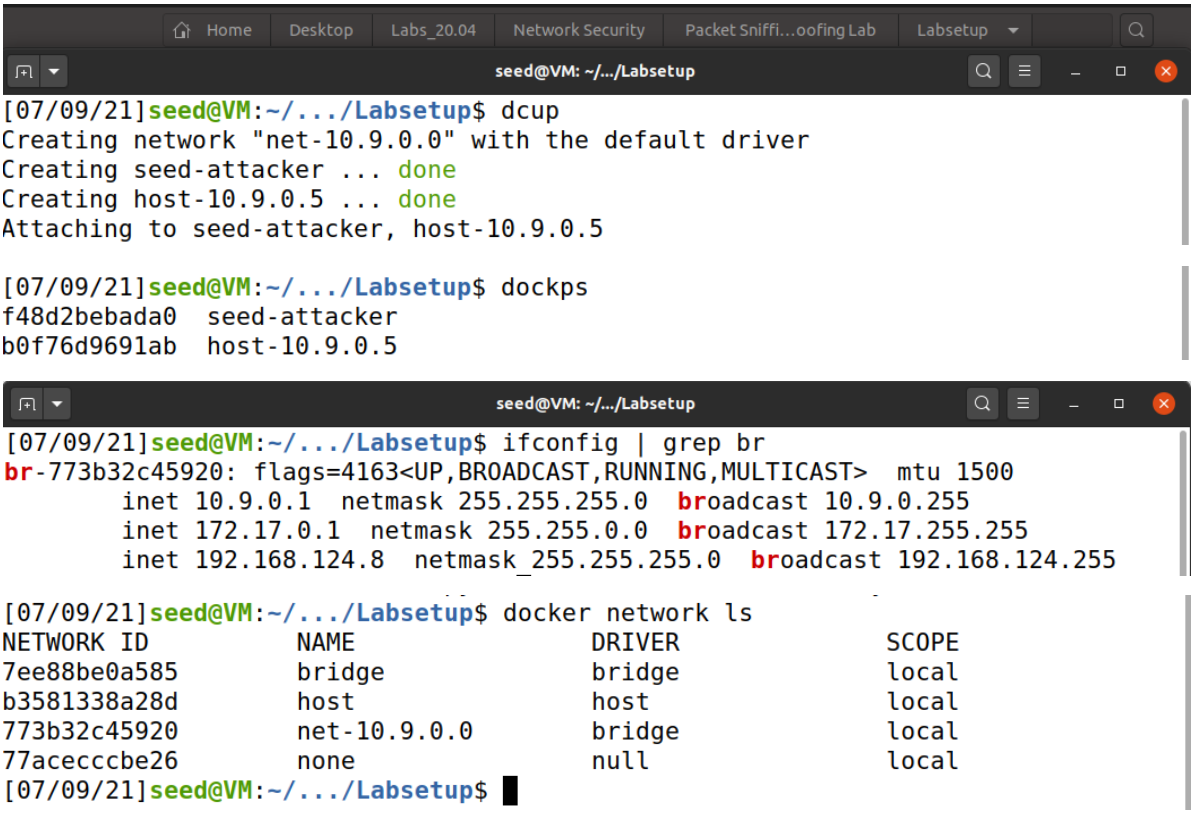
Lab1

57118202 付孜

Task 1.1: Sniffifing Packets

Task 1.1A

启动docker，查看网络ID以及容器ID。

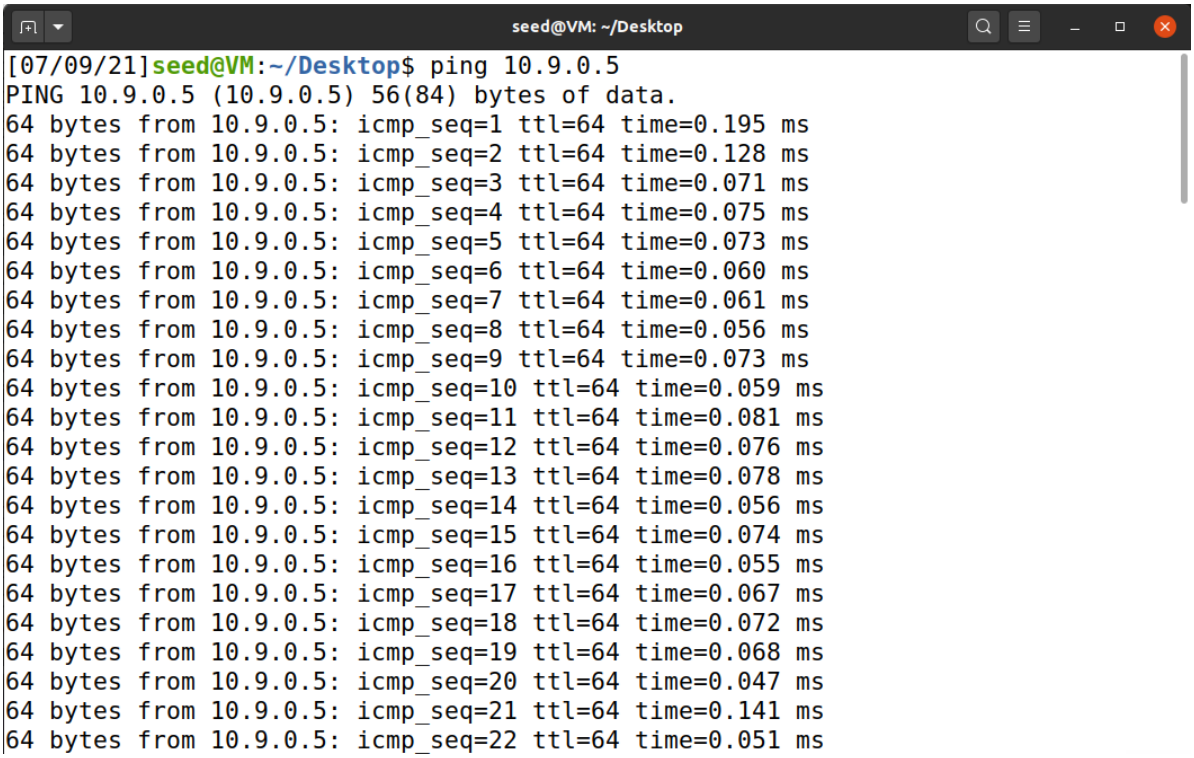


使用Scapy在Python程序中进行包嗅探的代码 `sniffer.py`：

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-773b32c45920', filter='icmp', prn=print_pkt)
```

使用root权限运行该程序，并在另一个terminal中对主机IP进行ping命令，运行结果如下图所示。

```
[07/09/21]seed@VM:~/Desktop$ touch sniffer.py
[07/09/21]seed@VM:~/Desktop$ chmod a+x sniffer.py
[07/09/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:f2:e6:b5:75
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 63946
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x2cc7
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
```



如果以seed用户而不用root权限运行`sniffer.py`，系统会报错，如下图所示。

```
[07/09/21]seed@VM:~/Desktop$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-773b32c45920', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in s
niff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in _r
un
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398, in
__init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)
)  # noqa: E501
  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
```

Task 1.1B

- **只抓取ICMP报文**，见Task 1.1A所示。
- **捕获任何来自特定IP的TCP数据包，目的端口为23。**

  将`sniffer.py`修改为：

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-773b32c45920', filter='tcp port 23 and host 10.9.0.5',
prn=print_pkt)
```

  利用docksh获取host的shell，telnet任意一个IP地址建立连接，并在运行`sniffer.py`的terminal中查看捕获的tcp数据包。

```
[07/09/21]seed@VM:~/.../Labsetup$ docksh b0
root@b0f76d9691ab:/# telnet 1.1.1.1
Trying 1.1.1.1...
telnet: Unable to connect to remote host: Connection timed out
```

