Lab1

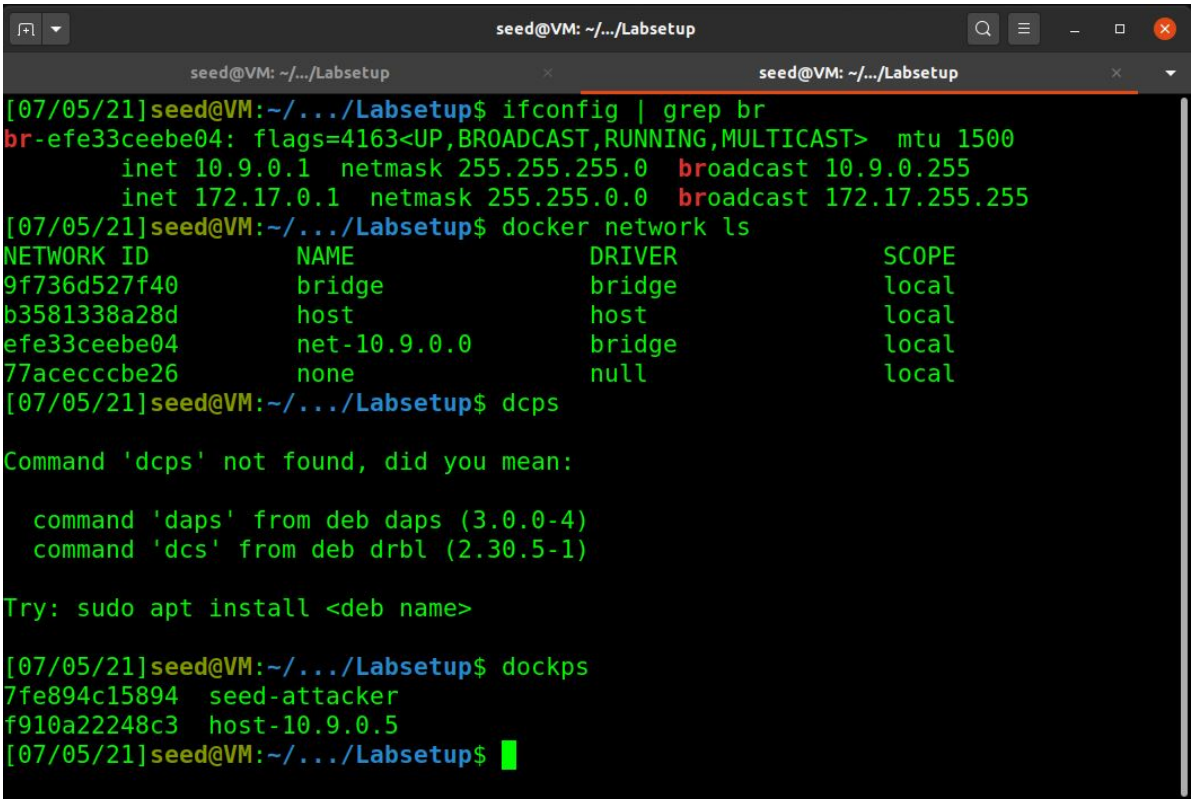
57118233 周厚霖

Task 1.1：Sniffing Packets

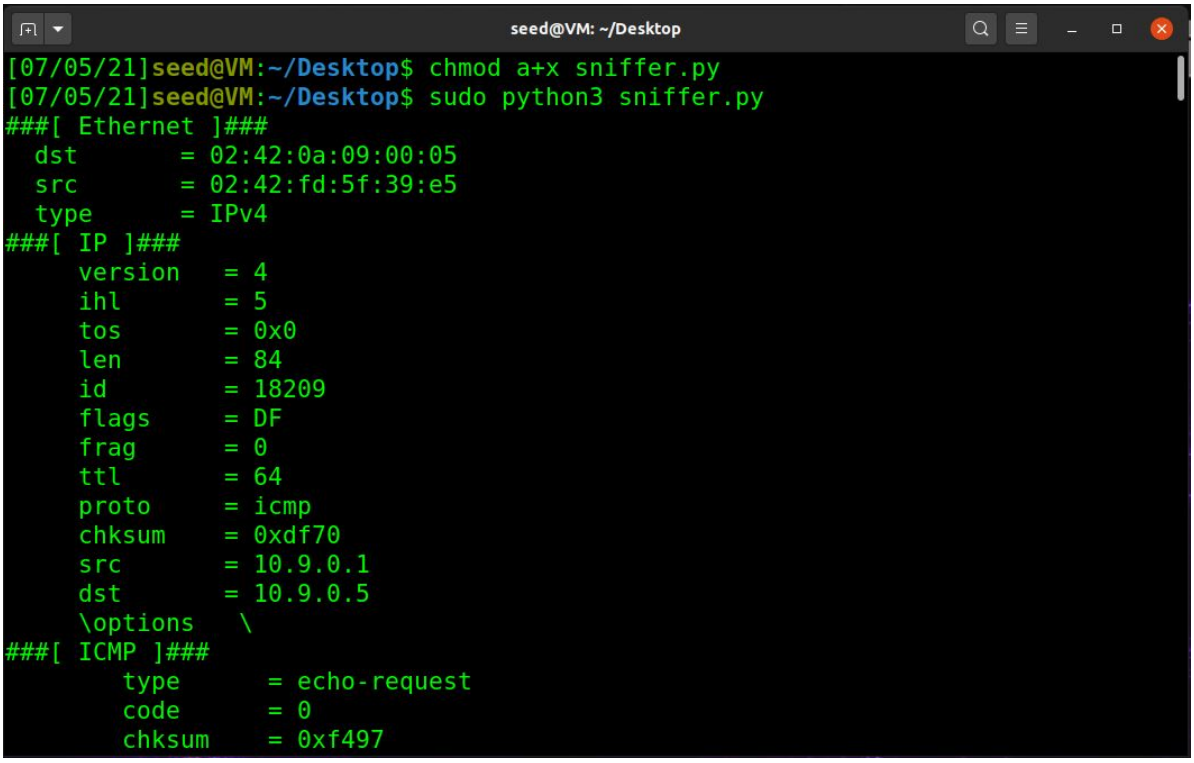
Task 1.1A

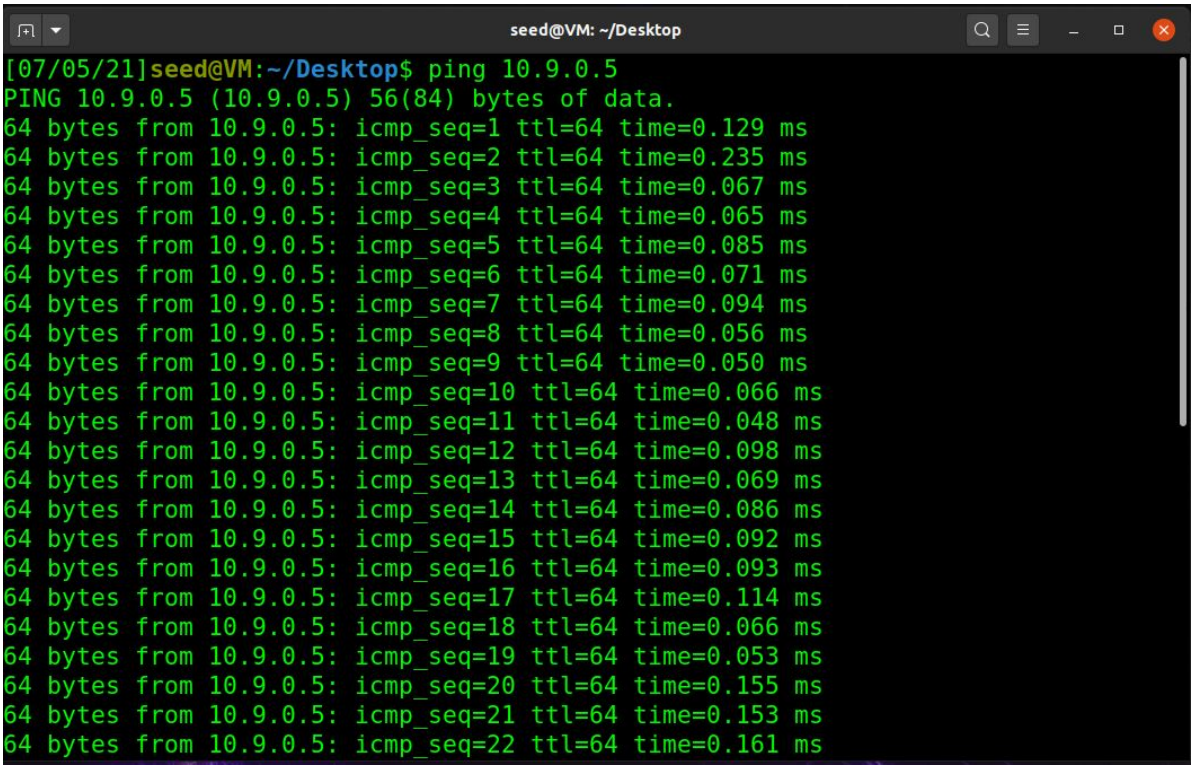
```
#!/usr/bin/evn python3
from scapy.all import *

def print_pkt(pkt):
    pkt.show()

pkt = sinff(iface='br-efe33ceebe04', filter='icmp', prn=print_pkt)
```

