Samantha Jackson
CSCI4406
11-03-2024

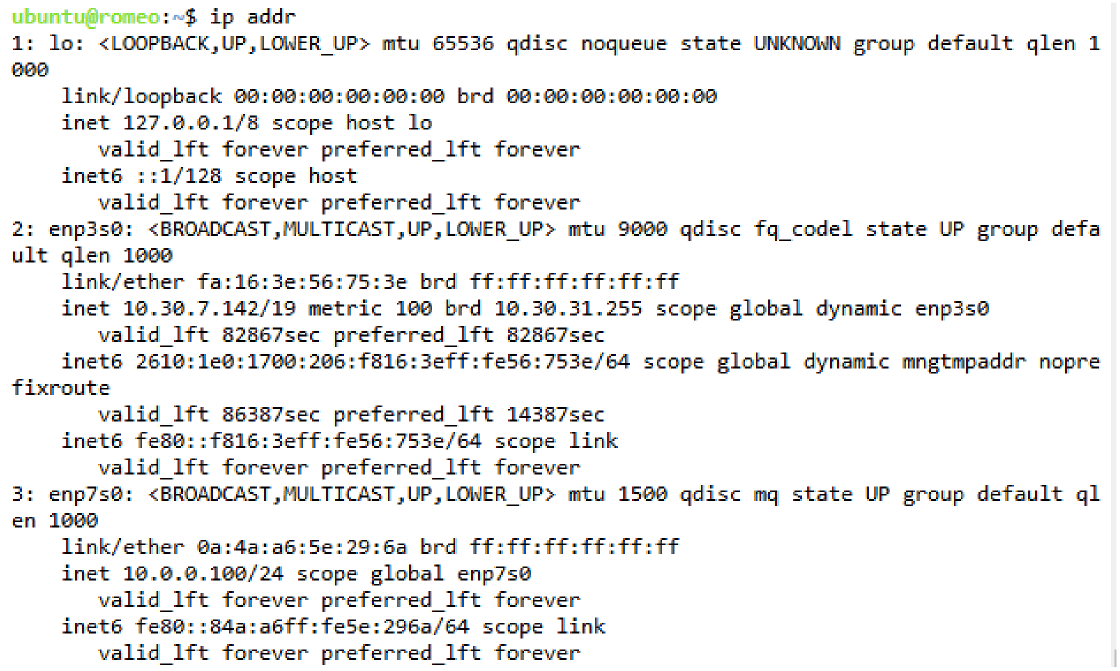
CLab 8 - Inspecting Network Traffic with TCPDump and Wireshark

Setting up the Experiment

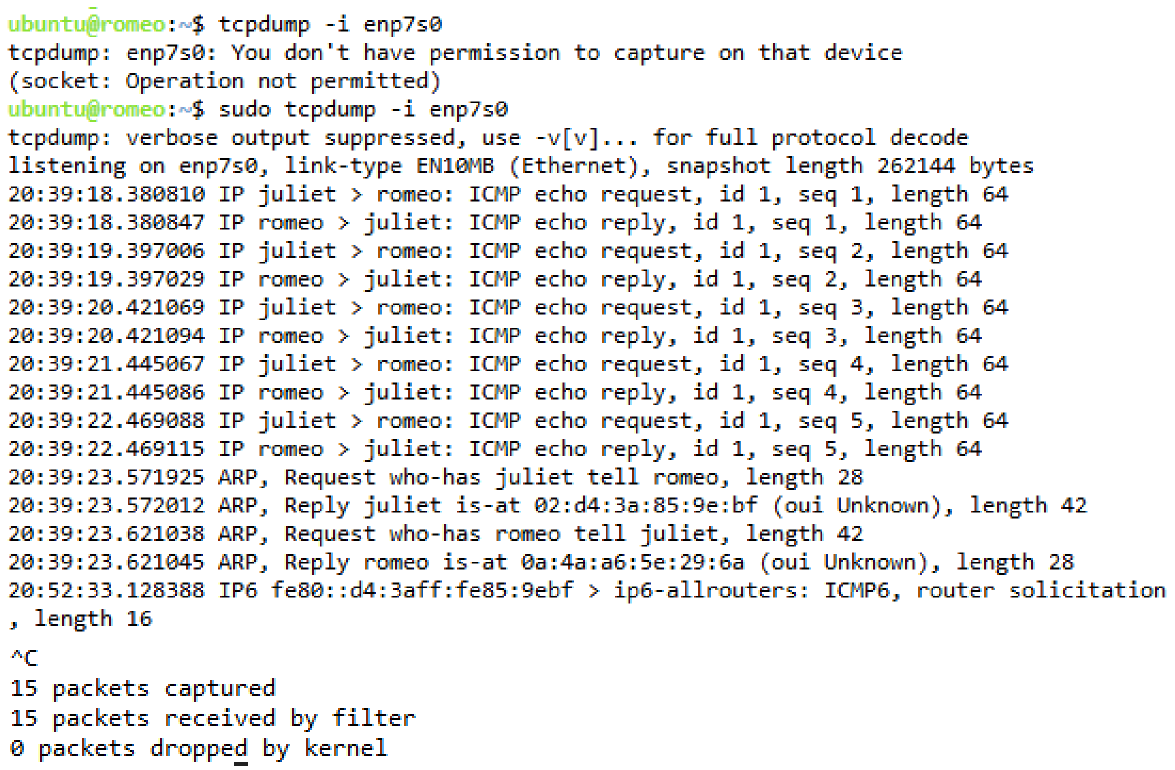
Observations/Issues: Redundancies from CLab 7 mean that the network topology doesn't match the example. I don't know why this is but I'm going to attempt to continue the lab as is. We already have the experiment set up.