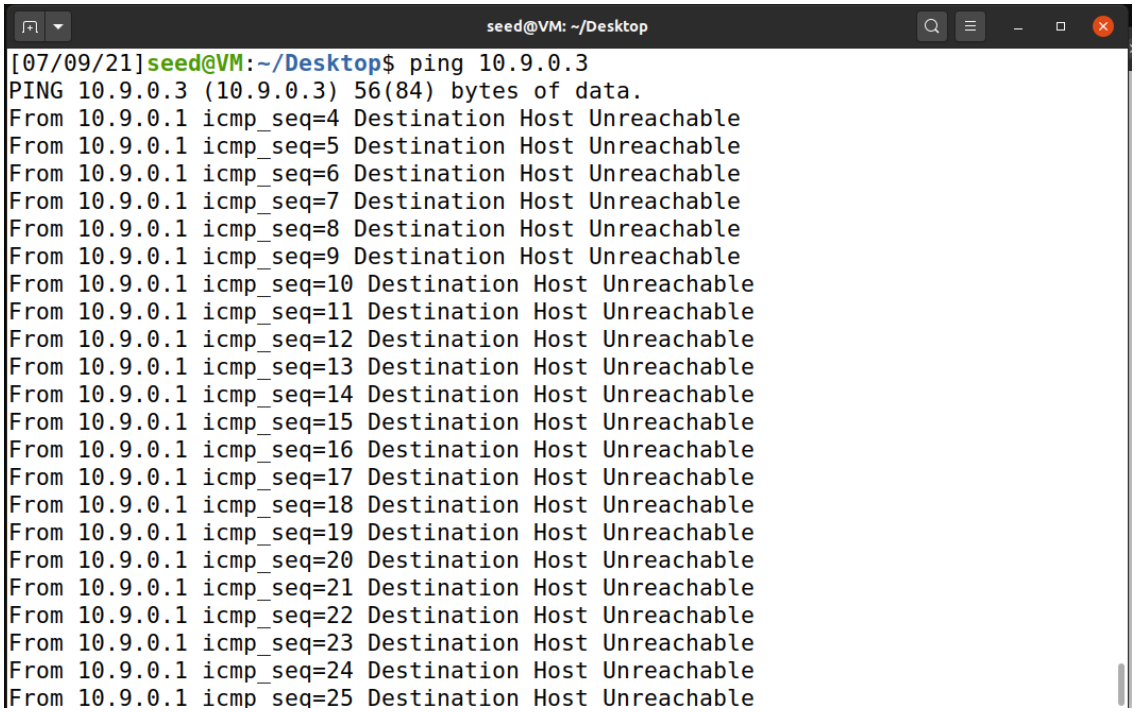
```
seed@VM: ~/Desktop
[07/09/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:f2:e6:b5:75
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 22747
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xd5c1
     src       = 10.9.0.5
     dst       = 1.1.1.1
     \options   \
###[ TCP ]###
        sport     = 49620
        dport     = telnet
        seq       = 324818138
        ack       = 0
```

