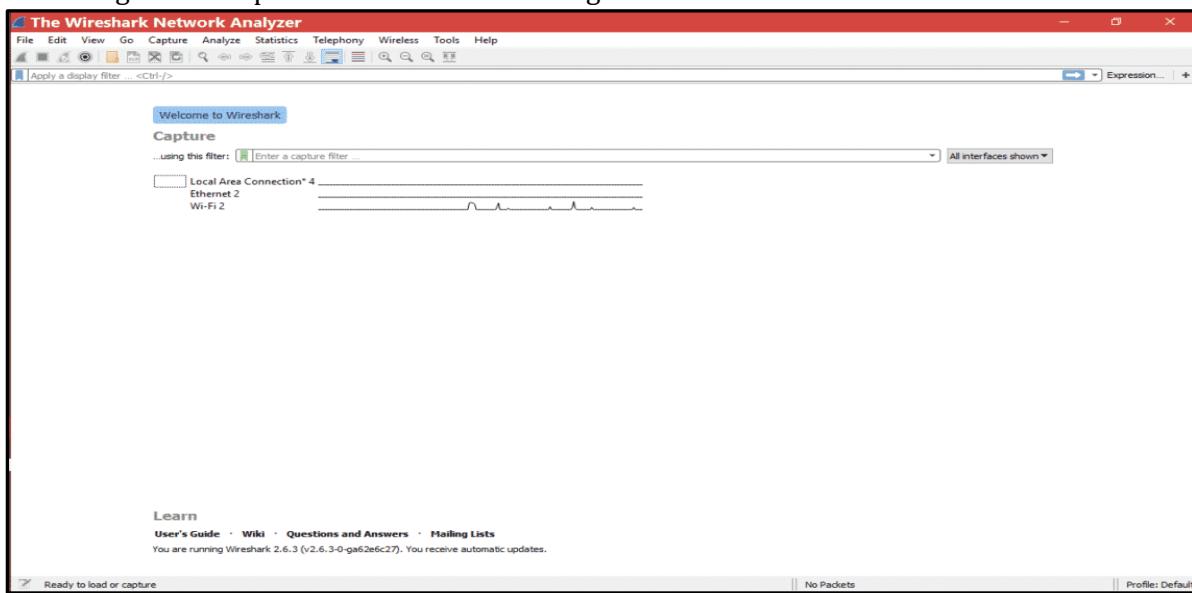
- Can listen on more than two interfaces simultaneously
- Packet coloring: Packets can be displayed in multiple colors for better analysis
- Filters
 - Capture filters: Capture packets based on some filter criteria
 - Display filters: Display only packets from the captured packets based on filter criteria.
- 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 Windows 10.



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

Wireshark - Protocol Hierarchy Statistics · Wi-Fi

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End
Frame	100.0	538	100.0	547241	227k	0	0	0
Ethernet	100.0	538	1.4	7532	3136	0	0	0
Internet Protocol Version 4	99.1	533	2.0	10676	4445	0	0	0
User Datagram Protocol	88.7	477	0.7	3816	1589	0	0	0
QUIC IETF	86.8	467	93.4	511260	212k	464	508854	2111
Domain Name System	0.4	2	0.1	523	217	2	523	217
Data	2.0	11	0.3	1669	695	11	1669	695
Transmission Control Protocol	9.7	52	2.4	12909	5375	32	6463	269
Transport Layer Security	3.2	17	2.1	11734	4886	17	11734	4886
Data	0.6	3	0.0	72	29	3	72	29
Internet Group Management Protocol	0.7	4	0.0	60	24	4	60	24
Address Resolution Protocol	0.9	5	0.0	140	58	5	140	58

No display filter.