Capturing Network Traffic with TCPDump

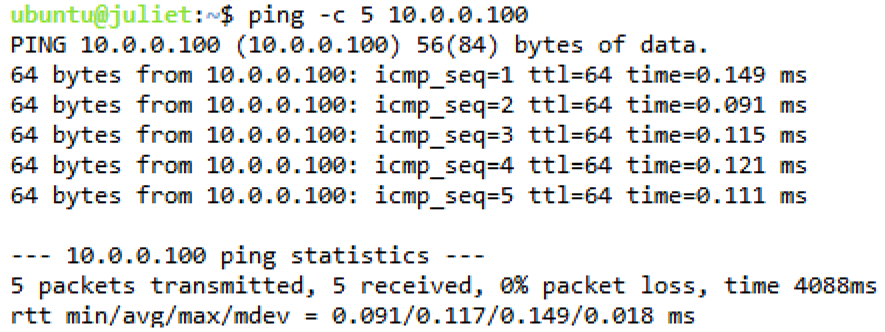
Observations/Issues: TCPDump is a terminal command to capture network traffic. Since my network topology is inexplicably broken we'll be TCP'ing IP addresses of my choice. In this instance I selected *'enp7s0'*.



```
ubuntu@romeo:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
000
    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 defa
ult qlen 1000
    link/ether fa:16:3e:56:75:3e brd ff:ff:ff:ff:ff:ff
    inet 10.30.7.142/19 metric 100 brd 10.30.31.255 scope global dynamic enp3s0
       valid_lft 82867sec preferred_lft 82867sec
    inet6 2610:1e0:1700:206:f816:3eff:fe56:753e/64 scope global dynamic mngtmpaddr nopre
fixroute
       valid_lft 86387sec preferred_lft 14387sec
    inet6 fe80::f816:3eff:fe56:753e/64 scope link
       valid_lft forever preferred_lft forever
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default ql
en 1000
    link/ether 0a:4a:a6:5e:29:6a brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.100/24 scope global enp7s0
       valid_lft forever preferred_lft forever
    inet6 fe80::84a:a6ff:fe5e:296a/64 scope link
       valid_lft forever preferred_lft forever
```



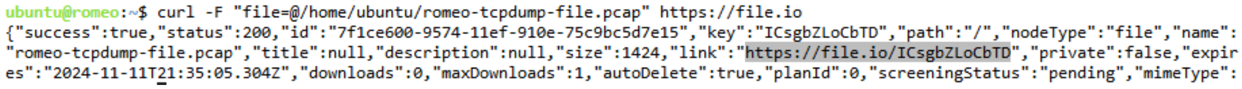
```
ubuntu@romeo:~$ tcpdump -i enp7s0
tcpdump: enp7s0: You don't have permission to capture on that device
(socket: Operation not permitted)
ubuntu@romeo:~$ sudo tcpdump -i enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:39:18.380810 IP juliet > romeo: ICMP echo request, id 1, seq 1, length 64
20:39:18.380847 IP romeo > juliet: ICMP echo reply, id 1, seq 1, length 64
20:39:19.397006 IP juliet > romeo: ICMP echo request, id 1, seq 2, length 64
20:39:19.397029 IP romeo > juliet: ICMP echo reply, id 1, seq 2, length 64
20:39:20.421069 IP juliet > romeo: ICMP echo request, id 1, seq 3, length 64
20:39:20.421094 IP romeo > juliet: ICMP echo reply, id 1, seq 3, length 64
20:39:21.445067 IP juliet > romeo: ICMP echo request, id 1, seq 4, length 64
20:39:21.445086 IP romeo > juliet: ICMP echo reply, id 1, seq 4, length 64
20:39:22.469088 IP juliet > romeo: ICMP echo request, id 1, seq 5, length 64
20:39:22.469115 IP romeo > juliet: ICMP echo reply, id 1, seq 5, length 64
20:39:23.571925 ARP, Request who-has juliet tell romeo, length 28
20:39:23.572012 ARP, Reply juliet is-at 02:d4:3a:85:9e:bf (oui Unknown), length 42
20:39:23.621038 ARP, Request who-has romeo tell juliet, length 42
20:39:23.621045 ARP, Reply romeo is-at 0a:4a:a6:5e:29:6a (oui Unknown), length 28
20:52:33.128388 IP6 fe80::d4:3aff:fe85:9ebf > ip6-allrouters: ICMP6, router solicitation
, length 16
^C
15 packets captured
15 packets received by filter
0 packets dropped by kernel
```