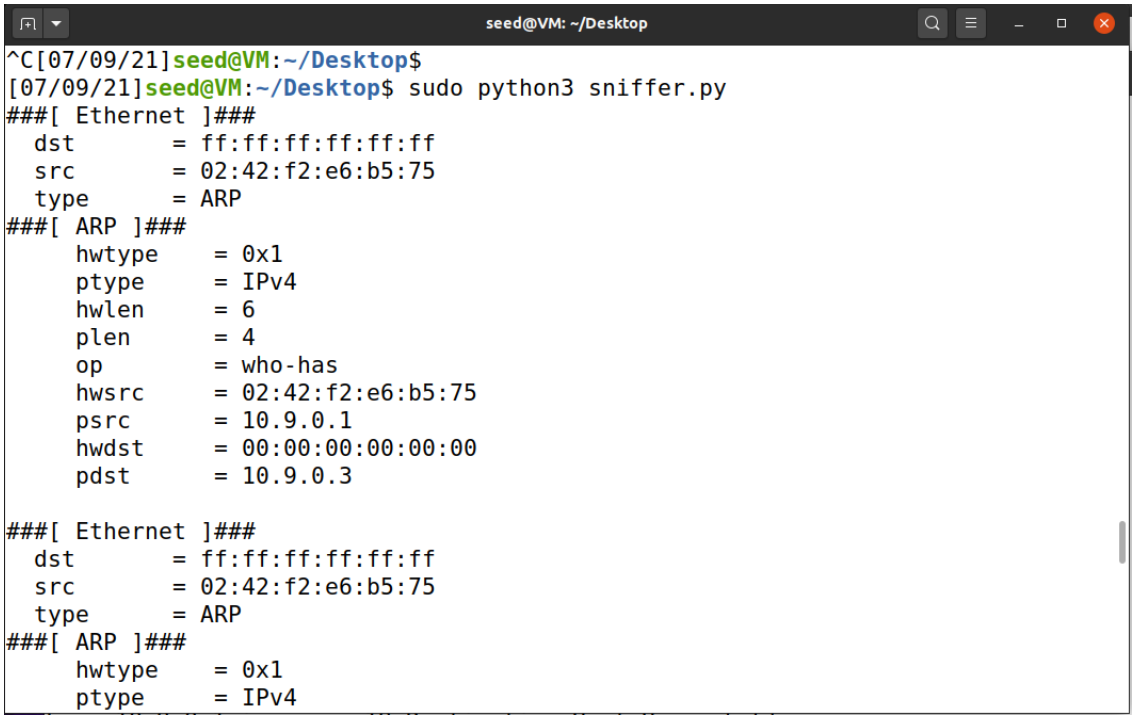
- **捕获来自或去特定子网的数据包。可以选择任何子网，如128.230.0.0/16；不应该选择VM所连接的子网。**

  将`sniffer.py`修改为：

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-773b32c45920', filter='host 10.9.0.3', prn=print_pkt)
```

ping 10.9.0.3，并在运行`sniffer.py`的terminal中查看捕获的tcp数据包。





Task 1.2: Spoofifing ICMP Packets

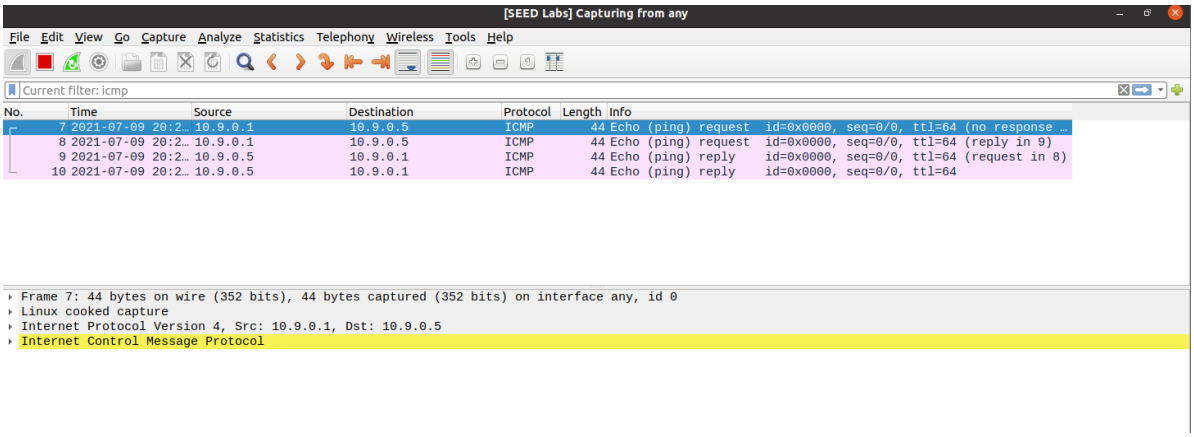
欺骗ICMP数据包的代码`spoofer.py`：

```
#!usr/bin/evn python3
from scapy.all import *
a = IP()                    //line 1
a.dst = '10.9.0.5'
b = ICMP()
p = a/b
send(p)
ls(a)
```

第一行创建一个IP对象，第二行设置目标IP地址字段。第三行创建了一个ICMP对象，默认类型为echo request。在第四行中，我们将a和b堆叠在一起形成了一个新对象，“/”操作符被重载，不再表示除法，它意味着将b添加为a的有效负载字段，并相应地修改a的字段。

最终我们得到一个表示ICMP数据包的新对象，报文重组后，向子网内的一个IP发送数据包，打开Wireshark可观测到发送数据包和响应数据包。

