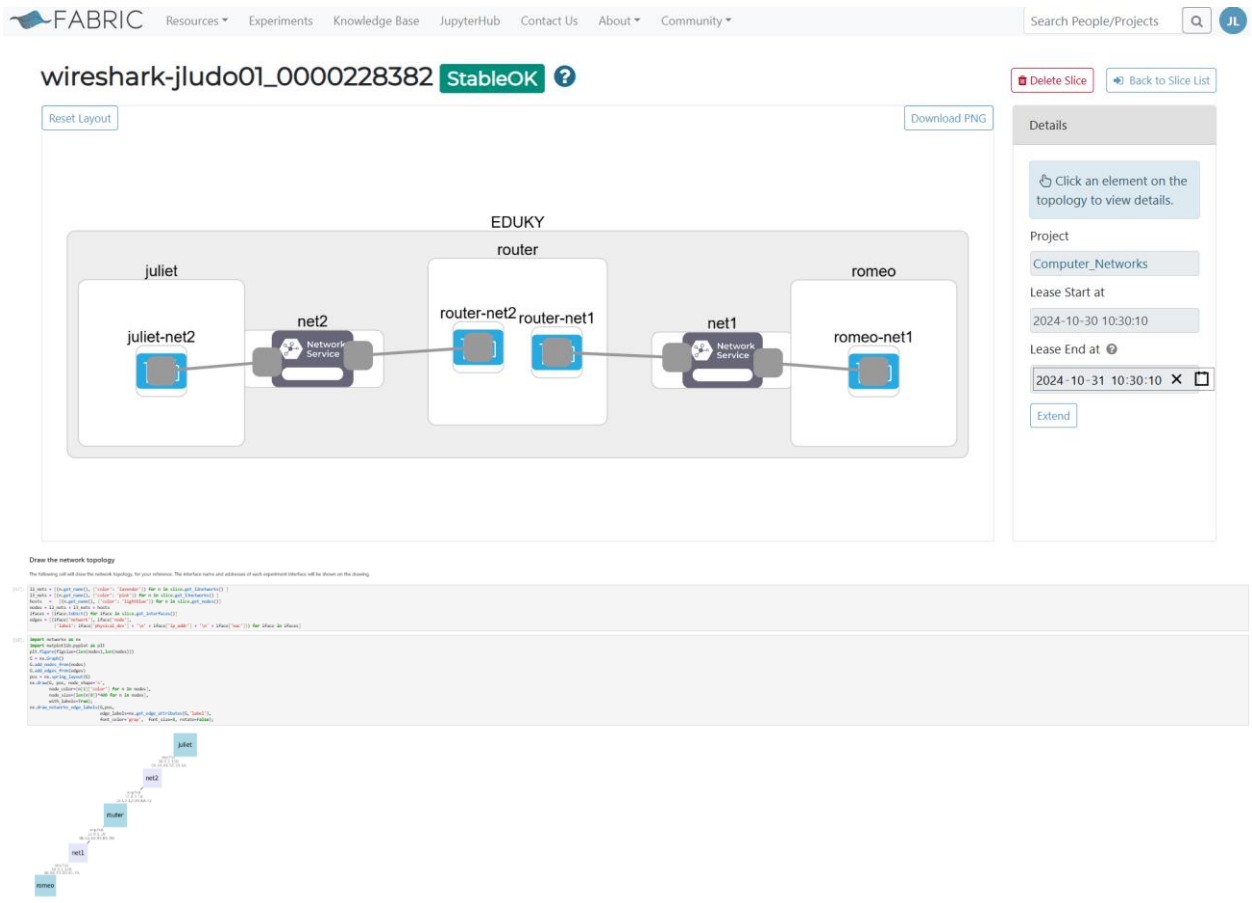
CLab 8 – Inspecting Network Traffic with TCDUMP and Wireshark

Joshua Ludolf
CSCI 4406 – Computer Networks

- To start this lab, I had to configure my fabric by running utilizing the blogs github repo (git checkout):



- From there, I followed the instructions from https://witestlab.poly.edu/blog/wireshark-tcpdump/, starting from Capture network traffic with tcpdump section:

