Lab 05 – IP Filters

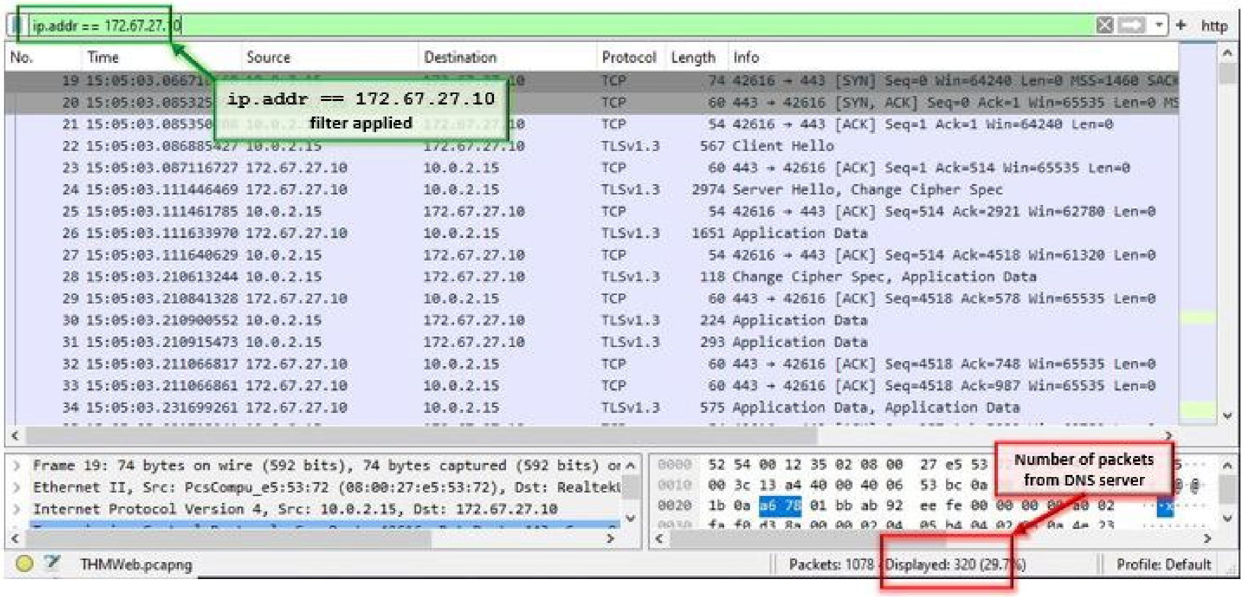
Author: Raymond Ng
Course Number/Section: IS 3413-006
Date: October 5, 2022

Introduction

The purpose of this lab was to allow the user to experiment with IP filters in Wireshark. Moreover, it allowed the user to become more familiarized with the use of Wireshark.