```
[07/09/21]seed@VM:~/Desktop$ sudo python3 spoofer.py
.
Sent 1 packets.
version    : BitField  (4 bits)                  = 4               (4)
ihl        : BitField  (4 bits)                  = None            (None)
tos        : XByteField                          = 0               (0)
len        : ShortField                          = None            (None)
id         : ShortField                          = 1               (1)
flags      : FlagsField  (3 bits)                = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField  (13 bits)                 = 0               (0)
ttl        : ByteField                           = 64              (64)
proto      : ByteEnumField                       = 0               (0)
chksum     : XShortField                         = None            (None)
src        : SourceIPField                       = '10.9.0.1'      (None)
dst        : DestIPField                         = '10.9.0.5'      (None)
options    : PacketListField                     = []              ([])
```



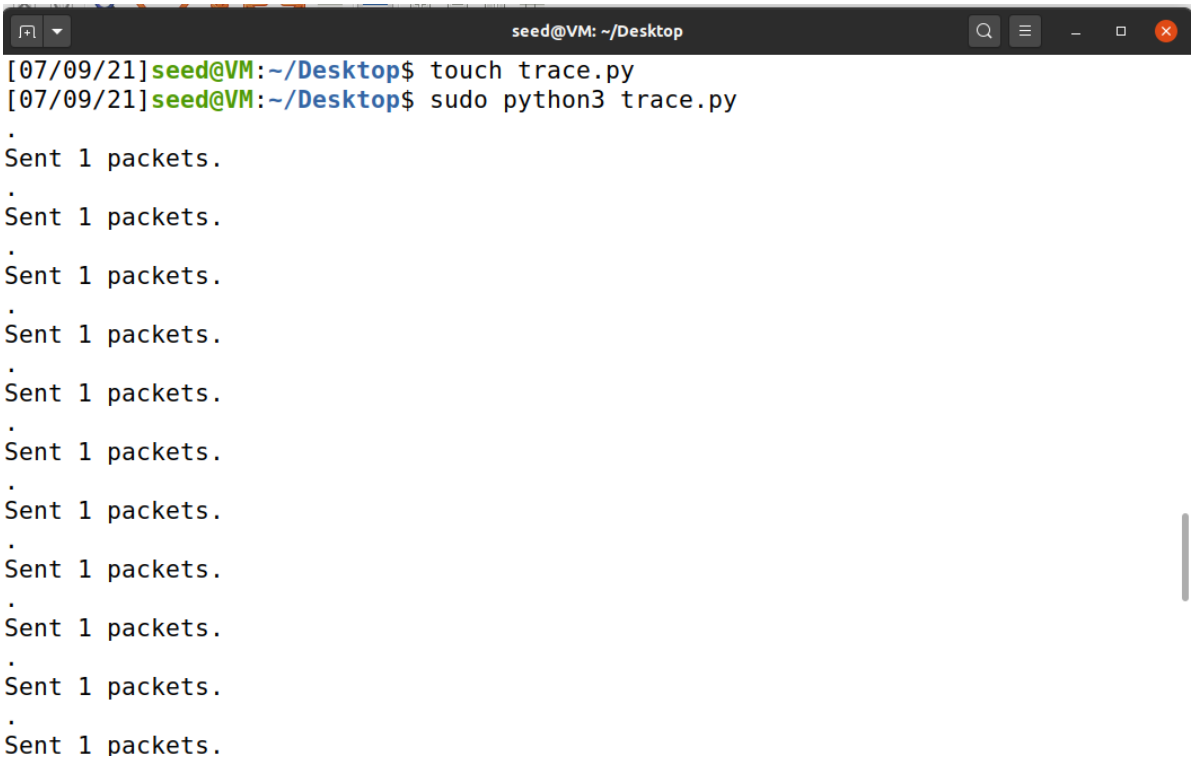
Task 1.3: Traceroute

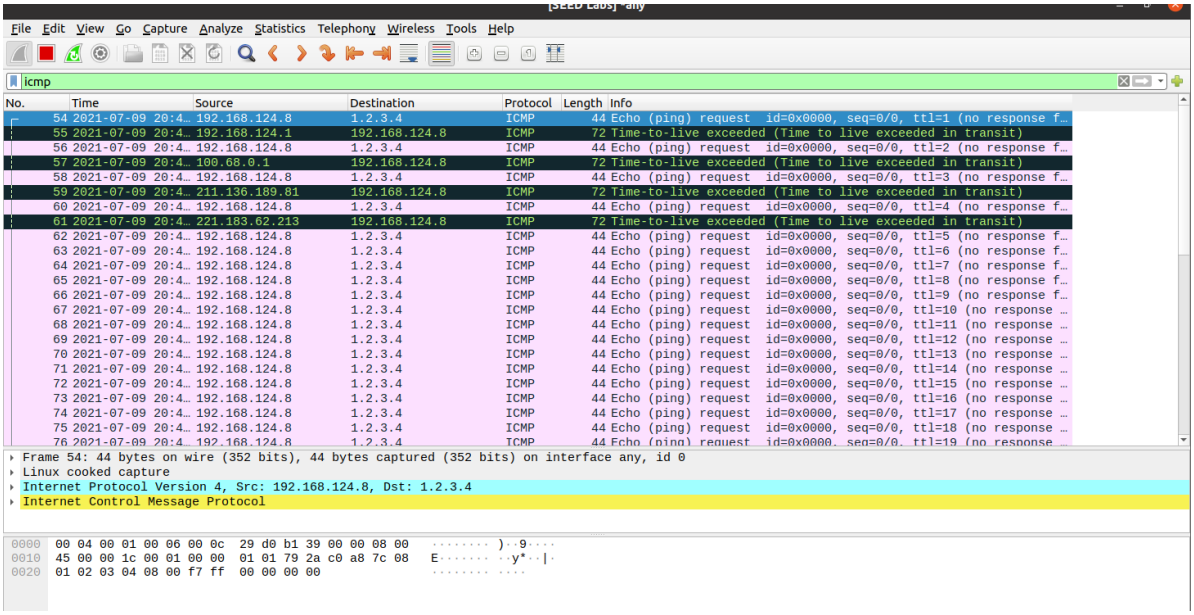
使用Scapy来估计虚拟机与选定目标之间的路由器数量之间的距离。编写工具`trace.py`：