Close Copy Help

Statistics -> Conversations

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

Wireshark - Conversations · Wi-Fi

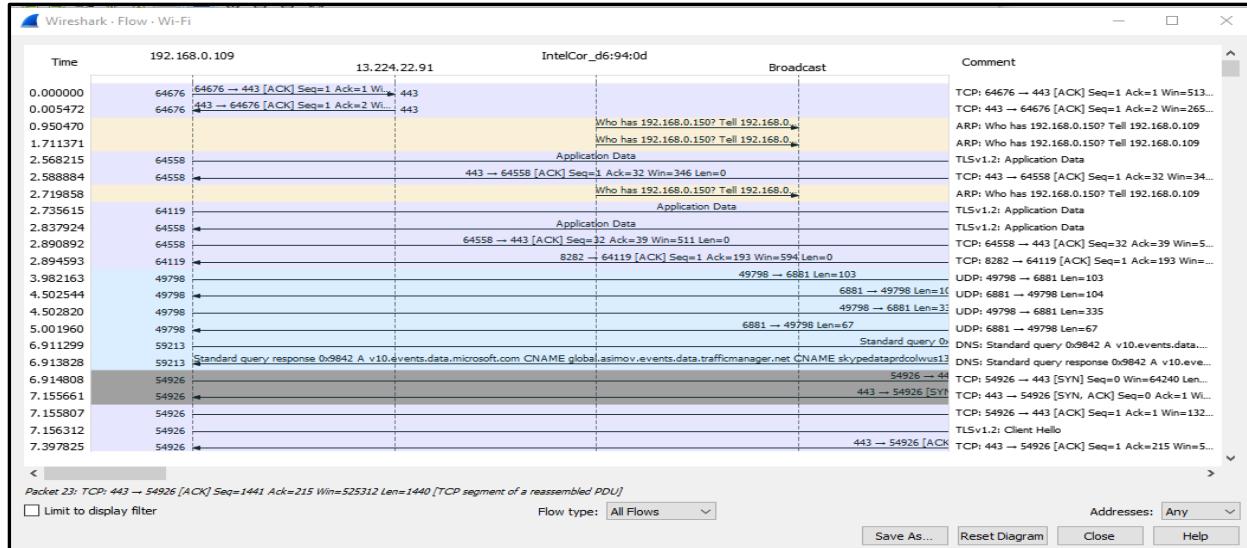
Ethernet · 6	IPv4 · 49	IPv6	TCP · 18	UDP · 49															
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A						
5.189.160.21	59176	192.168.0.109	49798	2	500	1	139	1	361	35.707985	0.0006	—	—						
105.112.101.150	137	192.168.0.109	137	3	276	0	0	3	276	65.620324	3.0218	0	730						
176.17.110.118	137	192.168.0.109	137	3	276	0	0	3	276	83.700050	3.0228	0	730						
188.143.94.195	59904	192.168.0.109	49798	2	476	1	145	1	331	59.696701	0.0005	—	—						
192.168.0.101	5353	224.0.0.251	5353	5	680	5	680	0	0	64.941035	22.0139	247	0						
192.168.0.109	49798	54.70.28.180	6881	2	254	1	145	1	109	3.982163	1.0198	1137	855						
192.168.0.109	49798	118.238.105.185	6881	2	523	1	377	1	146	4.502544	0.0003	—	—						
192.168.0.109	59213	192.168.0.1	53	2	607	1	89	1	518	6.911299	0.0025	—	—						
192.168.0.109	49798	159.196.228.199	3251	2	500	1	361	1	139	10.067895	0.0007	—	—						
192.168.0.109	49798	45.14.225.8	6339	1	145	1	145	0	0	0.965520	0.0000	—	—						
192.168.0.109	49798	185.157.221.247	25401	2	233	1	112	1	121	14.520769	0.0004	—	—						
192.168.0.109	59215	163.53.87.205	443	1,747	2128k	221	20k	1,526	2107k	16.560878	30.5271	5471	552k						
192.168.0.109	49798	131.147.215.124	22840	2	476	1	145	1	331	17.976309	0.1503	7718	17k						
192.168.0.109	59218	239.255.255.250	1900	4	860	4	860	0	0	21.159028	3.0222	2276	0						
192.168.0.109	49798	85.175.24.75	36090	2	505	1	359	1	146	24.322242	0.0005	—	—						
192.168.0.109	49798	41.190.31.14	26305	1	145	1	145	0	0	24.983319	0.0000	—	—						
192.168.0.109	59220	142.250.192.238	443	11	6933	4	2934	7	3999	26.473376	0.1109	211k	288k						
192.168.0.109	49798	131.147.215.124	22840	2	476	1	145	1	331	31.972508	1.0482	1106	831						
192.168.0.109	59218	93.104.54.130	49001	4	1392	2	1001	2	391	38.122057	19.0303	420	164						
192.168.0.109	49798	223.225.170.221	32023	1	145	1	145	0	0	38.961315	0.0000	—	—						
192.168.0.109	49798	188.242.253.229	41337	2	476	1	145	1	331	45.981992	0.1964	5906	13k						
192.168.0.109	49798	197.252.202.95	21046	2	505	1	359	1	146	45.996464	0.0006	—	—						
192.168.0.109	51987	192.168.0.1	53	2	235	1	87	1	148	47.987934	0.0713	9766	16k						
192.168.0.109	137	197.252.202.95	137	3	276	3	276	0	0	48.059949	3.0185	731	0						
192.168.0.109	49798	88.109.57.199	6895	2	505	1	359	1	146	48.716617	0.0006	—	—						
192.168.0.109	49798	194.187.220.200	9850	2	476	1	145	1	331	52.979140	0.2039	5688	12k						
192.168.0.109	49798	88.127.230.103	23712	2	505	1	359	1	146	55.5449494	0.0006	—	—						
192.168.0.109	49798	77.133.61.87	44900	2	476	1	331	1	145	56.046705	0.0005	—	—						
192.168.0.109	63221	192.168.0.1	53	3	373	2	170	1	203	57.475076	0.4599	2957	3531						
192.168.0.109	62884	192.168.0.1	53	2	518	1	77	1	441	57.599999	0.0047	—	—						
192.168.0.109	49798	91.144.176.221	40331	2	476	1	145	1	331	59.983077	0.2035	5700	13k						
192.168.0.109	52095	192.168.0.1	53	3	443	2	174	1	269	61.521663	0.5958	2336	3611						
192.168.0.109	49798	105.112.101.150	23853	4	1010	2	718	2	292	63.530956	25.9380	221	90						
192.168.0.109	58471	192.168.0.1	53	2	237	1	88	1	149	65.645202	0.0745	9450	16k						

Name resolution Limit to display filter Absolute start time Conversation Types

Copy Follow Stream... Graph... Close Help

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.

