

ANALYZING MY FIRST PACKET A LAB BY DARNELL MAIDEN

The screenshot displays a Wireshark capture of network traffic. The main pane shows a list of packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The traffic includes several ICMP Echo (ping) requests and replies, a TCP SYN-ACK handshake, and an HTTP GET request.

No.	Time	Source	Destination	Protocol	Length	Info
16	8.642690	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=1/256, ttl=64 (reply in 18)
18	8.643923	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=1/256, ttl=115 (request in 16)
25	9.644712	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=2/512, ttl=64 (reply in 26)
26	9.645078	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=2/512, ttl=115 (request in 25)
31	10.646049	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=3/768, ttl=64 (reply in 32)
32	10.646563	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=3/768, ttl=115 (request in 31)
64	18.032768	172.21.224.2	142.250.1.139	TCP	74	49652 → 80 [SYN] Seq=0 Win=65536 Len=0 MSS=1420 SACK_PERM TSval=2804123005 TSecr=0 WS=128
65	18.034210	142.250.1.139	172.21.224.2	TCP	74	80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM TSval=4069674930 TSecr=2804123005 WS=...
66	18.034238	172.21.224.2	142.250.1.139	TCP	66	49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2804123006 TSecr=4069674930
67	18.034291	172.21.224.2	142.250.1.139	HTTP	151	GET / HTTP/1.1
68	18.034724	142.250.1.139	172.21.224.2	HTTP	66	80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSval=4069674931 TSecr=2804123006
69	18.035927	142.250.1.139	172.21.224.2	HTTP	648	HTTP/1.1 301 Moved Permanently (text/html)

The packet details pane for packet 16 shows the following structure:

- Frame 16: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
- Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:01:ac:15:e0:01 (42:01:ac:15:e0:01)
- Internet Protocol Version 4, Src: 172.21.224.2, Dst: 142.250.1.139
- Internet Control Message Protocol

