

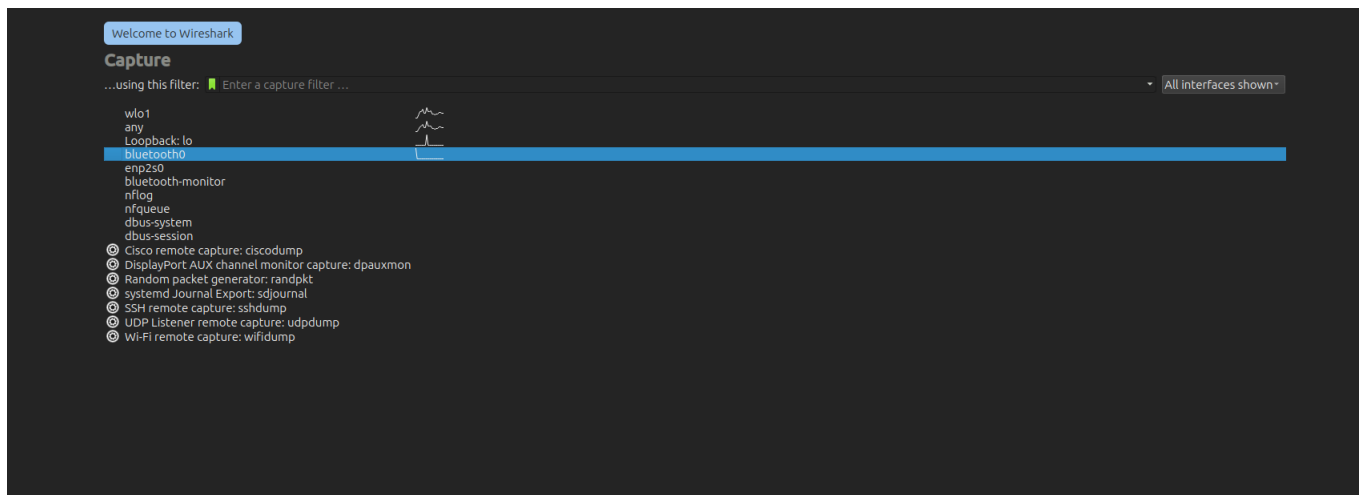
PCS Assignment-9

Name:Aditya Rathor

Roll No: B22AI044

Installing and starting wireshark:

- Below is the display page where all possible capture interface are visible



Selection of Network Interface: In this based on my IP address (generated using ipconfig) that is given to me based on the wifi network I am connected to I was able to select that network interface

Start and Stop of Packet Capture: I initiated the packet capture process at the onset of the session to begin monitoring network traffic. Following the completion of the data collection phase, I terminated the packet capture to ensure the integrity and completeness of the captured network traffic dataset.

I searched for a website ferrai.com and was able to see being made to that website on wireshark