- Ssh into Romeo node:

```
fabric@spring:CLab 8-16%$ ssh -i /home/fabric/work/fabric_config/slice_key -F /home/fabric/work/fabric_config/ssh_config ubuntu@2610:1e0:1700:206:f816:3eff:fe35:b97d
Warning: Permanently added 'bastion.fabric-testbed.net' (ED25519) to the list of known hosts.
Warning: Permanently added '2610:1e0:1700:206:f816:3eff:fe35:b97d' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Oct 29 22:00:59 UTC 2024

  System load:             0.02
  Usage of /:              15.2% of 9.51GB
  Memory usage:            6%
  Swap usage:              0%
  Processes:               161
  Users logged in:         0
  IPv4 address for enp3s0: 10.30.10.45
  IPv6 address for enp3s0: 2610:1e0:1700:206:f816:3eff:fe35:b97d


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Oct 29 21:49:01 2024 from 2610:1e0:1700:205::51
```

- Identifying interface with ip address 10.0.1.100:

```
ubuntu@romeo:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:61:40:f0 brd ff:ff:ff:ff:ff:ff
    inet 10.30.7.68/19 metric 100 brd 10.30.31.255 scope global dynamic enp3s0
       valid_lft 85826sec preferred_lft 85826sec
    inet6 2610:1e0:1700:206:f816:3eff:fe61:40f0/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 86391sec preferred_lft 14391sec
    inet6 fe80::f816:3eff:fe61:40f0/64 scope link
       valid_lft forever preferred_lft forever
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 06:a6:e3:d0:81:7a brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.100/24 scope global enp7s0
       valid_lft forever preferred_lft forever
    inet6 fe80::4a6:e3ff:fed0:817a/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@romeo:~$
```

- First Attempt of tcpdump command on interface enp7s0 (I knew this wasn't going to work but had to follow instructions…):

```
ubuntu@romeo:~$ tcpdump -i enp7s0
tcpdump: enp7s0: You don't have permission to capture on that device
(socket: Operation not permitted)
```

- Second Attempt of tcpdump command on interface enp7s0 (yay, I get to run it correctly! 😍):

```
ubuntu@romeo:~$ sudo tcpdump -i enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

- Now adding network packets by generating some traffic from Juliet node:

```
ubuntu@juliet:~$ ping -c 5 10.0.1.100
PING 10.0.1.100 (10.0.1.100) 56(84) bytes of data.
64 bytes from 10.0.1.100: icmp_seq=1 ttl=63 time=0.434 ms
64 bytes from 10.0.1.100: icmp_seq=2 ttl=63 time=0.147 ms
64 bytes from 10.0.1.100: icmp_seq=3 ttl=63 time=0.157 ms
64 bytes from 10.0.1.100: icmp_seq=4 ttl=63 time=0.147 ms
64 bytes from 10.0.1.100: icmp_seq=5 ttl=63 time=0.158 ms

--- 10.0.1.100 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4082ms
rtt min/avg/max/mdev = 0.147/0.208/0.434/0.112 ms
ubuntu@juliet:~$
```