Process

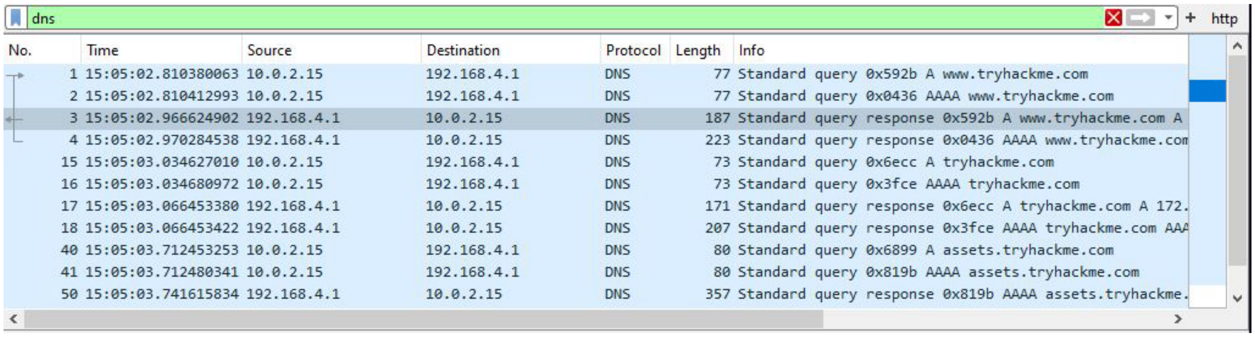
1) In the screen shot below (*Figure1*), I applied `ip.addr == 172.67.27.10` in Wireshark to filter all the packets to and from the host at `172.67.27.10`, resulting in `320` packets.



```
ip.addr == 172.67.27.10

No.  Time                Source          Destination     Protocol  Length  Info
  19 15:05:03.06671      10.0.2.15       172.67.27.10    TCP           74  42616 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK
  20 15:05:03.085325     [illegible]     [illegible]     TCP           60  443 → 42616 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
  21 15:05:03.085350     10.0.2.15       172.67.27.10    TCP           54  42616 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
  22 15:05:03.086885427  10.0.2.15       172.67.27.10    TLSv1.3      567  Client Hello
  23 15:05:03.087116727  172.67.27.10    10.0.2.15       TCP           60  443 → 42616 [ACK] Seq=1 Ack=514 Win=65535 Len=0
  24 15:05:03.111446469  172.67.27.10    10.0.2.15       TLSv1.3     2974  Server Hello, Change Cipher Spec
  25 15:05:03.111461785  10.0.2.15       172.67.27.10    TCP           54  42616 → 443 [ACK] Seq=514 Ack=2921 Win=62780 Len=0
  26 15:05:03.111633970  172.67.27.10    10.0.2.15       TLSv1.3     1651  Application Data
  27 15:05:03.111640629  10.0.2.15       172.67.27.10    TCP           54  42616 → 443 [ACK] Seq=514 Ack=4518 Win=61320 Len=0
  28 15:05:03.210613244  10.0.2.15       172.67.27.10    TLSv1.3      118  Change Cipher Spec, Application Data
  29 15:05:03.210841328  172.67.27.10    10.0.2.15       TCP           60  443 → 42616 [ACK] Seq=4518 Ack=578 Win=65535 Len=0
  30 15:05:03.210900552  10.0.2.15       172.67.27.10    TLSv1.3      224  Application Data
  31 15:05:03.210915473  10.0.2.15       172.67.27.10    TLSv1.3      293  Application Data
  32 15:05:03.211066817  172.67.27.10    10.0.2.15       TCP           60  443 → 42616 [ACK] Seq=4518 Ack=748 Win=65535 Len=0
  33 15:05:03.211066861  172.67.27.10    10.0.2.15       TCP           60  443 → 42616 [ACK] Seq=4518 Ack=987 Win=65535 Len=0
  34 15:05:03.231699261  172.67.27.10    10.0.2.15       TLSv1.3      575  Application Data, Application Data

> Frame 19: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) o
> Ethernet II, Src: PcsCompu_e5:53:72 (08:00:27:e5:53:72), Dst: Realtek
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 172.67.27.10

0000  52 54 00 12 35 02 08 00  27 e5 53
0010  00 3c 13 a4 40 00 40 06  53 bc 0a
0020  1b 0a a6 78 01 bb ab 92  ee fe 00 00 00 00 a0 02

THMWeb.pcapng        Packets: 1078 · Displayed: 320 (29.7%)        Profile: Default
```