No.	Time	Source	Destination	Protocol	Length	Info
3987	12.888217078	172.16.100.3	172.31.39.43	DNS	154	Standard query response 0xf57b HTTPS encrypted-tbn0.gstatic.com SOA ns1.google.com OPT
3988	12.888652961	172.16.100.3	172.31.39.43	DNS	143	Standard query response 0xf29d HTTPS ssl.gstatic.com SOA ns1.google.com OPT
4063	12.905460934	172.16.100.3	172.31.39.43	DNS	161	Standard query response 0x9064 HTTPS appstatusuggest-pa.googleapis.com SOA ns1.google.com OPT
4086	13.321676077	172.16.100.3	172.31.39.43	DNS	396	Standard query response 0xf5f1 A lns.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 142.250.194.161 NS n...
4210	13.322879446	172.16.100.3	172.31.39.43	DNS	182	Standard query response 0x4c2c HTTPS lns.googleusercontent.com CNAME googlehosted.l.googleusercontent.com SOA ns1.google.com...
4218	13.327338756	172.16.100.3	172.31.39.43	DNS	164	Standard query response 0x2eef HTTPS googlehosted.l.googleusercontent.com SOA ns1.google.com OPT
4372	13.786338076	172.16.100.3	172.31.39.43	DNS	216	Standard query response 0x16bc HTTPS data.grammarly.com CNAME prod-data.acquisition.grammarlyaws.com SOA ns-506.awsdns-63.co...
4374	13.788118227	172.16.100.3	172.31.39.43	DNS	187	Standard query response 0x0938 HTTPS prod-data.acquisition.grammarlyaws.com SOA ns-506.awsdns-63.com OPT
4379	13.811884666	172.16.100.3	172.31.39.43	DNS	435	Standard query response 0x8402 A data.grammarly.com CNAME prod-data.acquisition.grammarlyaws.com A 52.21.150.127 A 44.209.11...
4832	16.143504175	172.16.100.3	172.31.39.43	DNS	269	Standard query response 0x39d1 HTTPS ferrari-cdn.thron.com CNAME production-edgestatic-cdn.services.thron.com CNAME cdn.thro...
4814	16.146227285	172.16.100.3	172.31.39.43	DNS	150	Standard query response 0xf6f8 HTTPS a1874.d.akamai.net SOA n0d.akamai.net OPT
4819	16.299645333	172.16.100.3	172.31.39.43	DNS	261	Standard query response 0x73ce HTTPS www.ferrari.com CNAME www.ferrari.com CNAME www.ferrari.com CNAME dkehzerflwsi.cloudfr...
4821	16.303130332	172.16.100.3	172.31.39.43	DNS	186	Standard query response 0xf635 HTTPS dkehzerflwsi.cloudfront.net SOA ns-1877.awsdns-42.co.uk OPT
4833	16.439339394	172.16.100.3	172.31.39.43	DNS	232	Standard query response 0xazbb HTTPS service.maxymiser.net CNAME service.maxymiser.net.edgekey.net CNAME e242770.x.akamaiedg...
4835	16.442967566	172.16.100.3	172.31.39.43	DNS	156	Standard query response 0x1348 HTTPS e242770.x.akamaiedge.net SOA n0x.akamaiedge.net OPT
4840	16.580803059	172.16.100.3	172.31.39.43	DNS	543	Standard query response 0xe455 A ferrari-cdn.thron.com CNAME production-edgestatic-cdn.services.thron.com CNAME cdn.thron.co...
4881	16.753489142	172.16.100.3	172.31.39.43	DNS	439	Standard query response 0xe3ea A www.ferrari.com CNAME www.ferrari.com CNAME www.ferrari.com CNAME dkehzerflwsi.cloudfront...
4943	17.293988481	172.16.100.3	172.31.39.43	DNS	503	Standard query response 0x90ff A service.maxymiser.net CNAME service.maxymiser.net.edgekey.net CNAME e242770.x.akamaiedge.ne...
5056	18.320820844	172.16.100.3	172.31.39.43	DNS	366	Standard query response 0xb0f2 A www.googletagmanager.com A 142.250.77.232 NS ns4.google.com NS ns3.google.com NS ns2.google...
5057	18.321439887	172.16.100.3	172.31.39.43	DNS	152	Standard query response 0xe066 HTTPS www.googletagmanager.com SOA ns1.google.com OPT
5071	19.536583085	172.16.100.3	172.31.39.43	DNS	207	Standard query response 0x24e8 HTTPS cdn.ferrari.com CNAME d29ktrrh2p644.cloudfront.net SOA ns-148.awsdns-18.com OPT
5072	19.536615304	172.16.100.3	172.31.39.43	DNS	529	Standard query response 0xa02c HTTPS unpkg.com HTTPS ns.anurban.ns.cloudflare.com NS aron.ns.cloudflare.com A 172.64.32.69 A...
5074	19.537980707	172.16.100.3	172.31.39.43	DNS	479	Standard query response 0x9280 A unpkg.com A 184.17.249.203 A 184.17.245.203 A 184.17.246.203 A 184.17.248.203 A 184.17.247...
5076	19.538101045	172.16.100.3	172.31.39.43	HTTP	181	Standard query response 0x92b0 HTTPS d29ktrrh2p644.cloudfront.net CNAME ns-148.awsdns-18.com OPT

