

CN LAB ASSIGNMENT : 2

PANCHAL GUNGUN PARESH

U21CS052

Que-1. Packet sniffer and its structure

When any data has to be transmitted over the computer network, it is broken down into smaller units at the sender's node called *data packets* and reassembled at receiver's node in original format. It is the *smallest unit* of communication over a computer network. It is also called a block, a segment, a datagram or a cell. The act of capturing data packet across the computer network is called **packet sniffing**. It is similar to as wire tapping to a telephone network. It is mostly used by *crackers and hackers* to collect information illegally about network. It is also used by *ISPs, advertisers and governments*. **ISPs** use packet sniffing to track all your activities such as:

-
- what you looked on that website
- downloads from a site who is receiver of your email
- what is content of that email
- what you download
- sites you visit
 - streaming events like video, audio, etc.

Advertising agencies or internet advertising agencies are paid according to:

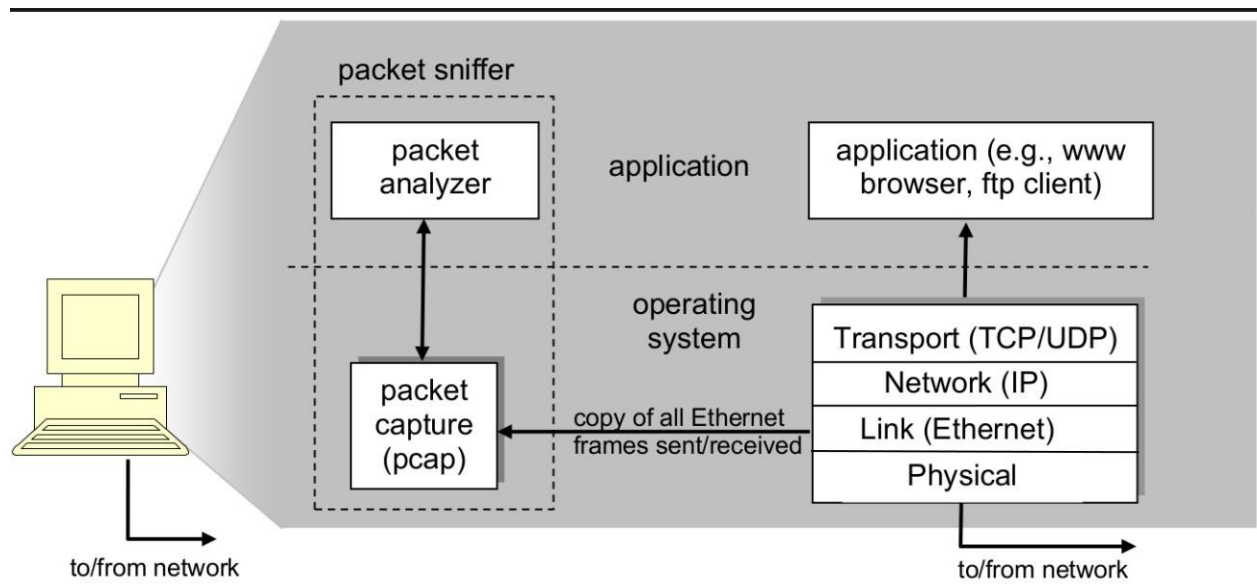
- number of ads shown by them.

- number of clicks on their ads also called PPC (pay per click).

To achieve this target, these agencies use packet sniffing to *inject advertisements* into the flowing packets. Most of the time these ads *contain malware*.

Government agencies use packet sniffing to:

- ensure security of data over the network.
- track an organisation's unencrypted data.