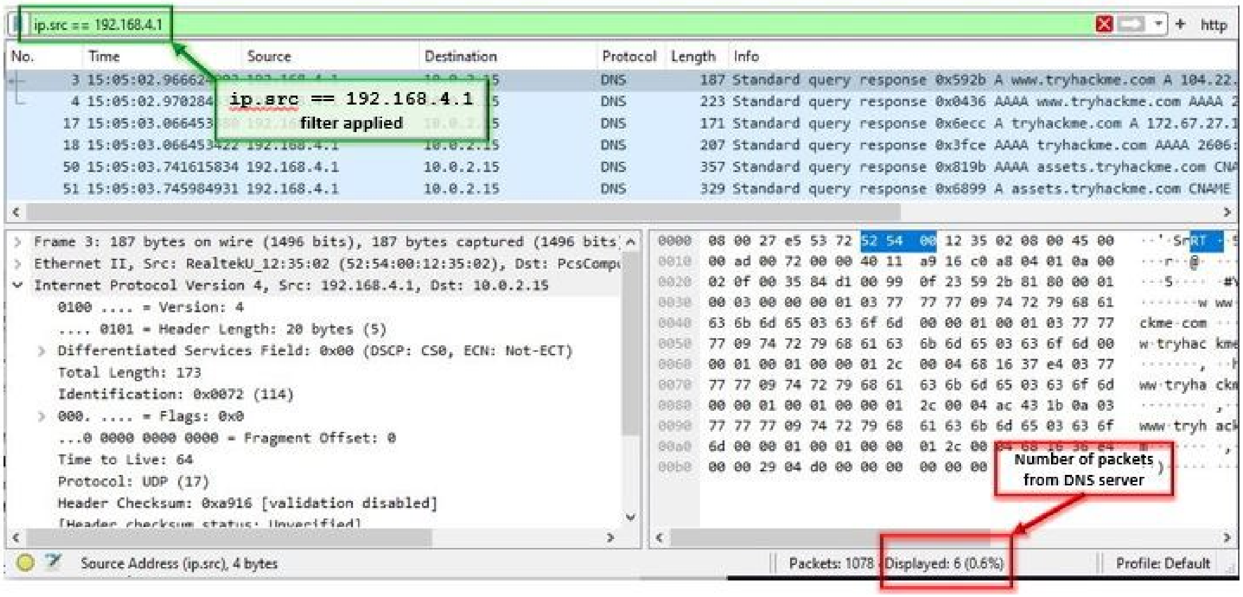
Figure 1: Filter applied in Wireshark filtering for all packets to and from host at 172.67.27.10, results located in the bottom right, `Displayed: 320`

2) In order to find the number of packets from the DNS server; first, I applied `dns` filter in Wireshark to filter for a DNS protocols (*Figure 2*). Next, I observed and analyzed the results that were replies coming back from that server. I observed `192.168.4.1` so I simply dragged one of the results from that source IP into the bar that filters packets and it produced `ip.src == 192.168.4.1` filter, resulting in `6` packets from DNS server (*Figure 3*).



```
No.   Time               Source        Destination   Protocol Length Info
    1 15:05:02.810380063 10.0.2.15     192.168.4.1   DNS        77 Standard query 0x592b A www.tryhackme.com
    2 15:05:02.810412993 10.0.2.15     192.168.4.1   DNS        77 Standard query 0x0436 AAAA www.tryhackme.com
    3 15:05:02.966624902 192.168.4.1   10.0.2.15     DNS       187 Standard query response 0x592b A www.tryhackme.com A
    4 15:05:02.970284538 192.168.4.1   10.0.2.15     DNS       223 Standard query response 0x0436 AAAA www.tryhackme.com
   15 15:05:03.034627010 10.0.2.15     192.168.4.1   DNS        73 Standard query 0x6ecc A tryhackme.com
   16 15:05:03.034680972 10.0.2.15     192.168.4.1   DNS        73 Standard query 0x3fce AAAA tryhackme.com
   17 15:05:03.066453380 192.168.4.1   10.0.2.15     DNS       171 Standard query response 0x6ecc A tryhackme.com A 172.
   18 15:05:03.066453422 192.168.4.1   10.0.2.15     DNS       207 Standard query response 0x3fce AAAA tryhackme.com AAA
   40 15:05:03.712453253 10.0.2.15     192.168.4.1   DNS        80 Standard query 0x6899 A assets.tryhackme.com
   41 15:05:03.712480341 10.0.2.15     192.168.4.1   DNS        80 Standard query 0x819b AAAA assets.tryhackme.com
   50 15:05:03.741615834 192.168.4.1   10.0.2.15     DNS       357 Standard query response 0x819b AAAA assets.tryhackme.
```

Figure 2: dns filter applied in Wireshark



```
No.   Time               Source        Destination   Protocol Length Info
    3 15:05:02.96662                              5  DNS       187 Standard query response 0x592b A www.tryhackme.com A 104.22.
    4 15:05:02.970284                             5  DNS       223 Standard query response 0x0436 AAAA www.tryhackme.com AAAA 2
   17 15:05:03.066453                             5  DNS       171 Standard query response 0x6ecc A tryhackme.com A 172.67.27.1
   18 15:05:03.066453422 192.168.4.1   10.0.2.15     DNS       207 Standard query response 0x3fce AAAA tryhackme.com AAAA 2606:
   50 15:05:03.741615834 192.168.4.1   10.0.2.15     DNS       357 Standard query response 0x819b AAAA assets.tryhackme.com CNA
   51 15:05:03.745984931 192.168.4.1   10.0.2.15     DNS       329 Standard query response 0x6899 A assets.tryhackme.com CNAME
```