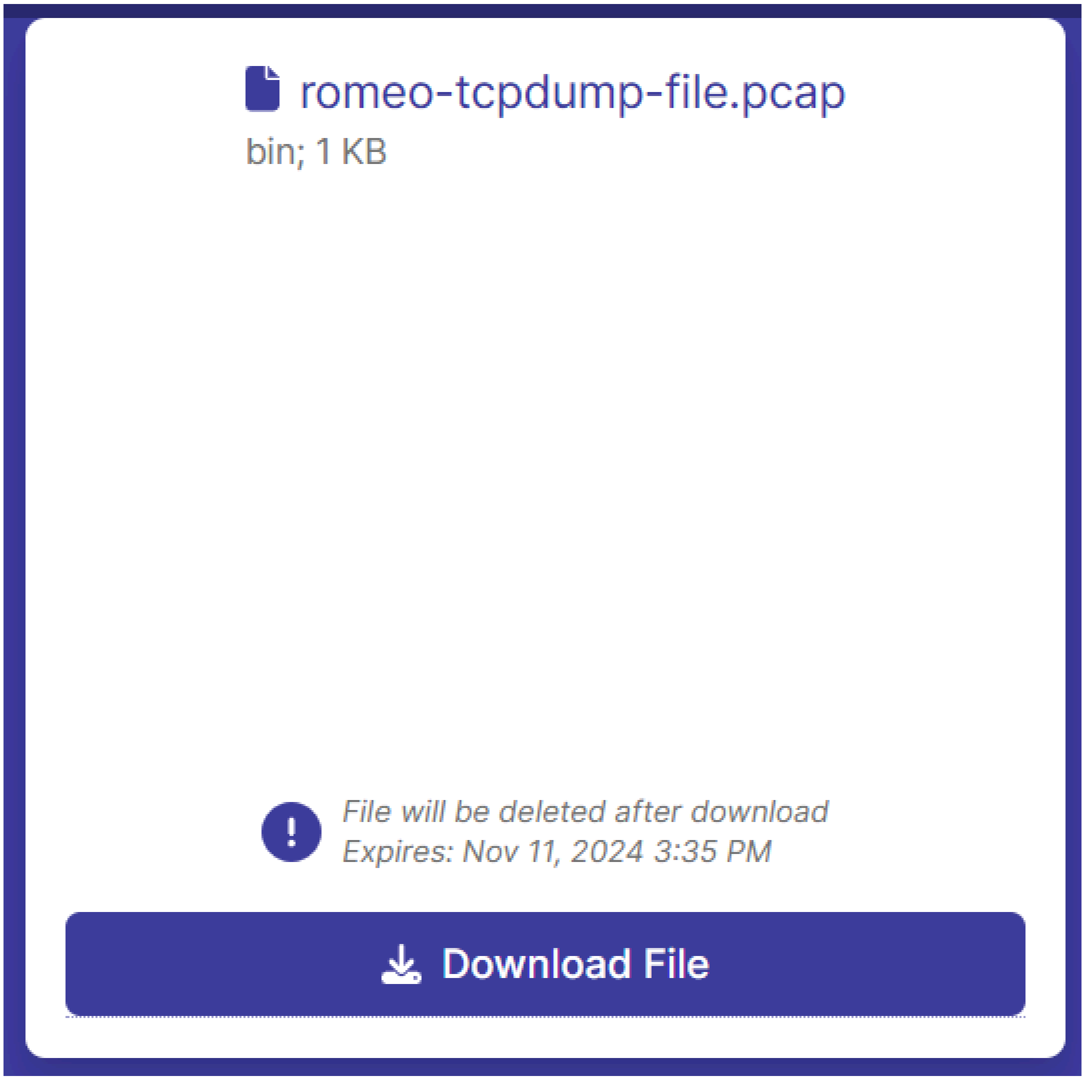
```
ubuntu@juliet:~$ ping -c 5 10.0.0.100
PING 10.0.0.100 (10.0.0.100) 56(84) bytes of data.
64 bytes from 10.0.0.100: icmp_seq=1 ttl=64 time=0.149 ms
64 bytes from 10.0.0.100: icmp_seq=2 ttl=64 time=0.091 ms
64 bytes from 10.0.0.100: icmp_seq=3 ttl=64 time=0.115 ms
64 bytes from 10.0.0.100: icmp_seq=4 ttl=64 time=0.121 ms
64 bytes from 10.0.0.100: icmp_seq=5 ttl=64 time=0.111 ms

--- 10.0.0.100 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4088ms
rtt min/avg/max/mdev = 0.091/0.117/0.149/0.018 ms
```

Saving a Packet Capture to Review it with TCPDump and Wireshark

Observations/Issues: All the other commands suggested in the documentation for this didn't work so I resorted to the solution suggested in the email '*curl -F "file=@/home/ubuntu/romeo-tcpdump-file.pcap"* *https://file.io*'



```
ubuntu@romeo:~$ curl -F "file=@/home/ubuntu/romeo-tcpdump-file.pcap" https://file.io
{"success":true,"status":200,"id":"7f1ce600-9574-11ef-910e-75c9bc5d7e15","key":"ICsgbZLoCbTD","path":"/","nodeType":"file","name":
"romeo-tcpdump-file.pcap","title":null,"description":null,"size":1424,"link":"https://file.io/ICsgbZLoCbTD","private":false,"expir
es":"2024-11-11T21:35:05.304Z","downloads":0,"maxDownloads":1,"autoDelete":true,"planId":0,"screeningStatus":"pending","mimeType":
```