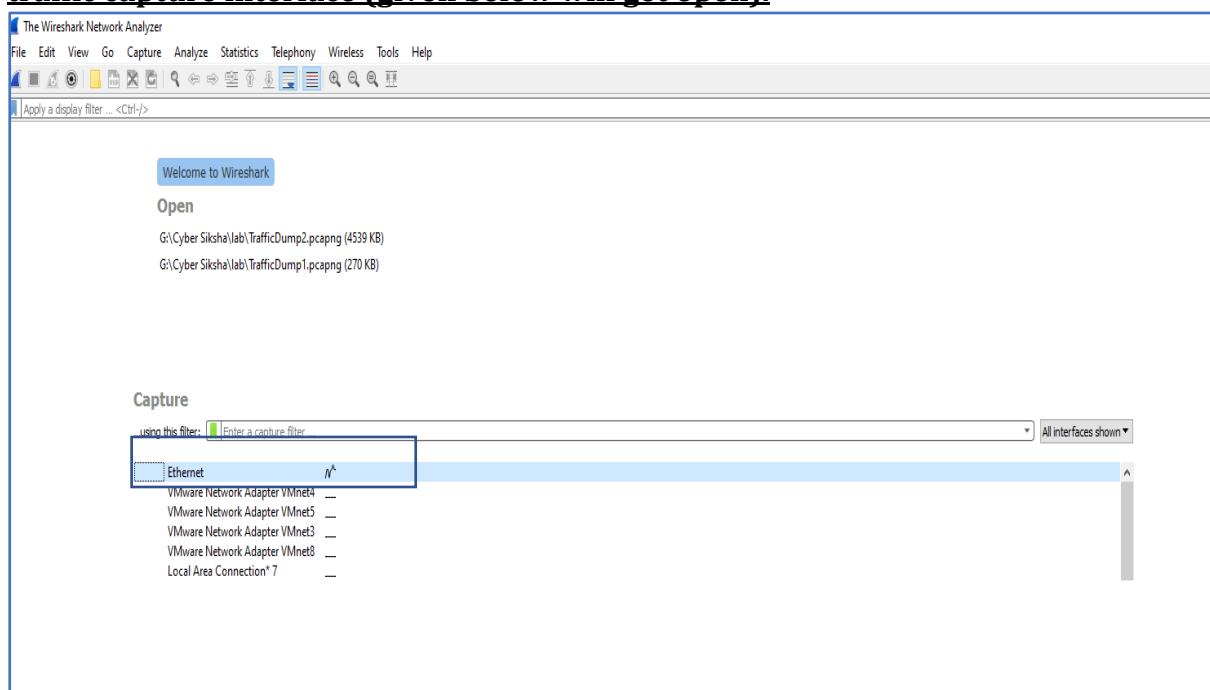
Tools

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

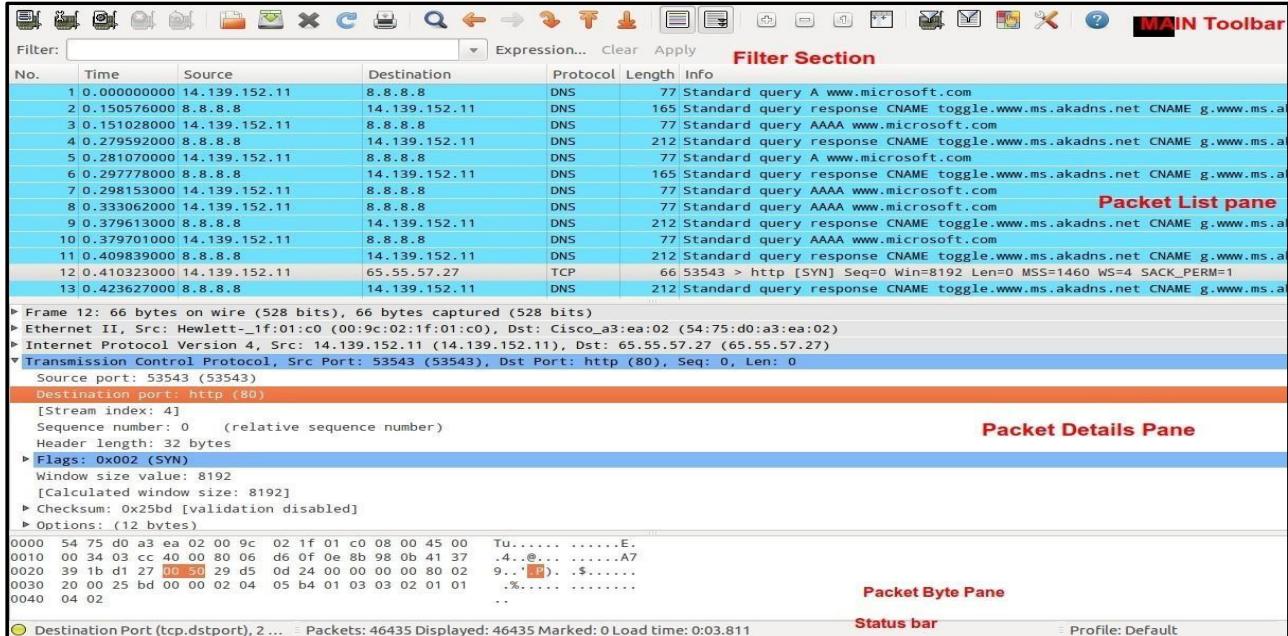
Help

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.