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 45 00 B-----B-----E-
0010 00 54 06 22 40 00 40 01 17 ea ac 15 e0 02 8e fa .T.@-----
0020 01 8b 08 00 42 fe 68 31 00 01 41 14 7e 63 00 00 .-B-h1-A-----
0030 00 00 cb 84 03 00 00 00 00 00 10 11 12 13 14 15 .....!-#5$
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .....+,-./012345
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 8()*+,-./012345
0060 36 37
```

[SUMMARY]:

AFTER GETTING MY BADGE FOR ENTRY, MY ONBOARDING PAPER WORK, AND A FLIRTATIOUS GOODBYE FROM DELORIS FROM THE HR DEPT. (YEAH, REAL CONUDRUM RIGHT?), I FINALLY GOT MY FIRST ASSIGNMENT. THE LEAD WANTED ME TO ANALYZE PACKETS TO TEST MY ANALYST SKILLS AND SEE WHERE I FIT IN THE COMPANY. SO I SAT AT MY NEW DESK, OPENED UP MY WINDOWS VIRTUAL MACHINE, AND GOT TO WORK.

[OPENING WIRESHARK]:



Recycle Bin



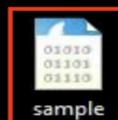
Google
Cloud S...



NVDA

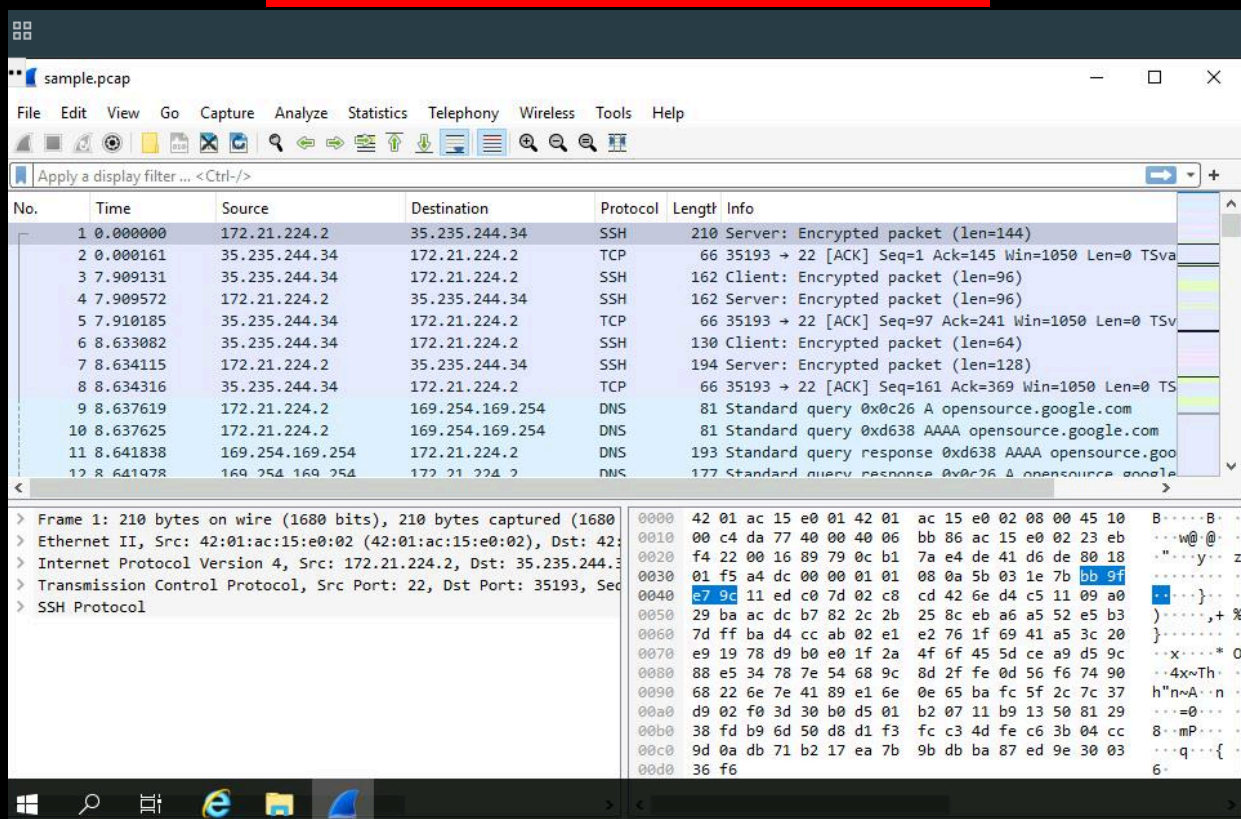


Wireshark



I OPENED UP MY **[VIRTUAL MACHINE]** AND FOUND THE SAMPLE PCAP FILE THEY SENT TO ME A FEW MINUTES AGO. I CLICK ON IT AND IT DISPLAYS THE DATA FROM A SYSTEM THAT MADE WEB REQUESTS TO A SITE. I OPENED UP **[WIRESHARK]** TO DISPLAY THE DATA.

[ANALYZE THE DATA]:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.21.224.2	35.235.244.34	SSH	210	Server: Encrypted packet (len=144)
2	0.000161	35.235.244.34	172.21.224.2	TCP	66	35193 → 22 [ACK] Seq=1 Ack=145 Win=1050 Len=0 TSva
3	7.909131	35.235.244.34	172.21.224.2	SSH	162	Client: Encrypted packet (len=96)
4	7.909572	172.21.224.2	35.235.244.34	SSH	162	Server: Encrypted packet (len=96)
5	7.910185	35.235.244.34	172.21.224.2	TCP	66	35193 → 22 [ACK] Seq=97 Ack=241 Win=1050 Len=0 TSv
6	8.633082	35.235.244.34	172.21.224.2	SSH	130	Client: Encrypted packet (len=64)
7	8.634115	172.21.224.2	35.235.244.34	SSH	194	Server: Encrypted packet (len=128)
8	8.634316	35.235.244.34	172.21.224.2	TCP	66	35193 → 22 [ACK] Seq=161 Ack=369 Win=1050 Len=0 TS
9	8.637619	172.21.224.2	169.254.169.254	DNS	81	Standard query 0xd638 A opensource.google.com
10	8.637625	172.21.224.2	169.254.169.254	DNS	81	Standard query 0xd638 AAAA opensource.google.com
11	8.641838	169.254.169.254	172.21.224.2	DNS	193	Standard query response 0xd638 AAAA opensource.goo
12	8.641978	169.254.169.254	172.21.224.2	DNS	177	Standard query response 0xd638 A opensource.goo

ONCE **[WIRESHARK]** WAS OPEN I TOOK A MOMENT TO REFRESH MYSELF ON THE COMPONENTS OF EACH PACKET, WHICH IS:

- **NO.:** THE INDEX OF THE NUMBER PACKET IN THIS PACKET CAPTURE FILE.
- **TIME:** THE TIMESTAMP OF THE PACKET.
- **SOURCE:** THE SOURCE IP ADDRESS.

- **DESTINATION:** THE DESTINATION IP ADDRESS.
- **PROTOCOL:** THE PROTOCOL CONTAINED IN THE PACKET
- **LENGTH:** THE TOTAL LENGTH OF THE PACKET
- **INFO:** SOME INFORMATION ABOUT THE DATA IN THE PACKET (THE PAYLOAD) . AS INTERPRETED BY WIRESHARK.

No.	Time	Source	Destination	Protocol	Length	Info
16	8.642690	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=1/256, ttl=64 (reply in 18)
18	8.643923	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=1/256, ttl=115 (request in 16)
25	9.644712	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=2/512, ttl=64 (reply in 26)
26	9.645078	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=2/512, ttl=115 (request in 25)
31	10.646049	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=3/768, ttl=64 (reply in 32)
32	10.646563	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=3/768, ttl=115 (request in 31)
64	18.032768	172.21.224.2	142.250.1.139	TCP	74	49652 → 80 [SYN] Seq=0 Win=65320 Len=0 MSS=1420 SACK_PERM TSval=2804123005 TSecr=0 WS=128
65	18.034210	142.250.1.139	172.21.224.2	TCP	74	80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM TSval=4069674930 TSecr=2804123005 WS=...
66	18.034238	172.21.224.2	142.250.1.139	TCP	66	49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2804123006 TSecr=4069674930
67	18.034291	172.21.224.2	142.250.1.139	HTTP	151	GET / HTTP/1.1