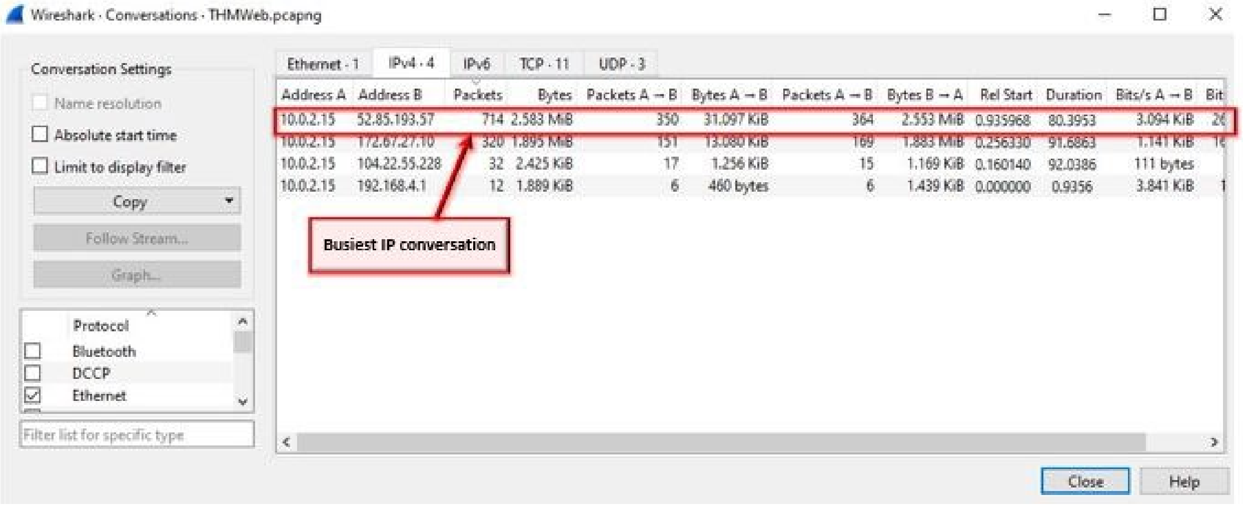
```
> Frame 3: 187 bytes on wire (1496 bits), 187 bytes captured (1496 bits
> Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu
v Internet Protocol Version 4, Src: 192.168.4.1, Dst: 10.0.2.15
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 173
    Identification: 0x0072 (114)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0xa916 [validation disabled]
    [Header checksum status: Unverified]
```

```
0000  08 00 27 e5 53 72 52 54  00 12 35 02 08 00 45 00   ··'·SrRT ··5
0010  00 ad 00 72 00 00 40 11  a9 16 c0 a8 04 01 0a 00   ···r··@· ····
0020  02 0f 00 35 84 d1 00 99  0f 23 59 2b 81 80 00 01   ···5···· ·#Y
0030  00 03 00 00 00 01 03 77  77 77 09 74 72 79 68 61   ·······w ww
0040  63 6b 6d 65 03 63 6f 6d  00 00 01 00 01 03 77 77   ckme·com ···
0050  77 09 74 72 79 68 61 63  6b 6d 65 03 63 6f 6d 00   w·tryhac kme
0060  00 01 00 01 00 00 01 2c  00 04 68 16 37 e4 03 77   ·······, ··h
0070  77 77 09 74 72 79 68 61  63 6b 6d 65 03 63 6f 6d   ww·tryha ckm
0080  00 00 01 00 01 00 00 01  2c 00 04 ac 43 1b 0a 03   ········ ,·
0090  77 77 77 09 74 72 79 68  61 63 6b 6d 65 03 63 6f   www·tryh ack
00a0  6d 00 00 01 00 01 00 00  01 2c 00
00b0  00 00 29 04 d0 00 00 00  00 00 00
```

Source Address (ip.src), 4 bytes | Packets: 1078 · Displayed: 6 (0.6%) | Profile: Default

Figure 3: Number of packets resulted after applying ip.src == 192.168.4.1 filter

3) To look for the busiest IP conversation in the given pcap file from TryHackMe. In Wireshark, I went to `Statistics`, clicked on `Conversations`. A new window generates. I then clicked on the `IPv4•4` Tab. I observed `714` packets for busiest IP conversation in the pcap file (*Figure 4*). From the same window I can apply a Filter and it will auto-generate a filter in the Wireshark to filter for the `714` packets.