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



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 the packet 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

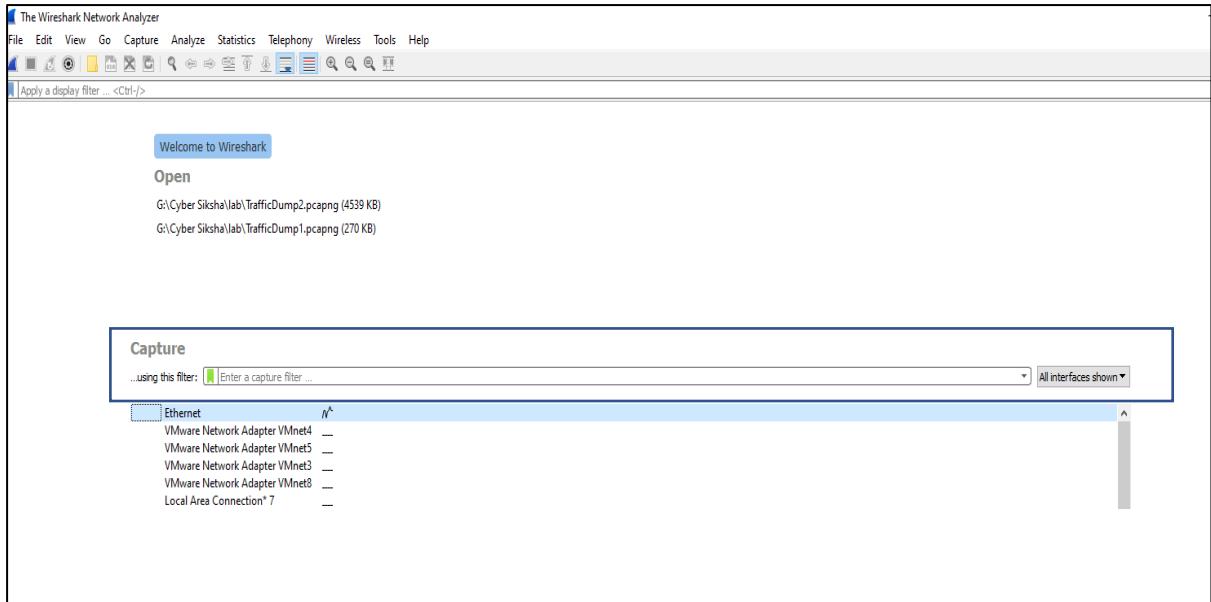
B. Packet Details pane

- 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.

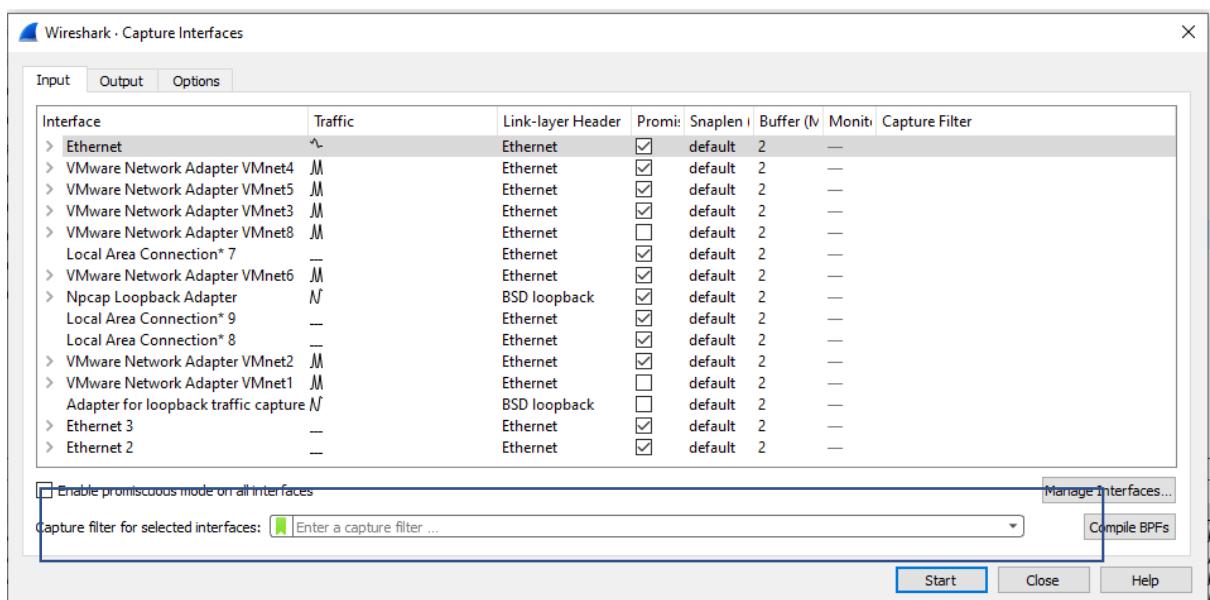
C. Capture Filter

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 of IPs 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