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.31.39.43	172.16.100.3	TCP	60	4172 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2958952389 TSecr=1622617034
2	0.005888801	172.31.39.43	172.16.100.3	TLSv1.2	98	Application Data
3	0.006962195	172.31.39.43	172.16.100.3	TLSv1.2	102	Application Data
4	0.010000000	172.31.39.43	172.16.100.3	TCP	60	4172 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2958952389 TSecr=1622617034
5	0.050813138	172.31.39.43	172.16.100.3	UDP	66	55522 → 21478 Len=24
6	0.090800449	172.31.39.43	172.16.100.3	TLSv1.2	97	Application Data
7	0.090369902	172.31.39.43	172.16.100.3	TLSv1.2	101	Application Data
8	0.255716146	172.31.39.43	172.16.100.3	UDP	58	> DISCOVERY_CMD
9	0.255980569	172.31.39.43	172.16.100.3	UDP	305	54915 → 54915 Len=263
10	0.378904282	172.31.39.43	172.16.100.3	TCP	60	443 → 40640 [ACK] Seq=64 Ack=37 Win=243 Len=0 TSval=169015003 TSecr=3734269893
11	0.378904883	172.31.39.43	172.16.100.3	TCP	60	443 → 40640 [ACK] Seq=64 Ack=37 Win=243 Len=0 TSval=169015003 TSecr=3734269893
12	0.379181850	172.31.39.43	172.16.100.3	TLSv1.2	97	Encrypted Alert
13	0.379181850	172.31.39.43	172.16.100.3	TCP	60	443 → 40640 [ACK] Seq=64 Ack=37 Win=243 Len=0 TSval=169015003 TSecr=3734269893
14	0.379439190	172.31.39.43	172.16.100.3	TCP	60	443 → 40640 [ACK] Seq=64 Ack=37 Win=243 Len=0 TSval=169015003 TSecr=3734269893
15	0.406817305	172.31.39.43	172.16.100.3	TCP	305	54915 → 54915 Len=263
16	0.641263917	172.31.39.43	172.16.100.3	TCP	60	443 → 40640 [ACK] Seq=64 Ack=37 Win=243 Len=0 TSval=169015003 TSecr=3734269893
17	0.665220841	172.31.39.43	172.16.100.3	UDP	82	64814 → 1947 Len=40
18	0.665324914	172.31.39.43	172.16.100.3	UDP	82	64814 → 1947 Len=40
19	0.767781131	172.31.39.43	172.16.100.3	DB-LSP	188	Dropbox LAM sync Discovery Protocol, JSON
20	0.767911087	172.31.39.43	172.16.100.3	DB-LSP	188	Dropbox LAM sync Discovery Protocol, JSON
21	0.768141673	172.31.39.43	172.16.100.3	DB-LSP	188	Dropbox LAM sync Discovery Protocol, JSON
22	0.768374624	172.31.39.43	172.16.100.3	DB-LSP	188	Dropbox LAM sync Discovery Protocol, JSON
23	0.768572076	172.31.39.43	172.16.100.3	DB-LSP	188	Dropbox LAM sync Discovery Protocol, JSON
24	0.972606269	172.31.39.43	172.16.100.3	UDP	305	54915 → 54915 Len=263
25	1.100000000	172.31.39.43	172.16.100.3	TCP	74	40640 → 443 [RST] Seq=0 Win=0 Len=0

The below bar display that 26183 packets were transmitted with not even single dropped

Within a few seconds so many packets were captured because Wireshark usually operates in “promiscuous mode” (on local networks), meaning it sees all the traffic passing by, even when it doesn't involve your machine.

Identification Based on IP Address: Using Wireshark's filtering capabilities, I employed IP address-based filters to focus on specific traffic flows within the captured data. By filtering packets based on IP addresses, I targeted communication originating from or destined to specific hosts or devices on the network.

Some of which were:

dns: to search for packets that followed the dns protocol

ip.src==<ip.addr> or ip.dest==<ip.addr> to see the packets of a particular ip address

Finding errors in packets as shown below:

630	43.141686038	172.31.39.43	74.125.68.188	TCP	66	48974 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3032166494 TSecr=131125819
637	43.141945052	172.31.39.43	74.125.68.188	TLSv1.3	639	Client Hello (SN=mtalk.google.com)
638	43.144740388	74.125.68.188	172.31.39.43	TCP	66	443 → 48974 [ACK] Seq=1 Ack=574 Win=1174784 Len=0 TSval=131125819 TSecr=3032166494
639	43.216143146	172.31.37.25	172.31.63.255	UDP	82	57621 → 57621 Len=40
640	43.216257573	172.31.37.25	172.31.63.255	UDP	82	57621 → 57621 Len=40
641	43.216396285	172.31.11.145	255.255.255.255	UDP	66	55522 → 21478 Len=24
642	43.228945338	74.125.68.188	172.31.39.43	TLSv1.3	2066	Server Hello, Change Cipher Spec
643	43.228946687	172.31.39.43	74.125.68.188	TCP	66	48974 → 443 [ACK] Seq=574 Ack=2801 Win=63368 Len=0 TSval=3032166581 TSecr=131125901
644	43.228945538	74.125.68.188	172.31.39.43	TCP	1263	[TCP Previous segment not captured] 443 → 48974 [PSH, ACK] Seq=5001 Ack=574 Win=66816 Len=1197 TSval=131125901 TSecr=3032166581
645	43.228976841	172.31.39.43	74.125.68.188	TCP	78	[TCP Dup ACK 645#1] 48974 → 443 [ACK] Seq=574 Ack=2801 Win=63368 Len=0 TSval=3032166581 TSecr=131125901 SLE=5001 SRE=6798
646	43.228946529	74.125.68.188	172.31.39.43	TCP	2066	[TCP Out-Of-Order] 443 → 48974 [PSH, ACK] Seq=2801 Ack=574 Win=66816 Len=2600 TSval=131125901 TSecr=3032166494
647	43.228988049	172.31.39.43	74.125.68.188	TCP	66	48974 → 443 [ACK] Seq=574 Ack=6798 Win=66832 Len=0 TSval=3032166581 TSecr=131125901
648	43.231706129	172.31.39.43	74.125.68.188	TLSv1.3	138	Change Cipher Spec, Application Data
649	43.240457941	172.31.39.43	74.125.68.188	TLSv1.3	312	Application Data
650	43.319543430	74.125.68.188	172.31.39.43	TCP	66	443 → 48974 [ACK] Seq=6798 Ack=638 Win=66816 Len=0 TSval=131125995 TSecr=3032166584
651	43.323245892	74.125.68.188	172.31.39.43	TCP	66	443 → 48974 [ACK] Seq=6798 Ack=884 Win=67840 Len=0 TSval=131125999 TSecr=3032166592
652	43.323777728	74.125.68.188	172.31.39.43	TLSv1.3	601	Application Data, Application Data

Out of order and duplicate packets were recieved which happens in TCP protocol

Traceroute to google's dns which is 8.8.8.8

```

adi_techbuddy ➤ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1 172.31.0.2 (172.31.0.2) 22.851 ms 22.815 ms 22.795 ms
 2 172.17.0.3 (172.17.0.3) 22.777 ms 22.759 ms 22.742 ms
 3 220.158.144.33.static-rajasthan.powertel.in (220.158.144.33) 25.955 ms 25.937 ms 24.520 ms
 4 * * *
 5 103.120.29.90.static-delhi.powertel.in (103.120.29.90) 39.113 ms 39.094 ms 35.134 ms
 6 72.14.205.20 (72.14.205.20) 30.185 ms 11.495 ms 12.315 ms
 7 * * *
 8 dns.google (8.8.8.8) 12.005 ms 11.464 ms 11.447 ms

```

It displays the hop number, the ip address and domain names for each hop, and 3 round trip hop time.

And the output in wireshark which is request for going to destination port and below is the output following the udp protocol

No.	Time	Source	Destination	Protocol	Length	Info
232	7.785377223	172.31.39.43	8.8.8.8	UDP	74	57933 → 33434 Len=3
233	7.785400958	172.31.39.43	8.8.8.8	UDP	74	36270 → 33435 Len=3
234	7.785415646	172.31.39.43	8.8.8.8	UDP	74	58512 → 33436 Len=3
235	7.785434942	172.31.39.43	8.8.8.8	UDP	74	55876 → 33437 Len=3
236	7.785452275	172.31.39.43	8.8.8.8	UDP	74	55721 → 33438 Len=3
237	7.785467013	172.31.39.43	8.8.8.8	UDP	74	42341 → 33439 Len=3
238	7.785482262	172.31.39.43	8.8.8.8	UDP	74	49773 → 33440 Len=3
239	7.785497390	172.31.39.43	8.8.8.8	UDP	74	42987 → 33441 Len=3
240	7.785512309	172.31.39.43	8.8.8.8	UDP	74	50774 → 33442 Len=3
241	7.785526756	172.31.39.43	8.8.8.8	UDP	74	58591 → 33443 Len=3
242	7.785541564	172.31.39.43	8.8.8.8	UDP	74	33465 → 33444 Len=3
243	7.785556282	172.31.39.43	8.8.8.8	UDP	74	34319 → 33445 Len=3
244	7.785571220	172.31.39.43	8.8.8.8	UDP	74	36168 → 33446 Len=3
245	7.785585567	172.31.39.43	8.8.8.8	UDP	74	60287 → 33447 Len=3
246	7.785600486	172.31.39.43	8.8.8.8	UDP	74	38219 → 33448 Len=3
247	7.785615374	172.31.39.43	8.8.8.8	UDP	74	56277 → 33449 Len=3
248	7.794245651	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceed
249	7.794245951	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceed
250	7.794246011	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceed
251	7.794246071	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceed
252	7.794246132	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceed
253	7.794246182	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceed
254	7.794742010	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceed
255	7.794742301	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceed
256	7.794891543	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceed

246	7.785600486	172.31.39.43	8.8.8.8	UDP
247	7.785615374	172.31.39.43	8.8.8.8	UDP
248	7.794245651	172.17.0.3	172.31.39.43	ICMP
249	7.794245951	172.17.0.3	172.31.39.43	ICMP
250	7.794246011	172.17.0.3	172.31.39.43	ICMP
251	7.794246071	172.31.0.2	172.31.39.43	ICMP
252	7.794246132	172.31.0.2	172.31.39.43	ICMP
253	7.794246182	172.31.0.2	172.31.39.43	ICMP
254	7.794742010	220.158.144.33	172.31.39.43	ICMP
255	7.794742301	220.158.144.33	172.31.39.43	ICMP
256	7.794891543	220.158.144.33	172.31.39.43	ICMP
262	7.800669090	72.14.205.20	172.31.39.43	ICMP
265	7.803174803	103.120.29.90	172.31.39.43	ICMP
266	7.803174954	103.120.29.90	172.31.39.43	ICMP
267	7.803175014	103.120.29.90	172.31.39.43	ICMP
269	7.803859600	172.31.39.43	8.8.8.8	UDP
270	7.803885579	172.31.39.43	8.8.8.8	UDP
271	7.803903743	172.31.39.43	8.8.8.8	UDP
272	7.803918691	172.31.39.43	8.8.8.8	UDP
273	7.803933309	172.31.39.43	8.8.8.8	UDP
274	7.803949830	172.31.39.43	8.8.8.8	UDP
277	7.809203085	172.31.39.43	8.8.8.8	UDP
278	7.809226880	172.31.39.43	8.8.8.8	UDP

Frame 278: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
 Ethernet II, Src: CloudNetwork_60:dc:93 (d8:80:83:00:00:00), Dst: 08:00:2b:01:02:03
 Internet Protocol Version 4, Src: 172.31.39.43, Destination: 8.8.8.8
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, Total Length: 60)
 Identification: 0xc478 (50296)
 0000 = Flags: 0x0
 ...0 0000 0000 0000 = Fragment Offset: 0
 Time to Live: 8
 Protocol: UDP (17)
 Header Checksum: 0x0adf [validation disabled]
 [Header checksum status: Unverified]
 Source Address: 172.31.39.43
 Destination Address: 8.8.8.8
 User Datagram Protocol, Src Port: 37840, Dst Port: 53
 Data (32 bytes)

Packet starts with the ttl(time to live)= 1 which represent the maximum hop value before reaching it dies and starts again if not able to reach the destination port then it starts again by increasing the hop value and does the same

I am able to see that with time to live as 8 it is able to reach the destination port

Troubleshooting

Now I pinged a random remote host

```
adi_techbuddy ping 192.168.50.1
PING 192.168.50.1 (192.168.50.1) 56(84) bytes of data.
^C
--- 192.168.50.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4079ms
```

