Name: Pratham Chintawar
Batch: 2
Roll No.: 16

Assignment No:6

Title: Understanding protocol stack of Intranet.

Problem Statement: Analyze packet formats of Ethernet, IP, TCP and UDP captured through Wireshark for wired networks.

Course Objective: To learn transport layer and application layer protocols used in the Internet.

Course Outcome: Develop Client-Servers by the means of correct standards, protocols and technologies.

Tools Required: Wireshark.

Theory:

1. Introduction

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible. You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course). In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

1.1 Features

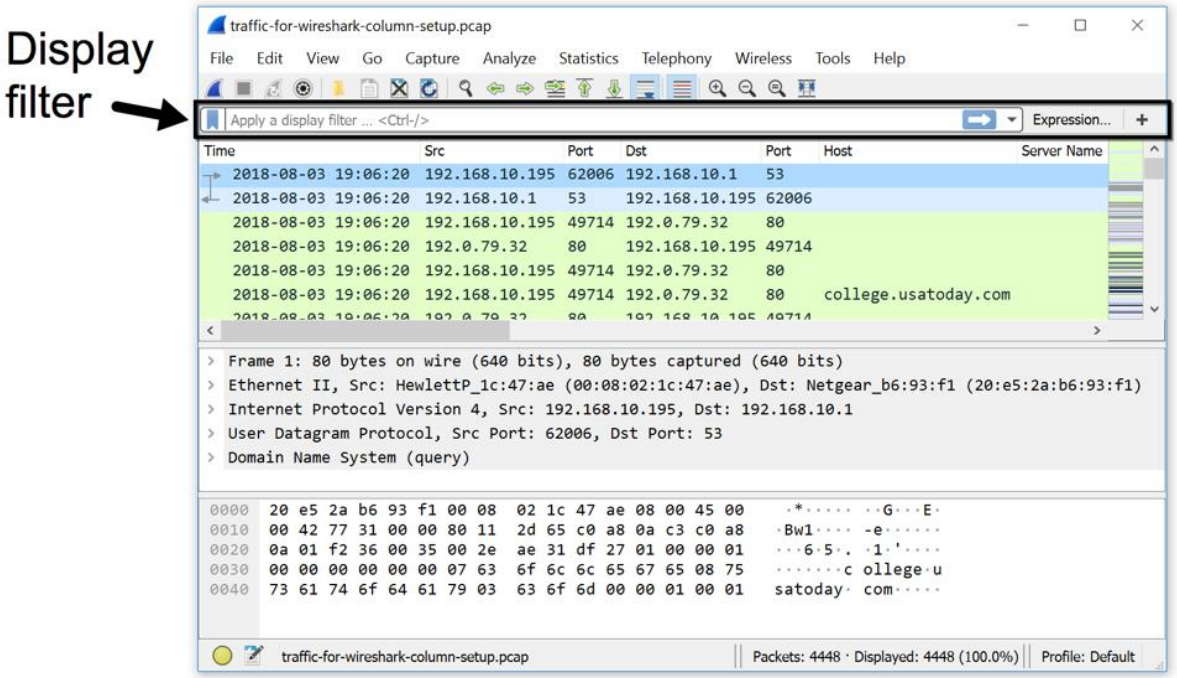
The following are some of the many features Wireshark provides:

- Available for *UNIX* and *Windows*.
- *Capture* live packet data from a network interface.
- *Open* files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- *Import* packets from text files containing hex dumps of packet data.
- Display packets with *very detailed protocol information*.
- *Save* packet data captured.
- *Export* some or all packets in a number of capture file formats.
- *Filter packets* on many criteria.
- *Search* for packets on many criteria.
- *Colorize* packet display based on filters.
- Create various *statistics*.

2. Wireshark Filter

2.1 Display Filter

Wireshark provides a display filter language that enables you to precisely control which packets are displayed. They can be used to check for the presence of a protocol or field, the value of a field, or even compare two fields to each other. These comparisons can be combined with logical operators, like "and" and "or", and parentheses into complex expressions.



Wireshark's display filter uses Boolean expressions, so you can specify values and chain them together. The following expressions are commonly used:

- Equals: == or ***eq***
- And: ***&&*** or ***and***
- Or: || (double pipe) or ***or***

2.1.1 Example

1. eth.src == 00:11:22:33:44:55: Source MAC address is 00:11:22:33:44:55
2. ip.addr == 10.0.0.1: Find all traffic that has IP of 10.0.0.1
3. tcp.dstport != 80: Destination tcp port is NOT 80
4. ip.addr==10.10.10.1
5. ip.addr==192.168.1.10 && ip.addr==192.168.1.20
6. !(ip.addr==192.168.1.10 && ip.addr==192.168.1.20)
7. (ip.addr==192.168.1.10 && ip.addr==192.168.1.20) && (tcp.port==445 || tcp.port==139)
8. ip.src==10.10.10.0/24
9. eth.addr==00:1b:17:00:01:31
10. ip.addr==10.10.10.1 && tcp.port==80
11. tcp.port==80
12. tcp.port==80 || tcp.port==3389
13. tcp.dstport==80
14. eth.dst=ff:ff:ff:ff:ff:ff
15. ip.addr==255.255.255.255
    ip.host contains "imap"