68	18.034724	142.250.1.139	172.21.224.2	TCP	66	80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSval=4069674931 TSecr=2804123006

PACKETS COME IN SPECIFIC COLORS TO CLASSIFY THE SPECIFIC TYPES OF DATA. FOR EXAMPLE, [LIGHT GREEN] CONTAINS A MIXTURE OF [TCP] AND [HTTP] TRAFFIC AND [LIGHT PURPLE] CONTAINS [ICMP] TRAFFIC.

No.	Time	Source	Destination	Protocol	Length	Info
16	8.642690	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=1/256, ttl=64 (reply in 18)
18	8.643923	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=1/256, ttl=115 (request in 16)
25	9.644712	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=2/512, ttl=64 (reply in 26)
26	9.645078	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=2/512, ttl=115 (request in 25)
31	10.646049	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=3/768, ttl=64 (reply in 32)
32	10.646563	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=3/768, ttl=115 (request in 31)
64	18.032768	172.21.224.2	142.250.1.139	TCP	74	49652 → 80 [SYN] Seq=0 Win=65320 Len=0 MSS=1420 SACK_PERM TSval=2804123005 TSecr=0 WS=128
65	18.034210	142.250.1.139	172.21.224.2	TCP	74	80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM TSval=4069674930 TSecr=2804123005 WS=...
66	18.034238	172.21.224.2	142.250.1.139	TCP	66	49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2804123006 TSecr=4069674930
67	18.034291	172.21.224.2	142.250.1.139	HTTP	151	GET / HTTP/1.1
68	18.034724	142.250.1.139	172.21.224.2	TCP	66	80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSval=4069674931 TSecr=2804123006

IF YOU LOOK CLOSELY AT THE FIRST PACKET AT THE TOP, IT BELONGS TO A GROUP OF ICMP TRAFFIC.

ICMP (INTERNET CONTROL MESSAGE PROTOCOL) IS A NETWORK LAYER PROTOCOL USED BY ROUTERS AND DEVICES TO SEND ERROR MESSAGES AND OPERATIONAL INFORMATION.

[FILTER AND INSPECT]:

AFTER GETTING A REFRESHER ON THE COMPONENTS AND DATA LAYOUTS I WENT TO THE “APPLY A DISPLAY FILTER” BOX AND TYPED IN A IP ADDRESS THAT WAS GIVEN TO ME BY THE COMPANY.

[IP.ADDR == 142.250.1.139]

The screenshot shows the Wireshark interface with a packet capture filter applied: `ip.addr == 142.250.1.139`. The packet list pane displays 16 filtered packets. The selected packet (No. 16) is an ICMP Echo (ping) request from 172.21.224.2 to 142.250.1.139. The packet details pane shows the structure of the ICMP Echo request, including the Ethernet II header, Internet Protocol Version 4 header, and Internet Control Message Protocol header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
16	8.642690	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=1/256, ttl=64 (reply in 18)
18	8.643923	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=1/256, ttl=115 (request in 16)
25	9.644712	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=2/512, ttl=64 (reply in 26)
26	9.645078	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=2/512, ttl=115 (request in 25)
31	10.646049	172.21.224.2	142.250.1.139	ICMP	98	Echo (ping) request id=0x6831, seq=3/768, ttl=64 (reply in 32)
32	10.646563	142.250.1.139	172.21.224.2	ICMP	98	Echo (ping) reply id=0x6831, seq=3/768, ttl=115 (request in 31)
64	18.032768	172.21.224.2	142.250.1.139	TCP	74	49652 → 80 [SYN] Seq=0 Win=65320 Len=0 MSS=1420 SACK_PERM TSval=2804123005 TSecr=0 WS=128
65	18.034210	142.250.1.139	172.21.224.2	TCP	74	80 → 49652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM TSval=4069674930 TSecr=2804123005 WS=...
66	18.034238	172.21.224.2	142.250.1.139	TCP	66	49652 → 80 [ACK] Seq=1 Ack=1 Win=65408 Len=0 TSval=2804123006 TSecr=4069674930
67	18.034291	172.21.224.2	142.250.1.139	HTTP	151	GET / HTTP/1.1
68	18.034724	142.250.1.139	172.21.224.2	TCP	66	80 → 49652 [ACK] Seq=1 Ack=86 Win=65536 Len=0 TSval=4069674931 TSecr=2804123006

AFTER ENTERING MY SEARCH I HEADED TO THE FIRST TCP ENTRY FOR THE IP ADDRESS GIVEN AND EXAMINED IT CAREFULLY USING THE BOTTOM LEFT WINDOW. THESE ARE THE COMPONENTS:

[FRAME]:

THIS IS THE FRAME. THESE ARE THE OVERALL DETAILS OF THE PACKET SUCH AS THE FRAME RATE AND ARRIVAL TIME OF THE PACKET ALONG WITH THE ENTIRETY OF THE PACKET DATA PRESENTED.