启动docker后，查看网络ID。

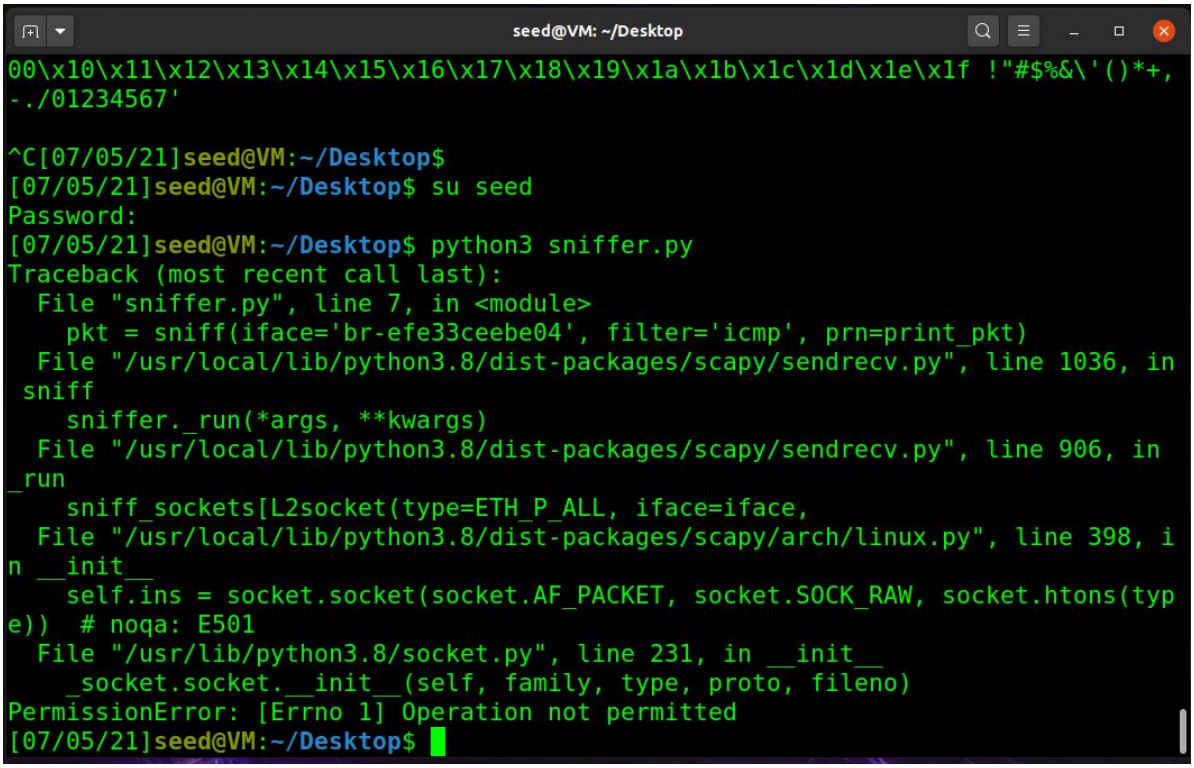


以root权限运行sniffer.py，新开一个命令行对主机IP进行ping命令。





以seed用户运行sniffer.py时，系统会报错。



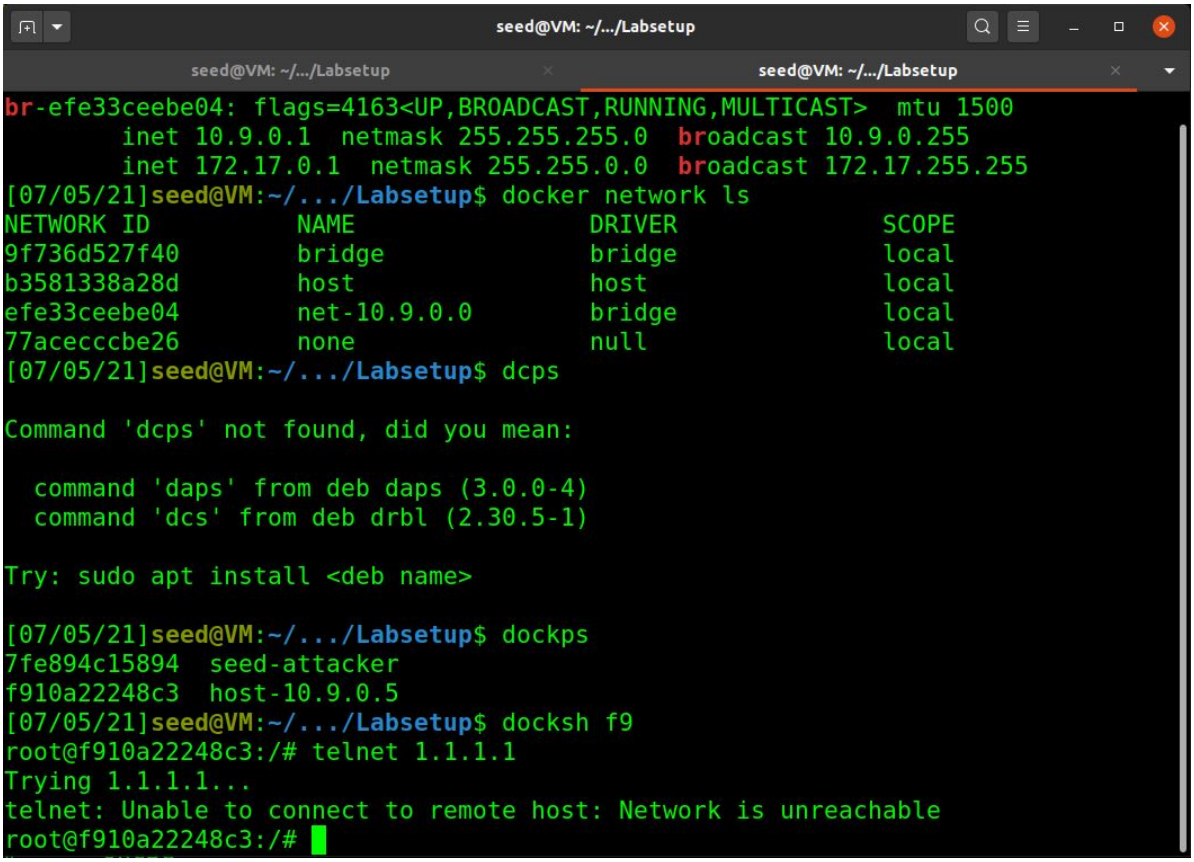
Task 1.1B

只抓取ICMP报文，见Task 1.1A所示。