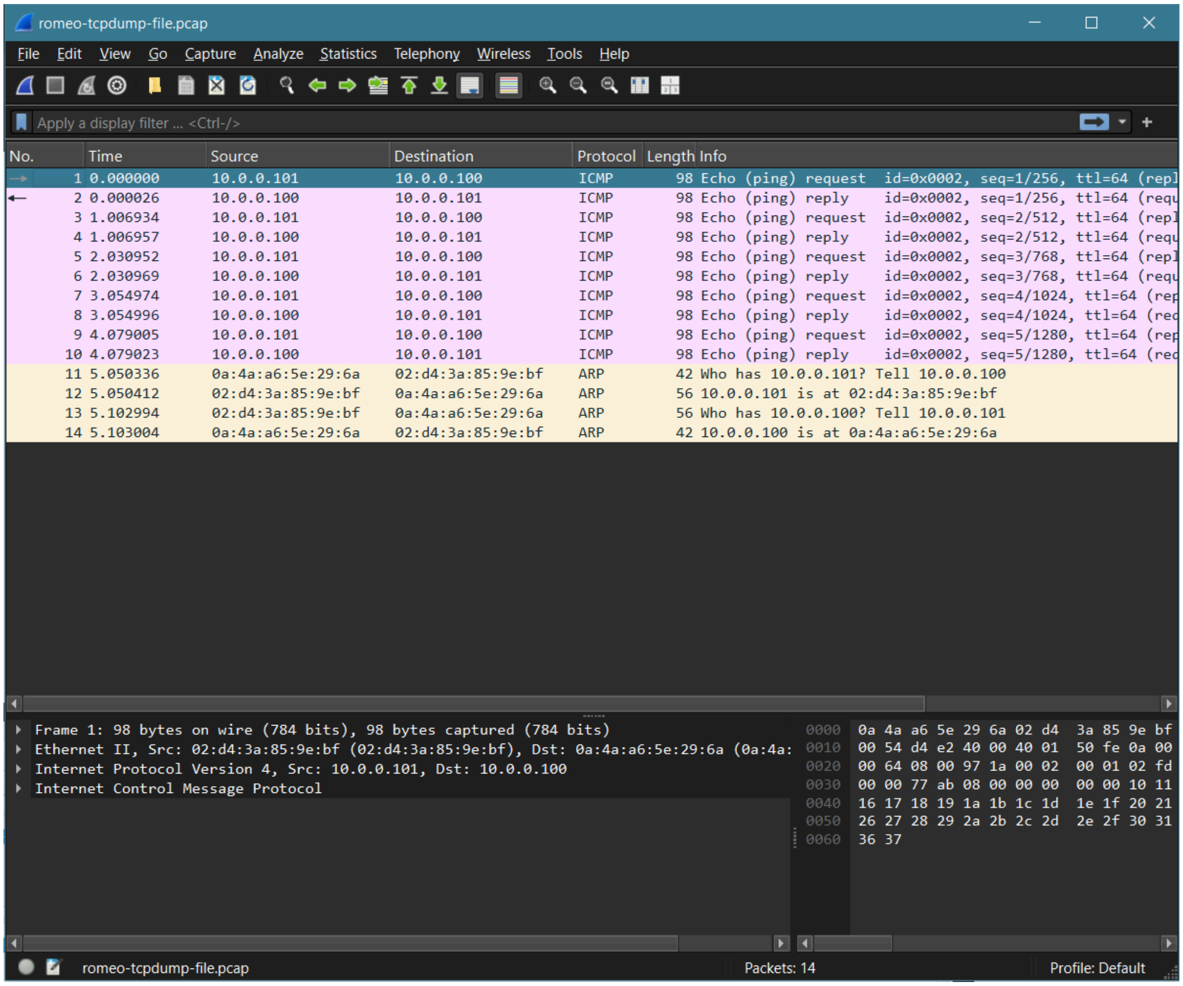
romeo-tcpdump-file.pcap

bin; 1 KB

File will be deleted after download
Expires: Nov 11, 2024 3:35 PM

Display Options for Wireshark

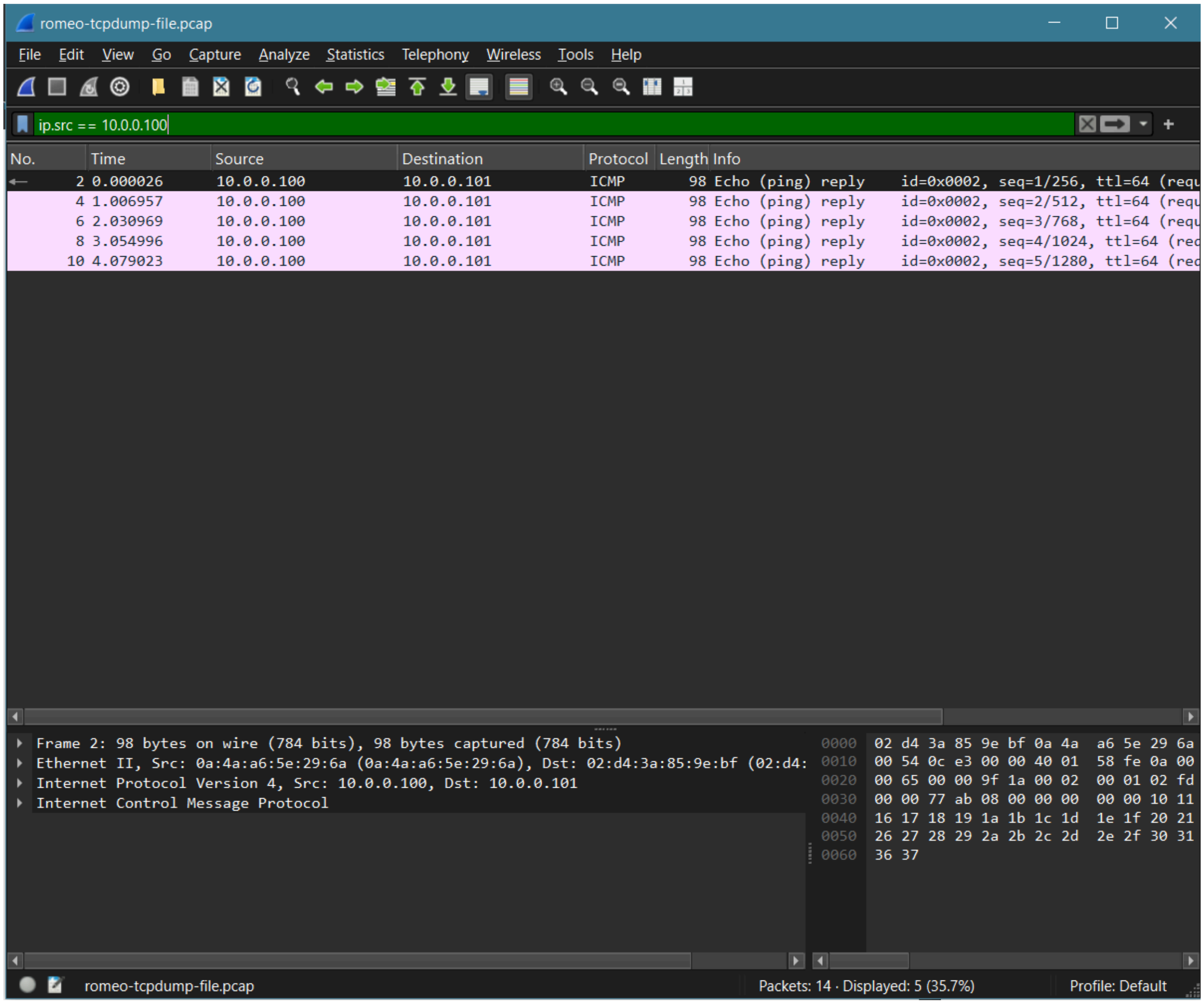
Observations/Issues: None. We have the pcap file displayed in Wireshark. We performed a ICMP response time check of > 0.1 and checked for ip's at 10.0.0.100.