```
ubuntu@romeo:~$ sudo tcpdump -i enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:44:23.847505 IP6 romeo > ip6-allrouters: ICMP6, router solicitation, length 16
15:44:24.668809 ARP, Request who-has romeo tell router, length 42
15:44:24.668837 ARP, Reply romeo is-at 06:a6:e3:d0:81:7a (oui Unknown), length 28
15:44:24.668903 IP juliet > romeo: ICMP echo request, id 1, seq 1, length 64
15:44:24.668919 IP romeo > juliet: ICMP echo reply, id 1, seq 1, length 64
15:44:25.678204 IP juliet > romeo: ICMP echo request, id 1, seq 2, length 64
15:44:25.678222 IP romeo > juliet: ICMP echo reply, id 1, seq 2, length 64
15:44:26.702167 IP juliet > romeo: ICMP echo request, id 1, seq 3, length 64
15:44:26.702197 IP romeo > juliet: ICMP echo reply, id 1, seq 3, length 64
15:44:27.726228 IP juliet > romeo: ICMP echo request, id 1, seq 4, length 64
15:44:27.726242 IP romeo > juliet: ICMP echo reply, id 1, seq 4, length 64
15:44:28.750170 IP juliet > romeo: ICMP echo request, id 1, seq 5, length 64
15:44:28.750197 IP romeo > juliet: ICMP echo reply, id 1, seq 5, length 64
15:44:29.735485 ARP, Request who-has router tell romeo, length 28
15:44:29.735551 ARP, Reply router is-at 0e:e1:e8:88:b5:9b (oui Unknown), length 42
15:44:42.013109 IP6 fe80::ce1:e8ff:fe88:b59b > ip6-allrouters: ICMP6, router solicitation, length 16
^C
16 packets captured
16 packets received by filter
0 packets dropped by kernel
```

- From there worked on Save a packet capture to a file and review it with tcpdump and Wireshark:
  - Running following command - sudo tcpdump -i XXX -w romeo-tcpdump-file.pcap:

    ```
    ubuntu@romeo:~$ sudo tcpdump -i enp7s0 -w romeo-tcpdump-file.pcap
    tcpdump: listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
    ```

  - Repeating ping command from Juliet as before:

    ```
    ubuntu@juliet:~$ ping -c 5 10.0.1.100
    PING 10.0.1.100 (10.0.1.100) 56(84) bytes of data.
    64 bytes from 10.0.1.100: icmp_seq=1 ttl=63 time=0.315 ms
    64 bytes from 10.0.1.100: icmp_seq=2 ttl=63 time=0.181 ms
    64 bytes from 10.0.1.100: icmp_seq=3 ttl=63 time=0.187 ms
    64 bytes from 10.0.1.100: icmp_seq=4 ttl=63 time=0.197 ms
    64 bytes from 10.0.1.100: icmp_seq=5 ttl=63 time=0.164 ms

    --- 10.0.1.100 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time 4100ms
    rtt min/avg/max/mdev = 0.164/0.208/0.315/0.054 ms
    ```

  - This time I couldn't see summary (blog mentions more why I wasn't suppose to 😎, however blog is incorrect, I get 14 packets as shown):

    ```
    ubuntu@romeo:~$ sudo tcpdump -i enp7s0 -w romeo-tcpdump-file.pcap
    tcpdump: listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
    ^C14 packets captured
    14 packets received by filter
    0 packets dropped by kernel
    ```

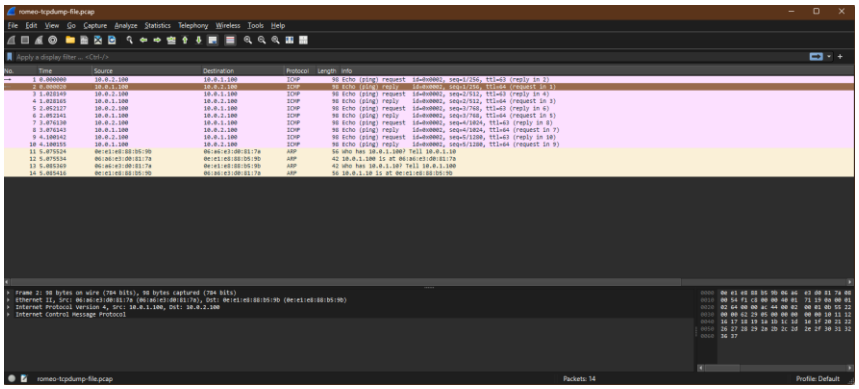
  - Now to show summary as before I ran following command - tcpdump -r romeo-tcpdump-file.pcap (Note that I didn't need special privileges to print back packet summaries from a file, only to capture live traffic from a network interface! That's why I didn't need sudo for this command.):

    ```
    ubuntu@romeo:~$ tcpdump -r romeo-tcpdump-file.pcap
    reading from file romeo-tcpdump-file.pcap, link-type EN10MB (Ethernet), snapshot length 262144
    15:47:23.338113 IP juliet > romeo: ICMP echo request, id 2, seq 1, length 64
    15:47:23.338133 IP romeo > juliet: ICMP echo reply, id 2, seq 1, length 64
    15:47:24.366262 IP juliet > romeo: ICMP echo request, id 2, seq 2, length 64
    15:47:24.366278 IP romeo > juliet: ICMP echo reply, id 2, seq 2, length 64
    15:47:25.390240 IP juliet > romeo: ICMP echo request, id 2, seq 3, length 64
    15:47:25.390254 IP romeo > juliet: ICMP echo reply, id 2, seq 3, length 64
    15:47:26.414243 IP juliet > romeo: ICMP echo request, id 2, seq 4, length 64
    15:47:26.414256 IP romeo > juliet: ICMP echo reply, id 2, seq 4, length 64
    15:47:27.438255 IP juliet > romeo: ICMP echo request, id 2, seq 5, length 64
    15:47:27.438268 IP romeo > juliet: ICMP echo reply, id 2, seq 5, length 64
    15:47:28.413637 ARP, Request who-has romeo tell router, length 42
    15:47:28.413647 ARP, Reply romeo is-at 06:a6:e3:d0:81:7a (oui Unknown), length 28
    15:47:28.423482 ARP, Request who-has router tell romeo, length 28
    15:47:28.423529 ARP, Reply router is-at 0e:e1:e8:88:b5:9b (oui Unknown), length 42
    ubuntu@romeo:~$
    ```

- At this point I now needed to transfer the packet capture (I realize that all the commands work, but took me so long because it was wrong filename xD):