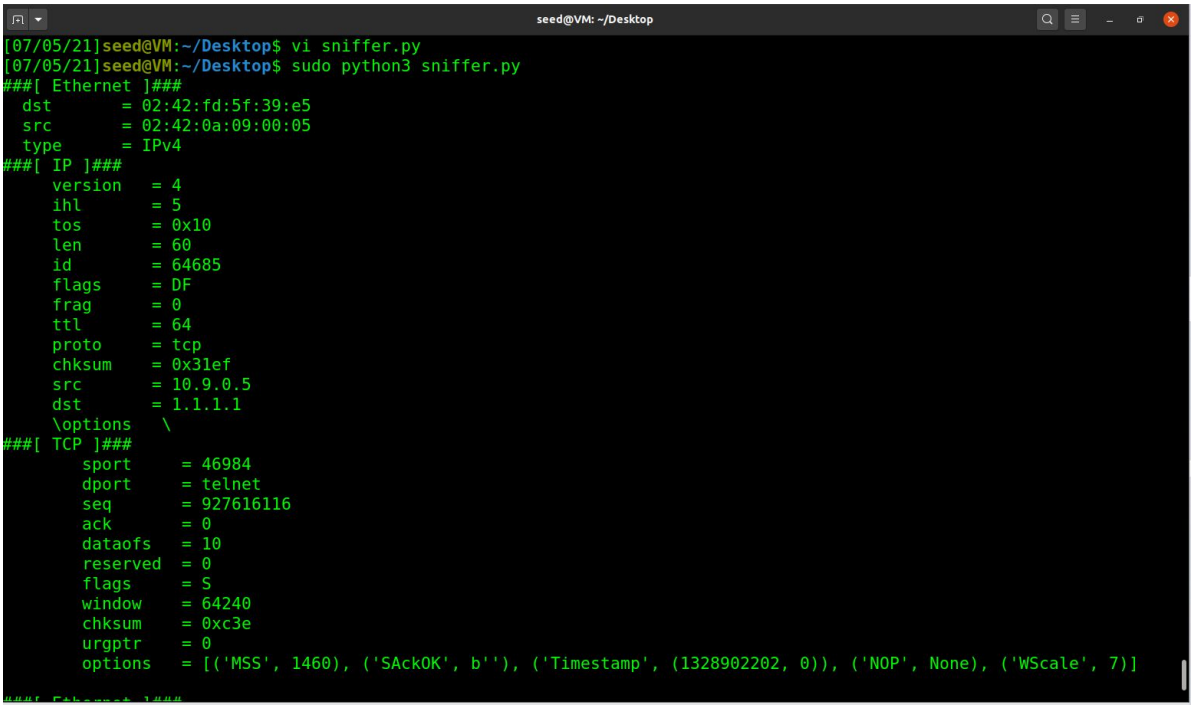
捕获任何来自特定IP的TCP数据包，目的端口为23。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-efe33ceebe04', 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。