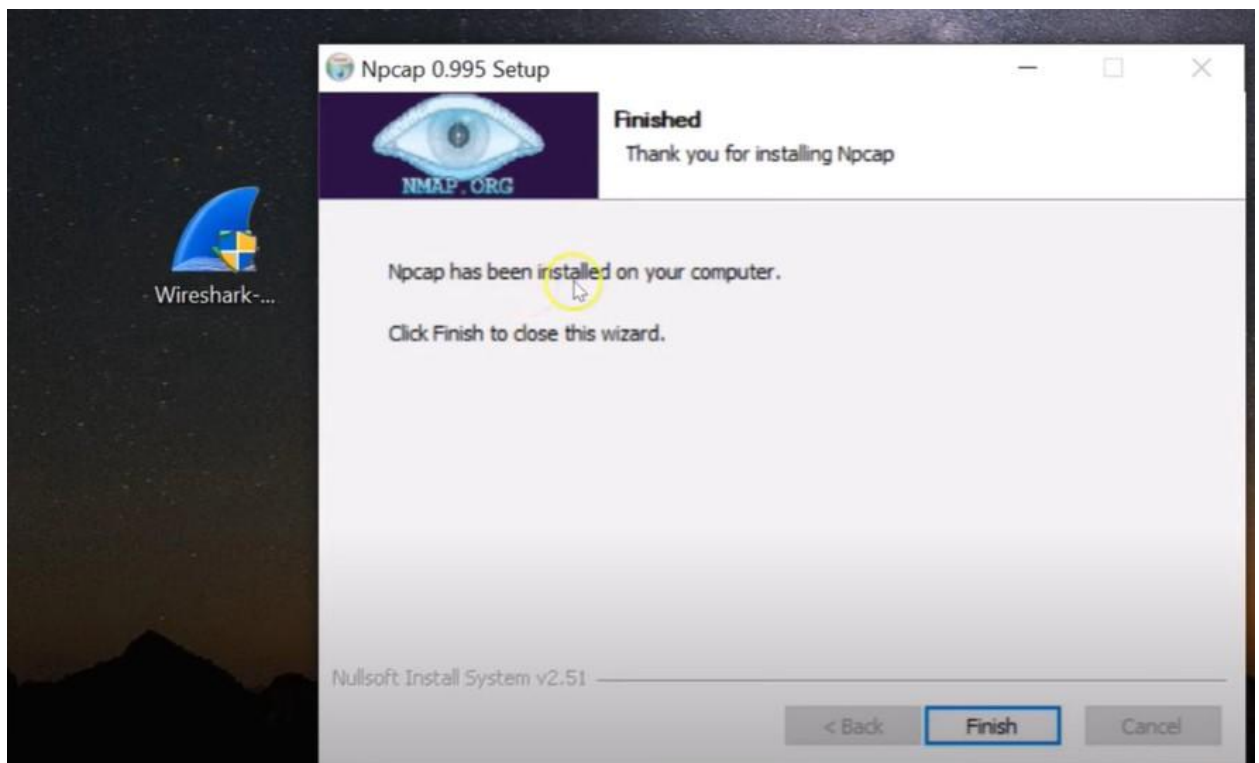


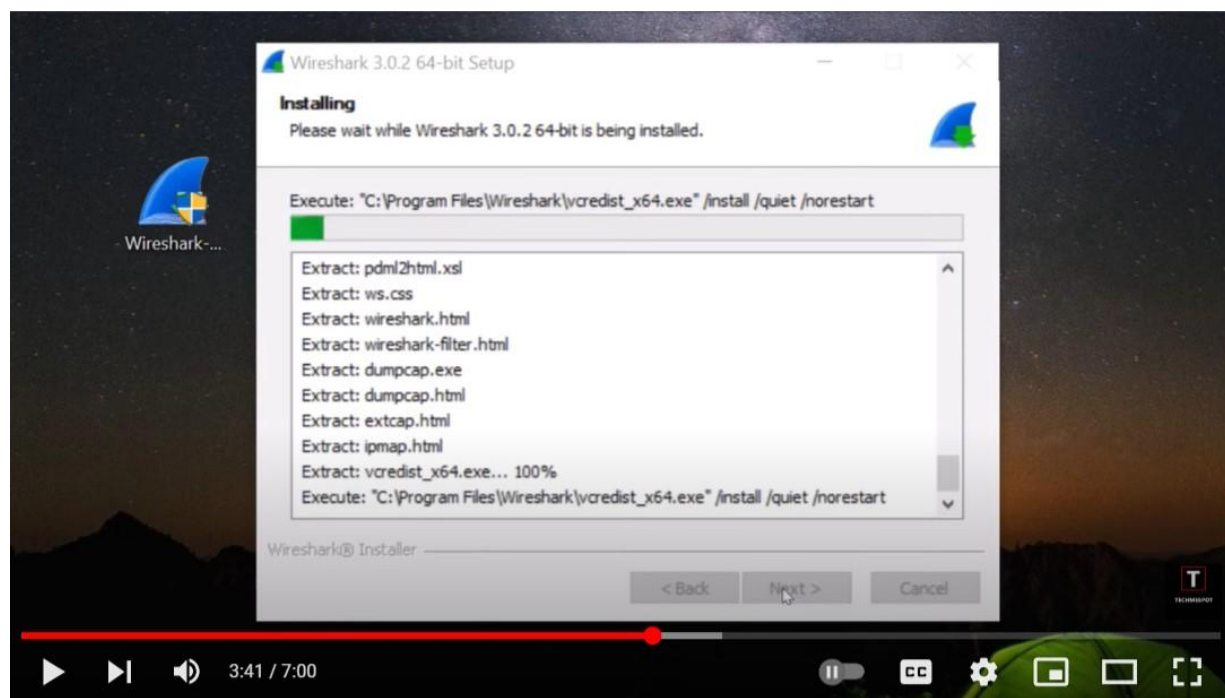