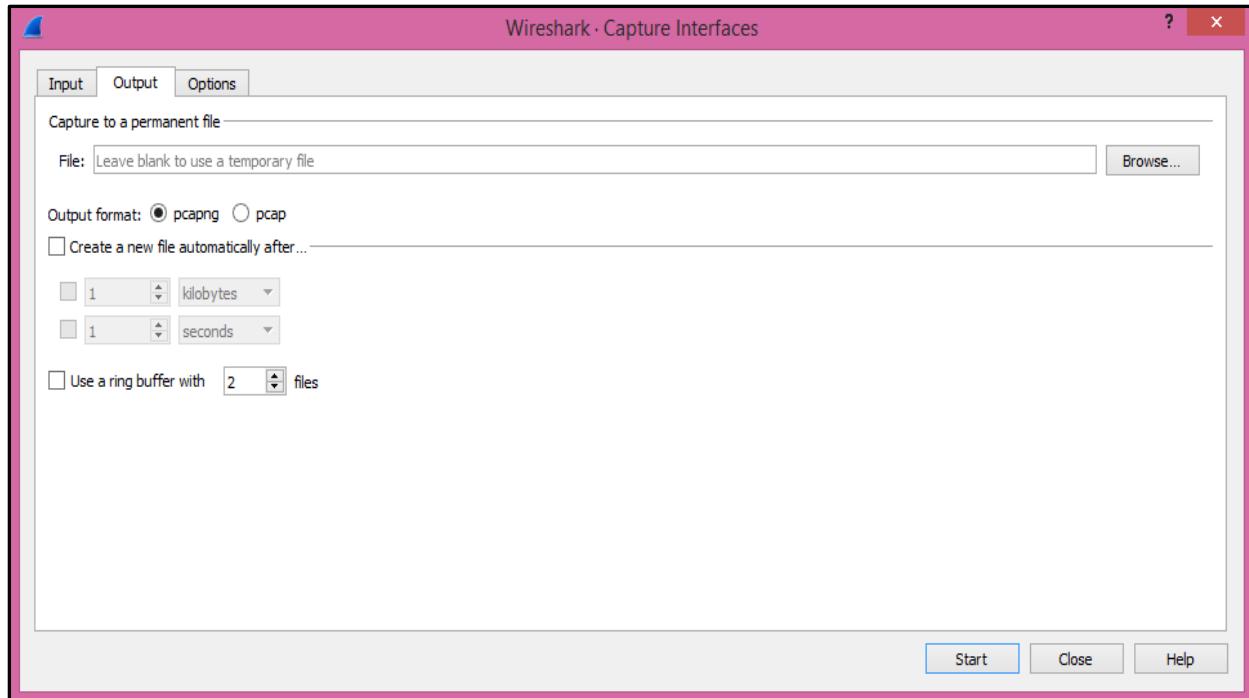
11. **not multicast:** to filter all multicast

Here are some examples of using Wireshark

a. List down the network interfaces connected to your host

No.	Time	Source	Destination	Protocol	Length	Info
565	10.743403	10.226.0.20	10.226.40.10	TCP	66	13111 → 58614 [SYN, ACK] Seq=1564811345 Ack=2377082522 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
570	10.807032	10.226.0.20	10.226.40.10	TCP	60	13111 → 58614 [ACK] Seq=1564811346 Ack=2377082805 Win=131328 Len=0
606	11.074579	10.226.0.20	10.226.40.10	TCP	173	13111 → 58614 [PSH, ACK] Seq=1564811346 Ack=2377082805 Win=131328 Len=119
756	14.183184	10.226.0.20	10.226.40.10	TCP	60	13111 → 58614 [ACK] Seq=1564811465 Ack=2377083088 Win=131072 Len=0
760	14.452668	10.226.0.20	10.226.40.10	TCP	350	13111 → 58614 [PSH, ACK] Seq=1564811465 Ack=2377083088 Win=131072 Len=296
815	16.792699	10.226.0.20	10.226.40.10	TCP	60	13111 → 58614 [ACK] Seq=1564811761 Ack=2377083610 Win=130560 Len=0
1788	36.750026	10.226.0.20	10.226.40.10	TCP	60	13111 → 58614 [ACK] Seq=1564811761 Ack=2377083611 Win=130560 Len=0
1789	36.750027	10.226.0.20	10.226.40.10	TCP	60	13111 → 58614 [FIN, ACK] Seq=1564811761 Ack=2377083611 Win=130560 Len=0
2484	48.969420	10.226.0.20	10.226.40.10	TCP	66	13111 → 58617 [SYN, ACK] Seq=4106498235 Ack=4247873905 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
2499	49.035276	10.226.0.20	10.226.40.10	TCP	60	13111 → 58617 [ACK] Seq=4106498236 Ack=4247874155 Win=131328 Len=0

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

Wi-Fi						
No.	Time	Source	Destination	Protocol	Length	Info
<code>ip.addr==172.217.166.238</code>						
274	64.670396	192.168.1.80	172.217.166.238	TCP	66	55843 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
275	64.675184	172.217.166.238	192.168.1.80	TCP	66	443 + 55843 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 WS=256
276	64.675366	192.168.1.80	172.217.166.238	TCP	54	55843 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
277	64.675872	192.168.1.80	172.217.166.238	TLSv1.3	630	Client Hello
278	64.679937	172.217.166.238	192.168.1.80	TCP	54	443 + 55843 [ACK] Seq=1 Ack=577 Win=66816 Len=0
279	64.745193	172.217.166.238	192.168.1.80	TLSv1.3	266	Server Hello, Change Cipher Spec, Application Data
280	64.747109	192.168.1.80	172.217.166.238	TLSv1.3	118	Change Cipher Spec, Application Data
281	64.747362	192.168.1.80	172.217.166.238	TLSv1.3	146	Application Data
282	64.747596	192.168.1.80	172.217.166.238	TLSv1.3	268	Application Data
283	64.747677	192.168.1.80	172.217.166.238	TCP	1466	55843 → 443 [ACK] Seq=947 Ack=213 Win=131072 Len=1412 [TCP segment of a reassembled PDU]
284	64.747677	192.168.1.80	172.217.166.238	TLSv1.3	1389	Application Data
285	64.753323	172.217.166.238	192.168.1.80	TCP	54	443 + 55843 [ACK] Seq=213 Ack=641 Win=66816 Len=0
286	64.753323	172.217.166.238	192.168.1.80	TCP	54	443 + 55843 [ACK] Seq=213 Ack=733 Win=66816 Len=0
287	64.753323	172.217.166.238	192.168.1.80	TCP	54	443 + 55843 [ACK] Seq=213 Ack=947 Win=67840 Len=0
288	64.755181	172.217.166.238	192.168.1.80	TLSv1.3	634	Application Data, Application Data