It was not able to send even single packet and then I traced its route

```
adi_techbuddy traceroute 192.168.50.1
traceroute to 192.168.50.1 (192.168.50.1): 30 hops max, 60 byte packets
 1 172.31.0.2 (172.31.0.2) 39.2701 ms 22.666 ms 22.981 ms
 2 172.17.0.3 (172.17.0.3) 22.946 ms 22.929 ms 22.907 ms
 3 220.158.144.33.static-rajasthan.powertel.in (220.158.144.33) 24.071 ms 24.534 ms 24.036 ms
 4 * * * 31.127042432 172.31.39.43 192.168.50.1 UDP 74 41757 - 33511 Len=32
 5 103.120.29.90.static-delhi.powertel.in (103.120.29.90) 31.577 ms 31.561 ms 31.545 ms
 6 * * * 35.827491459 172.31.39.43 192.168.50.1 UDP 74 54268 - 33514 Len=32
 7 * * * 35.827564858 172.31.39.43 192.168.50.1 UDP 74 60830 - 33515 Len=32
 8 * * * 35.832810084 172.31.39.43 192.168.50.1 UDP 74 37172 - 33516 Len=32
 9 * * * 35.832856591 172.31.39.43 192.168.50.1 UDP 74 59766 - 33517 Len=32
10 * * * 35.832871429 172.31.39.43 192.168.50.1 UDP 74 40758 - 33518 Len=32
11 * * * 35.832887129 172.31.39.43 192.168.50.1 UDP 74 34137 - 33519 Len=32
12 * * * 35.832902478 172.31.39.43 192.168.50.1 UDP 74 57825 - 33520 Len=32
13 * * * 35.832918288 172.31.39.43 192.168.50.1 UDP 74 44274 - 33521 Len=32
14 * * * 35.832934128 172.31.39.43 192.168.50.1 UDP 74 41091 - 33522 Len=32
15 * * * 35.832948525 172.31.39.43 192.168.50.1 UDP 74 57204 - 33523 Len=32
16 From 9*: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlo1, id 0
17 Ethernet II, Src: CloudNetwork_60:dc:93 (d8:80:83:60:dc:93), Dst: IETF-VRRP-VRID_0a (00:00:5e:00:01:0a)
18 Internet Protocol Version 4, Src: 172.31.39.43, Dst: 192.168.50.1
19 * * * = Version: 4
20 * * * * * = Header Length: 20 bytes (5)
21 * * * * * = Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
22 * * * * * = Total Length: 60
23 * * * * * = Identification: 0xb792 (46994)
24 * * * * * = Flags: 0x0
25 * * * * * = Fragment Offset: 0
26 * * * * * = Time to Live: 30
27 * * * * * = Protocol: UDP (17)
28 * * * * * = Checksum: 0x1f2b [validation disabled]
29 * * * * * [Header checksum status: Unverified]
30 * * * * * = Source Address: 172.31.39.43
    * * * * * = Destination Address: 192.168.50.1
    * * * * * = User Datagram Protocol, Src Port: 57204, Dst Port: 33523
    * * * * * = Data (33 bytes)
```

Now I checked for the same in wireshark and was able to see the protocol which is UDP and as it is not able to reach it in max 30 hops it stops after ttl value reaching 30

ip.addr==192.168.50.1

No.	Time	Source	Destination	Protocol	Length	Info
	409 15.796616441	172.31.39.43	192.168.50.1	UDP	74	55734 → 33434 Len=32
	410 15.796646658	172.31.39.43	192.168.50.1	UDP	74	60739 → 33435 Len=32
	411 15.796671906	172.31.39.43	192.168.50.1	UDP	74	36990 → 33436 Len=32
	412 15.796702705	172.31.39.43	192.168.50.1	UDP	74	43897 → 33437 Len=32
	413 15.796722592	172.31.39.43	192.168.50.1	UDP	74	55766 → 33438 Len=32
	414 15.796742418	172.31.39.43	192.168.50.1	UDP	74	60442 → 33439 Len=32
	415 15.796761746	172.31.39.43	192.168.50.1	UDP	74	51764 → 33440 Len=32
	416 15.796778809	172.31.39.43	192.168.50.1	UDP	74	55446 → 33441 Len=32
	417 15.796795801	172.31.39.43	192.168.50.1	UDP	74	37853 → 33442 Len=32
	418 15.796812312	172.31.39.43	192.168.50.1	UDP	74	59056 → 33443 Len=32
	419 15.796828784	172.31.39.43	192.168.50.1	UDP	74	45023 → 33444 Len=32
	420 15.796845365	172.31.39.43	192.168.50.1	UDP	74	35240 → 33445 Len=32
	421 15.796861556	172.31.39.43	192.168.50.1	UDP	74	37681 → 33446 Len=32
	422 15.796877686	172.31.39.43	192.168.50.1	UDP	74	55182 → 33447 Len=32
	423 15.796893927	172.31.39.43	192.168.50.1	UDP	74	43712 → 33448 Len=32
	424 15.796911280	172.31.39.43	192.168.50.1	UDP	74	60930 → 33449 Len=32
	425 15.819388748	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceeded (T
	426 15.819390909	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceeded (T
	427 15.819647631	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceeded (T
	428 15.819647741	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceeded (T
	429 15.819647791	172.17.0.3	172.31.39.43	ICMP	102	Time-to-live exceeded (T
	430 15.819647841	172.31.0.2	172.31.39.43	ICMP	70	Time-to-live exceeded (T
	432 15.820829862	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceeded (T
	433 15.820829993	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceeded (T
	434 15.821310975	220.158.144.33	172.31.39.43	ICMP	70	Time-to-live exceeded (T

Frame 409: 74 bytes on wire (592 bits), 74 bytes captured (592 bytes) on interface wlo1, id 0

