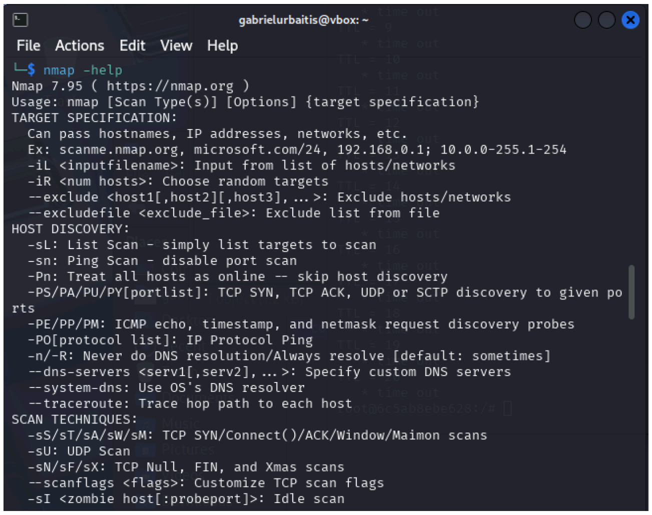
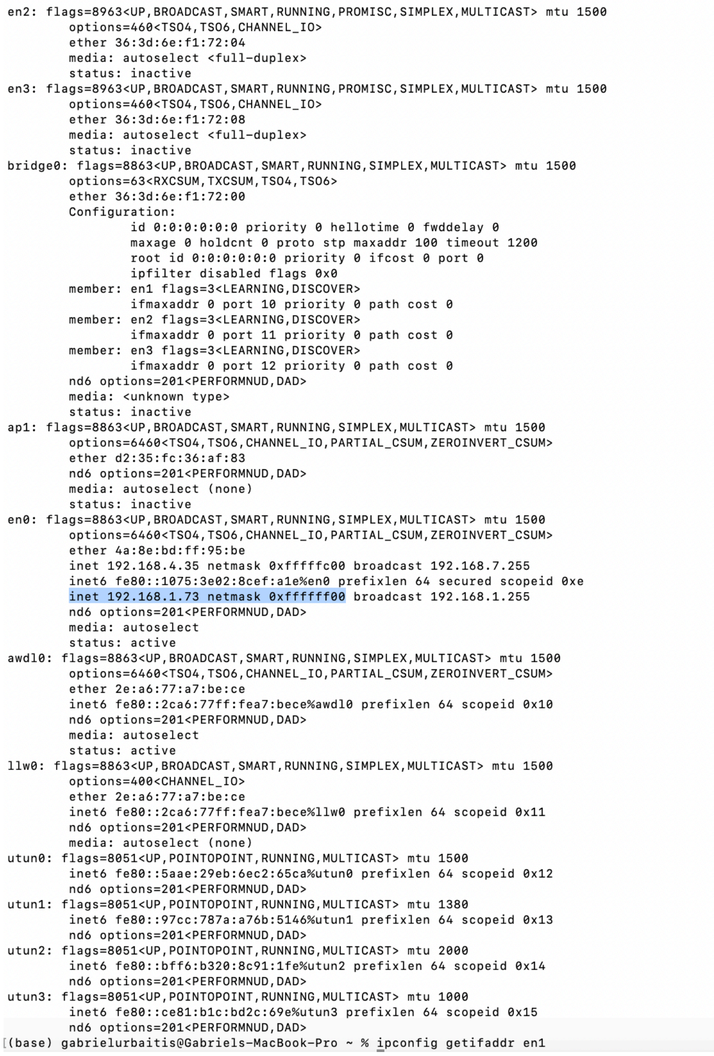
CS 544 Lab 1
Gabriel Urbaitis
Apr 10 2025

1.1



```
└─$ nmap -help
Nmap 7.95 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN, TCP ACK, UDP or SCTP discovery to given po
rts
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
```

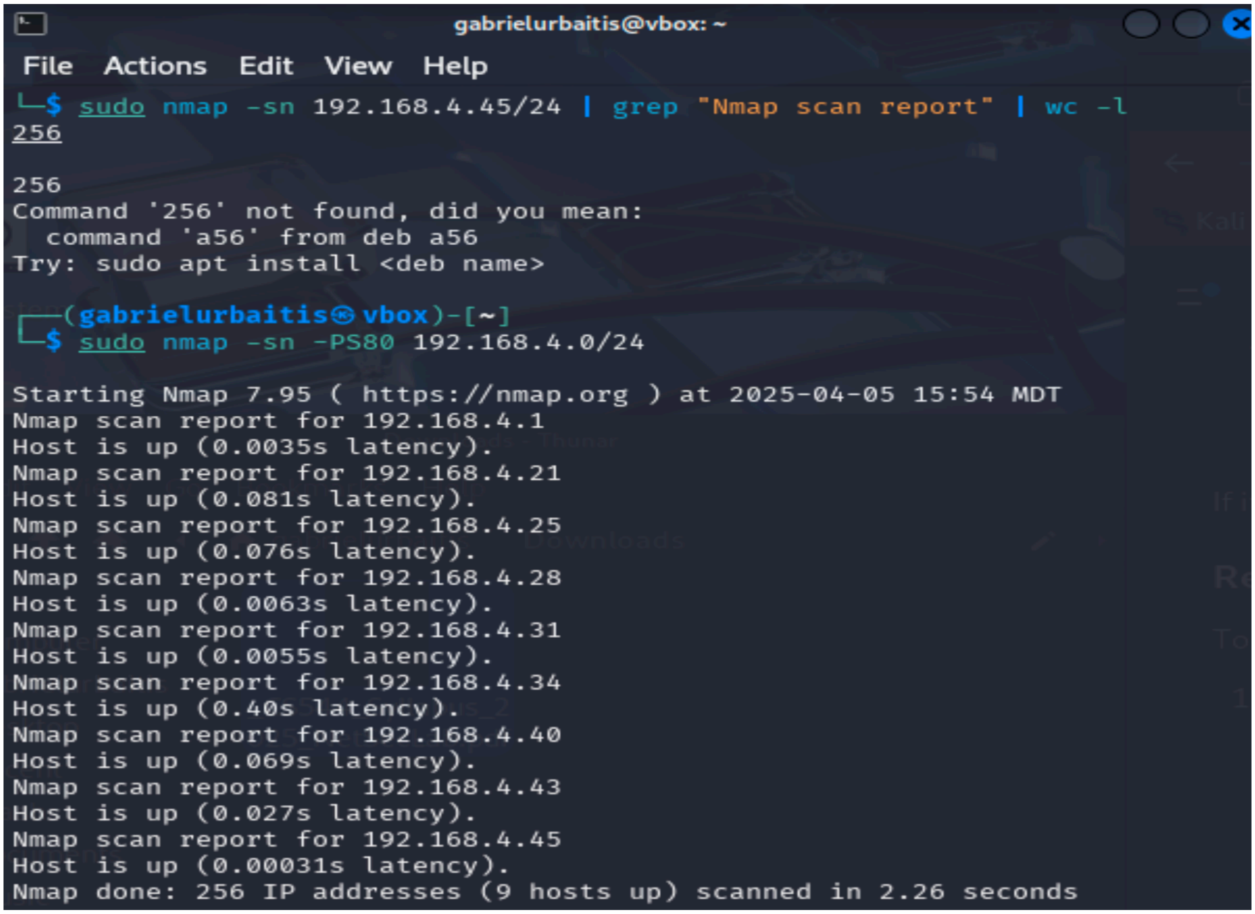
2.1



```
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=460<TSO4,TSO6,CHANNEL_IO>
        ether 36:3d:6e:f1:72:04
        media: autoselect <full-duplex>
        status: inactive
en3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=460<TSO4,TSO6,CHANNEL_IO>
        ether 36:3d:6e:f1:72:08
        media: autoselect <full-duplex>
        status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=63<RXCSUM,TXCSUM,TSO4,TSO6>
        ether 36:3d:6e:f1:72:00
        Configuration:
                id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
                maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
                root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
                ipfilter disabled flags 0x0
        member: en1 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 10 priority 0 path cost 0
        member: en2 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 11 priority 0 path cost 0
        member: en3 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 12 priority 0 path cost 0
        nd6 options=201<PERFORMNUD,DAD>
        media: <unknown type>
        status: inactive
ap1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether d2:35:fc:36:af:83
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (none)
        status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 4a:8e:bd:ff:95:be
        inet 192.168.4.35 netmask 0xfffffc00 broadcast 192.168.7.255
        inet6 fe80::1075:3e02:8cef:a1e%en0 prefixlen 64 secured scopeid 0xe
        inet 192.168.1.73 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
awdl0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 2e:a6:77:a7:be:ce
        inet6 fe80::2ca6:77ff:fea7:bece%awdl0 prefixlen 64 scopeid 0x10
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 2e:a6:77:a7:be:ce
        inet6 fe80::2ca6:77ff:fea7:bece%llw0 prefixlen 64 scopeid 0x11
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (none)
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet6 fe80::5aae:29eb:6ec2:65ca%utun0 prefixlen 64 scopeid 0x12
        nd6 options=201<PERFORMNUD,DAD>
utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1380
        inet6 fe80::97cc:787a:a76b:5146%utun1 prefixlen 64 scopeid 0x13
        nd6 options=201<PERFORMNUD,DAD>
utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
        inet6 fe80::bff6:b320:8c91:1fe%utun2 prefixlen 64 scopeid 0x14
        nd6 options=201<PERFORMNUD,DAD>
utun3: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1000
        inet6 fe80::ce81:b1c:bd2c:69e%utun3 prefixlen 64 scopeid 0x15
        nd6 options=201<PERFORMNUD,DAD>
(base) gabrielurbaitis@Gabriels-MacBook-Pro ~ % ipconfig getifaddr en1
```