Frame 294: 1053 bytes on wire (8424 bits), 1053 bytes captured (8424 bits) on interface \Device\NPF_{76C07C17-EFAF-499C-9EEF-D7A4BAE98AEA}, id 0
> Ethernet II, Src: NokiaSha_27:26:70 (78:17:35:27:26:70), Dst: Tp-LinkT_e9:db:44 (50:3e:aa:e9:db:44)
> Internet Protocol Version 4, Src: 172.217.166.238, Dst: 192.168.1.80
> Transmission Control Protocol, Src Port: 443, Dst Port: 55843, Seq: 824, Ack: 3725, Len: 999
> Transport Layer Security

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.

```
> Frame 79: 182 bytes on wire (1456 bits), 182 bytes captured (1456 bits) on interface \Device\NPF_{76C07C17-EFAF-499C-9EEF-D
> Ethernet II, Src: NokiaSha_27:26:70 (78:17:35:27:26:70), Dst: Tp-LinkT_e9:db:44 (50:3e:aa:e9:db:44)
> Internet Protocol Version 4, Src: 220.156.184.6, Dst: 192.168.1.80
└ Transmission Control Protocol, Src Port: 443, Dst Port: 49780 Seq: 1569348788 Ack: 535296358, Len: 128
    Source Port: 443
    Destination Port: 49780
    [Stream index: 0]
    [TCP Segment Len: 128]
    Sequence number: 1569348788
    [Next sequence number: 1569348916]
    Acknowledgment number: 535296358
    0101 .... = Header Length: 20 bytes (5)
    > Flags: 0x018 (PSH, ACK)
    Window size value: 199
    [Calculated window size: 199]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xbbf7 [unverified]
```

e. Finding packets with a particular ttl value

Set display filter ip.ttl == 128

ip.ttl==128						
No.	Time	Source	Destination	Protocol	Length	Info
2 0.043745	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295357 Ack=1569347252 Win=508 Len=0
3 0.043865	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
5 0.229863	192.168.1.80	52.114.159.35		TCP	54	64202 + 443 [FIN, ACK] Seq=1922059455 Ack=3619039193 Win=1022 Len=0
7 0.477021	192.168.1.80	52.114.159.35		TCP	54	64202 + 443 [ACK] Seq=1922059456 Ack=3619039193 Win=1022 Len=0
9 1.056180	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295434 Ack=1569347380 Win=507 Len=0
10 1.056340	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
16 3.062547	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295511 Ack=1569347508 Win=507 Len=0
17 3.062703	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
20 5.088119	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295588 Ack=1569347636 Win=512 Len=0
21 5.088298	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
24 6.087816	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295665 Ack=1569347764 Win=512 Len=0
25 6.087989	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
32 8.109377	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295742 Ack=1569347892 Win=511 Len=0
33 8.109532	192.168.1.80	220.156.184.6		TLSv1.2	131	Application Data
37 10.113633	192.168.1.80	220.156.184.6		TCP	54	49780 + 443 [ACK] Seq=535295819 Ack=1569348020 Win=511 Len=0

f. Finding a string in the capture

Use display filter frame containing “string”

frame contains "yahoo"						
No.	Time	Source	Destination	Protocol	Length	Info
220 34.359162	192.168.0.1	192.168.0.2		TELNET	117	Telnet Data ...
249 46.424756	192.168.0.1	192.168.0.2		TELNET	223	Telnet Data ...

<						
> Frame 220: 117 bytes on wire (936 bits), 117 bytes captured (936 bits)						
> Ethernet II, Src: WesternD_9f:a0:97 (00:00:c0:9f:a0:97), Dst: Lite-OnU_3b:bf:fa (00:a0:cc:3b:bf:fa)						
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2						
> Transmission Control Protocol, Src Port: 23, Dst Port: 1254, Seq: 3225455365, Ack: 72604012, Len: 51						
└ Telnet						
Data: PING www.yahoo.com (204.71.200.74): 56 data bytes\r\n\ n						