Wireshark · Conversations · THMWeb.pcapng

Conversation Settings

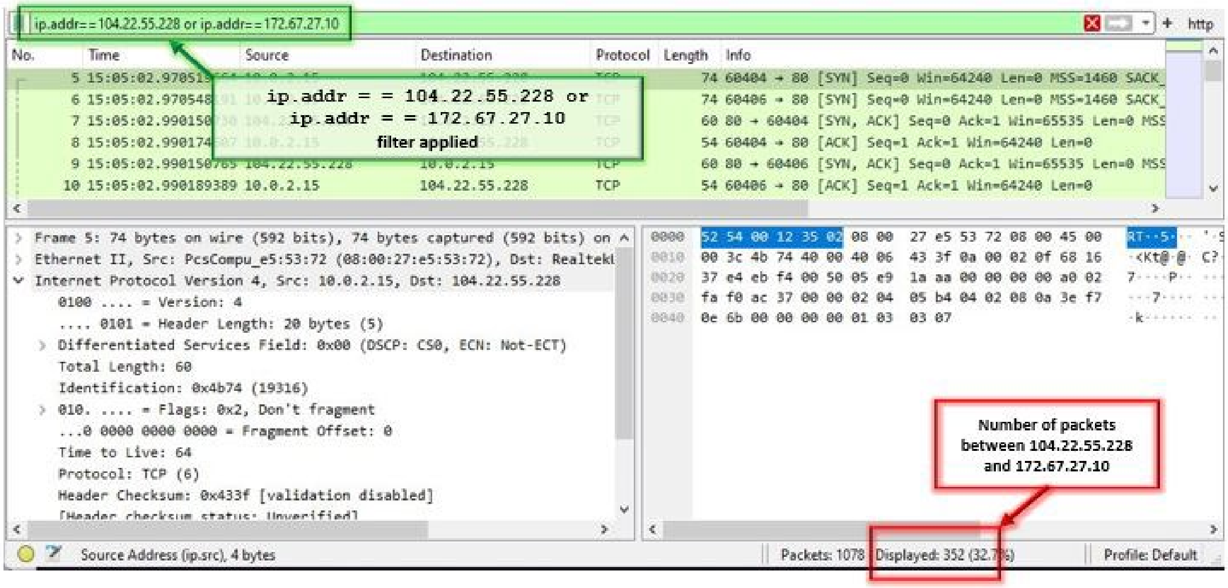
□ Name resolution
□ Absolute start time
□ Limit to display filter

	Protocol
□	Bluetooth
□	DCCP
☑	Ethernet

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets A → B	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bit
10.0.2.15	52.85.193.57	714	2.583 MiB	350	31.097 KiB	364	2.553 MiB	0.935968	80.3953	3.094 KiB	26
10.0.2.15	172.67.27.10	320	1.895 MiB	151	13.080 KiB	169	1.883 MiB	0.256330	91.6863	1.141 KiB	16
10.0.2.15	104.22.55.228	32	2.425 KiB	17	1.256 KiB	15	1.169 KiB	0.160140	92.0386	111 bytes	
10.0.2.15	192.168.4.1	12	1.889 KiB	6	460 bytes	6	1.439 KiB	0.000000	0.9356	3.841 KiB	1

Figure 4: Busiest IP conversation from PCAP file in Wireshark

4) Here, I filtered for all the traffic to and from `104.22.44.228` and `172.67.27.10`, resulting in `352` packets (*Figure 5*) by applying `ip.addr==104.22.55.228 or ip.addr==172.67.27.10`. Another way to execute the filter for the same results, per Chris Greer's YouTube video, is using the filter `ip.addr in {104.22.5.228, 172.67.27.10}` [1].