My local network in CIDR format is 192.168.1.73/24.
(inet 192.168.1.73 netmask 0xffffff00)

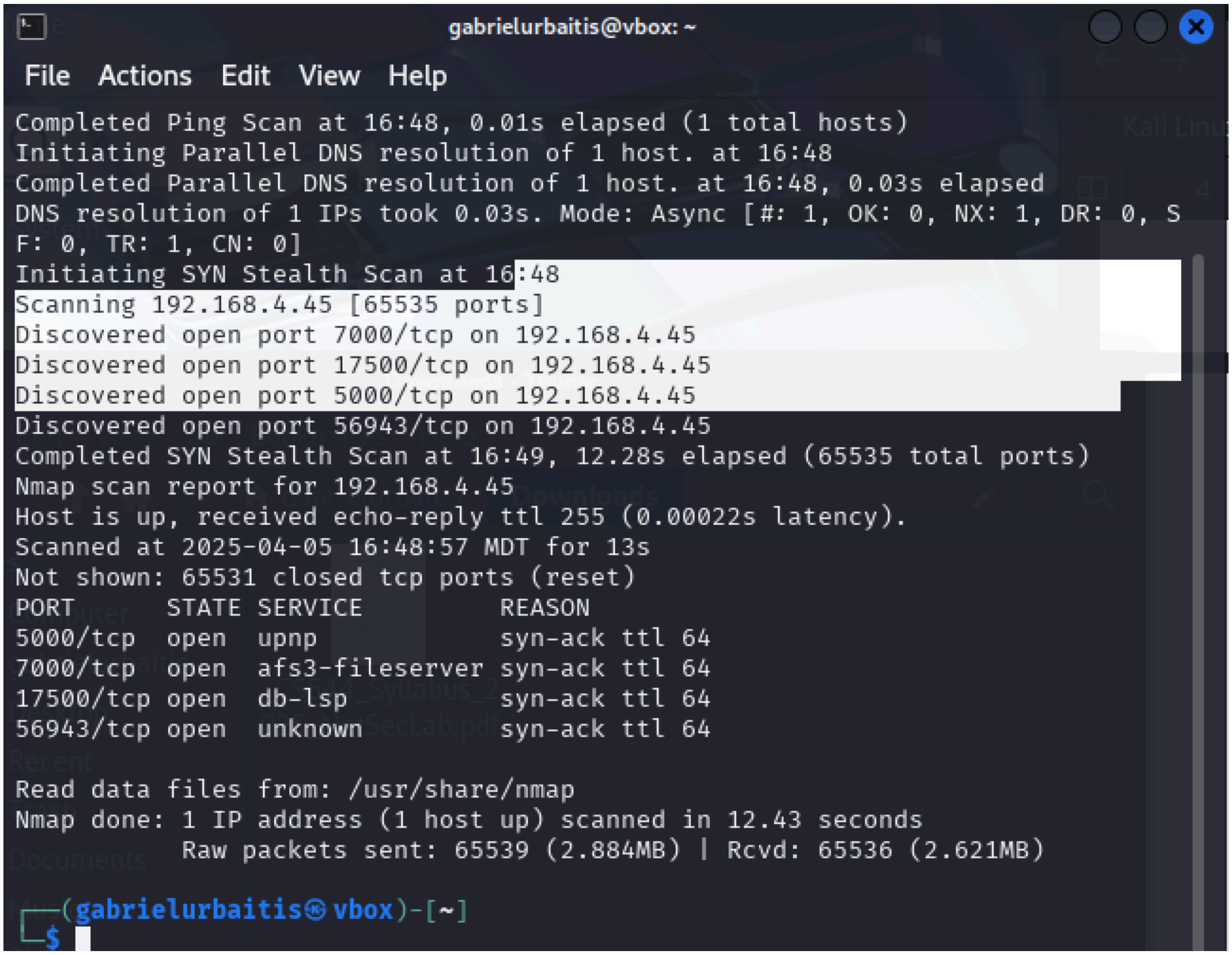
2.2, 2.3



```
└─$ sudo nmap -sn 192.168.4.45/24 | grep "Nmap scan report" | wc -l
256

256
Command '256' not found, did you mean:
  command 'a56' from deb a56
Try: sudo apt install <deb name>

┌──(gabrielurbaitis㉿vbox)-[~]
└─$ sudo nmap -sn -PS80 192.168.4.0/24

Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-05 15:54 MDT
Nmap scan report for 192.168.4.1
Host is up (0.0035s latency).
Nmap scan report for 192.168.4.21
Host is up (0.081s latency).
Nmap scan report for 192.168.4.25
Host is up (0.076s latency).
Nmap scan report for 192.168.4.28
Host is up (0.0063s latency).
Nmap scan report for 192.168.4.31
Host is up (0.0055s latency).
Nmap scan report for 192.168.4.34
Host is up (0.40s latency).
Nmap scan report for 192.168.4.40
Host is up (0.069s latency).
Nmap scan report for 192.168.4.43
Host is up (0.027s latency).
Nmap scan report for 192.168.4.45
Host is up (0.00031s latency).
Nmap done: 256 IP addresses (9 hosts up) scanned in 2.26 seconds
```

Apparently if ICMP is disabled or blocked on the firewall or OS all 256 show as up, so using the -PS80 flag sends a TCP SYN packet to port 80 which shows 9 on my home network (much more reasonable). 192.16585.4.45 is my work computer that I was doing the lab on.