Ethernet II, Src: CloudNetwork 60:dc:93:73 (d8:80:83:60:dc:93), Dst: IETF-VRRP-VRID 00 (00:00:5e:00:01:0a)

Internet Protocol Version 4, Src: 172.31.39.43, Dst: 192.168.50.1

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 60

Identification: 0x23e3 (9187)

0000 = Flags: 0x0

...0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 1

Protocol: UDP (17)

Header Checksum: 0xcfd4 [validation disabled]

[Header checksum status: Unverified]

Source Address: 172.31.39.43

Destination Address: 192.168.50.1

User Datagram Protocol, Src Port: 55734, Dst Port: 33434

Data (32 bytes)

14	0.986044275	172.31.39.43	192.168.50.1	UDP	74 59717 → 33436 Len=32
15	0.986072518	172.31.39.43	192.168.50.1	UDP	74 45868 → 33437 Len=32
16	0.986095602	172.31.39.43	192.168.50.1	UDP	74 51777 → 33438 Len=32
17	0.986113286	172.31.39.43	192.168.50.1	UDP	74 50546 → 33439 Len=32
18	0.986132211	172.31.39.43	192.168.50.1	UDP	74 40916 → 33440 Len=32
19	0.986151268	172.31.39.43	192.168.50.1	UDP	74 60192 → 33441 Len=32
20	0.986169181	172.31.39.43	192.168.50.1	UDP	74 47680 → 33442 Len=32
21	0.986187466	172.31.39.43	192.168.50.1	UDP	74 49067 → 33443 Len=32
22	0.986205861	172.31.39.43	192.168.50.1	UDP	74 43457 → 33444 Len=32
23	0.986224616	172.31.39.43	192.168.50.1	UDP	74 43738 → 33445 Len=32
24	0.986243262	172.31.39.43	192.168.50.1	UDP	74 56762 → 33446 Len=32
25	0.986261166	172.31.39.43	192.168.50.1	UDP	74 34267 → 33447 Len=32
26	0.986280773	172.31.39.43	192.168.50.1	UDP	74 58124 → 33448 Len=32
27	0.986298446	172.31.39.43	192.168.50.1	UDP	74 46702 → 33449 Len=32
28	1.005180797	172.17.0.3	172.31.39.43	ICMP	102 Time-to-live exceeded (
29	1.005181178	172.31.0.2	172.31.39.43	ICMP	70 Time-to-live exceeded (
30	1.005181278	172.17.0.3	172.31.39.43	ICMP	102 Time-to-live exceeded (
31	1.005181348	172.17.0.3	172.31.39.43	ICMP	102 Time-to-live exceeded (
32	1.005181429	172.31.0.2	172.31.39.43	ICMP	70 Time-to-live exceeded (
33	1.005181499	172.31.0.2	172.31.39.43	ICMP	70 Time-to-live exceeded (
35	1.005969361	220.158.144.33	172.31.39.43	ICMP	70 Time-to-live exceeded (
36	1.005969431	220.158.144.33	172.31.39.43	ICMP	70 Time-to-live exceeded (
37	1.006757243	220.158.144.33	172.31.39.43	ICMP	70 Time-to-live exceeded (
44	1.013339419	103.120.29.90	172.31.39.43	ICMP	70 Time-to-live exceeded (
48	1.016955849	172.31.39.43	192.168.50.1	UDP	74 51290 → 33450 Len=32
49	1.016977369	172.31.39.43	192.168.50.1	UDP	74 54055 → 33451 Len=32
50	1.016994732	172.31.39.43	192.168.50.1	UDP	74 34843 → 33452 Len=32
51	1.017012225	172.31.39.43	192.168.50.1	UDP	74 41202 → 33453 Len=32
52	1.017031843	172.31.39.43	192.168.50.1	UDP	74 45011 → 33454 Len=32
53	1.017061819	172.31.39.43	192.168.50.1	UDP	74 43613 → 33455 Len=32
55	1.019397473	103.120.29.90	172.31.39.43	ICMP	70 Time-to-live exceeded (
57	1.020561547	172.31.39.43	192.168.50.1	UDP	74 48997 → 33456 Len=32
58	1.020582156	172.31.39.43	192.168.50.1	UDP	74 51764 → 33457 Len=32
59	1.020599980	172.31.39.43	192.168.50.1	UDP	74 49261 → 33458 Len=32
60	1.020617473	172.31.39.43	192.168.50.1	UDP	74 45249 → 33459 Len=32
61	1.020671245	172.31.39.43	192.168.50.1	UDP	74 58433 → 33460 Len=32
62	1.021349279	103.120.29.90	172.31.39.43	ICMP	70 Time-to-live exceeded (
63	1.021386760	172.31.39.43	192.168.50.1	UDP	74 50882 → 33461 Len=32
68	1.274101878	172.31.39.43	192.168.50.1	UDP	74 57769 → 33462 Len=32
69	1.274149207	172.31.39.43	192.168.50.1	UDP	74 52545 → 33463 Len=32
70	1.274181108	172.31.39.43	192.168.50.1	UDP	74 54790 → 33464 Len=32
114	5.988516673	172.31.39.43	192.168.50.1	UDP	74 37017 → 33465 Len=32
119	6.017680515	172.31.39.43	192.168.50.1	UDP	74 53045 → 33466 Len=32
120	6.017700393	172.31.39.43	192.168.50.1	UDP	74 54308 → 33467 Len=32
121	6.017716163	172.31.39.43	192.168.50.1	UDP	74 57193 → 33468 Len=32
122	6.017731431	172.31.39.43	192.168.50.1	UDP	74 35480 → 33469 Len=32
123	6.017745839	172.31.39.43	192.168.50.1	UDP	74 46646 → 33470 Len=32
124	6.017761178	172.31.39.43	192.168.50.1	UDP	74 55599 → 33471 Len=32
125	6.020934357	172.31.39.43	192.168.50.1	UDP	74 60957 → 33472 Len=32
126	6.020960927	172.31.39.43	192.168.50.1	UDP	74 50496 → 33473 Len=32