2.2 Capture Filters

Capture filters are used to decrease the size of captures by filtering out packets before they are added. **Capture filters** enable you to capture only traffic that you want to be captured, eliminating an unwanted stream of packets. Capturing packets is a processor-intensive task, and packet analyzers use a good amount of primary memory while they are running. Packets are only sent to the capture engine if they meet a certain criterion (capture filter expressions). Capture filters do not facilitate advanced filtering options, as in display filters.

2.2.1 Examples

Capture only traffic to or from IP address 172.18.5.4:

- host 172.18.5.4

Capture traffic to or from a range of IP addresses:

- net 192.168.0.0/24

or

- net 192.168.0.0 mask 255.255.255.0

Capture traffic from a range of IP addresses:

- src net 192.168.0.0/24

or

- src net 192.168.0.0 mask 255.255.255.0

Capture traffic to a range of IP addresses:

- dst net 192.168.0.0/24

or

- dst net 192.168.0.0 mask 255.255.255.0

Capture only DNS (port 53) traffic:

- port 53

Capture non-HTTP and non-SMTP traffic on your server (both are equivalent):

- host www.example.com and not (port 80 or port 25)

host www.example.com and not port 80 and not port 25

Capture except all ARP and DNS traffic:

- port not 53 and not arp

Capture traffic within a range of ports

- (tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550)

or, with newer versions of libpcap (0.9.1 and later):

- tcp portrange 1501-1549

Capture only Ethernet type EAPOL:

- ether proto 0x888e

Reject ethernet frames towards the Link Layer Discovery Protocol Multicast group:

- not ether dst 01:80:c2:00:00:0e

Capture only IPv4 traffic - the shortest filter, but sometimes very useful to get rid of lower layer protocols like ARP and STP:

- ip

Capture only unicast traffic - useful to get rid of noise on the network if you only want to see traffic to and from your machine, not, for example, broadcast and multicast announcements:

- not broadcast and not multicast

Capture IPv6 "all nodes" (router and neighbor advertisement) traffic. Can be used to find rogue RAs:

- dst host ff02::1

Conclusion:

Hence, we have Successfully Capture packets using Wireshark and wrote the exact packet capture filter expressions.

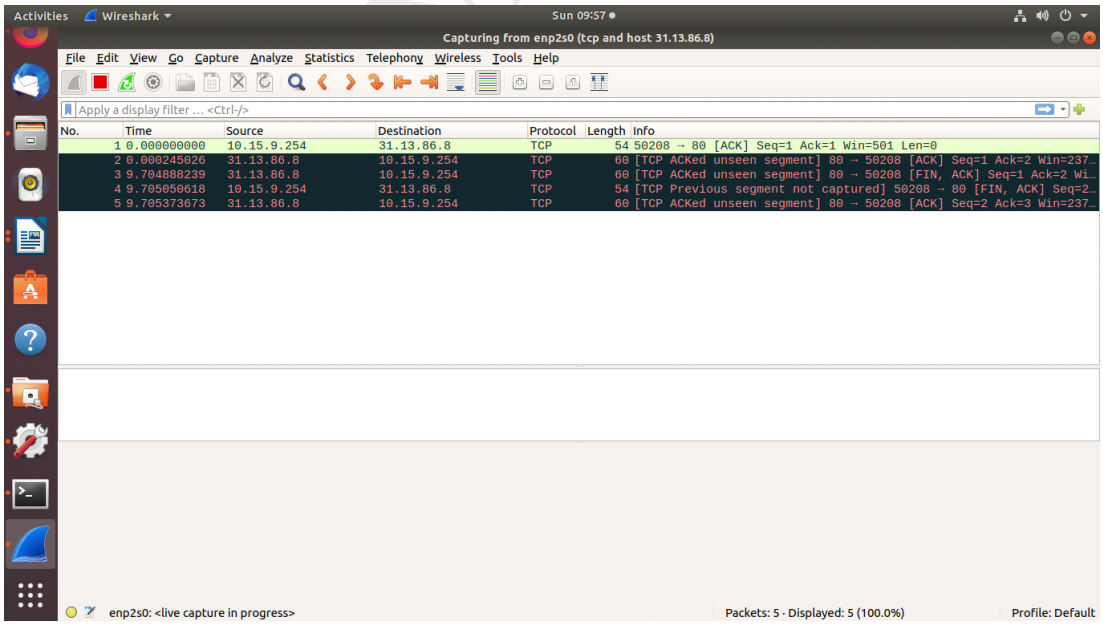
Example:

1. Identify Facebook.com IP Address

sudo apt-get install traceroute
traceroute facebook.com
traceroute to facebook.com (31.13.86.8), 30 hops max, 60 byte packets

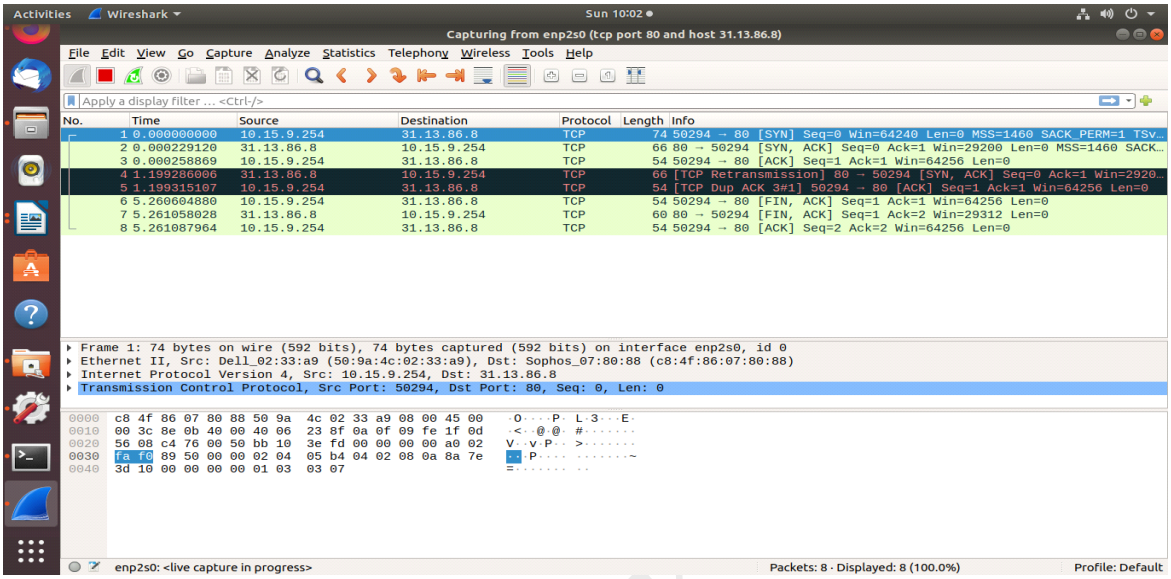
2. Capture all TCP traffic to/from Facebook, during the time when you log in to your Facebook account.

Capture filter: tcp and host 31.13.86.8



3. Capture all HTTP traffic to/from Facebook, when you log in to your Facebook account

Capture filter: tcp port 80 and host 31.13.86.8



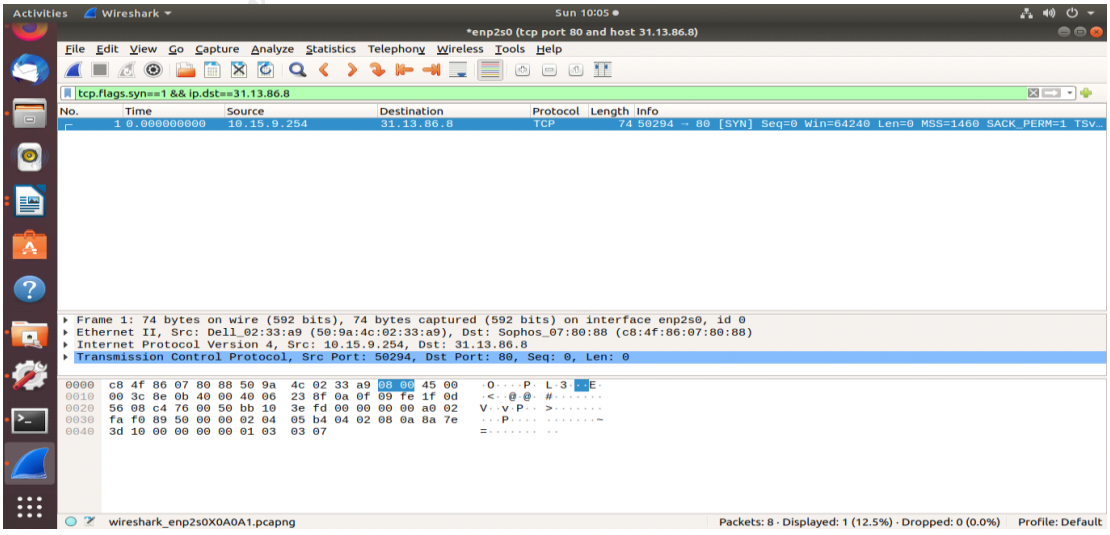
4. Write a DISPLAY filter expression to count all TCP packets (captured under item #1) that have the flags SYN, PSH, and RST set. Show the fraction of packets that had each flag set.