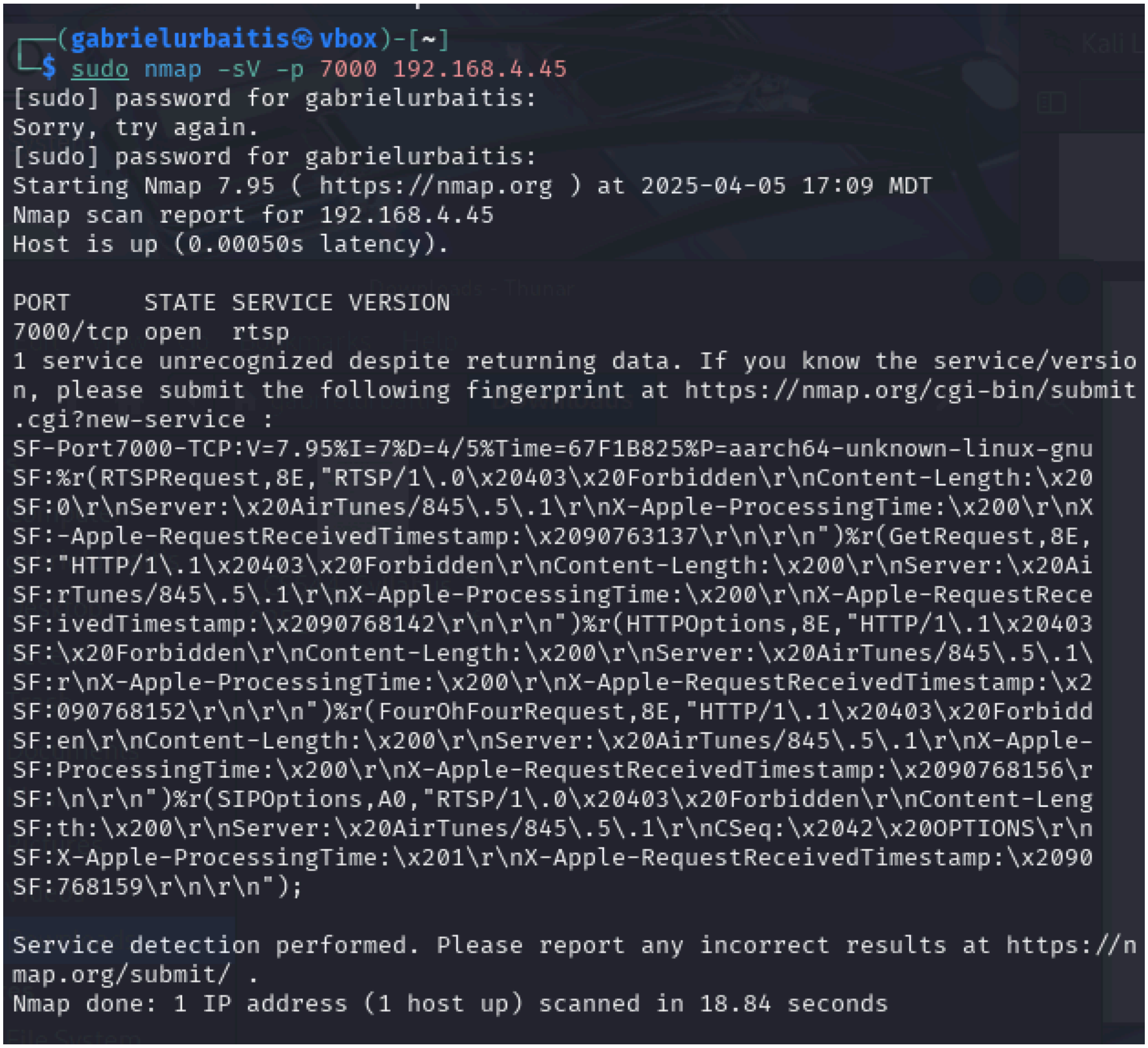
2.4



```
Completed Ping Scan at 16:48, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 16:48
Completed Parallel DNS resolution of 1 host. at 16:48, 0.03s elapsed
DNS resolution of 1 IPs took 0.03s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, S
F: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 16:48
Scanning 192.168.4.45 [65535 ports]
Discovered open port 7000/tcp on 192.168.4.45
Discovered open port 17500/tcp on 192.168.4.45
Discovered open port 5000/tcp on 192.168.4.45
Discovered open port 56943/tcp on 192.168.4.45
Completed SYN Stealth Scan at 16:49, 12.28s elapsed (65535 total ports)
Nmap scan report for 192.168.4.45
Host is up, received echo-reply ttl 255 (0.00022s latency).
Scanned at 2025-04-05 16:48:57 MDT for 13s
Not shown: 65531 closed tcp ports (reset)
PORT      STATE SERVICE         REASON
5000/tcp  open  upnp            syn-ack ttl 64
7000/tcp  open  afs3-fileserver syn-ack ttl 64
17500/tcp open  db-lsp          syn-ack ttl 64
56943/tcp open  unknown         syn-ack ttl 64

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 12.43 seconds
           Raw packets sent: 65539 (2.884MB) | Rcvd: 65536 (2.621MB)

┌──(gabrielurbaitis㉿vbox)-[~]
└─$
```

-p- tells nmap to scan all 65,535 TCP ports
-sS sends SYN packets and watches for SYN-ACKs, without completing the TCP handshake
—open only shows ports that are open
--min-rate 5000 sets minimum scan rate to 5000 packets per second
-vvv very verbose mode, includes each probe sent, port status, and more.
Yes:
5000 upnp Universal Plug and Play, used for device discovery
7000 afs3-fileserver, Andrew File System
17500 db-lsp Dropbox LAN sync protocol
56943 unknown

2.5



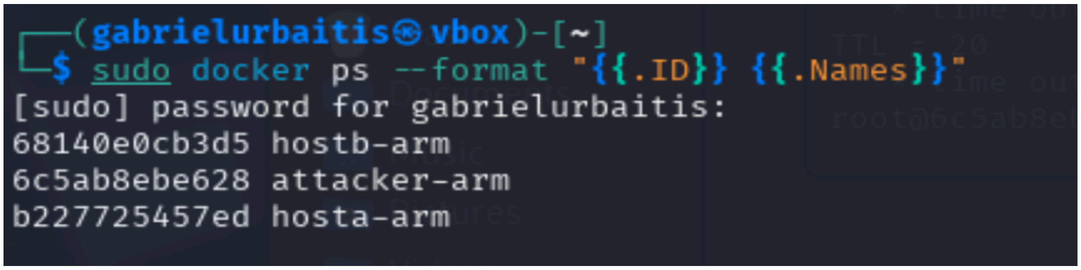
```
┌──(gabrielurbaitis㉿vbox)-[~]
└─$ sudo nmap -sV -p 7000 192.168.4.45
[sudo] password for gabrielurbaitis:
Sorry, try again.
[sudo] password for gabrielurbaitis:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-05 17:09 MDT
Nmap scan report for 192.168.4.45
Host is up (0.00050s latency).

PORT     STATE SERVICE VERSION
7000/tcp open  rtsp
1 service unrecognized despite returning data. If you know the service/versio
n, please submit the following fingerprint at https://nmap.org/cgi-bin/submit
.cgi?new-service :
SF-Port7000-TCP:V=7.95%I=7%D=4/5%Time=67F1B825%P=aarch64-unknown-linux-gnu
SF:%r(RTSPRequest,8E,"RTSP/1\.0\x20403\x20Forbidden\r\nContent-Length:\x20
SF:0\r\nServer:\x20AirTunes/845\.5\.1\r\nX-Apple-ProcessingTime:\x200\r\nX
SF:-Apple-RequestReceivedTimestamp:\x2090763137\r\n\r\n")%r(GetRequest,8E,
SF:"HTTP/1\.1\x20403\x20Forbidden\r\nContent-Length:\x200\r\nServer:\x20Ai
SF:rTunes/845\.5\.1\r\nX-Apple-ProcessingTime:\x200\r\nX-Apple-RequestRece
SF:ivedTimestamp:\x2090768142\r\n\r\n")%r(HTTPOptions,8E,"HTTP/1\.1\x20403
SF:\x20Forbidden\r\nContent-Length:\x200\r\nServer:\x20AirTunes/845\.5\.1\
SF:r\nX-Apple-ProcessingTime:\x200\r\nX-Apple-RequestReceivedTimestamp:\x2
SF:090768152\r\n\r\n")%r(FourOhFourRequest,8E,"HTTP/1\.1\x20403\x20Forbidd
SF:en\r\nContent-Length:\x200\r\nServer:\x20AirTunes/845\.5\.1\r\nX-Apple-
SF:ProcessingTime:\x200\r\nX-Apple-RequestReceivedTimestamp:\x2090768156\r
SF:\n\r\n")%r(SIPOptions,A0,"RTSP/1\.0\x20403\x20Forbidden\r\nContent-Leng
SF:th:\x200\r\nServer:\x20AirTunes/845\.5\.1\r\nCSeq:\x2042\x20OPTIONS\r\n
SF:X-Apple-ProcessingTime:\x201\r\nX-Apple-RequestReceivedTimestamp:\x2090
SF:768159\r\n\r\n");

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.84 seconds
```

RTSP, AirTunes

2.6 Open ports increase your attack surface, and open services could be misconfigured or have known vulnerabilities that hackers could exploit