```
ubuntu@romeo:~$ curl -F "file=@/home/ubuntu/romeo-tcpdump-file.pcap" https://file.io
{"success":true,"status":200,"id":"8c7376c0-96e1-11ef-908b-7da7c4195438","key":"hQI5279RFzW5","path":"/","nodeType":"file","name":"romeo-tcpdump-file.pcap","title":null,"description":null,"size":1424,"link":"https://file.io/hQI5279RFzW5","private":false,"expires":"2024-11-13T17:08:13.985Z","downloads":0,"maxDownloads":1,"autoDelete":true,"planId":0,"screeningStatus":"pending","mimeType":"application/octet-stream","created":"2024-10-30T17:08:13.985Z","modified":"2024-10-30T17:08:13.985Z"}ubuntu@romeo:~$
```

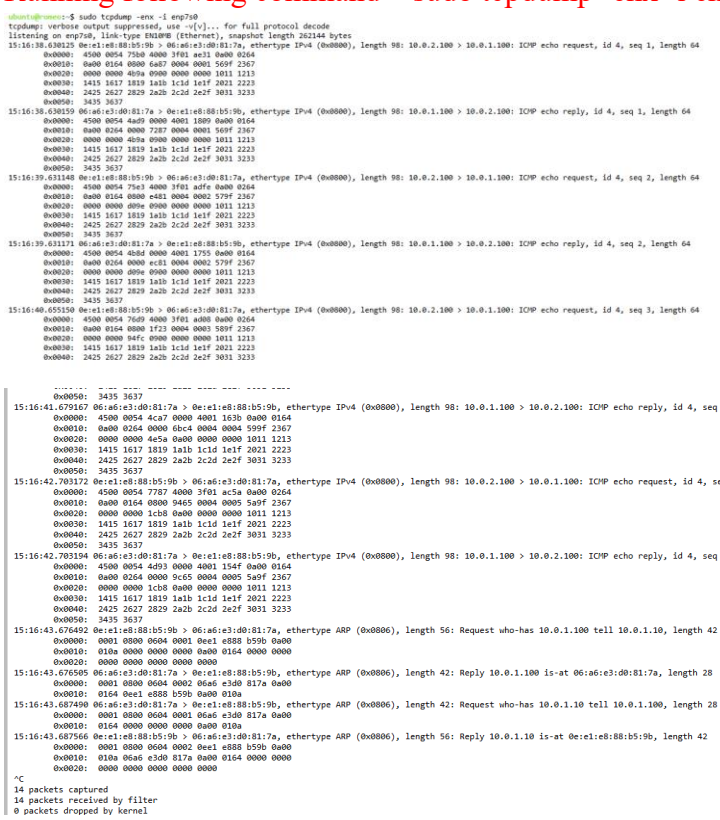
- Then I opened the packet capture on my actual machine:



- Useful display options and capture options in tcpdump
  - Running following command – man tcpdump:

```
TCPDUMP(8)                    System Manager's Manual                    TCPDUMP(8)