Searching for a http request method in the capture

Set display filter http.request.Method==POST

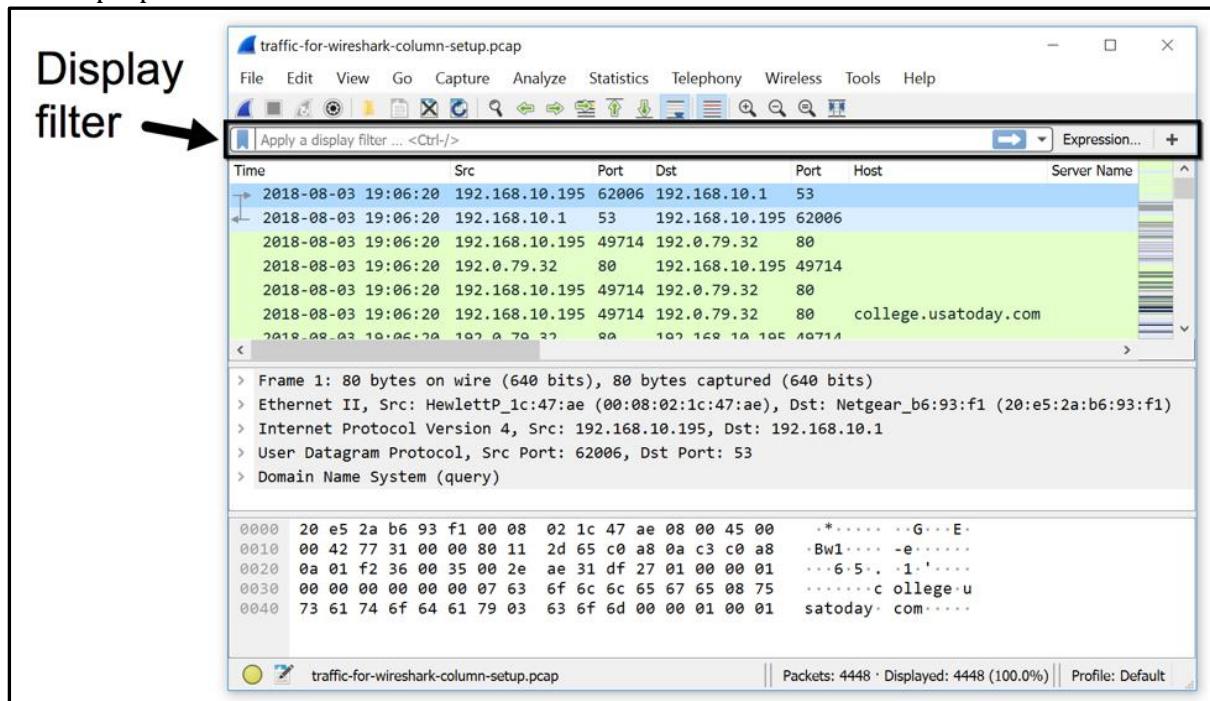
http.request.method == POST						
No.	Time	Source	Destination	Protocol	Length	Info
839	33.619829	192.168.1.108	209.196.28.153	HTTP	1141	1141 POST /ssframework/log/log.png HTTP/1.1 (application/x-www-form-urlencoded)
1367	194.916693	192.168.1.108	174.129.196.71	HTTP	1049	1049 POST /click.php HTTP/1.1 (application/x-www-form-urlencoded)
3129	206.927181	192.168.1.108	174.129.196.71	HTTP	1043	1043 POST /click.php HTTP/1.1 (application/x-www-form-urlencoded)

< [Frame 839: 1141 bytes on wire (9128 bits), 1141 bytes captured (9128 bits) on interface unknown, id 0]

- > Ethernet II, Src: HonHaiPr_68:74:f6 (90:4c:e5:68:74:f6), Dst: D-Link ff:1a:bc (00:26:5a:ff:1a:bc)
- > Internet Protocol Version 4, Src: 192.168.1.108, Dst: 209.196.28.153
- > Transmission Control Protocol, Src Port: 50897, Dst Port: 80, Seq: 3154919908, Ack: 2596290179, Len: 1087
- > [2 Reassembled TCP Segments (2327 bytes): #838(1240), #839(1087)]
- > Hypertext Transfer Protocol
- > HTML Form URL Encoded: application/x-www-form-urlencoded

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.



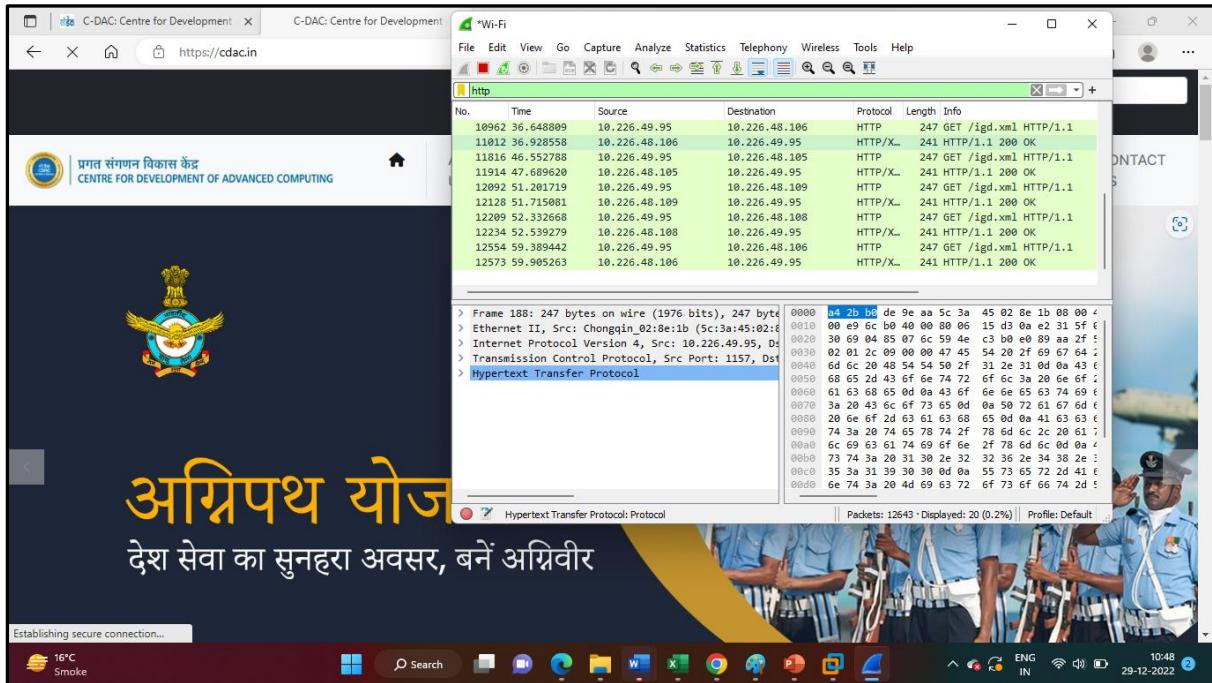
Below are some examples given for display filters.