```
▼ Frame 64: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 23, 2022 12:38:34.620693000 Greenwich Standard Time
  UTC Arrival Time: Nov 23, 2022 12:38:34.620693000 UTC
  Epoch Arrival Time: 1669207114.620693000
  [Time shift for this packet: 0.000000000 seconds]
  [Time delta from previous captured frame: 0.000389000 seconds]
  [Time delta from previous displayed frame: 7.386205000 seconds]
  [Time since reference or first frame: 18.032768000 seconds]
  Frame Number: 64
  Frame Length: 74 bytes (592 bits)
  Capture Length: 74 bytes (592 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp]
  [Coloring Rule Name: HTTP]
  [Coloring Rule String: http || tcp.port == 80 || http2]
```

[ETHERNET II]:

THIS SECTION CONTAINS THE DETAILS ABOUT THE PACKET AT THE ETHERNET LEVEL, INCLUDING THE SOURCE AND DESTINATION MAC ADDRESSES AND THE TYPE OF INTERNAL PROTOCOL THAT THE ETHERNET PACKET CONTAINS.

```
▼ Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 42:01:ac:15:e0:01 (42:01:ac:15:e0:01)
  > Destination: 42:01:ac:15:e0:01 (42:01:ac:15:e0:01)
  > Source: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02)
  Type: IPv4 (0x0800)
```

[IPV4]:

THIS IS THE IP DATA SECTION WHICH CONTAINS INFORMATION ABOUT THE TIME TO LIVE, THE PROTOCOL, SOURCE ADDRESS, AND THE DESTINATION ADDRESS AS WELL.

```
Internet Protocol Version 4, Src: 172.21.224.2, Dst: 142.250.1.139
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0xe4a8 (58536)
  > 010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: TCP (6)
  Header Checksum: 0x3976 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.21.224.2
  Destination Address: 142.250.1.139
```

[PROTOCOL INFORMATION]:

THIS SECTION CONTAINS PROTOCOL INFORMATION FOR THE PACKET YOU'RE INSPECTING SUCH AS THE SOURCE PORTS, DESTINATION PORTS, ETC.

```
Transmission Control Protocol, Src Port: 49652, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 49652
  Destination Port: 80
  [Stream index: 4]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3412824992
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
```

[FLAG INFORMATION]:

THIS SECTION DISPLAYS THE PROTOCOL FLAG INFORMATION.