Thus wireshark helps in troubleshooting at which port it is giving error and not able to move forward and from above it is port 103.120.39.90 after which no icmp protocol is followed to our src port

ICMP stands for Internet Control Message Protocol. It's a network layer protocol in the Internet Protocol (IP) suite, primarily used for diagnostic and control purposes. ICMP is used by network devices, such as routers and hosts, to communicate error messages, status updates, and other control information to one another.

Packet Inspection: Wireshark allows for the capture and analysis of network packets in real-time or from stored capture files. By inspecting packet payloads, headers, and metadata, security analysts can identify anomalies or patterns indicative of security threats.

Protocol Analysis: Wireshark supports the decoding and analysis of various network protocols, including TCP, UDP, ICMP, DNS, HTTP, and more. Analyzing protocol interactions can reveal abnormalities or unauthorized behaviors that may indicate security breaches or malicious activities.

Signature Detection: Wireshark can be used to apply signature-based detection techniques to identify known patterns associated with malware infections, suspicious activities, or common attack vectors. This involves comparing captured packets against predefined signatures or rulesets to flag potential threats.

Behavioral Analysis: Wireshark enables security analysts to conduct behavioral analysis by monitoring network traffic over time and identifying deviations from normal patterns. Sudden spikes in traffic, unusual communication patterns, or unexpected protocol usage may indicate security incidents or unauthorized access attempts.

Anomaly Detection: Wireshark supports anomaly detection techniques to identify irregularities in network traffic that may signify security threats. This includes detecting unusual packet sizes, high rates of packet loss, unexpected port usage, or abnormal communication flows.

DNS Resolution: Wireshark can capture and analyze DNS traffic, allowing analysts to detect DNS-based attacks such as DNS spoofing, DNS hijacking, or domain generation algorithms (DGAs) used by malware for command and control.

Encryption Inspection: While Wireshark cannot decrypt encrypted traffic without access to encryption keys, it can still analyze encrypted protocols and metadata. Analysts can look for signs of encrypted communication that may be indicative of data exfiltration or unauthorized access.

Traffic Profiling: Wireshark facilitates traffic profiling by categorizing network traffic based on protocols, source/destination IP addresses, port numbers, and other attributes. This helps identify abnormal traffic patterns or unexpected network behaviors that may indicate security threats.

Forensic Analysis: Wireshark can be used for forensic analysis of network traffic to reconstruct events, trace the origins of security incidents, and gather evidence for incident response or legal purposes.

Actions that I will take to tackle malware threats:

Analyze captured packets in Wireshark to identify patterns associated with known malware infections or suspicious activities.

Implement firewall rules or intrusion prevention systems (IPS) to block traffic originating from malicious IP addresses or exhibiting malicious behavior.

Analyze network traffic in Wireshark to identify plaintext transmission of sensitive data, such as login credentials or confidential information.

Implement encryption protocols (e.g., TLS/SSL) to encrypt sensitive data in transit and protect against eavesdropping or data interception.