```
#!usr/bin/evn python3
from scapy.all import *
a = IP()
b = ICMP()
a.dst = '1.2.3.4'
for i in range(30):
    a.ttl = i + 1
    p = a / b
    send(p)
```

这个程序向目标IP发送 ICMP 数据包，一开始设置 TTL 值为 1, 这个数据包将被第一个路由器丢弃，它将向我们发送一个ICMP错误消息，告诉我们生存时间已经超过了时间。这就是我们如何获得第一个路由器的IP地址。然后，不断增加 TTL 的值，直到使得数据包到达目的地。

用Wireshark观测数据包发送情况。





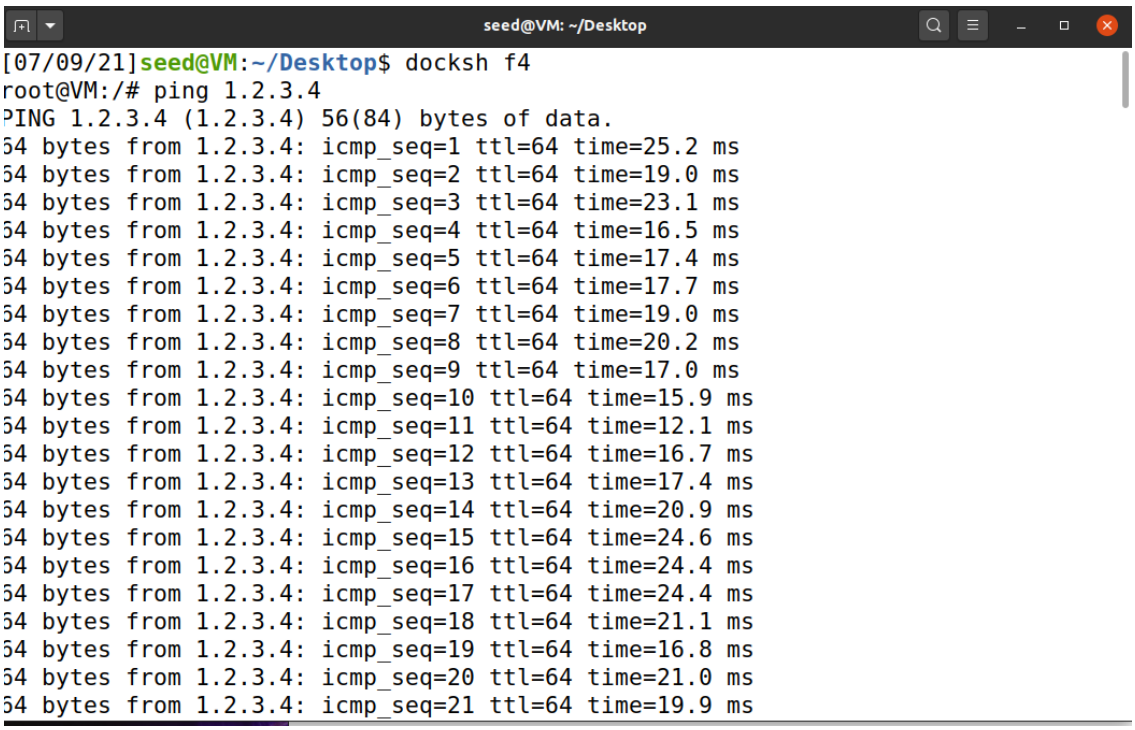
从Wireshark中我们可以观察到，途径的IP有192.168.124.1，100.68.0.1，211.136.189.81，221.183.62.213，最后到达1.2.3.4，即目的地。

Task 1.4: Sniffing and-then Spoofing

结合嗅探和欺骗技术，编写`sniff-and-then-spoof.py`：

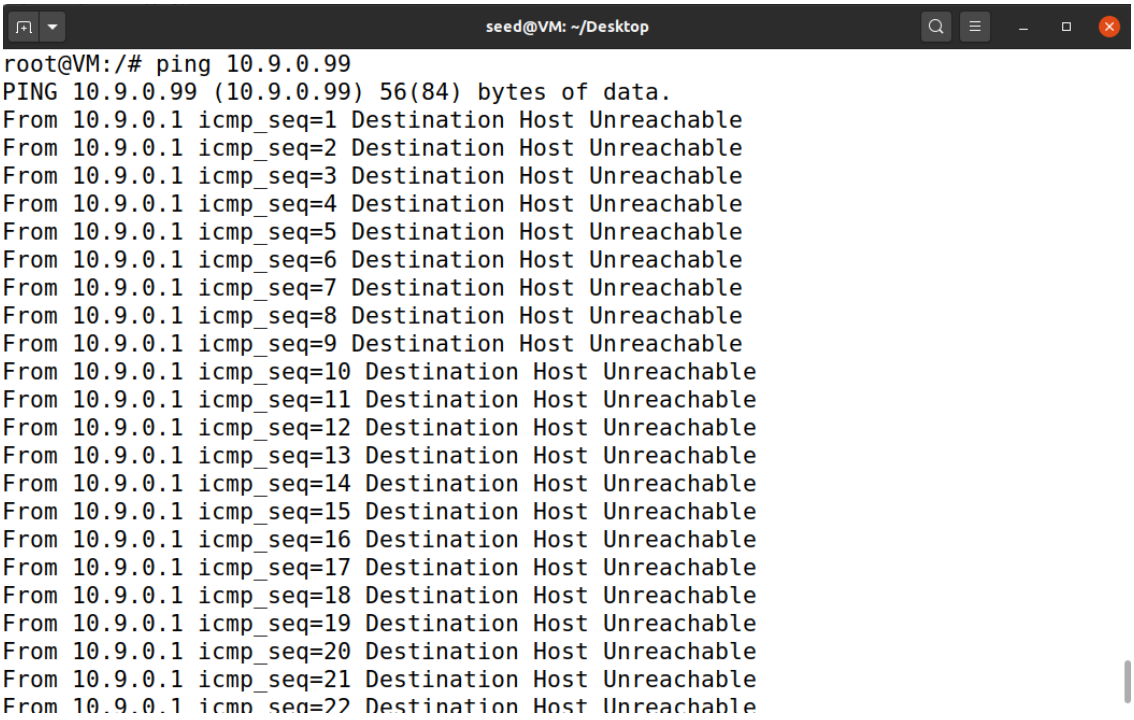
```python
#!/usr/bin/evn python3
from scapy.all import *