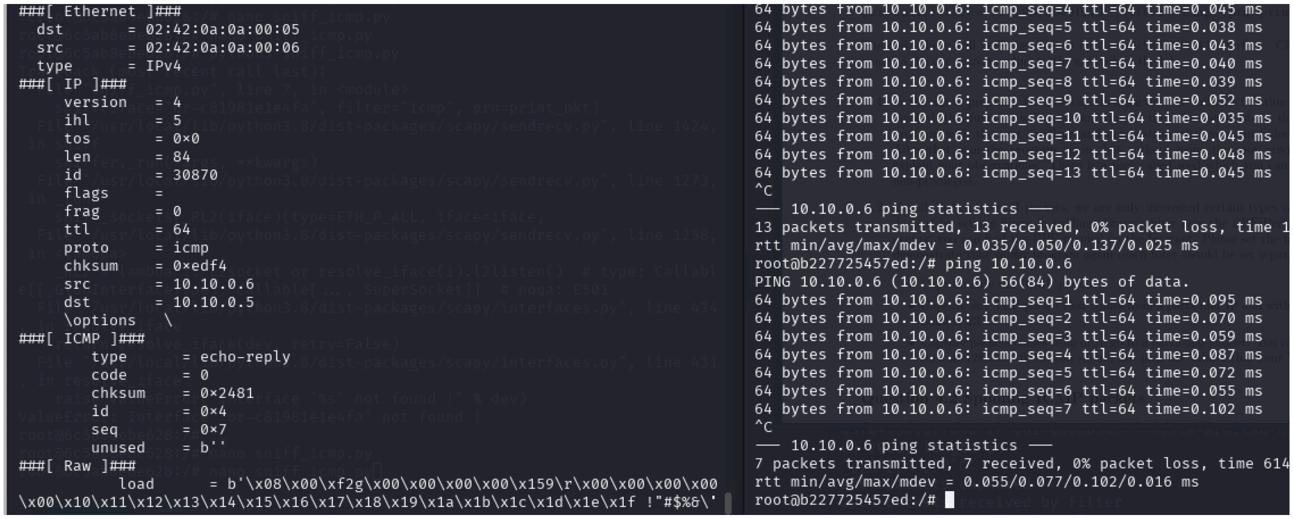
2.7 It is possible, but there are consequences. The Computer Fraud and Abuse Act prohibits unauthorized access or attempted access to protected systems. Port scanning can be interpreted as probing for vulnerabilities.

3. I had to use custom docker files to get it to work on my MacBook M4, which I will attach in Canvas if it allows. Feel free to use it next year with your students with Apple machines, the main change is using Ubuntu 20.04 which supports ARM.



```
┌──(gabrielurbaitis㉿vbox)-[~]
└─$ sudo docker ps --format "{{.ID}} {{.Names}}"
[sudo] password for gabrielurbaitis:
68140e0cb3d5 hostb-arm
6c5ab8ebe628 attacker-arm
b227725457ed hosta-arm
```

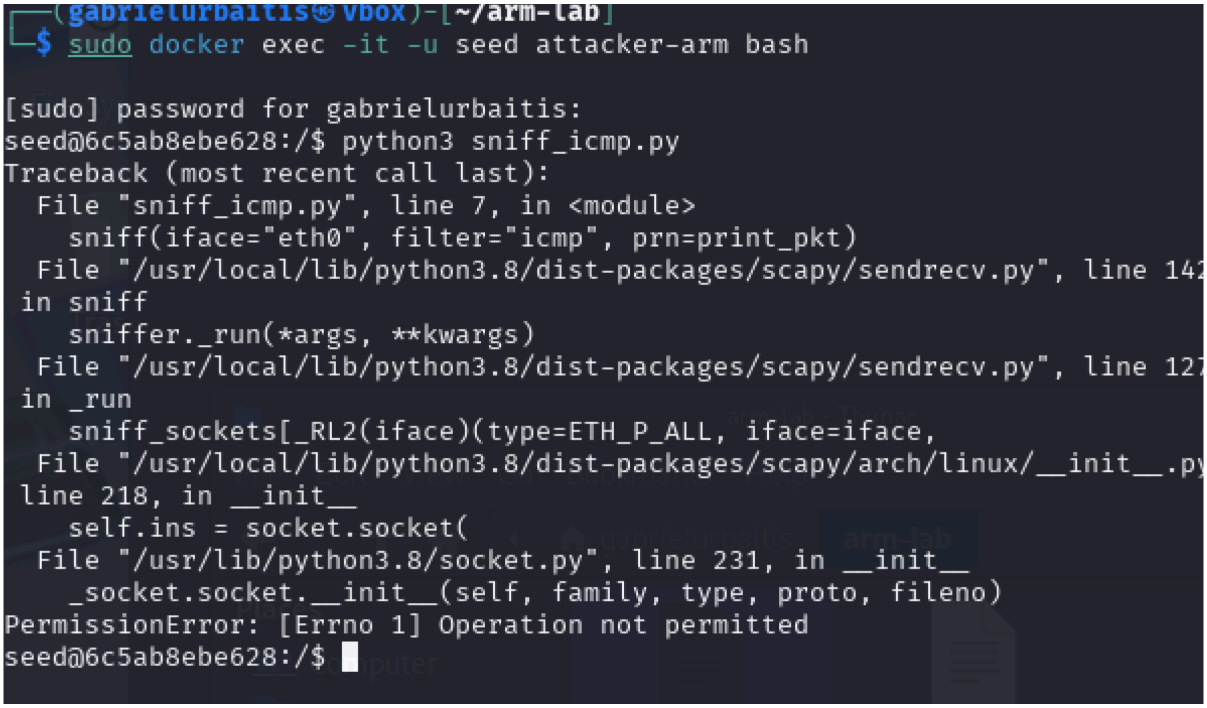
4.1.1
Root:



```
###[ Ethernet ]###
  dst       = 02:42:0a:0a:00:05
  src       = 02:42:0a:0a:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 30870
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xedf4
     src       = 10.10.0.6
     dst       = 10.10.0.5
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0x2481
        id        = 0x4
        seq       = 0x7
        unused    = b''
###[ Raw ]###
           load      = b'\x08\x00\xf2g\x00\x00\x00\x00\x159\r\x00\x00\x00\x00
\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'
```

```
64 bytes from 10.10.0.6: icmp_seq=4 ttl=64 time=0.045 ms
64 bytes from 10.10.0.6: icmp_seq=5 ttl=64 time=0.038 ms
64 bytes from 10.10.0.6: icmp_seq=6 ttl=64 time=0.043 ms
64 bytes from 10.10.0.6: icmp_seq=7 ttl=64 time=0.040 ms
64 bytes from 10.10.0.6: icmp_seq=8 ttl=64 time=0.039 ms
64 bytes from 10.10.0.6: icmp_seq=9 ttl=64 time=0.052 ms
64 bytes from 10.10.0.6: icmp_seq=10 ttl=64 time=0.035 ms
64 bytes from 10.10.0.6: icmp_seq=11 ttl=64 time=0.045 ms
64 bytes from 10.10.0.6: icmp_seq=12 ttl=64 time=0.048 ms
64 bytes from 10.10.0.6: icmp_seq=13 ttl=64 time=0.045 ms
^C
--- 10.10.0.6 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 1
rtt min/avg/max/mdev = 0.035/0.050/0.137/0.025 ms
root@b227725457ed:/# ping 10.10.0.6
PING 10.10.0.6 (10.10.0.6) 56(84) bytes of data.
64 bytes from 10.10.0.6: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 10.10.0.6: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 10.10.0.6: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 10.10.0.6: icmp_seq=4 ttl=64 time=0.087 ms
64 bytes from 10.10.0.6: icmp_seq=5 ttl=64 time=0.072 ms
64 bytes from 10.10.0.6: icmp_seq=6 ttl=64 time=0.055 ms
64 bytes from 10.10.0.6: icmp_seq=7 ttl=64 time=0.102 ms
^C
--- 10.10.0.6 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 614
rtt min/avg/max/mdev = 0.055/0.077/0.102/0.016 ms
root@b227725457ed:/# 
```