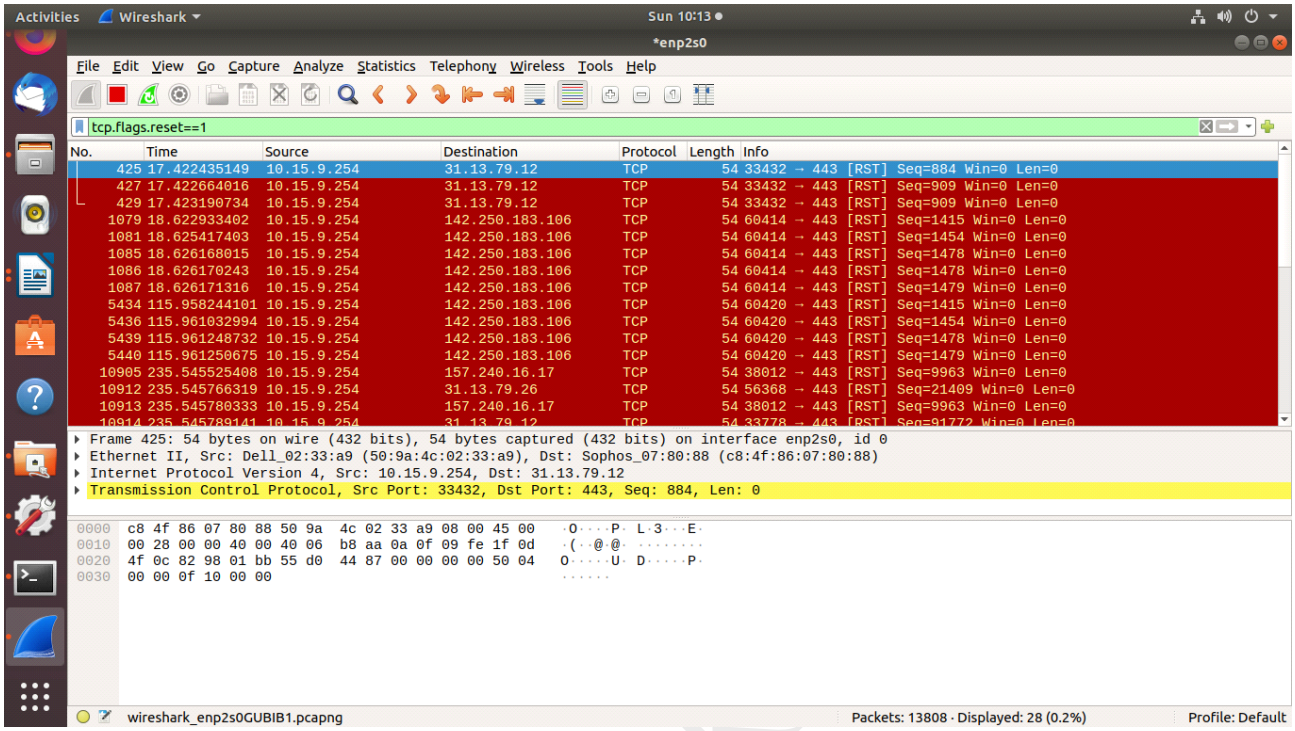
DISPLAY filter:

tcp.flags.syn==1 && ip.dst==31.13.86.8

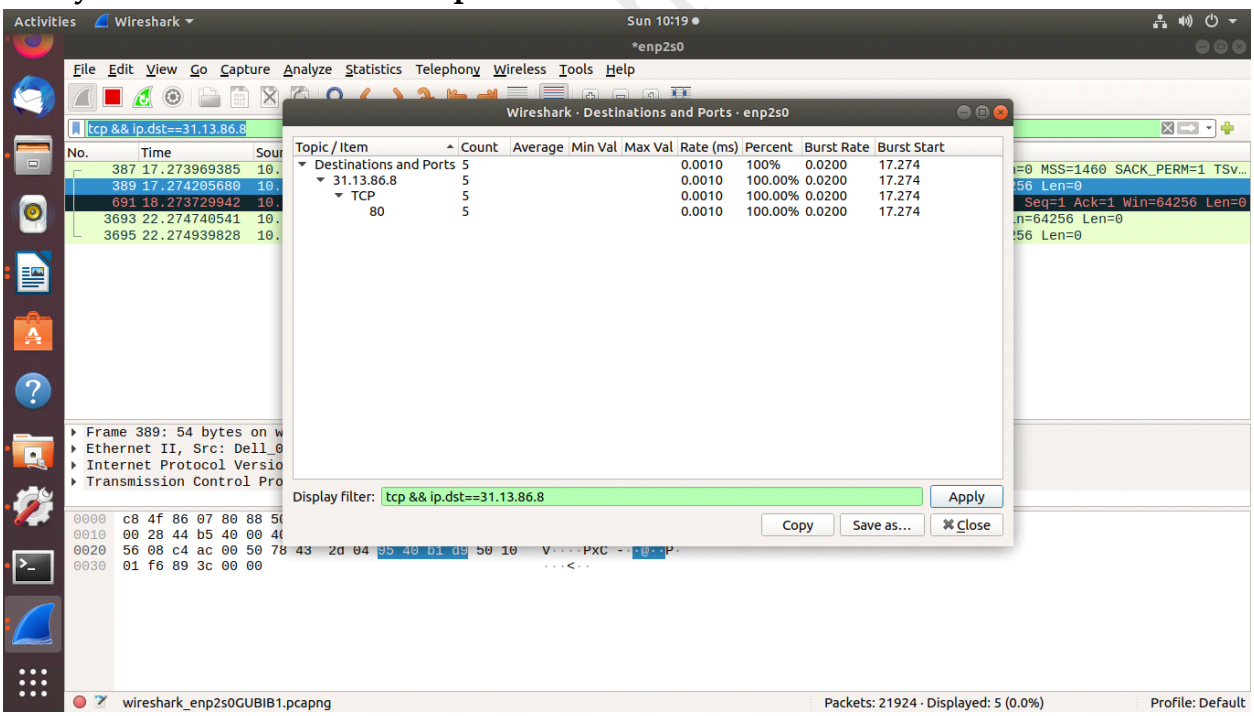
tcp.flags.push==1 && ip.dst==31.13.86.8

tcp.flags.reset==1 && ip.dst==31.13.86.8