```

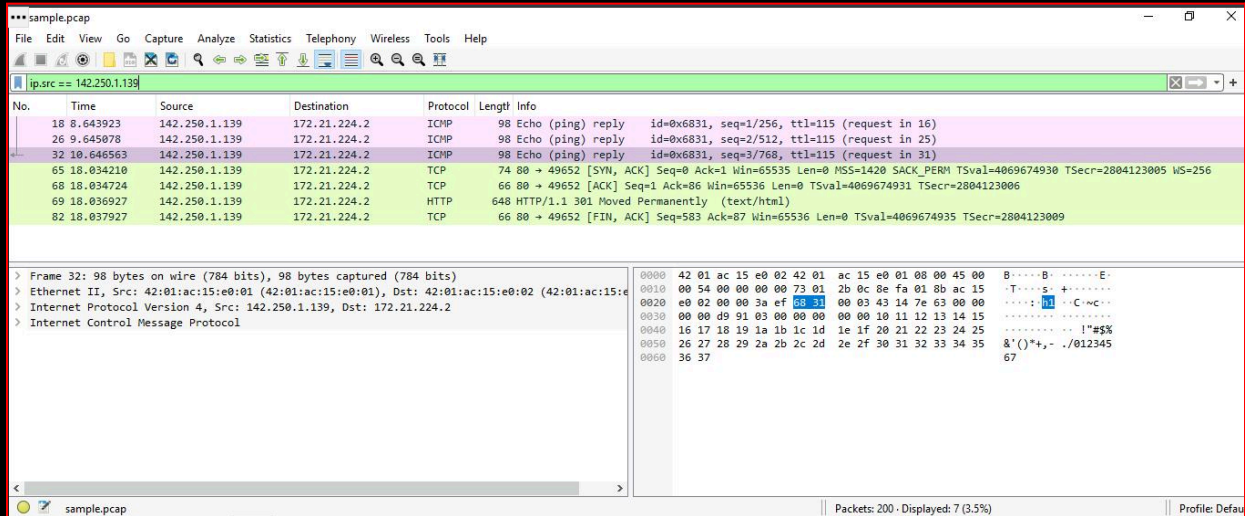
▼ Flags: 0x002 (SYN)
  000. .... = Reserved: Not set
  ...0 .... = Accurate ECN: Not set
  .... 0... = Congestion Window Reduced: Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .....0... = Acknowledgment: Not set
  ..... 0... = Push: Not set
  ..... .0.. = Reset: Not set
  > ..... ..1. = Syn: Set
  ..... ...0 = Fin: Not set
  [TCP Flags: .....S.]
Window: 65320
[Calculated window size: 65320]
Checksum: 0x1ccc [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP)
> [Timestamps]

```

[FILTER AND SELECT]:

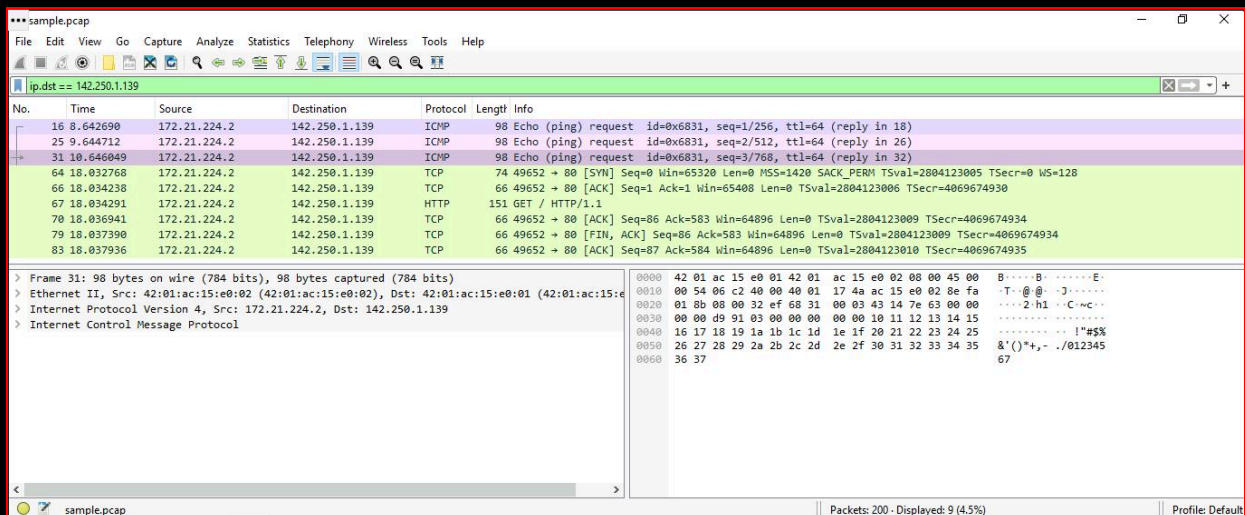
MY NEXT ASSIGNMENT IS ANALYZING SPECIFIC NETWORK PACKETS BASED ON WHERE THEY CAME FROM OR WHERE THEY WERE SENT TO. I'LL BE SELECTING PACKETS BASED ON THEIR PHYSICAL ETHERNET MEDIA ACCESS CONTROL (MAC) ADDRESS OR THEIR INTERNET PROTOCOL (IP) ADDRESS. LET'S TAKE THIS IP SOURCE HERE FOR EXAMPLE.