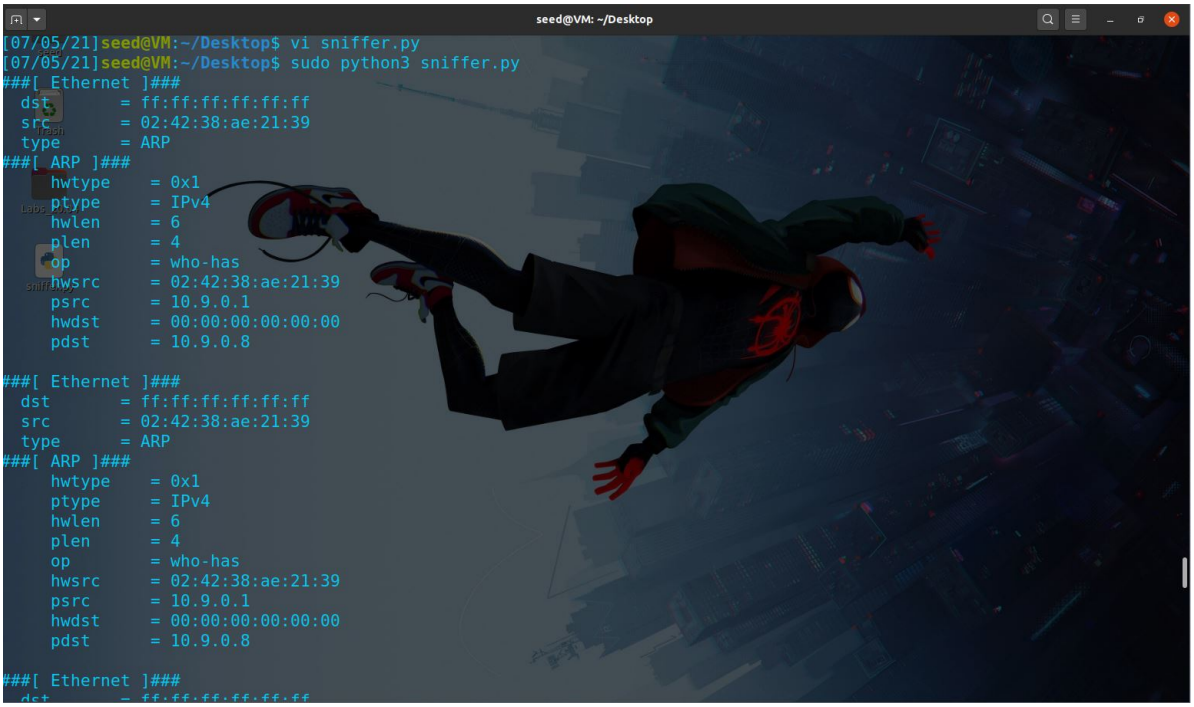
在另一处可看到tcp数据包。



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所绑定的子网。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-13a5b79724e2', filter='host 