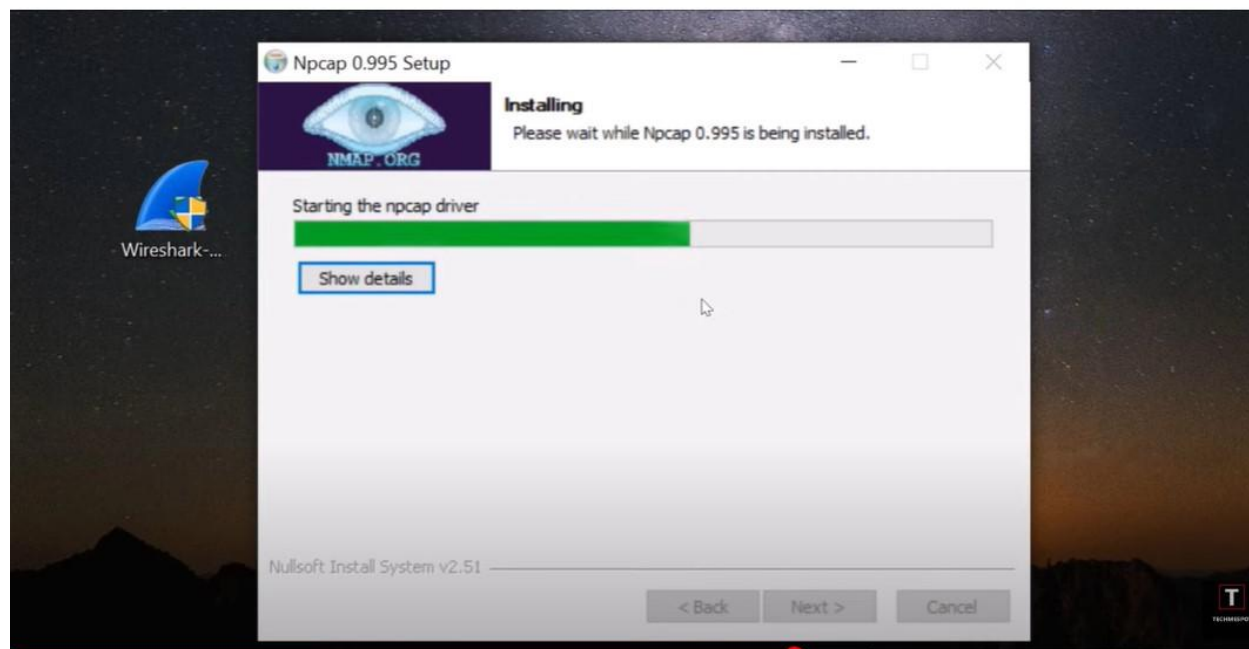
2. Steps involved in installation and running the wireshark with its components.