romeo-tcpdump-file.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
romeo-tcpdump-file.pcap
Packets: 14
Profile: Default

```
No.   Time      Source             Destination        Protocol Length Info
    1 0.000000  10.0.0.101         10.0.0.100         ICMP        98 Echo (ping) request  id=0x0002, seq=1/256, ttl=64 (repl
    2 0.000026  10.0.0.100         10.0.0.101         ICMP        98 Echo (ping) reply    id=0x0002, seq=1/256, ttl=64 (requ
    3 1.006934  10.0.0.101         10.0.0.100         ICMP        98 Echo (ping) request  id=0x0002, seq=2/512, ttl=64 (repl
    4 1.006957  10.0.0.100         10.0.0.101         ICMP        98 Echo (ping) reply    id=0x0002, seq=2/512, ttl=64 (requ
    5 2.030952  10.0.0.101         10.0.0.100         ICMP        98 Echo (ping) request  id=0x0002, seq=3/768, ttl=64 (repl
    6 2.030969  10.0.0.100         10.0.0.101         ICMP        98 Echo (ping) reply    id=0x0002, seq=3/768, ttl=64 (requ
    7 3.054974  10.0.0.101         10.0.0.100         ICMP        98 Echo (ping) request  id=0x0002, seq=4/1024, ttl=64 (rep
    8 3.054996  10.0.0.100         10.0.0.101         ICMP        98 Echo (ping) reply    id=0x0002, seq=4/1024, ttl=64 (req
    9 4.079005  10.0.0.101         10.0.0.100         ICMP        98 Echo (ping) request  id=0x0002, seq=5/1280, ttl=64 (rep
   10 4.079023  10.0.0.100         10.0.0.101         ICMP        98 Echo (ping) reply    id=0x0002, seq=5/1280, ttl=64 (req
   11 5.050336  0a:4a:a6:5e:29:6a  02:d4:3a:85:9e:bf  ARP         42 Who has 10.0.0.101? Tell 10.0.0.100
   12 5.050412  02:d4:3a:85:9e:bf  0a:4a:a6:5e:29:6a  ARP         56 10.0.0.101 is at 02:d4:3a:85:9e:bf
   13 5.102994  02:d4:3a:85:9e:bf  0a:4a:a6:5e:29:6a  ARP         56 Who has 10.0.0.100? Tell 10.0.0.101
   14 5.103004  0a:4a:a6:5e:29:6a  02:d4:3a:85:9e:bf  ARP         42 10.0.0.100 is at 0a:4a:a6:5e:29:6a
```

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: 02:d4:3a:85:9e:bf (02:d4:3a:85:9e:bf), Dst: 0a:4a:a6:5e:29:6a (0a:4a:
▸ Internet Protocol Version 4, Src: 10.0.0.101, Dst: 10.0.0.100
▸ Internet Control Message Protocol
```

```
0000  0a 4a a6 5e 29 6a 02 d4  3a 85 9e bf
0010  00 54 d4 e2 40 00 40 01  50 fe 0a 00
0020  00 64 08 00 97 1a 00 02  00 01 02 fd
0030  00 00 77 ab 08 00 00 00  00 00 10 11
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31
0060  36 37
```



romeo-tcpdump-file.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

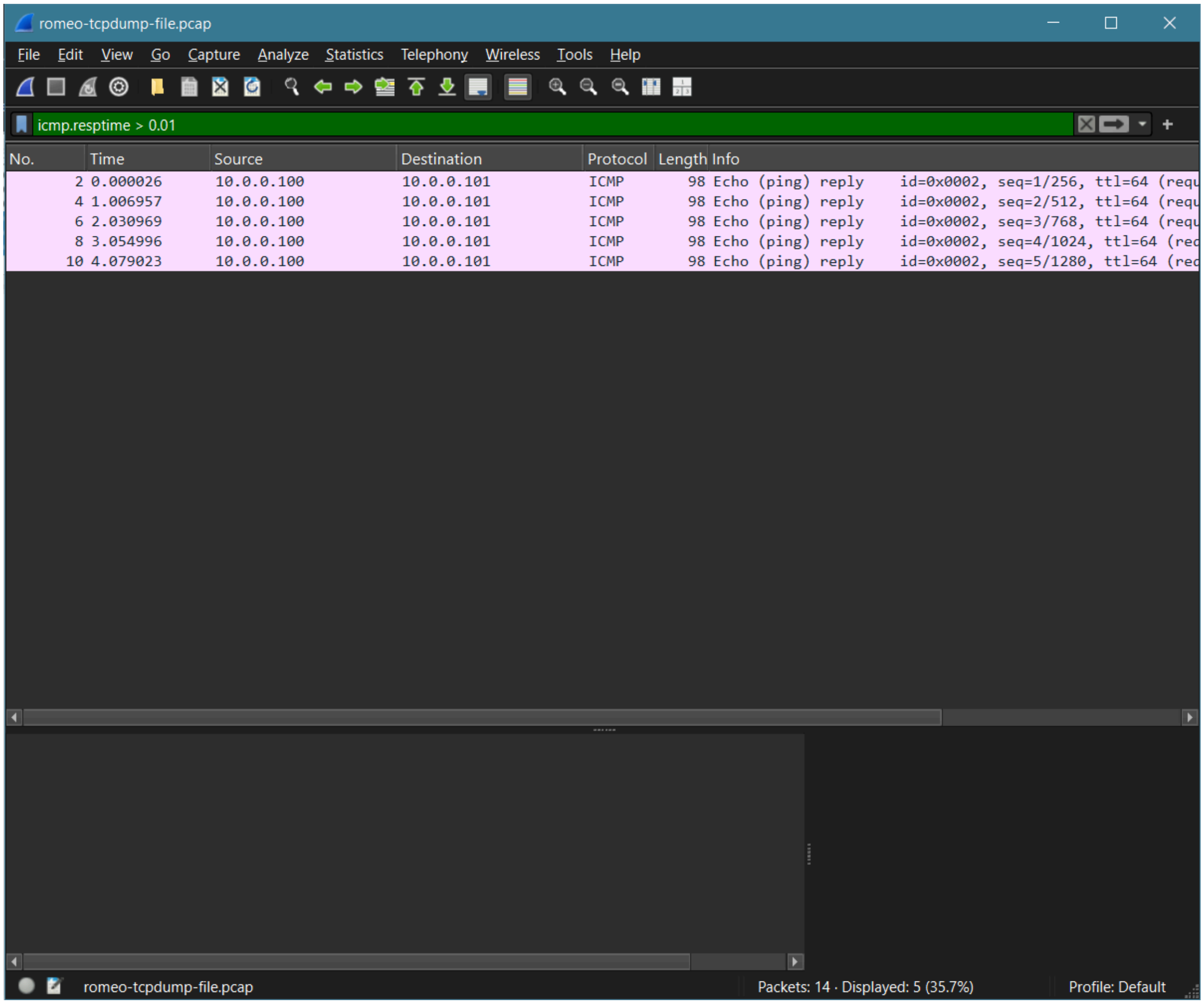
```
ip.src == 10.0.0.100
```