10.9.0.8', prn=print_pkt)
```

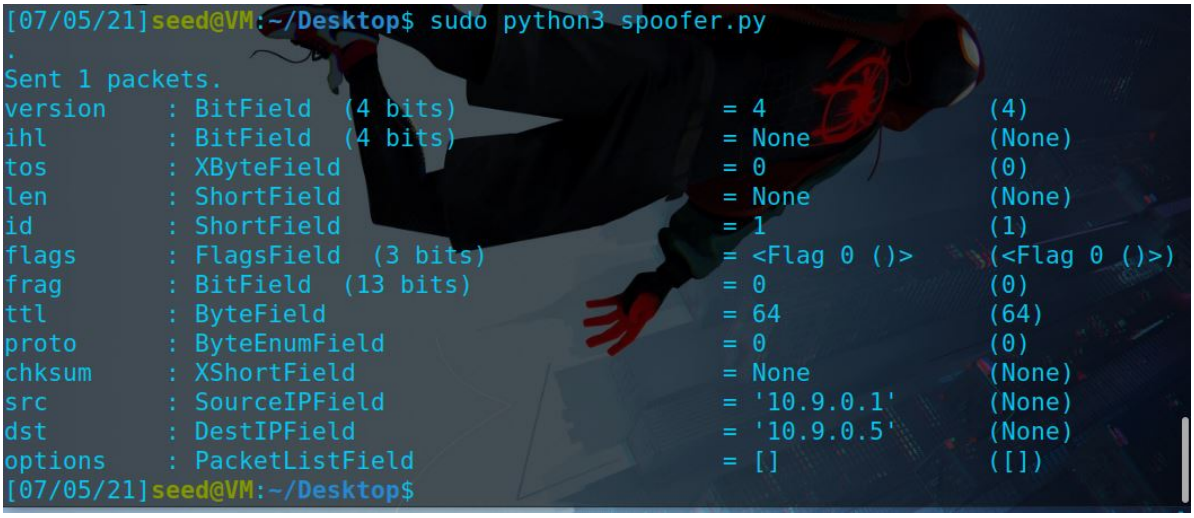
直接ping 10.9.0.8，可得到捕获的数据包。

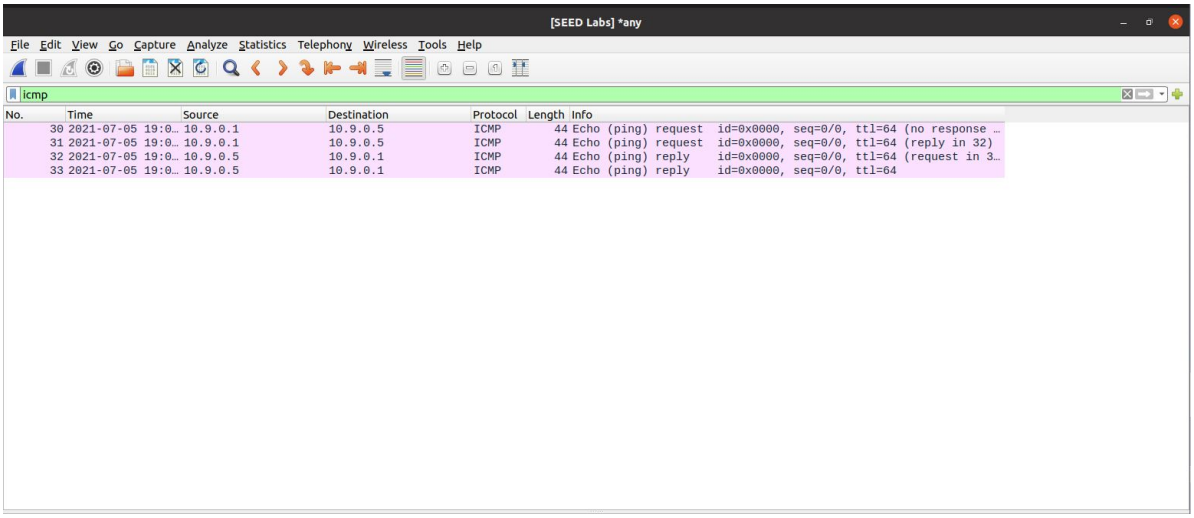


Task 1.2：Spoofing ICMP Packets