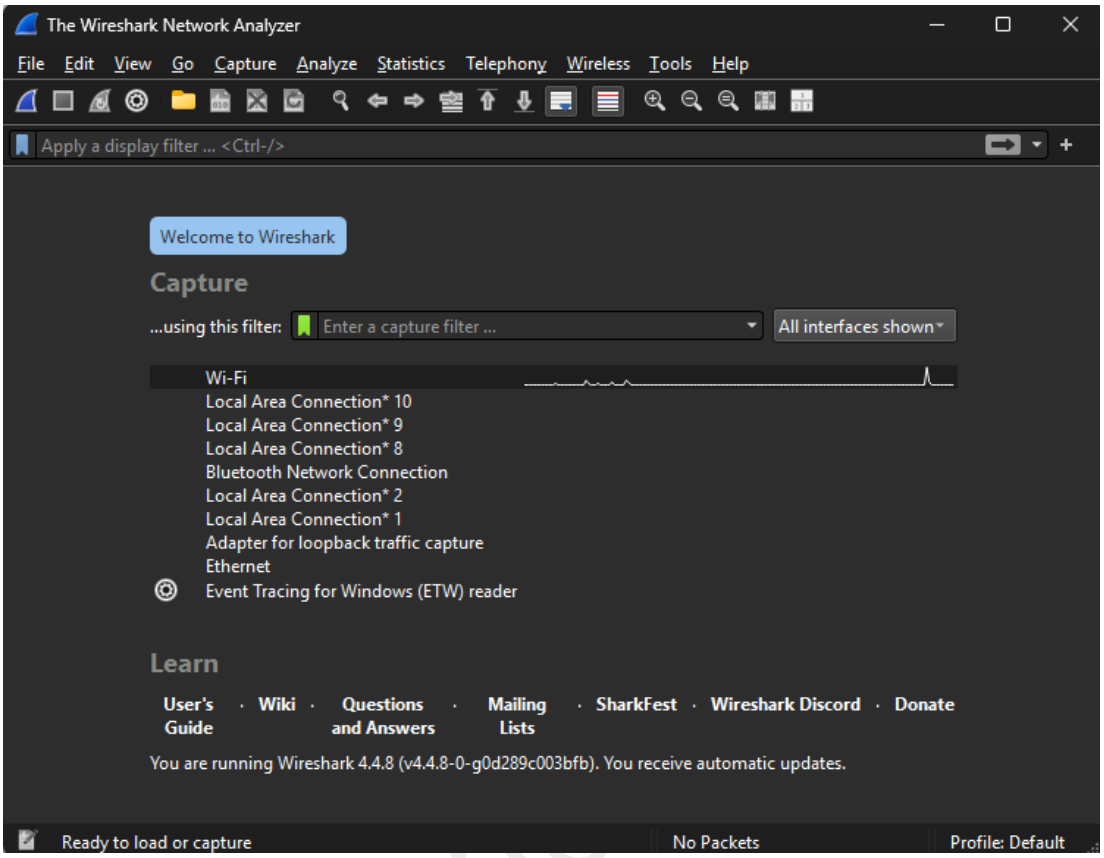




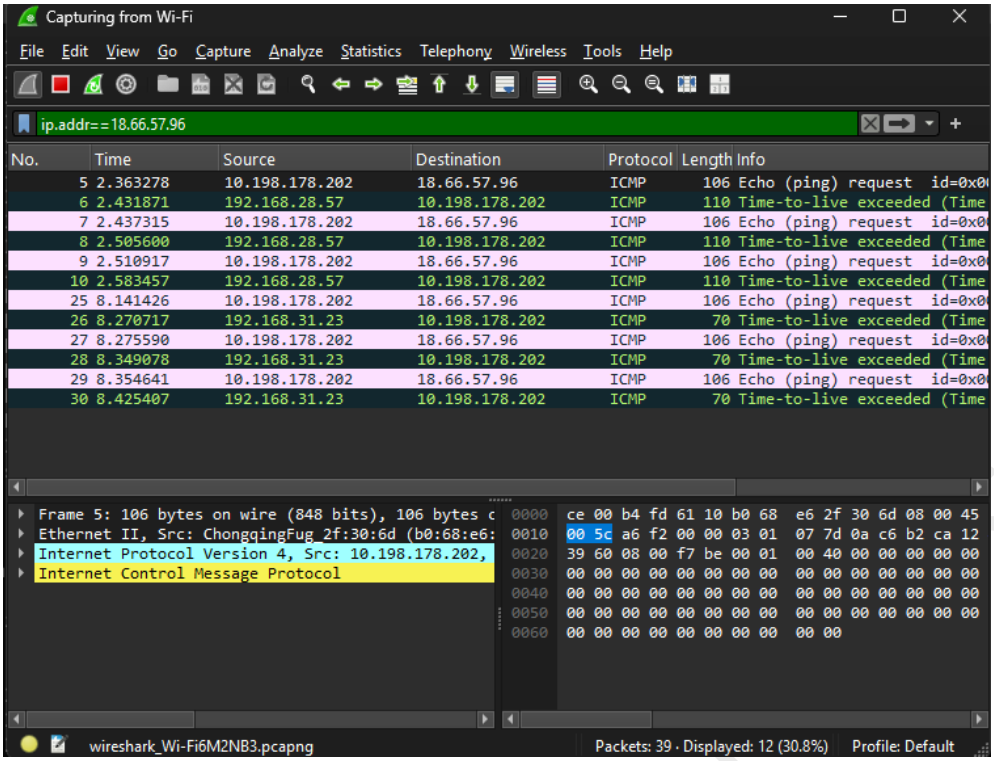
5. Count how many TCP packets you received from / sent to Face book, and how many of each were also HTTP packets.