def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data
        send(newpkt)

pkt = sniff(filter='icmp', prn=spoof_pkt)
```

- **ping 1.2.3.4 # a non-existing host on the Internet**



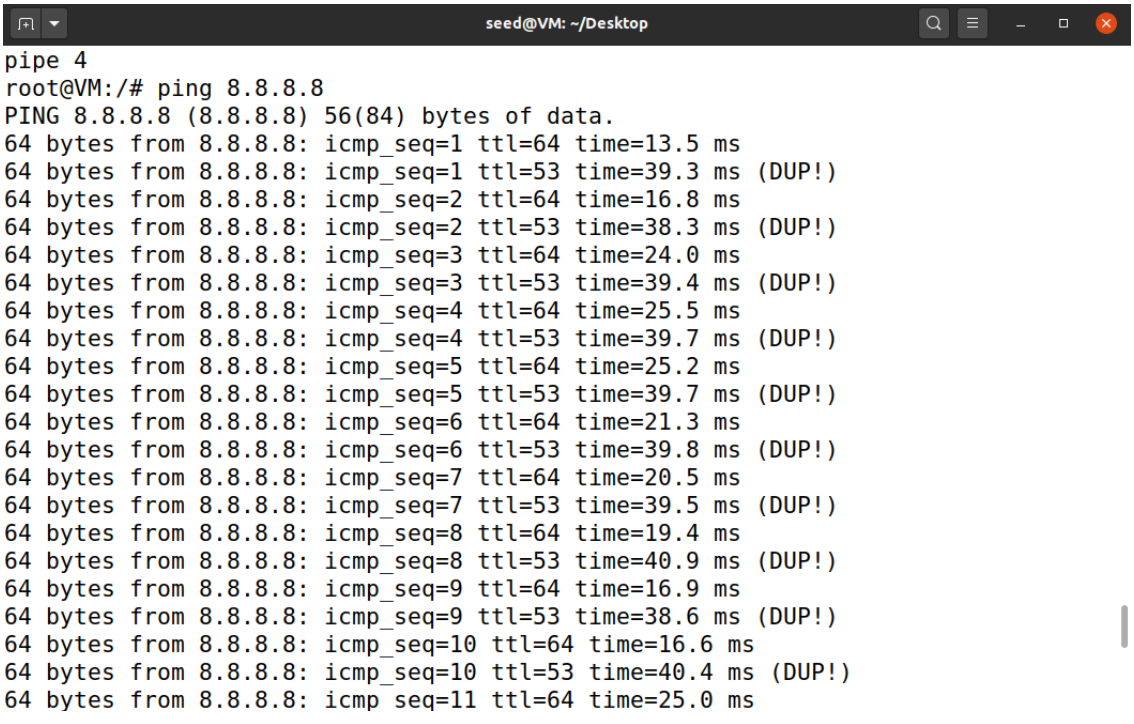
可以看到当我们ping一个网络上不存在的IP时，由于伪造报文，我们仍可以接收到响应。

- **ping 10.9.0.99 # a non-existing host on the LAN**



对于局域网内不存在的主机，先利用ARP进行MAC地址询问，由于一直得不到结果，所以没有ICMP报文，也就不存在报文欺骗。

- **ping 8.8.8.8 # an existing host on the Internet**



对于网络上存在的主机，我们可以看到每个序列号的报文都存在一个重复报文，我们可以知道TTL=64，且时间较短的那个报文是伪造的报文。