```
#!usr/bin/evn python3
from scapy.all import *
a = IP()
a.dst = '10.9.0.3'
b = ICMP()
p = a/b
send(p)
ls(a)
```

第一行创建了一个ICMP对象，默认类型为echo request。在第六行中，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，不在表示除法，而是将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可观测发送数据包和响应数据包。



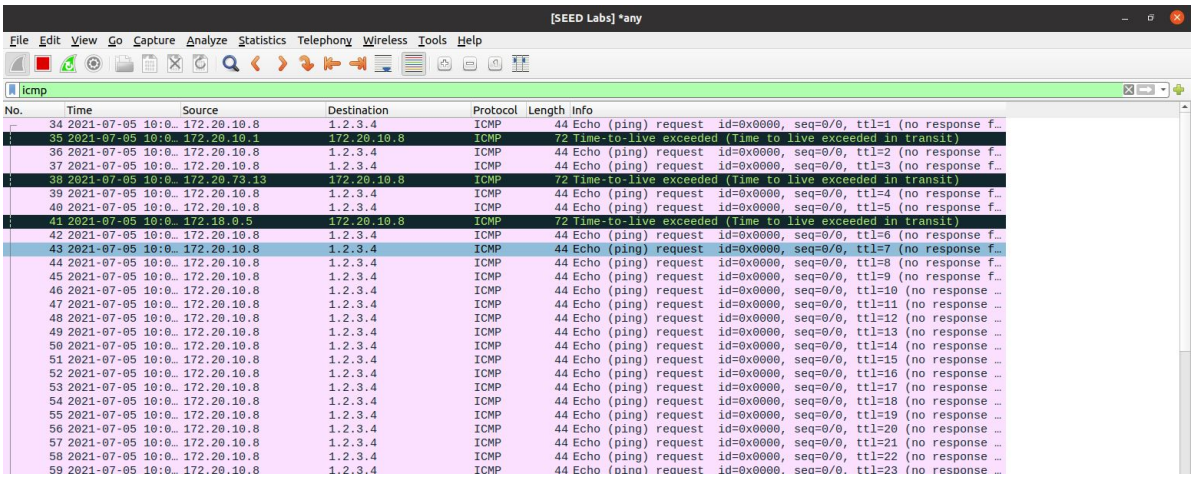


Task 1.3：Traceroute