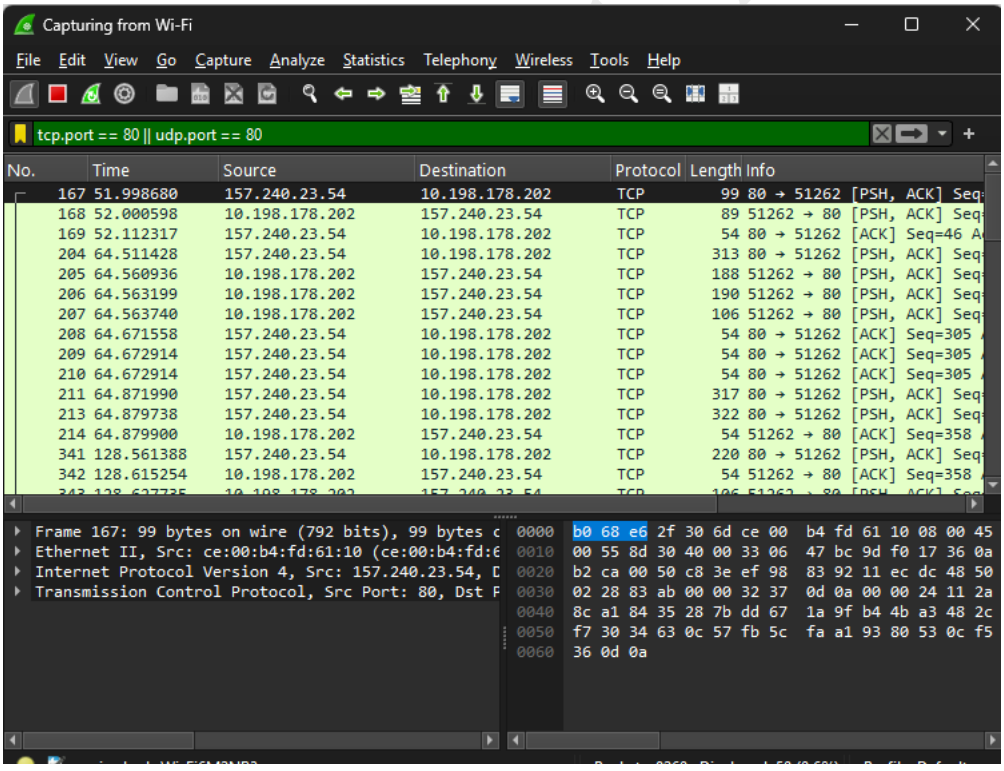


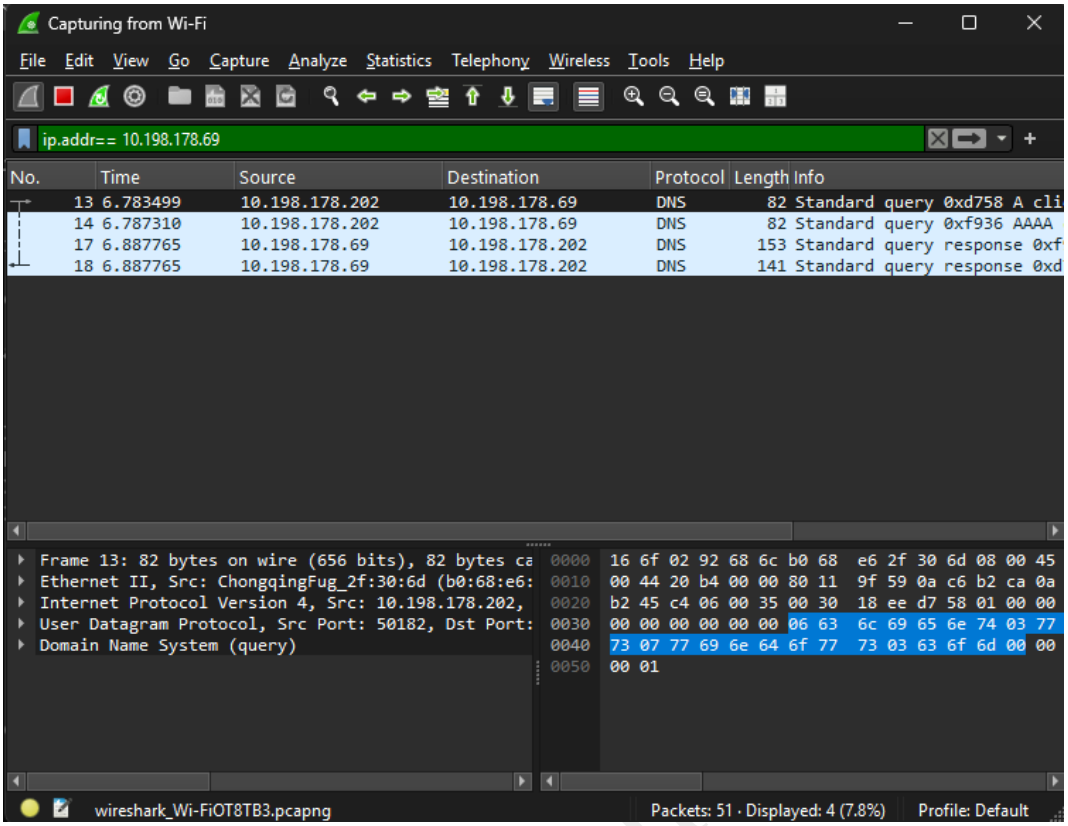
Output:




```
C:\Users\prath>tracert classroom.volp.in

Tracing route to classroom.volp.in [18.66.57.62]
over a maximum of 30 hops:

  1     9 ms     4 ms     5 ms  10.198.178.69
  2   305 ms   229 ms   181 ms  192.168.29.10
  3    38 ms    64 ms    35 ms  192.168.28.61
  4    80 ms    69 ms   120 ms  192.168.31.23
  5    96 ms    69 ms    76 ms  192.168.31.49
  6   138 ms    67 ms   124 ms  192.168.251.99
  7   122 ms    81 ms    71 ms  125.20.207.126
  8    76 ms    74 ms    75 ms  125.20.207.125
  9   123 ms   114 ms    78 ms  116.119.36.22
 10   109 ms   114 ms   115 ms  182.79.177.216
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
 13     *        *        *     Request timed out.
 14     *        *        *     Request timed out.
 15     *        *        *     Request timed out.
 16     *        *        *     Request timed out.
 17    68 ms   102 ms    93 ms  15.230.203.3
 18   113 ms   113 ms   115 ms  server-18-66-57-62.bom78.r.cloudfront.net [18.66.57.62]

Trace complete.
```


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.addr==18.66.57.96
No. Time Source Destination Protocol Length Info
5 2.363278 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
6 2.431871 192.168.28.57 10.198.178.202 ICMP 110 Time-to-live exceeded (Time
7 2.437315 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
8 2.505600 192.168.28.57 10.198.178.202 ICMP 110 Time-to-live exceeded (Time
9 2.510917 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
10 2.583457 192.168.28.57 10.198.178.202 ICMP 110 Time-to-live exceeded (Time
25 8.141426 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
26 8.270717 192.168.31.23 10.198.178.202 ICMP 70 Time-to-live exceeded (Time
27 8.275590 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
28 8.349078 192.168.31.23 10.198.178.202 ICMP 70 Time-to-live exceeded (Time
29 8.354641 10.198.178.202 18.66.57.96 ICMP 106 Echo (ping) request id=0x0
30 8.425407 192.168.31.23 10.198.178.202 ICMP 70 Time-to-live exceeded (Time
Frame 5: 106 bytes on wire (848 bits), 106 bytes c
Ethernet II, Src: ChongqingFug_2f:30:6d (b0:68:e6:
Internet Protocol Version 4, Src: 10.198.178.202,
Internet Control Message Protocol
0000 ce 00 b4 fd 61 10 b0 68 e6 2f 30 6d 08 00 45
0010 00 5c a6 f2 00 00 03 01 07 7d 0a c6 b2 ca 12
0020 39 60 08 00 f7 be 00 01 00 40 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00
wireshark_Wi-Fi6M2NB3.pcapng
Packets: 39 · Displayed: 12 (30.8%)
Profile: Default


