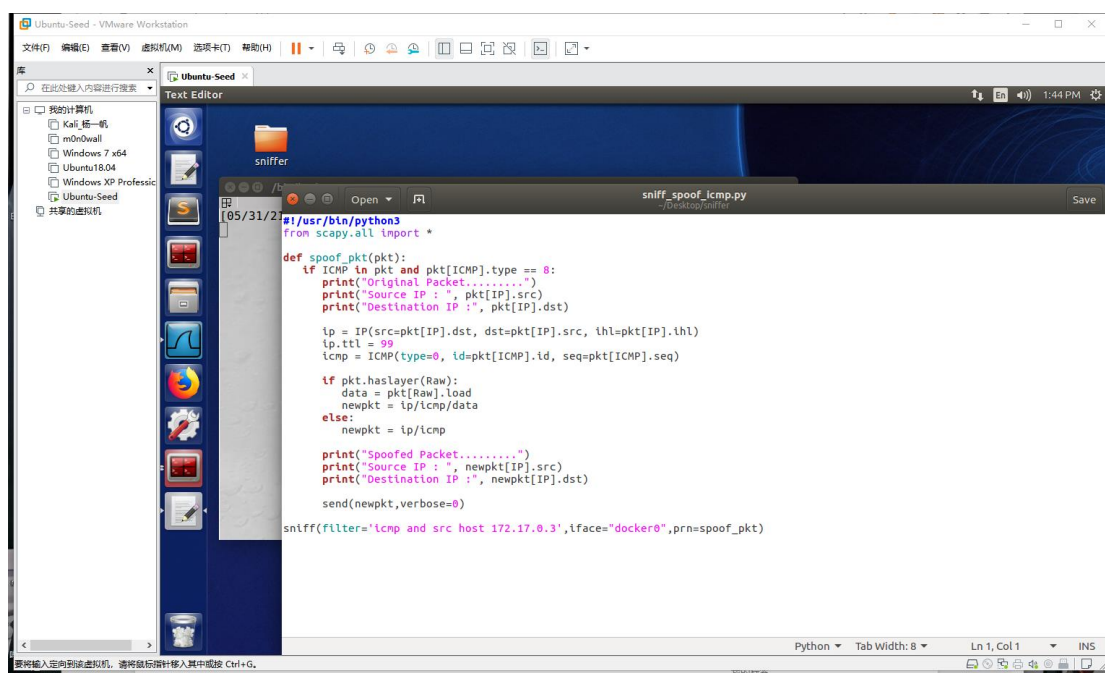


五、

借用老师发的使用 scapy 编写的 icmp 嗅探、欺骗脚本完成内容，首先查看脚本 sniff_spoof_icmp.py，其中通过 sniff 来嗅探网卡 docker0 上的网络流量，使用 BPF 过滤规则来过滤来源为 172.17.0.2 的 icmp 数据包，然后将数据包传入回调函数 spoof_pkt 中。在回调函数的开始，首先判断数据包是不是 ICMP，且 ICMP 包的首字节 type 是不是 0x08，即是不是 ICMP request 包，如果是请求包则继续处理，然后后续构造 ICMP reply 包。首先新建一个 IP 头部、记为 ip（这会同时构造好以太网头部），src ip 和 dst ip 与 pkt 的 IP 头的两个 ip 地址相反，ttl 和 pkt 的 IP 头的 ttl 相同，然后构造 ICMP 层，首先新建了一个 ICMP 头部、记为 icmp，将 type 设置成 0x00（即 reply 包），然后开始填写 data 字段。首先判断原始数据包是否含有 Raw 数据包，如果有则将其构造成 data 字段，并构造数据包 newpkt = ip/icmp/data，如果没有 Raw 数据包，则构造数据包 newpkt = ip/icmp，最后将其发送，则可以针对每个 icmp request 包来模拟出 icmp reply 包，进而达到 icmp 欺骗。脚本内容如图 5.1 所示：



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

def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        print("Original Packet.....")
        print("Source IP : ", pkt[IP].src)
        print("Destination IP : ", pkt[IP].dst)