```
No.   Time       Source       Destination   Protocol Length Info
←   2 0.000026   10.0.0.100   10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=1/256, ttl=64 (requ
    4 1.006957   10.0.0.100   10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=2/512, ttl=64 (requ
    6 2.030969   10.0.0.100   10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=3/768, ttl=64 (requ
    8 3.054996   10.0.0.100   10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=4/1024, ttl=64 (rec
   10 4.079023   10.0.0.100   10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=5/1280, ttl=64 (rec
```

```
▸ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: 0a:4a:a6:5e:29:6a (0a:4a:a6:5e:29:6a), Dst: 02:d4:3a:85:9e:bf (02:d4:
▸ Internet Protocol Version 4, Src: 10.0.0.100, Dst: 10.0.0.101
▸ Internet Control Message Protocol
```

```
0000  02 d4 3a 85 9e bf 0a 4a  a6 5e 29 6a
0010  00 54 0c e3 00 00 40 01  58 fe 0a 00
0020  00 65 00 00 9f 1a 00 02  00 01 02 fd
0030  00 00 77 ab 08 00 00 00  00 00 10 11
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31
0060  36 37
```

romeo-tcpdump-file.pcap   Packets: 14 · Displayed: 5 (35.7%)   Profile: Default


romeo-tcpdump-file.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
+
romeo-tcpdump-file.pcap
Profile: Default

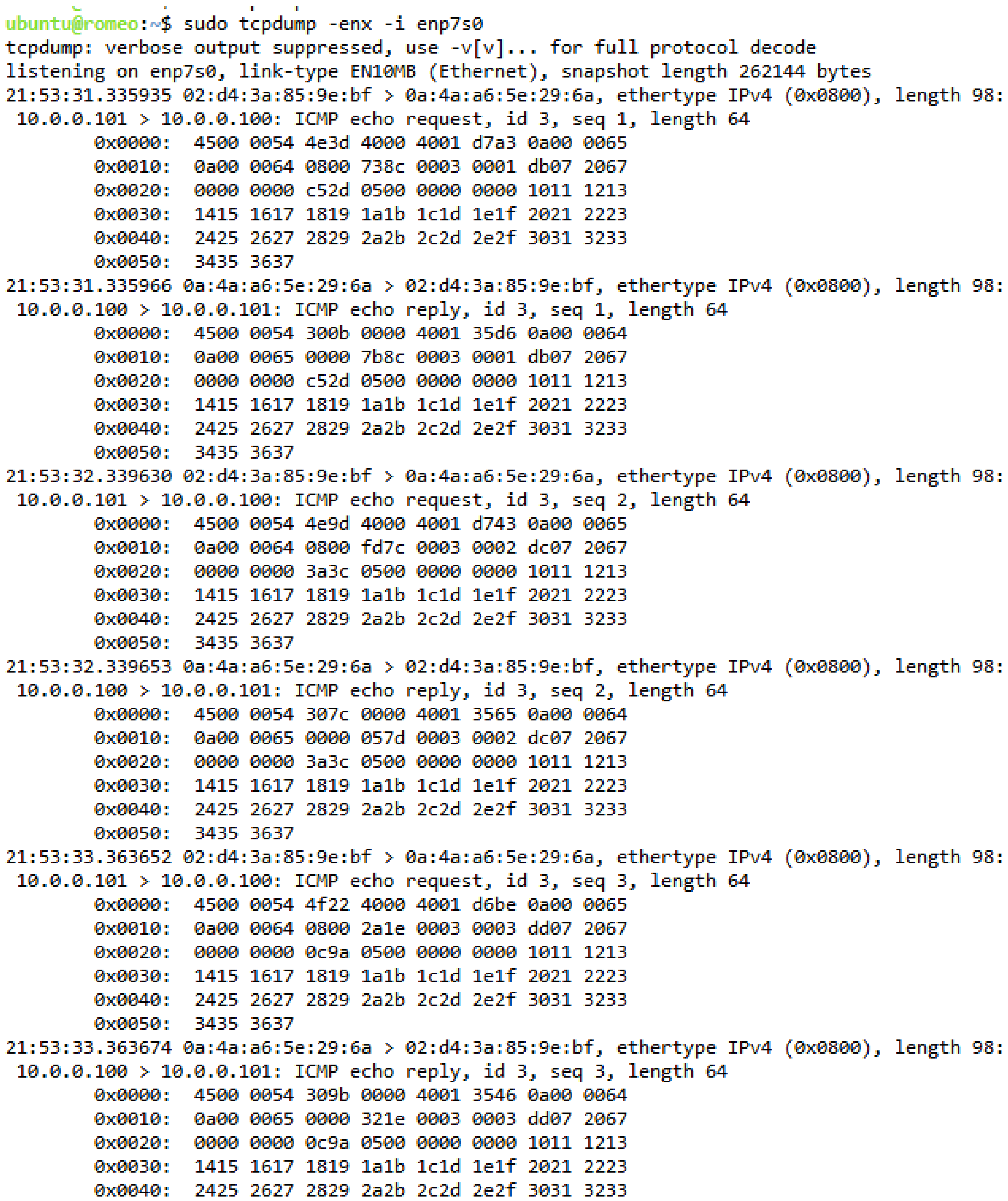
icmp.resptime > 0.01