```
#!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)
```

创建一个文件trace.py，向目标IP发送 ICMP 数据包，一开始设置 TTL (Time-To-Live)值 为 1, 那么发出的 ICMP 数据包在经历一个路由结点后, 就会失活被抛弃,我们利用循环, 不断增加TTL的值，最终使得数据包到达目的地。

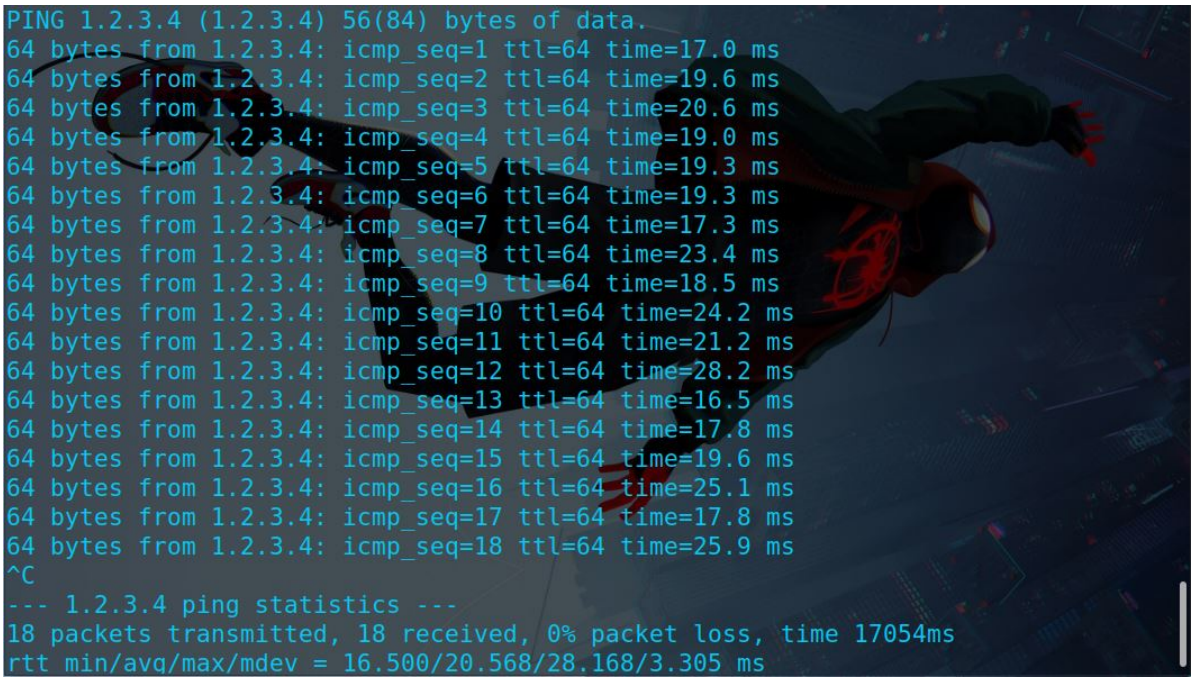
从Wireshark中我们可以观察到，途径的IP有172.20.10.1，172.20.73.13，172.18.0.5，最后到达1.2.3.4，即目的地。



Task 1.4：Sniffing and-then Spoofing

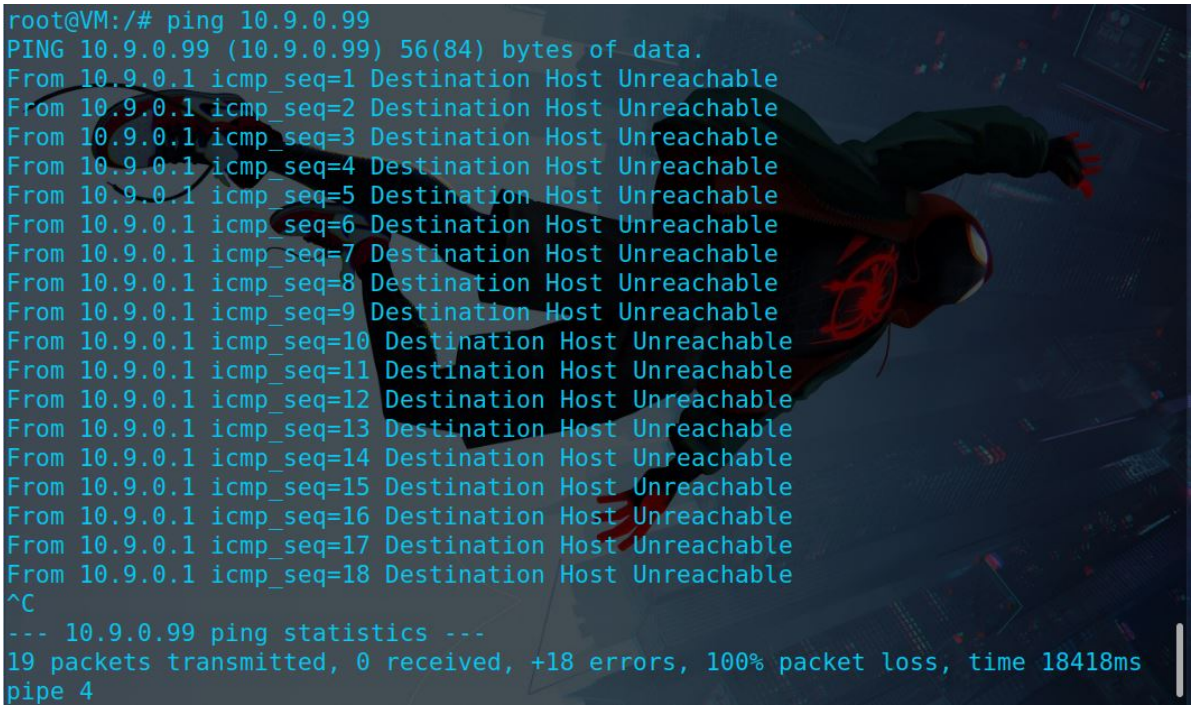
```
#!/usr/bin/evn python3
from scapy.all import *

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。