        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ttl=pkt[IP].ttl)
        icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)

        if pkt.haslayer(Raw):
            data = pkt[Raw].load
            newpkt = ip/icmp/data
        else:
            newpkt = ip/icmp

        print("Spoofed Packet.....")
        print("Source IP : ", newpkt[IP].src)
        print("Destination IP : ", newpkt[IP].dst)

        send(newpkt, verbose=0)

sniff(filter='icmp and src host 172.17.0.3', iface="docker0", prn=spoof_pkt)
```

图 5.1 sniff_spoof_icmp.py 脚本内容

下面进行实验：

首先查看 client 机的 ip 地址，如图 5.2 所示，client 机的 ip 为 172.17.0.3：

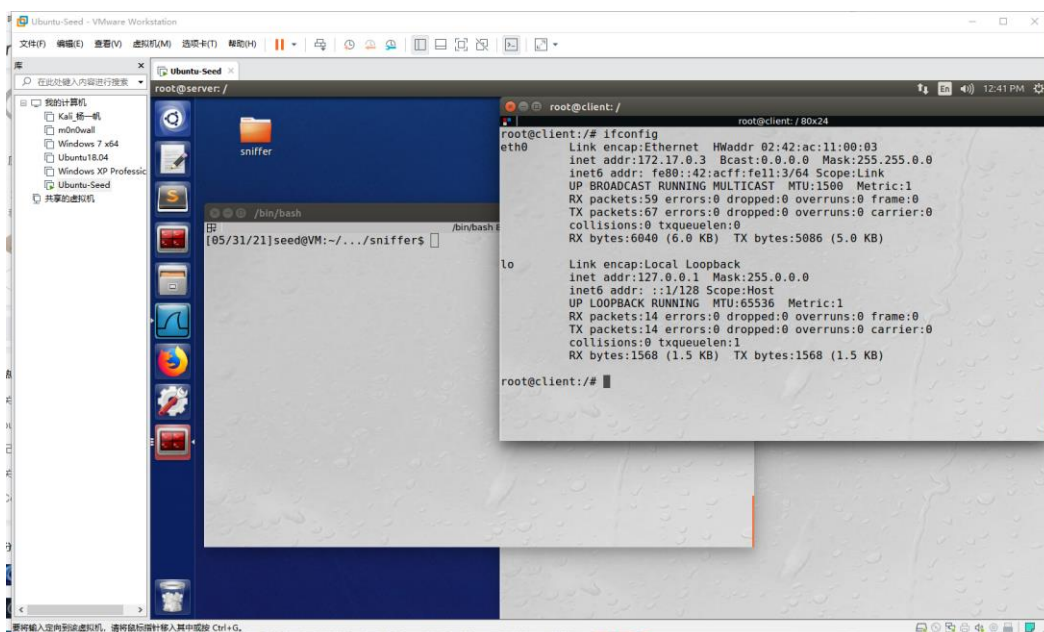


图 5.2 client 机的 ip 地址

然后在 VM 机上运行脚本 `sniff_spoof_icmp.py` (因为 VM 机在众多 docker 机中充当网关的位置, 所以可以监听到所有的数据包, 也在 docker 局域网内发送数据包), 并 ping 一个不存在的 ip 地址 (这里随便敲了一个 123.124.125.126), 发现 client 机成功接受到了 icmp reply 包, 如图 5.3 所示:

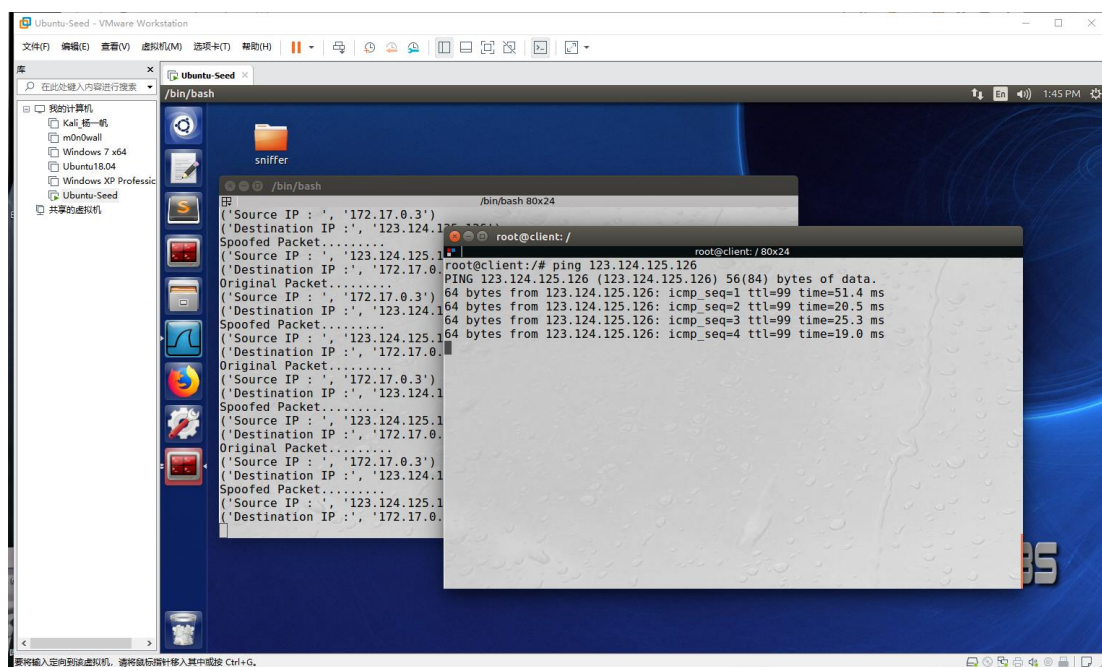