```
No.   Time      Source        Destination   Protocol Length Info
    2 0.000026  10.0.0.100    10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=1/256, ttl=64 (requ
    4 1.006957  10.0.0.100    10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=2/512, ttl=64 (requ
    6 2.030969  10.0.0.100    10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=3/768, ttl=64 (requ
    8 3.054996  10.0.0.100    10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=4/1024, ttl=64 (req
   10 4.079023  10.0.0.100    10.0.0.101    ICMP        98 Echo (ping) reply    id=0x0002, seq=5/1280, ttl=64 (req
```

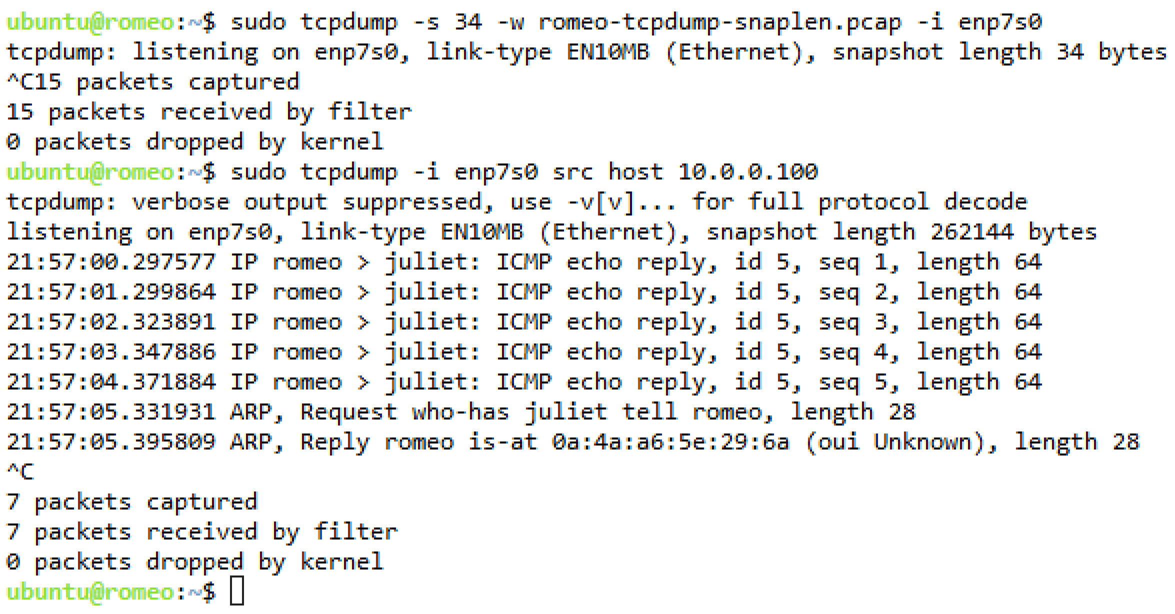
Packets: 14 · Displayed: 5 (35.7%)

Display and Capture Options for TCPDump

Observations/Issues: None. We have a verbose tcpdump, byte limiter of 34 bytes, and capture filters.



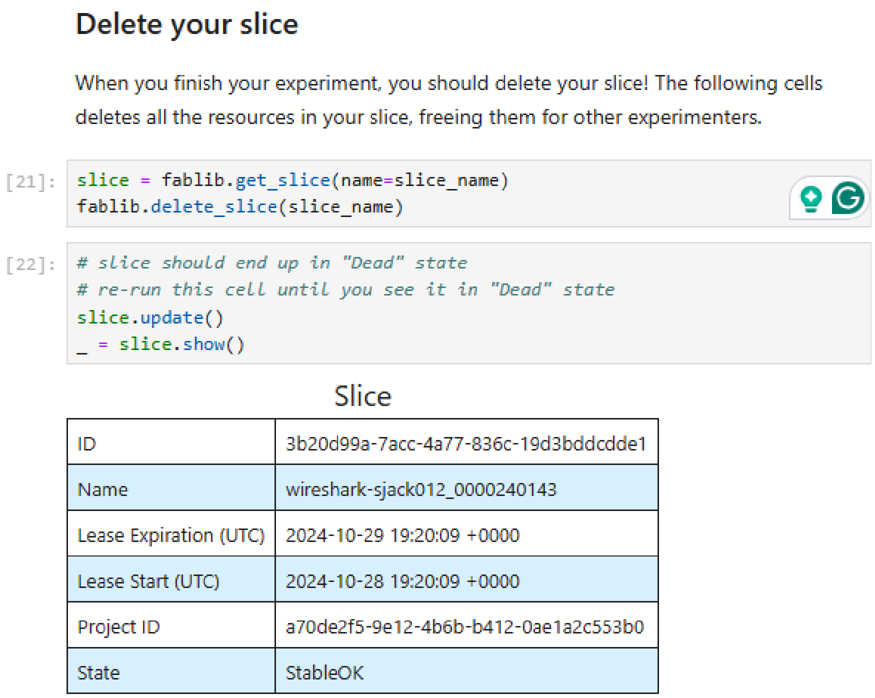
```
ubuntu@romeo:~$ sudo tcpdump -enx -i enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:53:31.335935 02:d4:3a:85:9e:bf > 0a:4a:a6:5e:29:6a, ethertype IPv4 (0x0800), length 98:
 10.0.0.101 > 10.0.0.100: ICMP echo request, id 3, seq 1, length 64
        0x0000:  4500 0054 4e3d 4000 4001 d7a3 0a00 0065
        0x0010:  0a00 0064 0800 738c 0003 0001 db07 2067
        0x0020:  0000 0000 c52d 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
21:53:31.335966 0a:4a:a6:5e:29:6a > 02:d4:3a:85:9e:bf, ethertype IPv4 (0x0800), length 98:
 10.0.0.100 > 10.0.0.101: ICMP echo reply, id 3, seq 1, length 64
        0x0000:  4500 0054 300b 0000 4001 35d6 0a00 0064
        0x0010:  0a00 0065 0000 7b8c 0003 0001 db07 2067
        0x0020:  0000 0000 c52d 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
21:53:32.339630 02:d4:3a:85:9e:bf > 0a:4a:a6:5e:29:6a, ethertype IPv4 (0x0800), length 98:
 10.0.0.101 > 10.0.0.100: ICMP echo request, id 3, seq 2, length 64
        0x0000:  4500 0054 4e9d 4000 4001 d743 0a00 0065
        0x0010:  0a00 0064 0800 fd7c 0003 0002 dc07 2067
        0x0020:  0000 0000 3a3c 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
21:53:32.339653 0a:4a:a6:5e:29:6a > 02:d4:3a:85:9e:bf, ethertype IPv4 (0x0800), length 98:
 10.0.0.100 > 10.0.0.101: ICMP echo reply, id 3, seq 2, length 64
        0x0000:  4500 0054 307c 0000 4001 3565 0a00 0064
        0x0010:  0a00 0065 0000 057d 0003 0002 dc07 2067
        0x0020:  0000 0000 3a3c 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
21:53:33.363652 02:d4:3a:85:9e:bf > 0a:4a:a6:5e:29:6a, ethertype IPv4 (0x0800), length 98:
 10.0.0.101 > 10.0.0.100: ICMP echo request, id 3, seq 3, length 64
        0x0000:  4500 0054 4f22 4000 4001 d6be 0a00 0065
        0x0010:  0a00 0064 0800 2a1e 0003 0003 dd07 2067
        0x0020:  0000 0000 0c9a 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
        0x0050:  3435 3637
21:53:33.363674 0a:4a:a6:5e:29:6a > 02:d4:3a:85:9e:bf, ethertype IPv4 (0x0800), length 98:
 10.0.0.100 > 10.0.0.101: ICMP echo reply, id 3, seq 3, length 64
        0x0000:  4500 0054 309b 0000 4001 3546 0a00 0064
        0x0010:  0a00 0065 0000 321e 0003 0003 dd07 2067
        0x0020:  0000 0000 0c9a 0500 0000 0000 1011 1213
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
```