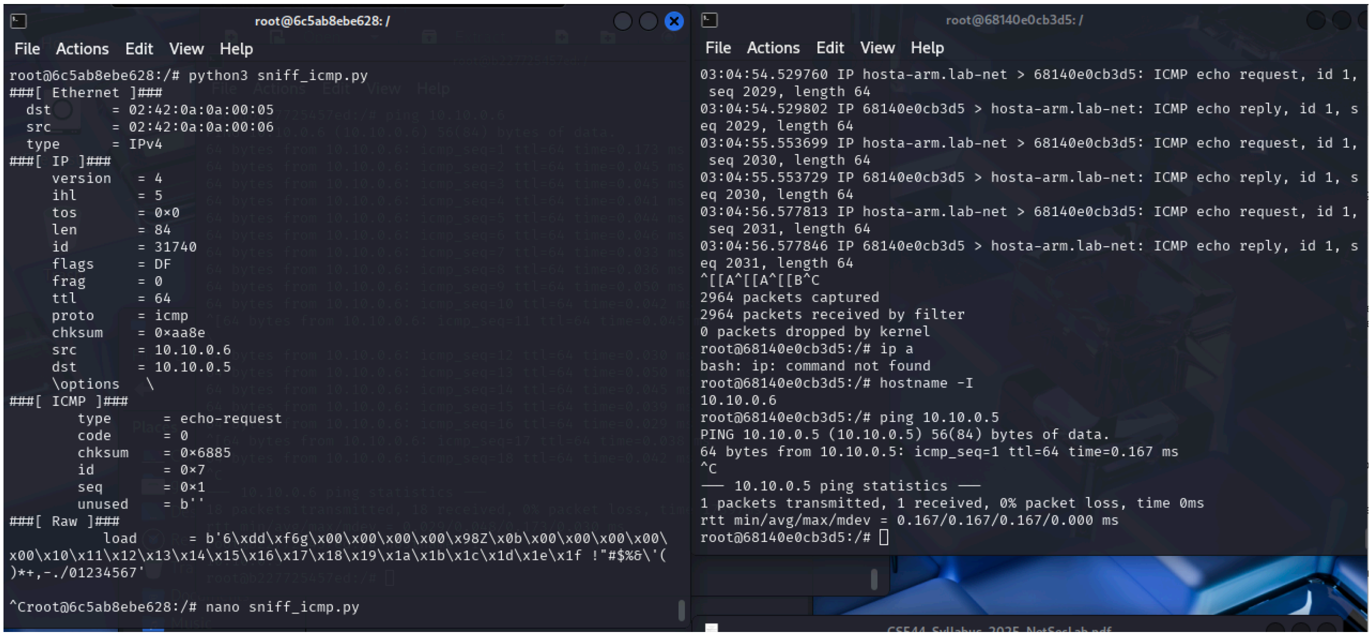
Seed:



```
┌──(gabrielurbaitis㉿vbox)-[~/arm-lab]
└─$ sudo docker exec -it -u seed attacker-arm bash

[sudo] password for gabrielurbaitis:
seed@6c5ab8ebe628:/$ python3 sniff_icmp.py
Traceback (most recent call last):
  File "sniff_icmp.py", line 7, in <module>
    sniff(iface="eth0", filter="icmp", prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 142
 in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 127
 in _run
    sniff_sockets[_RL2(iface)(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux/__init__.py
 line 218, in __init__
    self.ins = socket.socket(
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
seed@6c5ab8ebe628:/$ █
```

Scapy needs access to raw sockets, which are restricted to root, so the operation is not permitted.

4.1.2
ICMP:
Script: Same as in Assignment
Capture:

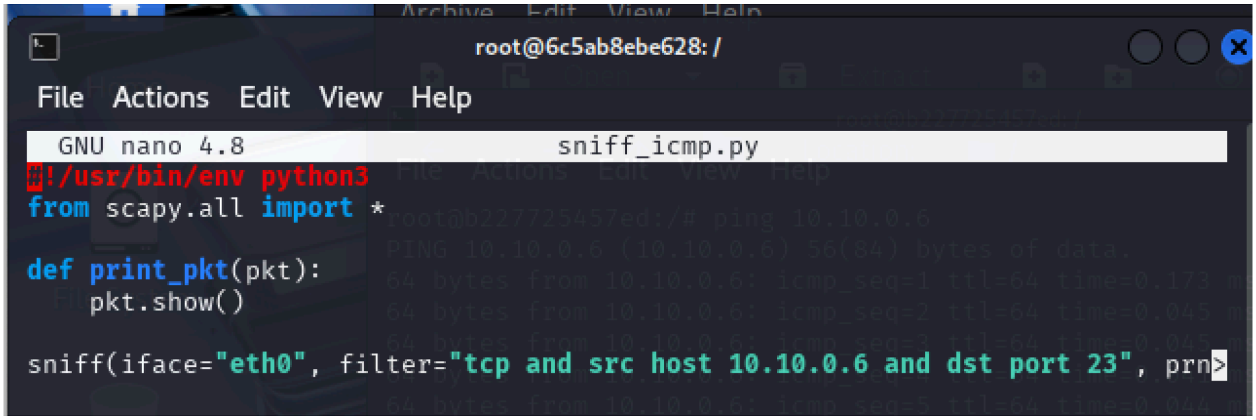


```
root@6c5ab8ebe628:/# python3 sniff_icmp.py
###[ Ethernet ]###
  dst       = 02:42:0a:0a:00:05
  src       = 02:42:0a:0a:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 31740
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xaa8e
     src       = 10.10.0.6
     dst       = 10.10.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x6885
        id        = 0x7
        seq       = 0x1
        unused    = b''
###[ Raw ]###
           load      = b'6\xdd\xf6g\x00\x00\x00\x00\x98Z\x0b\x00\x00\x00\x00\
x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'(
)*+,-./01234567'

^Croot@6c5ab8ebe628:/# nano sniff_icmp.py
```

```
03:04:54.529760 IP hosta-arm.lab-net > 68140e0cb3d5: ICMP echo request, id 1,
 seq 2029, length 64
03:04:54.529802 IP 68140e0cb3d5 > hosta-arm.lab-net: ICMP echo reply, id 1, s
eq 2029, length 64
03:04:55.553699 IP hosta-arm.lab-net > 68140e0cb3d5: ICMP echo request, id 1,
 seq 2030, length 64
03:04:55.553729 IP 68140e0cb3d5 > hosta-arm.lab-net: ICMP echo reply, id 1, s
eq 2030, length 64
03:04:56.577813 IP hosta-arm.lab-net > 68140e0cb3d5: ICMP echo request, id 1,
 seq 2031, length 64
03:04:56.577846 IP 68140e0cb3d5 > hosta-arm.lab-net: ICMP echo reply, id 1, s
eq 2031, length 64
^[[A^[[A^[[B^C
2964 packets captured
2964 packets received by filter
0 packets dropped by kernel
root@68140e0cb3d5:/# ip a
bash: ip: command not found
root@68140e0cb3d5:/# hostname -I
10.10.0.6
root@68140e0cb3d5:/# ping 10.10.0.5
PING 10.10.0.5 (10.10.0.5) 56(84) bytes of data.
64 bytes from 10.10.0.5: icmp_seq=1 ttl=64 time=0.167 ms
^C
—— 10.10.0.5 ping statistics ——
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.167/0.167/0.167/0.000 ms
root@68140e0cb3d5:/# ▯
```