NAME
       tcpdump - dump traffic on a network

SYNOPSIS
       tcpdump [ -AbdDefhHIJK1lnNOpq$tuUvxX# ] [ -B buffer_size ]
               [ -c count ] [ --count ] [ -C file_size ]
               [ -E spi@ipaddr algo:secret,... ]
               [ -F file ] [ -G rotate_seconds ] [ -i interface ]
               [ --immediate-mode ] [ -j tstamp_type ] [ -m module ]
               [ -M secret ] [ --number ] [ --print ] [ -Q in|out|inout ]
               [ -r file ] [ -s snaplen ] [ -T type ] [ --version ]
               [ -V file ] [ -w file ] [ -W filecount ] [ -y datalinktype ]
               [ -z postrotate-command ] [ -Z user ]
               [ --time-stamp-precision=tstamp_precision ]
               [ --micro ] [ --nano ]
               [ expression ]

DESCRIPTION
       Tcpdump  prints  out  a description of the contents of packets on a network interface that match the Boolean expression; the description is preceded by a time
       stamp, printed, by default, as hours, minutes, seconds, and fractions of a second since midnight.  It can also be run with the -w flag,  which  causes  it  to
       save  the packet data to a file for later analysis, and/or with the -r flag, which causes it to read from a saved packet file rather than to read packets from
       a network interface.  It can also be run with the -V flag, which causes it to read a list of saved packet files. In all cases, only packets that match  expres-
       sion will be processed by tcpdump.

       Tcpdump  will,  if not run with the -c flag, continue capturing packets until it is interrupted by a SIGINT signal (generated, for example, by typing your in-
       terrupt character, typically control-C) or a SIGTERM signal (typically generated with the kill(1) command); if run with the -c flag, it will  capture  packets
       until it is interrupted by a SIGINT or SIGTERM signal or the specified number of packets have been processed.

       When tcpdump finishes capturing packets, it will report counts of:

              packets ``captured'' (this is the number of packets that tcpdump has received and processed);