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 80 || udp.port == 80
No. Time Source Destination Protocol Length Info
167 51.998680 157.240.23.54 10.198.178.202 TCP 99 80 → 51262 [PSH, ACK] Seq
168 52.000598 10.198.178.202 157.240.23.54 TCP 89 51262 → 80 [PSH, ACK] Seq
169 52.112317 157.240.23.54 10.198.178.202 TCP 54 80 → 51262 [ACK] Seq=46 A
204 64.511428 157.240.23.54 10.198.178.202 TCP 313 80 → 51262 [PSH, ACK] Seq
205 64.560936 10.198.178.202 157.240.23.54 TCP 188 51262 → 80 [PSH, ACK] Seq
206 64.563199 10.198.178.202 157.240.23.54 TCP 190 51262 → 80 [PSH, ACK] Seq
207 64.563740 10.198.178.202 157.240.23.54 TCP 106 51262 → 80 [PSH, ACK] Seq
208 64.671558 157.240.23.54 10.198.178.202 TCP 54 80 → 51262 [ACK] Seq=305
209 64.672914 157.240.23.54 10.198.178.202 TCP 54 80 → 51262 [ACK] Seq=305
210 64.672914 157.240.23.54 10.198.178.202 TCP 54 80 → 51262 [ACK] Seq=305
211 64.871990 157.240.23.54 10.198.178.202 TCP 317 80 → 51262 [PSH, ACK] Seq
213 64.879738 157.240.23.54 10.198.178.202 TCP 322 80 → 51262 [PSH, ACK] Seq
214 64.879900 10.198.178.202 157.240.23.54 TCP 54 51262 → 80 [ACK] Seq=358
341 128.561388 157.240.23.54 10.198.178.202 TCP 220 80 → 51262 [PSH, ACK] Seq
342 128.615254 10.198.178.202 157.240.23.54 TCP 54 51262 → 80 [ACK] Seq=358
Frame 167: 99 bytes on wire (792 bits), 99 bytes c
Ethernet II, Src: ce:00:b4:fd:61:10 (ce:00:b4:fd:6
Internet Protocol Version 4, Src: 157.240.23.54, D
Transmission Control Protocol, Src Port: 80, Dst P
0000 b0 68 e6 2f 30 6d ce 00 b4 fd 61 10 08 00 45
0010 00 55 8d 30 40 00 33 06 47 bc 9d f0 17 36 0a
0020 b2 ca 00 50 c8 3e ef 98 83 92 11 ec dc 48 50
0030 02 28 83 ab 00 00 32 37 0d 0a 00 00 24 11 2a
0040 8c a1 84 35 28 7b dd 67 1a 9f b4 4b a3 48 2c
0050 f7 30 34 63 0c 57 fb 5c fa a1 93 80 53 0c f5
0060 36 0d 0a


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.addr== 10.198.178.69
No. Time Source Destination Protocol Length Info
13 6.783499 10.198.178.202 10.198.178.69 DNS 82 Standard query 0xd758 A cli
14 6.787310 10.198.178.202 10.198.178.69 DNS 82 Standard query 0xf936 AAAA
17 6.887765 10.198.178.69 10.198.178.202 DNS 153 Standard query response 0xf
18 6.887765 10.198.178.69 10.198.178.202 DNS 141 Standard query response 0xd
Frame 13: 82 bytes on wire (656 bits), 82 bytes ca
Ethernet II, Src: ChongqingFug_2f:30:6d (b0:68:e6:
Internet Protocol Version 4, Src: 10.198.178.202,
User Datagram Protocol, Src Port: 50182, Dst Port:
Domain Name System (query)
0000 16 6f 02 92 68 6c b0 68 e6 2f 30 6d 08 00 45
0010 00 44 20 b4 00 00 80 11 9f 59 0a c6 b2 ca 0a
0020 b2 45 c4 06 00 35 00 30 18 ee d7 58 01 00 00
0030 00 00 00 00 00 00 06 63 6c 69 65 6e 74 03 77
0040 73 07 77 69 6e 64 6f 77 73 03 63 6f 6d 00 00
0050 00 01
wireshark_Wi-FiOT8TB3.pcapng
Packets: 51 · Displayed: 4 (7.8%)
Profile: Default