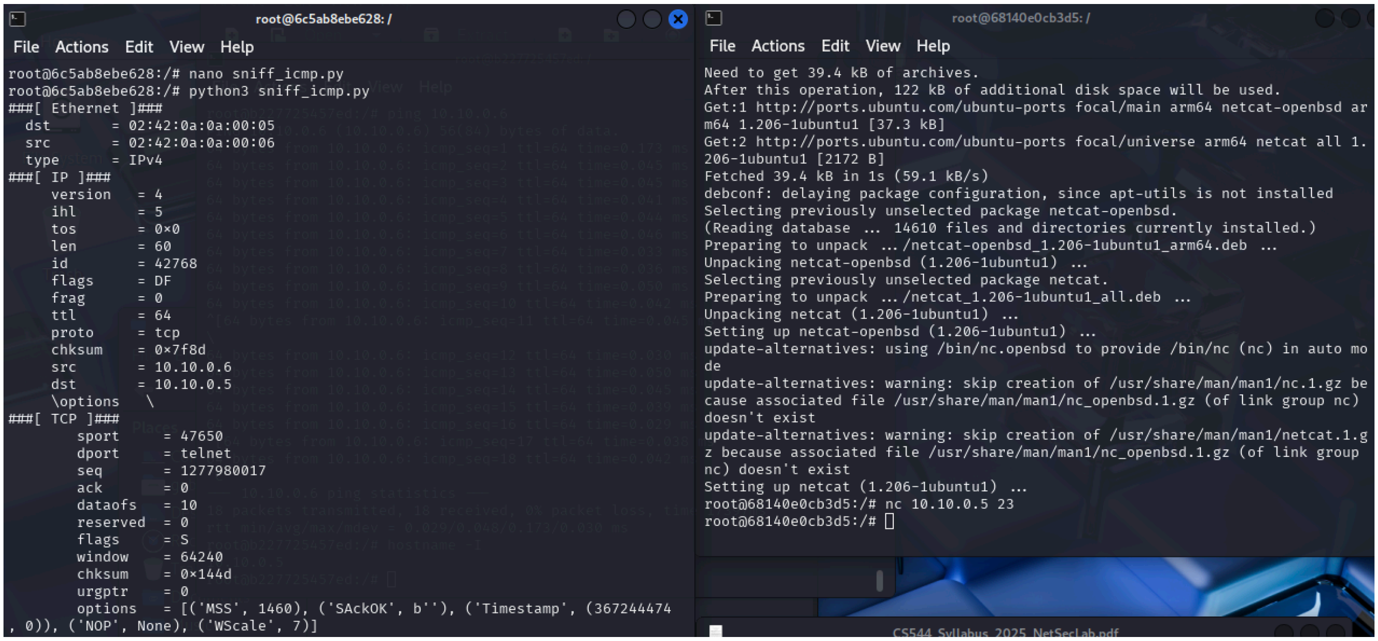
TCP:

Script:



```
  GNU nano 4.8                        sniff_icmp.py
#!/usr/bin/env python3
from scapy.all import *

def print_pkt(pkt):
    pkt.show()

sniff(iface="eth0", filter="tcp and src host 10.10.0.6 and dst port 23", prn>
```

Capture:

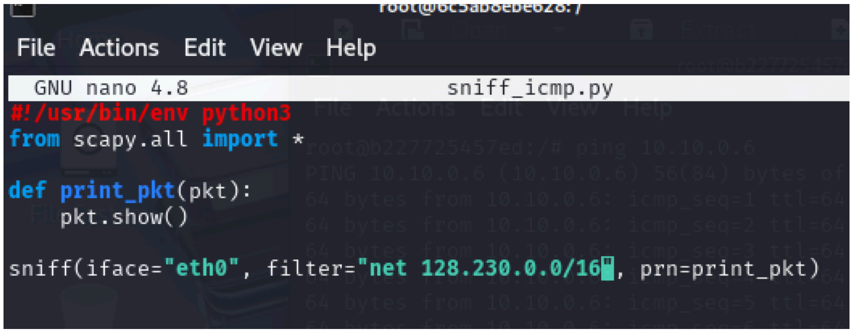


```
root@6c5ab8ebe628:/# nano sniff_icmp.py
root@6c5ab8ebe628:/# python3 sniff_icmp.py
###[ Ethernet ]###
  dst       = 02:42:0a:0a:00:05
  src       = 02:42:0a:0a:00:06
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 60
     id        = 42768
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x7f8d
     src       = 10.10.0.6
     dst       = 10.10.0.5
     \options   \
###[ TCP ]###
        sport     = 47650
        dport     = telnet
        seq       = 1277980017
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x144d
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (367244474
, 0)), ('NOP', None), ('WScale', 7)]
```

```
Need to get 39.4 kB of archives.
After this operation, 122 kB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports focal/main arm64 netcat-openbsd ar
m64 1.206-1ubuntu1 [37.3 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports focal/universe arm64 netcat all 1.
206-1ubuntu1 [2172 B]
Fetched 39.4 kB in 1s (59.1 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package netcat-openbsd.
(Reading database ... 14610 files and directories currently installed.)
Preparing to unpack .../netcat-openbsd_1.206-1ubuntu1_arm64.deb ...
Unpacking netcat-openbsd (1.206-1ubuntu1) ...
Selecting previously unselected package netcat.
Preparing to unpack .../netcat_1.206-1ubuntu1_all.deb ...
Unpacking netcat (1.206-1ubuntu1) ...
Setting up netcat-openbsd (1.206-1ubuntu1) ...
update-alternatives: using /bin/nc.openbsd to provide /bin/nc (nc) in auto mo
de
update-alternatives: warning: skip creation of /usr/share/man/man1/nc.1.gz be
cause associated file /usr/share/man/man1/nc_openbsd.1.gz (of link group nc)
doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/netcat.1.g
z because associated file /usr/share/man/man1/nc_openbsd.1.gz (of link group
nc) doesn't exist
Setting up netcat (1.206-1ubuntu1) ...
root@68140e0cb3d5:/# nc 10.10.0.5 23
root@68140e0cb3d5:/# 
```

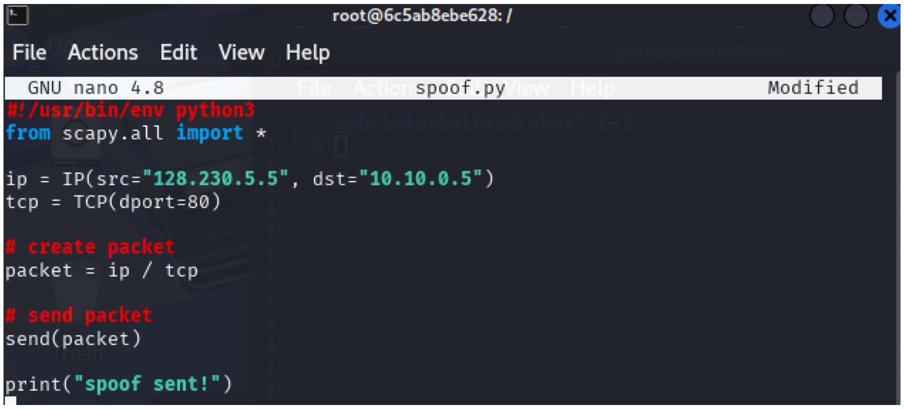
Subnet 128.230.0.0/16:

Script: I realized I had to do this on another docker, so I retitled it sniff_spoof as you'll see in the capture



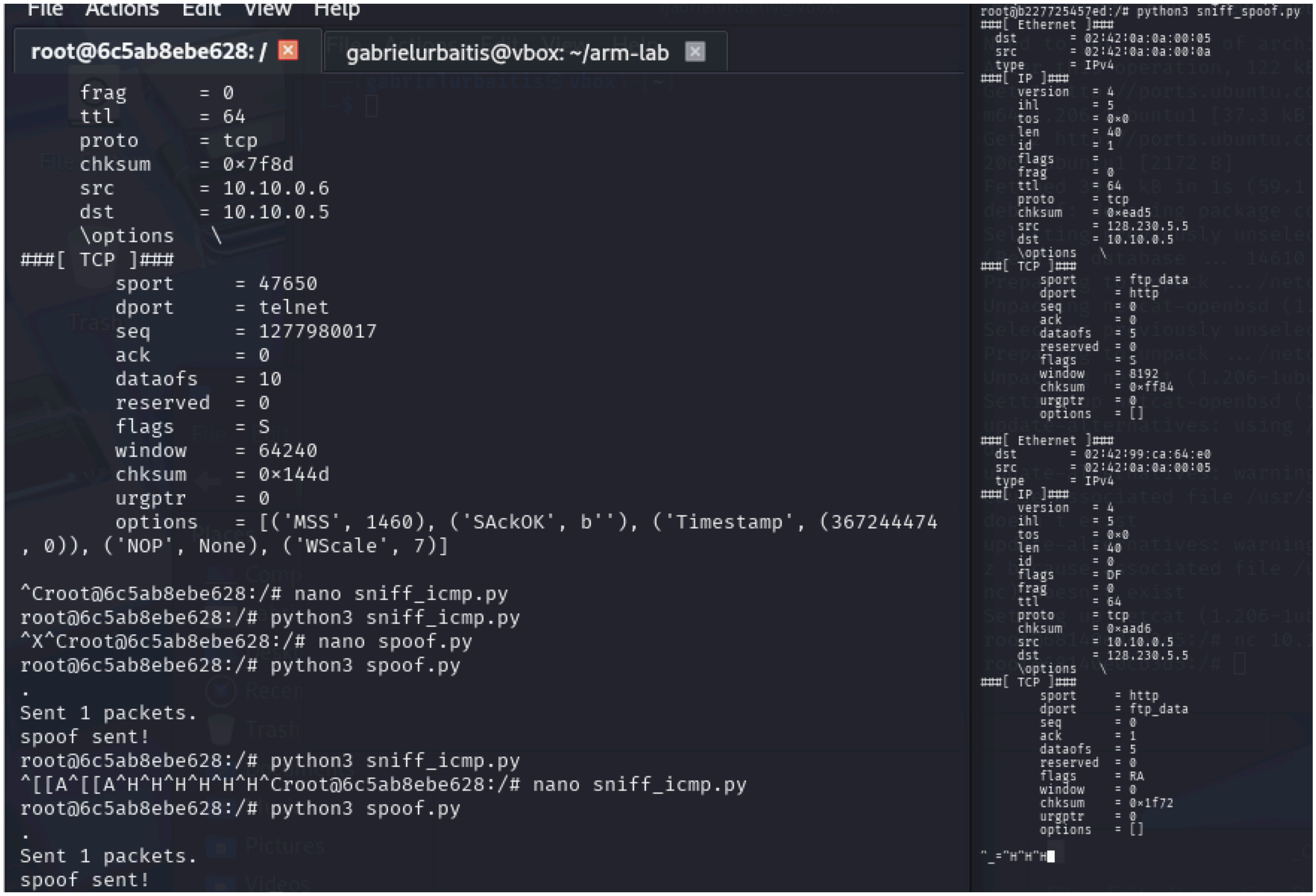
```
#!/usr/bin/env python3
from scapy.all import *

def print_pkt(pkt):
    pkt.show()

sniff(iface="eth0", filter="net 128.230.0.0/16", prn=print_pkt)
```