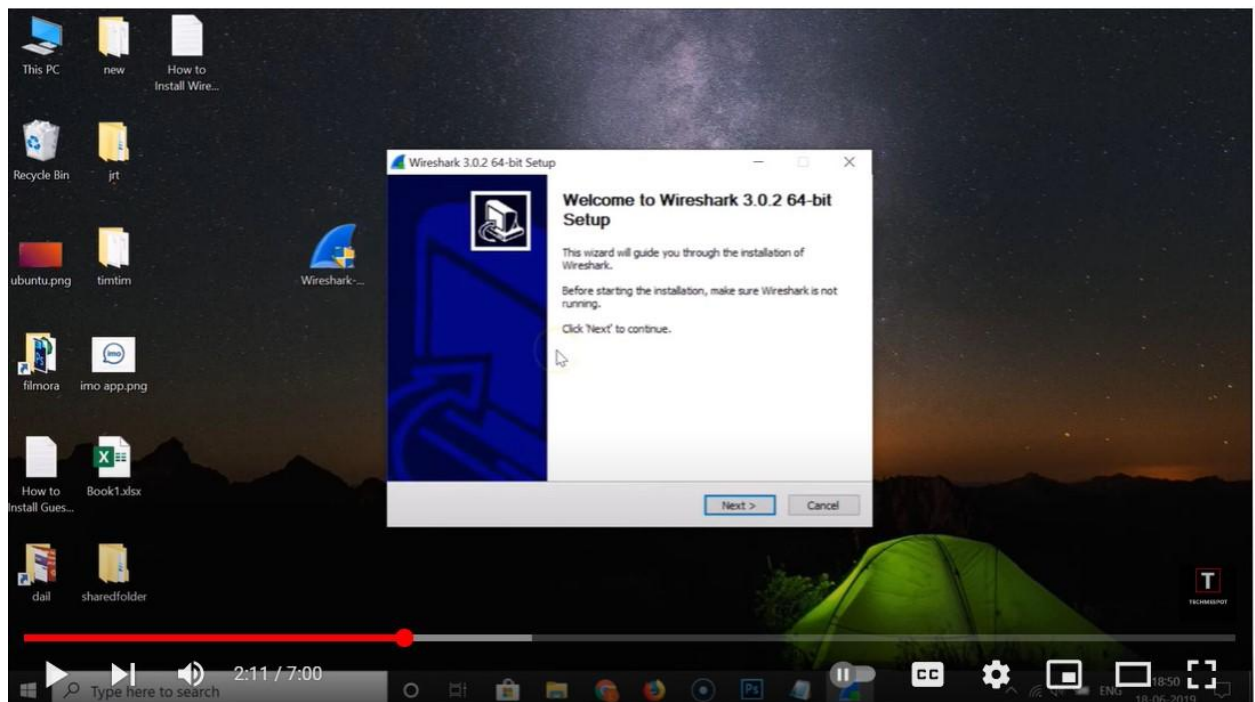
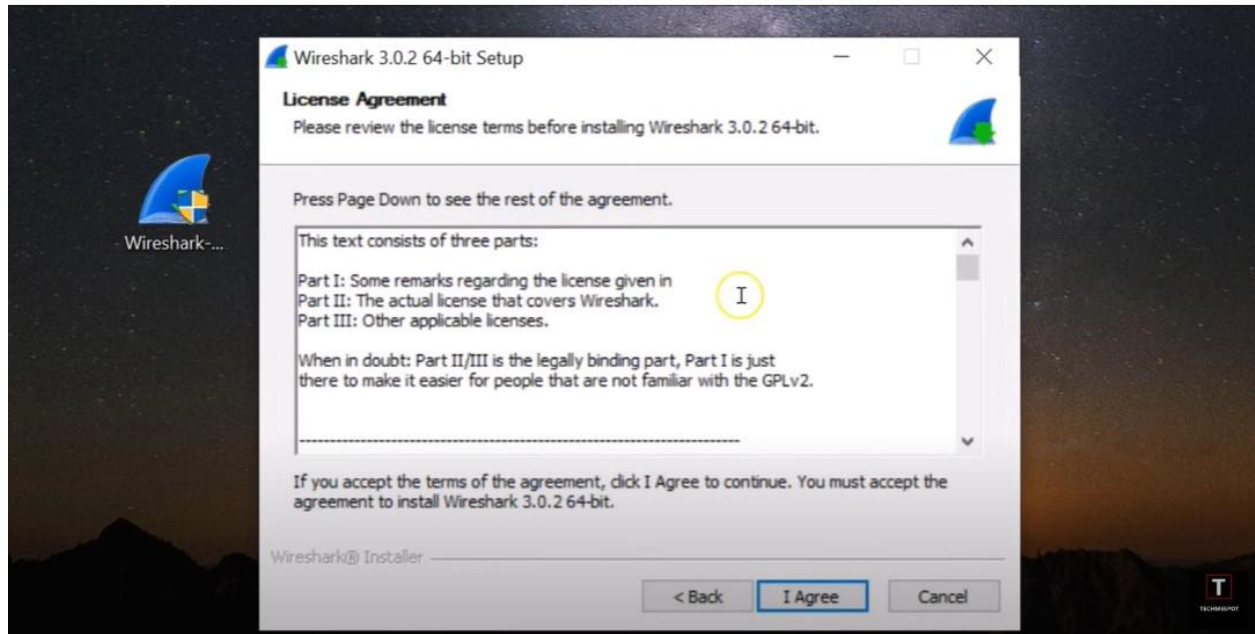
1. Go to the following website