a. Filter by Protocol

It's 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

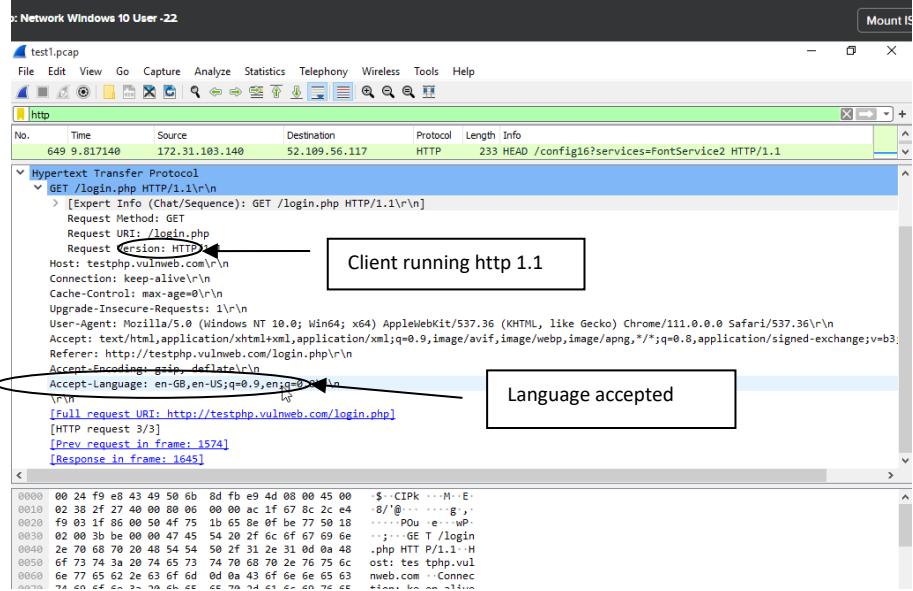
Remember that when deciding which protocol the transmitted data belongs to, the program considers the used port number. If a non-standard port is used, the program will not be able to find the necessary data. For example, if you connect to SSH on port 1234, the **ssh** filter will not find SSH traffic.

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:	<ol style="list-style-type: none">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:	<ol style="list-style-type: none">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-urlencoded"

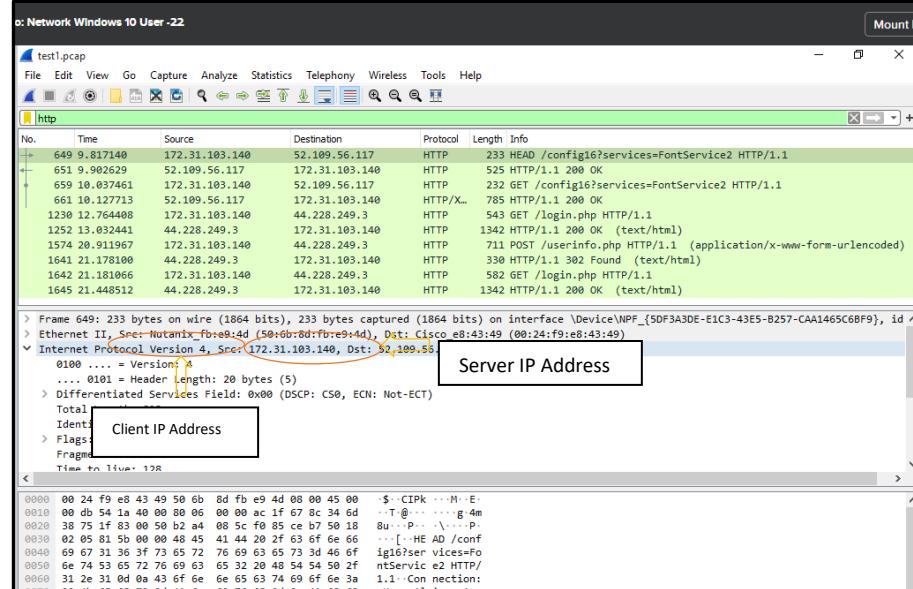
		<p>7. http.content_type contains "compressed"</p> <p>8. http.content_type contains "application"</p>
	Search for requests for files of a certain type. For example, to search for transferred ZIP archives:	http.request.uri contains "zip"

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?