              packets  ``received by filter'' (the meaning of this depends on the OS on which you're running tcpdump, and possibly on the way the OS was configured -
              if a filter was specified on the command line, on some OSes it counts packets regardless of whether they were matched by  the  filter  expression  and,
              even if they were matched by the filter expression, regardless of whether tcpdump has read and processed them yet, on other OSes it counts only packets
Manual page tcpdump(8) line 1 (press h for help or q to quit)
```

- Running following command – sudo tcpdump -enx -i enp7s0:



- Running following command - sudo tcpdump -s 34 -w romeo-tcpdump-snaplen.pcap -i enp7s0

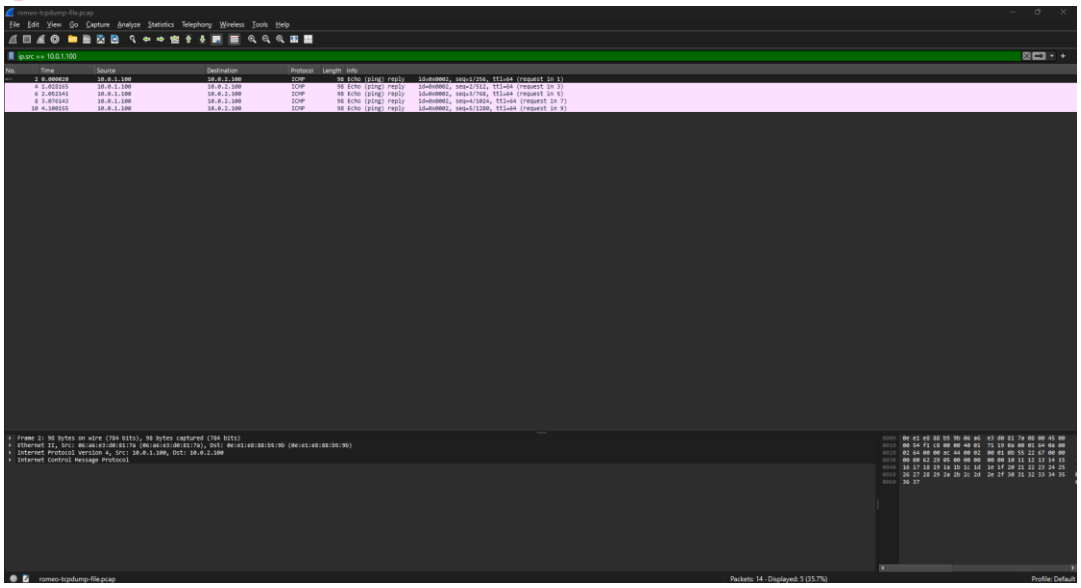
```
ubuntu@romeo:~$ sudo tcpdump -s 34 -w romeo-tcpdump-snaplen.pcap -i enp7s0
tcpdump: listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 34 bytes
^C14 packets captured
14 packets received by filter
0 packets dropped by kernel
```

- Then I curl the Romeo-tcpdump-snaplen.pcap file:

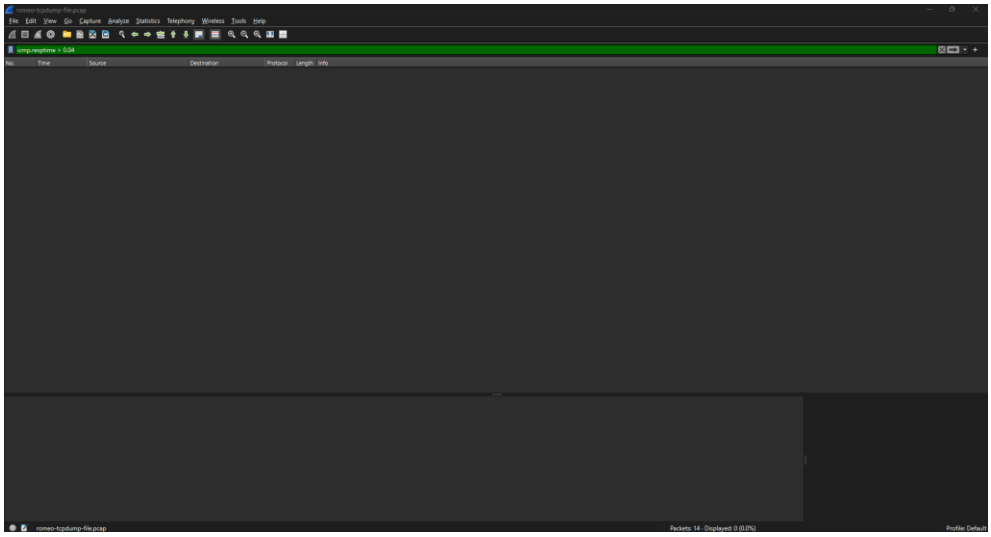
- Running sudo tcpdump -i enp7s0 src host 10.0.1.100

```
ubuntu@romeo:~$ sudo tcpdump -i enp7s0 src host 10.0.1.100
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
15:25:08.180701 IP romeo > juliet: ICMP echo reply, id 7, seq 1, length 64
15:25:09.199262 IP romeo > juliet: ICMP echo reply, id 7, seq 2, length 64
15:25:10.223234 IP romeo > juliet: ICMP echo reply, id 7, seq 3, length 64
15:25:11.247234 IP romeo > juliet: ICMP echo reply, id 7, seq 4, length 64
15:25:12.271241 IP romeo > juliet: ICMP echo reply, id 7, seq 5, length 64
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

- Useful display options in Wireshark:
  - I opened my first capture file and applied following display filter – ip.src == 10.0.1.100:



  - Then I applied icmp.resptime > 0.04, but I didn't get any packets like the blog did:



  - None of the packets were ICMP that I got so…

Through this lab on inspecting network traffic with tcpdump and Wireshark, I've gained a lot of valuable insights. Using tcpdump, I learned how to capture network traffic efficiently. It was fascinating to

see real-time data and understand how this tool can be crucial for monitoring network issues.
Wireshark took this learning further. Delving into the details of packets, I saw how protocols like ICMP, TCP, and HTTP are structured. Analyzing the "Type" field in an ICMP packet, for example, highlighted how much detail is involved in each packet.
One of the key takeaways was troubleshooting. By examining packet flows, I could identify anomalies and errors, sharpening my skills in network diagnostics. The practical experience solidified theoretical concepts, making them much more comprehensible and applicable.
Overall, this lab equipped me with hands-on skills in network analysis and troubleshooting, paving the way for more advanced tasks in the future.