Spoof traffic:



```
#!/usr/bin/env python3
from scapy.all import *

ip = IP(src="128.230.5.5", dst="10.10.0.5")
tcp = TCP(dport=80)

# create packet
packet = ip / tcp

# send packet
send(packet)

print("spoof sent!")
```

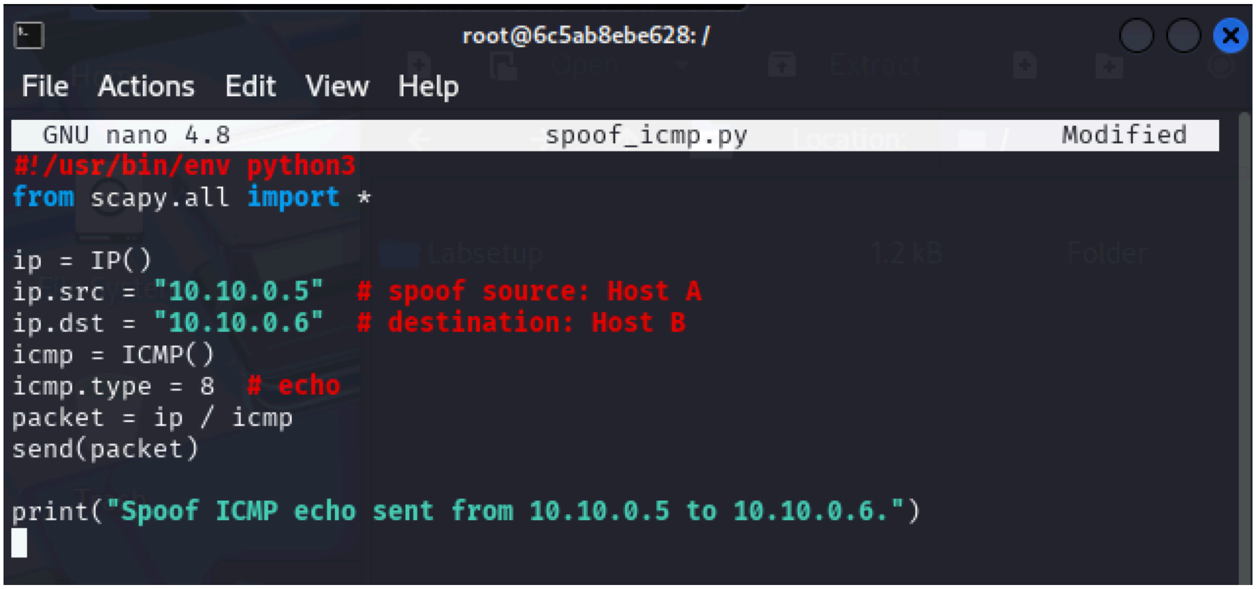
Capture:



```
     frag       = 0
     ttl        = 64
     proto      = tcp
     chksum     = 0x7f8d
     src        = 10.10.0.6
     dst        = 10.10.0.5
     \options    \
###[ TCP ]###
        sport     = 47650
        dport     = telnet
        seq       = 1277980017
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x144d
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (367244474
, 0)), ('NOP', None), ('WScale', 7)]

^Croot@6c5ab8ebe628:/# nano sniff_icmp.py
root@6c5ab8ebe628:/# python3 sniff_icmp.py
^X^Croot@6c5ab8ebe628:/# nano spoof.py
root@6c5ab8ebe628:/# python3 spoof.py
.
Sent 1 packets.
spoof sent!
root@6c5ab8ebe628:/# python3 sniff_icmp.py
^[[A^[[A^H^H^H^H^H^H^Croot@6c5ab8ebe628:/# nano sniff_icmp.py
root@6c5ab8ebe628:/# python3 spoof.py
.
Sent 1 packets.
spoof sent!
```

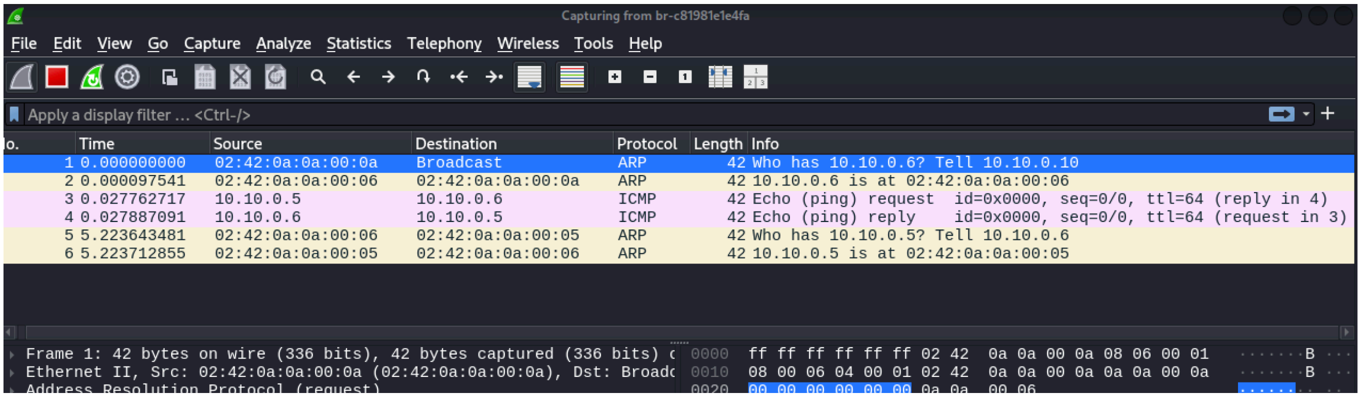
```
root@b227725457ed:/# python3 sniff_spoof.py
###[ Ethernet ]###
  dst       = 02:42:0a:0a:00:05
  src       = 02:42:0a:0a:00:0a
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xead5
     src       = 128.230.5.5
     dst       = 10.10.0.5
     \options   \
###[ TCP ]###
        sport     = ftp_data
        dport     = http
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0xff84
        urgptr    = 0
        options   = []

###[ Ethernet ]###
  dst       = 02:42:99:ca:64:e0
  src       = 02:42:0a:0a:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 0
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xaad6
     src       = 10.10.0.5
     dst       = 128.230.5.5
     \options   \
###[ TCP ]###
        sport     = http
        dport     = ftp_data
        seq       = 0
        ack       = 1
        dataofs   = 5
        reserved  = 0
        flags     = RA
        window    = 0
        chksum    = 0x1f72
        urgptr    = 0
        options   = []

^_=^H^H^H
```