```
[IP.SRC == 142.250.1.139]
```



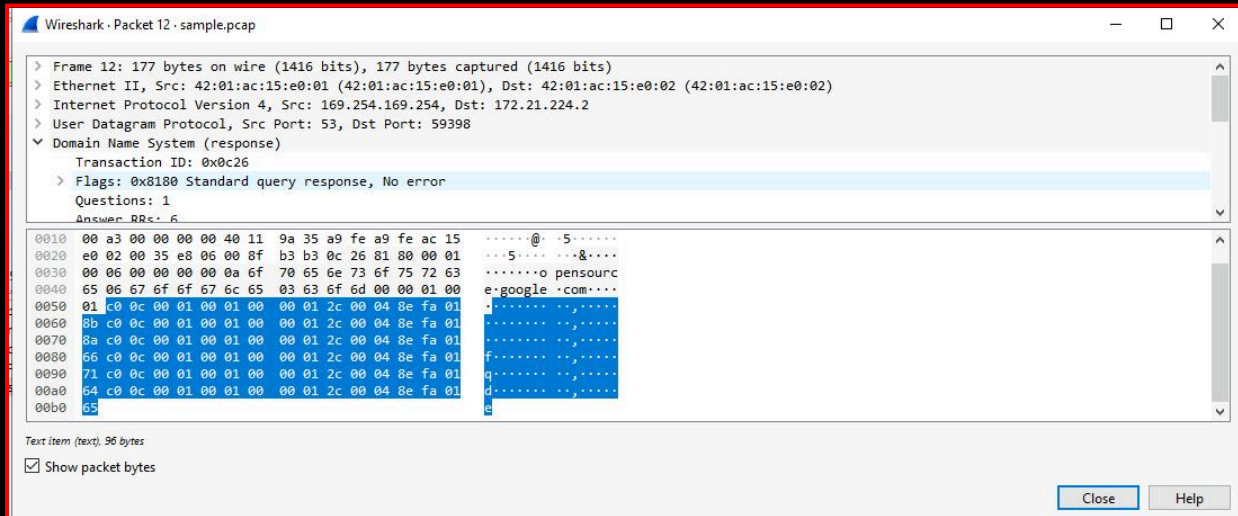
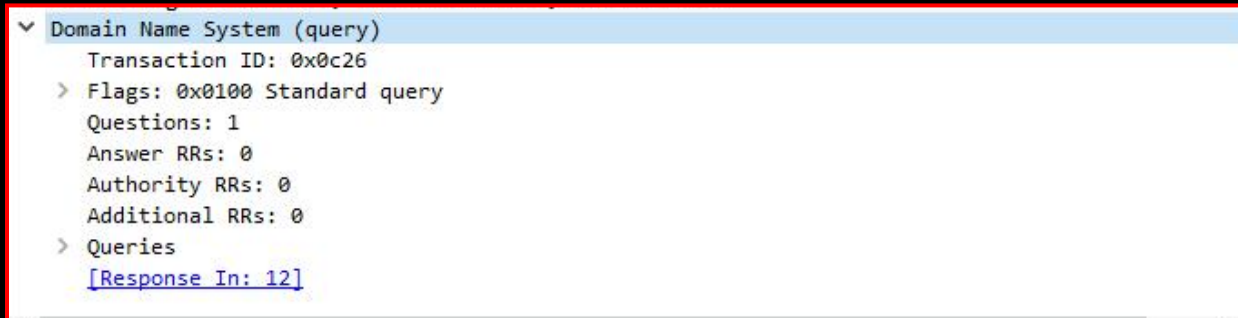
I HAVE THE OVERVIEW OF EVERY ENTRY THAT CAME FROM
142.250.1.139
 I CAN USE THIS INFORMATION FOR ANY ANALYSIS , SUCH AS UNUSUAL SOURCE
 TRAFFIC.

NOW LET'S FILTER THAT SAME IP FOR THE DESTINATION TRAFFIC.
[IP.DST == 142.250.1.139]

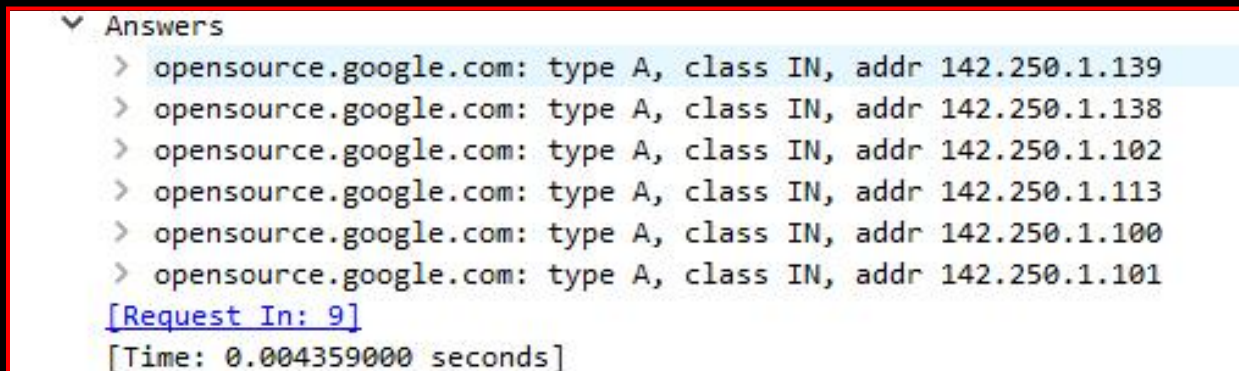
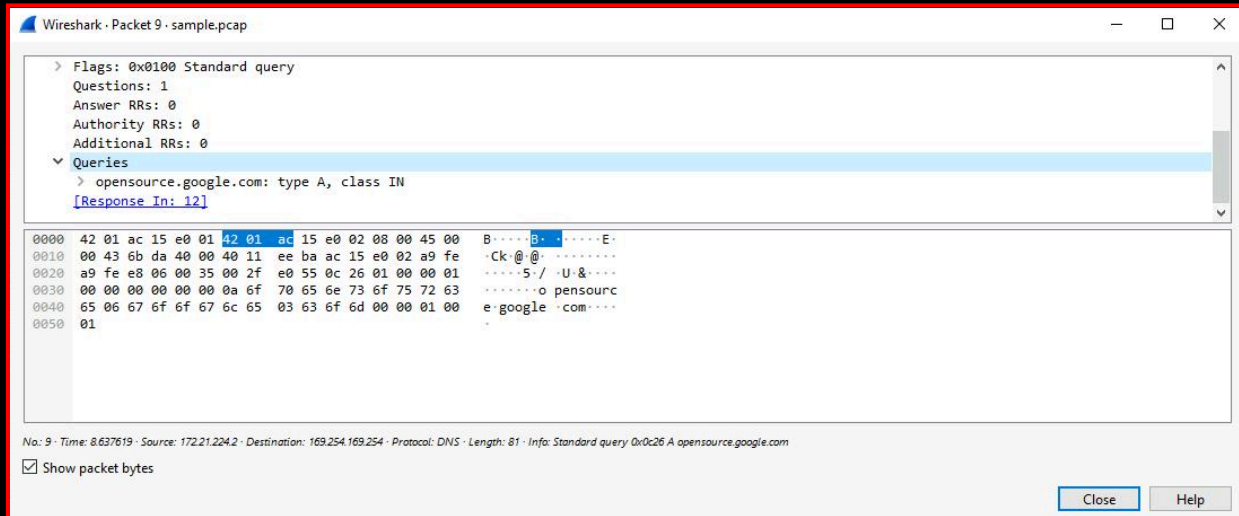


NOW IT CONTAINS ALL THE PACKETS THAT WE'RE SENT TO
142.250.1.139

THIS IS A GOOD START BUT I NEED THE TO KNOW WHERE THE SITE IS, SO I DOUBLE CLICK THE **DOMAIN NAME SYSTEM (QUERY)** , THEN CLICK THE QUERIES SECTION...