<https://www.wireshark.org/download.html>

2. Choose the version according to your desktop specifications







3. Steps for capturing packets using URL given below:

<http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>

4. After your browser has displayed the INTRO-wireshark-file1.html page, stop Wireshark packet capture by selecting stop in the Wireshark capture window. Type in “http” into the display filter specification window at the top of the main Wireshark window. Then select Apply or just hit return. Find the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server.

a) In the unfiltered packet-listing window in step above, list the three distinct protocols that are present in the protocol column

1. TCP
2. HTTP
3. SSDP

b) How long did it take between sending the HTTP GET message and receiving the HTTP OK response? (By default, the Time column in the packet listing window displays the duration of Wireshark tracing in seconds. Select the Wireshark View pull-down menu, then click Time Show Format, then click Time-of-day to display the Time field in time-of-day format.)

```
13301      15:42:16.687738 2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff
2600:140f:1c00::1740:8ca9  HTTP 186  GET /connecttest.txt HTTP/1.1
```

```
360  15:38:46.422844 2600:140f:1c00::1740:8ca9
2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff  HTTP 261  HTTP/1.1 200 OK
(text/plain)
```

Time difference = 00:03:30 (3 minutes and 30 seconds)

c) What is the Internet address of the gaia.cs.umass.edu (also known as www.net.cs.umass.edu)? What is the Internet address of your computer?

=> 192.168.217.44

No.	Time	Source	Destination	Protocol	Length	Info
623	16:01:03.513248	192.168.217.44	192.168.217.76	DNS	77	Standard query 0x77e9 HTTPS gaia.cs.umass.edu
634	16:01:03.521439	192.168.217.44	192.168.217.76	DNS	77	Standard query 0xb306 AAAA gaia.cs.umass.edu
635	16:01:03.521687	192.168.217.44	192.168.217.76	DNS	77	Standard query 0xf981 A gaia.cs.umass.edu
942	16:01:03.802251	192.168.217.76	192.168.217.44	DNS	93	Standard query response 0xb004 A gaia.cs.umass.edu A 128.119.245.12
943	16:01:03.804365	192.168.217.76	192.168.217.44	DNS	93	Standard query response 0xf981 A gaia.cs.umass.edu A 128.119.245.12
1365	16:01:04.460340	192.168.217.44	192.168.217.76	DNS	71	Standard query 0x826e AAAA api.msn.com
1366	16:01:04.460603	192.168.217.44	192.168.217.76	DNS	71	Standard query 0x7fc4 A api.msn.com
1367	16:01:04.460962	192.168.217.44	192.168.217.76	DNS	71	Standard query 0xa205 HTTPS api.msn.com

> Frame 623: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface \Device\NPF_{B75F4D26-0000-0000-0000-000000000000} on Interface 192.168.217.44

> Ethernet II, Src: IntelCor_99:c0:c3 (a8:7e:ea:99:c0:c3), Dst: 66:0a:bd:19:60:42 (66:0a:bd:19:60:42)