ip.addr==104.22.55.228 or ip.addr==172.67.27.10

```
No.  Time               Source          Destination     Protocol Length Info
   5 15:05:02.97051                                              74 60404 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
   6 15:05:02.970548                                              74 60406 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
   7 15:05:02.990150                                              60 80 → 60404 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS
   8 15:05:02.990174                                              54 60404 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
   9 15:05:02.990150765 104.22.55.228   10.0.2.15       TCP      60 80 → 60406 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS
  10 15:05:02.990189389 10.0.2.15       104.22.55.228   TCP      54 60406 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

```
> Frame 5: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on
> Ethernet II, Src: PcsCompu_e5:53:72 (08:00:27:e5:53:72), Dst: RealtekU
v Internet Protocol Version 4, Src: 10.0.2.15, Dst: 104.22.55.228
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x4b74 (19316)
  > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0x433f [validation disabled]
    [Header checksum status: Unverified]
```

```
0000  52 54 00 12 35 02 08 00  27 e5 53 72 08 00 45 00   RT··5···'·S
0010  00 3c 4b 74 40 00 40 06  43 3f 0a 00 02 0f 68 16   ·<Kt@·@·C?
0020  37 e4 eb f4 00 50 05 e9  1a aa 00 00 00 00 a0 02   7····P····
0030  fa f0 ac 37 00 00 02 04  05 b4 04 02 08 0a 3e f7   ···7······
0040  0e 6b 00 00 00 00 01 03  03 07                     ·k········
```

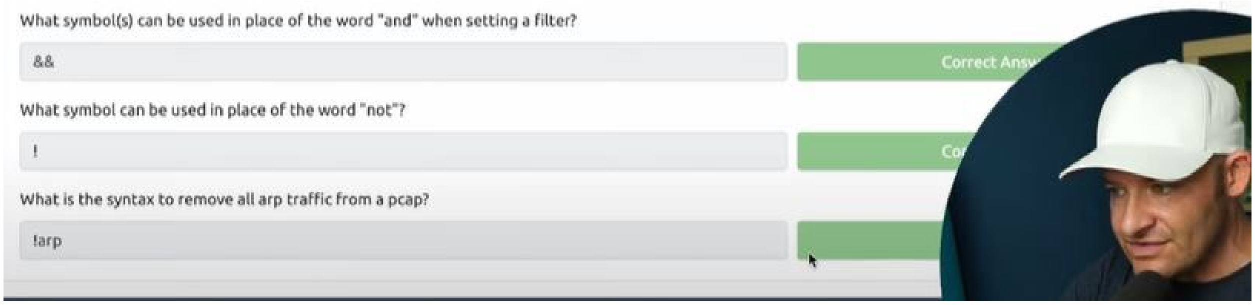
Source Address (ip.src), 4 bytes | Packets: 1078 | Displayed: 352 (32.7%) | Profile: Default

Figure 5: Filter applied to filter for the packets associated with the traffice to and from 104.22.55.228 and 172.67.27.10

5) The `&&` symbol can be used in place of the word "`and`" when setting a filter in Wireshark (*Figure 6*) [1].

6) The `!` symbol can be used in place of the word "`not`" when setting a filter in Wireshark (*Figure 6*) [1].

7) `!arp` is the syntax to remove all arp traffic from the TryHackMe pcap file in Wireshark (*Figure 6*) [1].



What symbol(s) can be used in place of the word "and" when setting a filter?

&&

What symbol can be used in place of the word "not"?

!

What is the syntax to remove all arp traffic from a pcap?

!arp

Figure 6: Chris Greer' Youtube video, TryHackMe WIRESHARK Filters Walkthrough, showing users how to use proper symbols and syntax to execute filters in Wireshark

LIMITATIONS/CONCLUSION

As an introductory experiment for the novice user like myself, I thought the lab's difficulty was simple. I do not think there were any limitations because everything was executed in a live environment versus a controlled environment, like on a virtual machine. The biggest takeaway from this lab was learning how to use proper symbols and syntax to execute filters for the user's desired results.

REFERENCES

[1] Greer, *YouTube* [Online]. "TryHackMe WIRESHARK Filters Walkthrough", September 6, 2022. Available: https://www.youtube.com/watch?v=-MLkdg4s4ew [Accessed: 6-Oct-2022]

COLLABORATION

The entirety of this lab was executed independently by the author. No additional collaboration to report.