....AND VOLIA, THE ANSWERS DATA SHOWS ME THAT THE QUERIED NAME WAS **OPENSOURCE.GOOGLE.COM** AND THE IP ADDRESS WAS 142.250.1.139

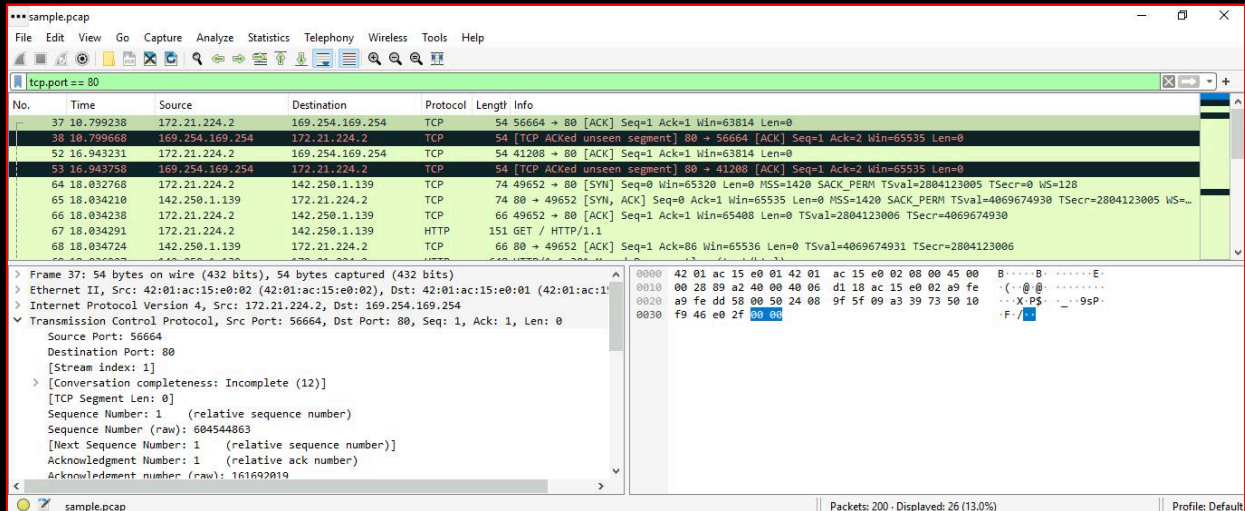


[TCP EXPLORATION]

THAT DNS DATA DIDN'T SIT RIGHT WITH ME. SO I DID MORE DIGGING AND FILTERED THROUGH TCP PACKETS AS WELL. MY LEAD GAVE ME THIS TO SEARCH

FOR:

TCP.PORT == 80



A FEW PACKETS WERE MADE AFTER THE USER ACCESSED THE TARGET SITE. I CLICKED THE FIRST PACKET TO EXAMINE THE COMPONENTS OF IT.

- **THE DESTINATION IP: [169.254.169.254]**
- **THE SOURCE IP: [172.21.224.2]**
- **THE TIME TO LIVE VALUE: [64]**
- **HEADER LENGTH: [20 BYTES]**