```
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
^C15 packets captured
15 packets received by filter
0 packets dropped by kernel
ubuntu@romeo:~$ sudo tcpdump -i enp7s0 src host 10.0.0.100
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:57:00.297577 IP romeo > juliet: ICMP echo reply, id 5, seq 1, length 64
21:57:01.299864 IP romeo > juliet: ICMP echo reply, id 5, seq 2, length 64
21:57:02.323891 IP romeo > juliet: ICMP echo reply, id 5, seq 3, length 64
21:57:03.347886 IP romeo > juliet: ICMP echo reply, id 5, seq 4, length 64
21:57:04.371884 IP romeo > juliet: ICMP echo reply, id 5, seq 5, length 64
21:57:05.331931 ARP, Request who-has juliet tell romeo, length 28
21:57:05.395809 ARP, Reply romeo is-at 0a:4a:a6:5e:29:6a (oui Unknown), length 28
^C
7 packets captured
7 packets received by filter
0 packets dropped by kernel
ubuntu@romeo:~$ 
```

Deleting the Slice

Observations/Issues: None. The experiment is complete so we can delete this slice.



Delete your slice

When you finish your experiment, you should delete your slice! The following cells deletes all the resources in your slice, freeing them for other experimenters.

[21]:
```
slice = fablib.get_slice(name=slice_name)
fablib.delete_slice(slice_name)
```

[22]:
```
# slice should end up in "Dead" state
# re-run this cell until you see it in "Dead" state
slice.update()
_ = slice.show()
```

Slice

ID	3b20d99a-7acc-4a77-836c-19d3bddcdde1
Name	wireshark-sjack012_0000240143
Lease Expiration (UTC)	2024-10-29 19:20:09 +0000
Lease Start (UTC)	2024-10-28 19:20:09 +0000
Project ID	a70de2f5-9e12-4b6b-b412-0ae1a2c553b0
State	StableOK

In this lab I learned how to use TCPDump shell commands in Fabric. I performed several different TCPDump commands and learned how to export those files to be used in Wireshark for further analysis. I then proceeded to delete the slice for this project since CLab 7 and Clab 8 are completed.