> Internet Protocol Version 4, Src: 192.168.217.44, Dst: 192.168.217.76

> User Datagram Protocol, Src Port: 57697, Dst Port: 53

> Domain Name System (query)

```

0000  66 0a bd 19 60 42 a8 7e  ea 99 c0 c3 08 00 45 00  f....B.....E-
0010  00 3f 00 ad 00 00 80 11  00 00 c0 a8 d9 2c c0 a8  ?.....
0020  d9 4c e1 61 00 35 00 2b  34 07 77 e9 01 00 00 01  .L.a.S.+4.w....
0030  00 00 00 00 00 00 04 67  61 69 61 02 63 73 05 75  .....g.aia.cs.u
0040  6d 61 73 73 03 65 64 75  00 00 41 00 01          mass.edu --A-

```

- The computer is going to have many ip addresses according the packets that are being accessed => so which one to chose

No.	Source	Time	Destination	Protocol	Length	Info
1	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:18.936740	2600:140e:6:aa7::11a6	TCP	75	64195 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=
2	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:18.936740	2600:140e:6:aa7::11a6	TCP	75	64194 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=
3	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:18.936743	2600:140e:6:aa7::11a6	TCP	75	64193 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=
4	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:18.951824	2600:140e:6:aa7::11a6	TCP	75	64192 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=
5	2600:140e:6:aa7::11a6	16:21:19.080163	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	TCP	86	443 → 64194 [ACK] Seq=1 Ack=2 Win=501 Len=
6	2600:140e:6:aa7::11a6	16:21:19.080908	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	TCP	86	443 → 64192 [ACK] Seq=1 Ack=2 Win=501 Len=
7	2600:140e:6:aa7::11a6	16:21:19.108160	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	TCP	86	443 → 64193 [ACK] Seq=1 Ack=2 Win=501 Len=
8	2600:140e:6:aa7::11a6	16:21:19.112087	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	TCP	86	443 → 64195 [ACK] Seq=1 Ack=2 Win=501 Len=
9	64:ff9b::4101:2eb7	16:21:19.613968	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	TLSv1.2	105	Encrypted Alert
10	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:19.666409	64:ff9b::4101:2eb7	TCP	74	64298 → 443 [ACK] Seq=1 Ack=32 Win=258 Len=
11	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:21.053987	2404:6800:4009:801::200a	UDP	91	53952 → 443 Len=29
12	2404:6800:4009:801::200a	16:21:21.109201	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	UDP	87	443 → 53952 Len=25
13	57.128.101.85	16:21:21.577242	192.168.217.44	TCP	54	80 → 63891 [ACK] Seq=1 Ack=1 Win=0 Len=0
14	192.168.217.44	16:21:21.577307	57.128.101.85	TCP	54	[TCP ACKed unseen segment] 63891 → 80 [ACK
15	fe80::640a:bfff:fe19:6042	16:21:23.249471	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	ICMPv6	86	Neighbor Solicitation for 2409:40c1:10bf:926a
16	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:23.249538	fe80::640a:bfff:fe19:6042	ICMPv6	86	Neighbor Advertisement 2409:40c1:10bf:926a
17	2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff	16:21:23.489090	2404:6800:4009:830::200e	QUIC	1292	Initial, DCID=08cf0f4cc4151f85, PKT: 1, PA

> Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{B75F4D26-A0-0000-0000-000000000000} on Interface 192.168.217.44

> Ethernet II, Src: IntelCor_99:c0:c3 (a8:7e:ea:99:c0:c3), Dst: 66:0a:bd:19:60:42 (66:0a:bd:19:60:42)

> Internet Protocol Version 6, Src: 2409:40c1:10bf:926a:50d6:f19a:5bf4:b7ff, Dst: 2600:140e:6:aa7::11a6

> Transmission Control Protocol, Src Port: 64195, Dst Port: 443, Seq: 1, Ack: 1, Len: 1

```

0000  66 0a bd 19 60 42 a8 7e  ea 99 c0 c3 86 dd 60 0e  f....B.....E-
0010  f6 01 00 15 06 3f 24 09  40 c1 10 bf 92 6a 50 d6  .a...?..@....JP.
0020  f1 9a 5b f4 b7 ff 26 00  14 0e 00 06 0a a7 00 00  ..[...&.....N.F...
0030  00 00 00 00 11 a6 fa c3  01 bb 4e 10 46 d9 e5 c0  .....gAP.....
0040  67 41 50 10 01 00 b4 d5  00 00 00

```