4.2 Script:



```
#!/usr/bin/env python3
from scapy.all import *

ip = IP()
ip.src = "10.10.0.5"  # spoof source: Host A
ip.dst = "10.10.0.6"  # destination: Host B
icmp = ICMP()
icmp.type = 8  # echo
packet = ip / icmp
send(packet)

print("Spoof ICMP echo sent from 10.10.0.5 to 10.10.0.6.")
```

Wireshark:

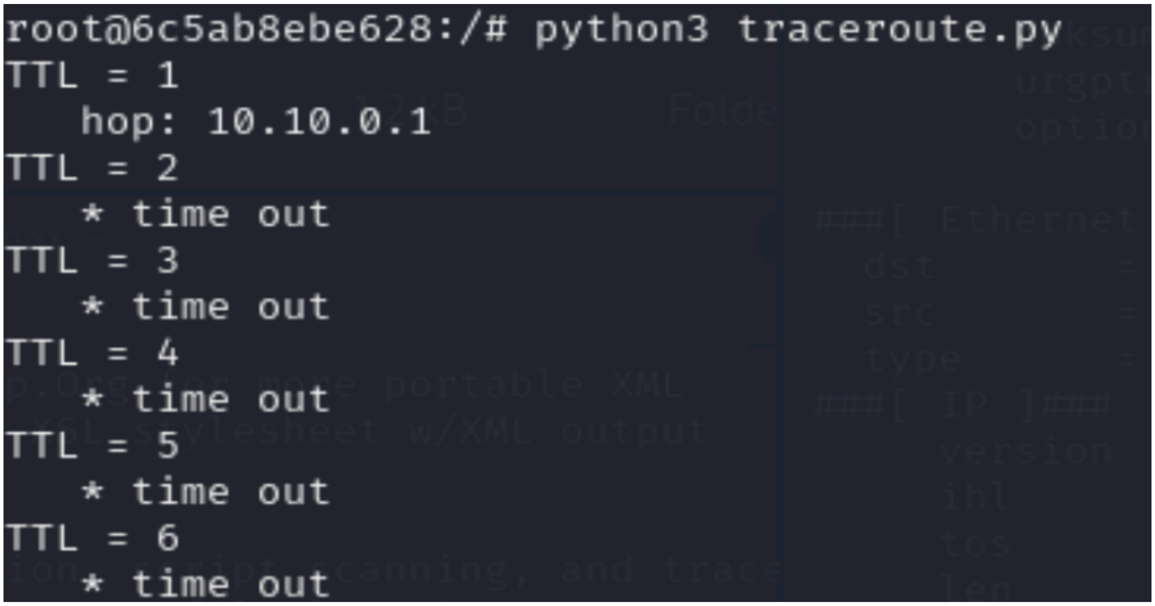


o.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	02:42:0a:0a:00:0a	Broadcast	ARP	42	Who has 10.10.0.6? Tell 10.10.0.10
2	0.000097541	02:42:0a:0a:00:06	02:42:0a:0a:00:0a	ARP	42	10.10.0.6 is at 02:42:0a:0a:00:06
3	0.027762717	10.10.0.5	10.10.0.6	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 4)
4	0.027887091	10.10.0.6	10.10.0.5	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 3)
5	5.223643481	02:42:0a:0a:00:06	02:42:0a:0a:00:05	ARP	42	Who has 10.10.0.5? Tell 10.10.0.6
6	5.223712855	02:42:0a:0a:00:05	02:42:0a:0a:00:06	ARP	42	10.10.0.5 is at 02:42:0a:0a:00:05

```
Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) c
Ethernet II, Src: 02:42:0a:0a:00:0a (02:42:0a:0a:00:0a), Dst: Broadc
Address Resolution Protocol (request)
```

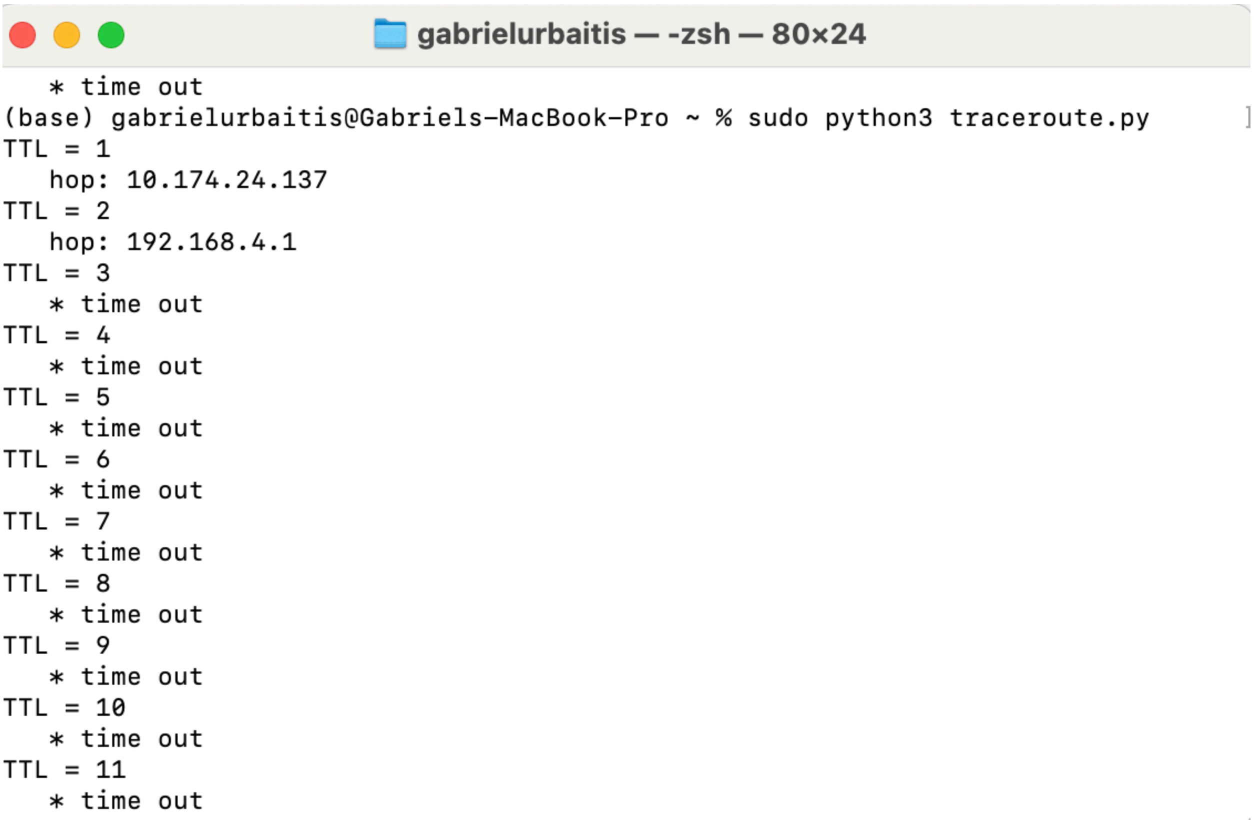
```
0000  ff ff ff ff ff ff 02 42  0a 0a 00 0a 08 06 00 01   ·······B ···
0010  08 00 06 04 00 01 02 42  0a 0a 00 0a 0a 0a 00 0a   ·······B ···
0020  00 00 00 00 00 00 0a 0a  00 06                     ······ · ··
```

4.3
Code attached separately per instructions.
On my virtual machine I can only get one hop before timeout



```
root@6c5ab8ebe628:/# python3 traceroute.py
TTL = 1
   hop: 10.10.0.1
TTL = 2
   * time out
TTL = 3
   * time out
TTL = 4
   * time out
TTL = 5
   * time out
TTL = 6
   * time out
```

If I use a hotspot on my Mac I can get two hops.



```
   * time out
(base) gabrielurbaitis@Gabriels-MacBook-Pro ~ % sudo python3 traceroute.py
TTL = 1
   hop: 10.174.24.137
TTL = 2
   hop: 192.168.4.1
TTL = 3
   * time out
TTL = 4
   * time out
TTL = 5
   * time out
TTL = 6
   * time out
TTL = 7
   * time out
TTL = 8
   * time out
TTL = 9
   * time out
TTL = 10
   * time out
TTL = 11
   * time out
```

This is likely because of the ICMP blocking I referred to earlier. It seems very common, I experienced something similar at work.