```
Internet Protocol Version 4, Src: 172.21.224.2, Dst: 169.254.169.254
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x89a2 (35234)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xd118 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.21.224.2
    Destination Address: 169.254.169.254
  > Transmission Control Protocol, Src Port: 56664, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 45 00  B.....B.....E.
0010 00 28 89 a2 40 00 40 06 d1 18 ac 15 e0 02 a9 fe  .(..@.@.....
0020 a9 fe dd 58 00 50 24 08 9f 5f 09 a3 39 73 50 10  ...X.P$. _...9sP.
0030 f9 46 e0 2f 00 00                                -F./..
```

● THE FRAME LENGTH: [54 BYTES]

```
Frame 37: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 23, 2022 12:38:27.387163000 Greenwich Standard Time
  UTC Arrival Time: Nov 23, 2022 12:38:27.387163000 UTC
  Epoch Arrival Time: 1669207107.387163000
  [Time shift for this packet: 0.000000000 seconds]
  [Time delta from previous captured frame: 0.151417000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 10.799238000 seconds]
  Frame Number: 37
  Frame Length: 54 bytes (432 bits)
  Capture Length: 54 bytes (432 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp]
  [Coloring Rule Name: HTTP]
  [Coloring Rule String: http || tcp.port == 80 || http2]
```

AFTER GIVING THIS INFORMATION TO THE LEAD, THEY HAD ONE MORE TASK FOR ME. THEY WANTED ME TO SEARCH FOR A SPECIFIC PACKET THAT CONTAINED THE CURL COMMAND. IT CAME BACK WITH THE TWO RESULTS BELOW.

The screenshot shows the Wireshark interface with a search filter 'tcp contains "curl"'. Two packets are listed in the packet list pane:

No.	Time	Source	Destination	Protocol	Length	Info
67	18.034291	172.21.224.2	142.250.1.139	HTTP	151	GET / HTTP/1.1
148	42.369093	172.21.224.2	142.250.1.102	HTTP	151	GET / HTTP/1.1

The details pane for the first packet (No. 67) shows the following information:

- [Time delta from previous captured frame: ...]
- [Time delta from previous displayed frame: ...]
- [Time since reference or first frame: 18.034291]
- Frame Number: 67
- Frame Length: 151 bytes (1208 bits)
- Capture Length: 151 bytes (1208 bits)
- [Frame is marked: False]
- [Frame is ignored: False]
- [Protocols in frame: eth:ethertype:ip:tcp]
- [Coloring Rule Name: HTTP]
- [Coloring Rule String: http || tcp.port]
- > Ethernet II, Src: 42:01:ac:15:e0:02 (42:01:ac:15:e0:02), Dst: 02:00:14:00:00:00
- ▼ Internet Protocol Version 4, Src: 172.21.224.2, Dst: 142.250.1.139
- = Version: 4

The packet bytes pane shows the raw data of the packet:

```
0000 42 01 ac 15 e0 01 42 01 ac 15 e0 02 08 00 00 00
0010 00 89 e4 aa 40 00 40 06 39 27 ac 15 e0 02 00 00 00
0020 01 8b c1 f4 00 50 cb 6b 93 a1 60 64 ec 2d 00 00 00
0030 01 ff 1d 19 00 00 01 01 08 0a a7 23 85 7d 00 00 00
0040 4f b2 47 45 54 20 2f 20 48 54 54 50 2f 3e 00 00 00
0050 0d 0a 48 6f 73 74 3a 20 6f 70 65 6e 73 6f 6f 6f 6f
0060 63 65 2e 67 6f 6f 67 6c 65 2e 63 6f 6d 6f 6f 6f 6f
0070 73 65 72 2d 41 67 65 6e 74 3a 20 63 75 7d 00 00 00
0080 37 2e 37 34 2e 30 0d 0a 41 63 63 65 70 7d 00 00 00
0090 2a 2f 2a 0d 0a 0d 0a
```

[CONCLUSION]

AFTER GIVING ALL THE INFORMATION TO THE LEAD, THEY WERE QUITE IMPRESSED. THEY SAID PACKET SNIFFING IS A QUALITY SKILL THAT MANY SECURITY ANALYSTS NEED IN DIRE TIMES OF EXPLORATION AND DATA ANALYSIS. THE PURPOSE OF PACKET SNIFFING IS TO MONITOR TRAFFIC AND SEARCH FOR ANYTHING SUSPICIOUS AS WELL.

IF YOU ENJOYED THIS LAB, CLICK THE LINK AND SHOOT ME A FOLLOW OR A MESSAGE ON LINKEDIN. I'M HAPPY TO COLLABORATE AND WORK. THANK YOU FOR READING.