图 5.3 client 机成功接受到了 icmp reply 包

接下来我们停止运行 `sniff_spoof_icmp.py` 脚本, 再次 ping 123.124.125.126, 发现 client 无法接收到 icmp reply 包了, 如图 5.4 所示:

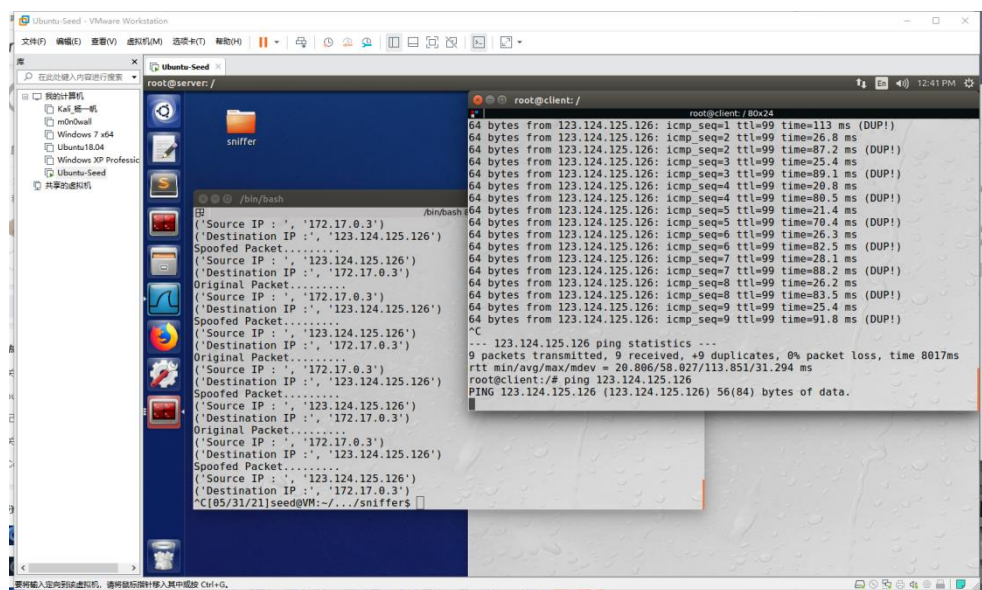


图 5.4 没有运行脚本时，client 无法接收到 icmp reply 包

然后我们来通过 wireshark 来分析数据包，在一开始开启脚本再 ping 的时候，wireshark 查看到 docker0 网卡上有大量的 icmp request 和 reply 包，且 reply 包有重复，如图 5.5 所示：

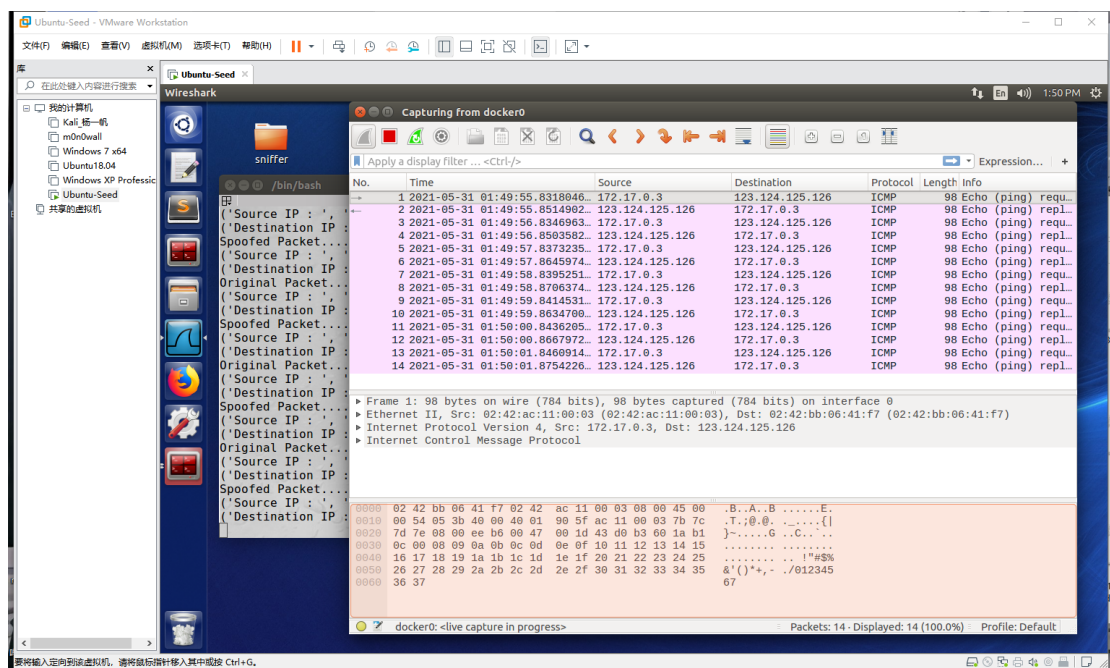


图 5.5 开启脚本时 wireshark 监听情况

然后关闭脚本，再 ping 原来的地址，wireshark 上监听到只有 icmp request 包，并无 reply 包，如图 5.6 所示：

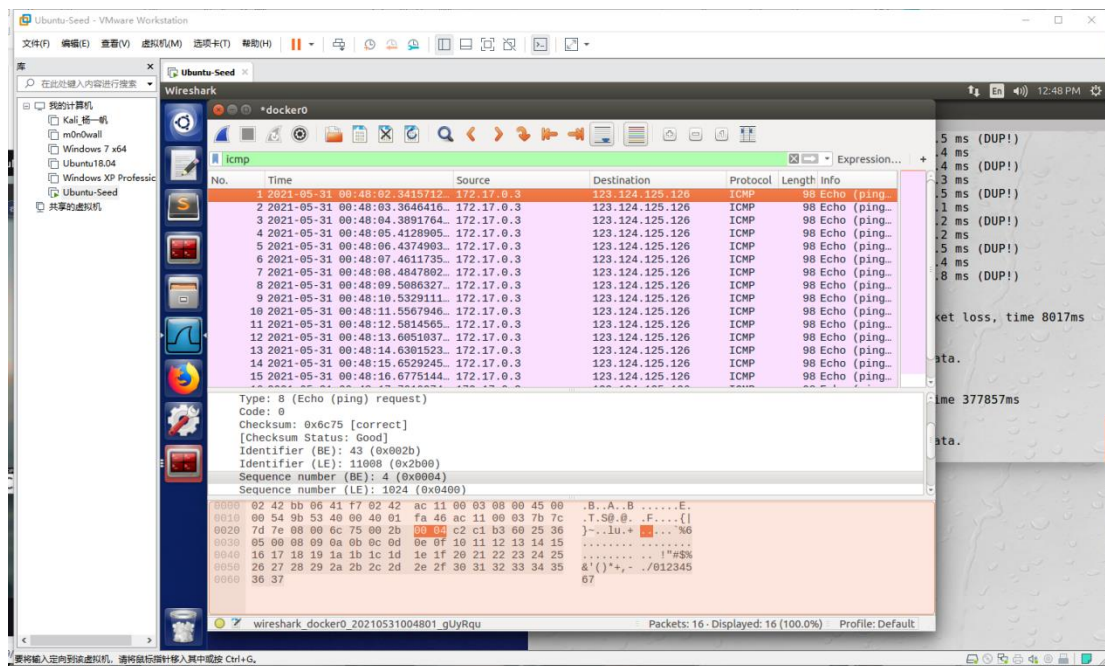


图 5.6 不开启脚本时 wireshark 监听情况

可以说明 icmp reply 包就是脚本 sniff_spoof_icmp.py 发送出来的。

(1) 在没有 root 的情况下运行该程序,程序在 sniff 处开始报错,如图 5.6 所示:

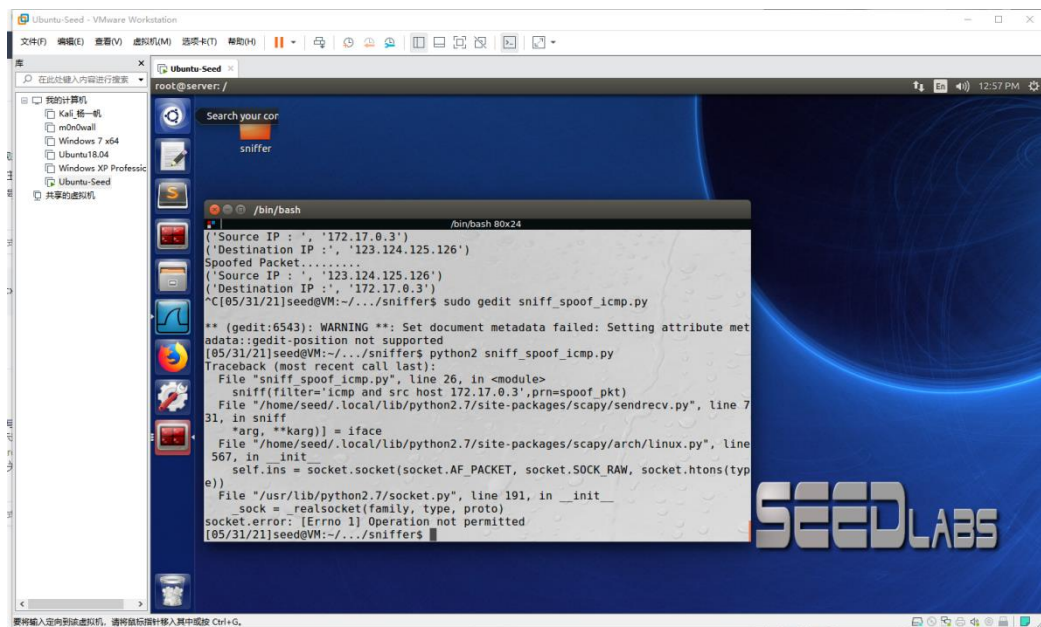


图 5.6 在没有 root 的情况下运行程序

这是因为非 root 用户无法使用开启混杂模式来接受所有经过它的网络流量。

(2) 在关闭混杂模式后，网卡将不再接收目的地址不是它的数据流，这也就导致 VM 的 docker0 网卡无法监听到 client 机发送给未知地址的 icmp request 包，也就无法进一步构造 icmp reply 数据包，从而导致 icmp 欺骗失效。