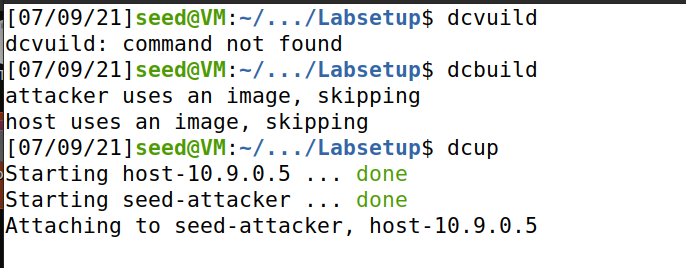
**Packet Sniffing and Spoofing Lab**

57118112 王怡乐

**Task1.1A**

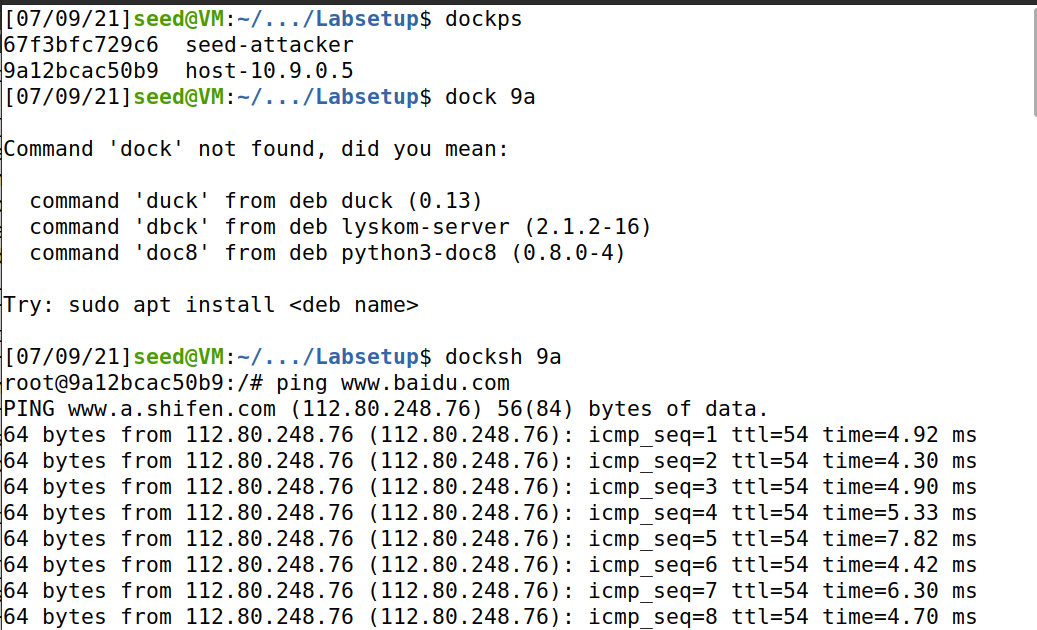
打开docker



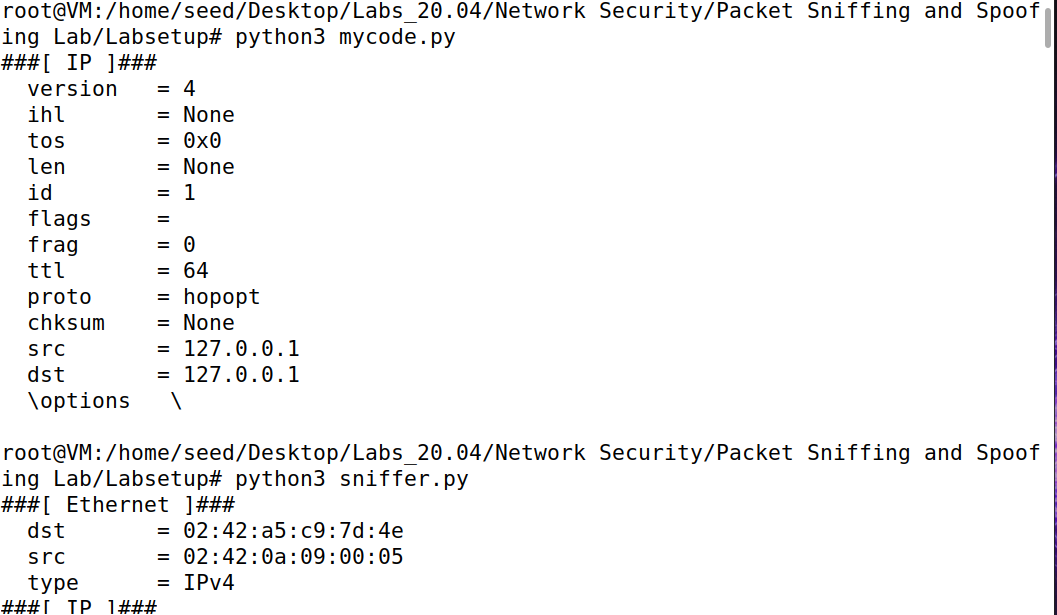
监听程序sniffer.py代码如下，此时过滤得到的是icmp报文



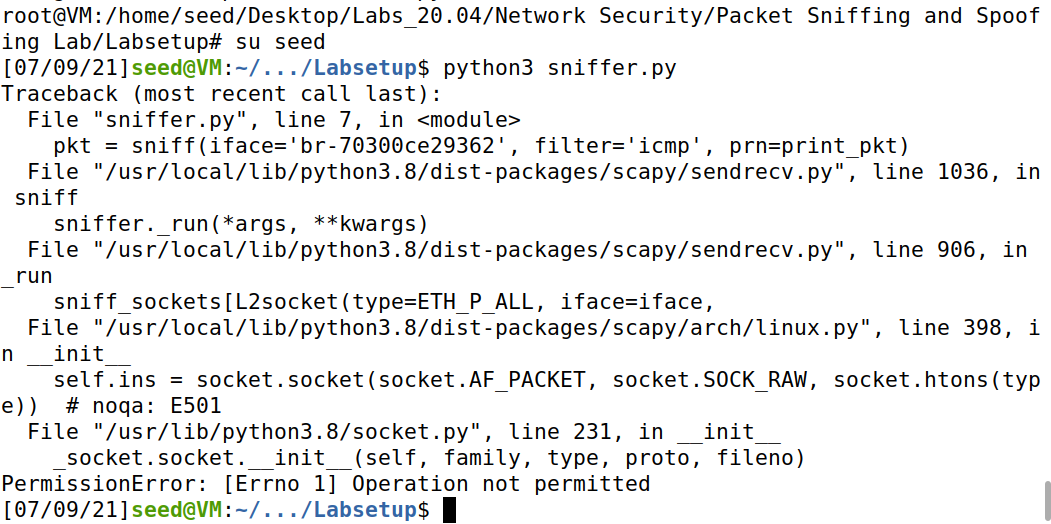
在另一个shell下ping某个IP



在root权限下运行监听程序，可正常抓包



普通用户下运行监听程序显示权限不够



**Task 1.1B**

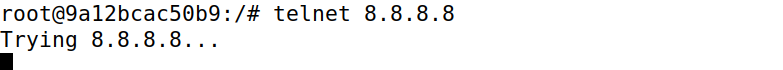


同Task1.1A

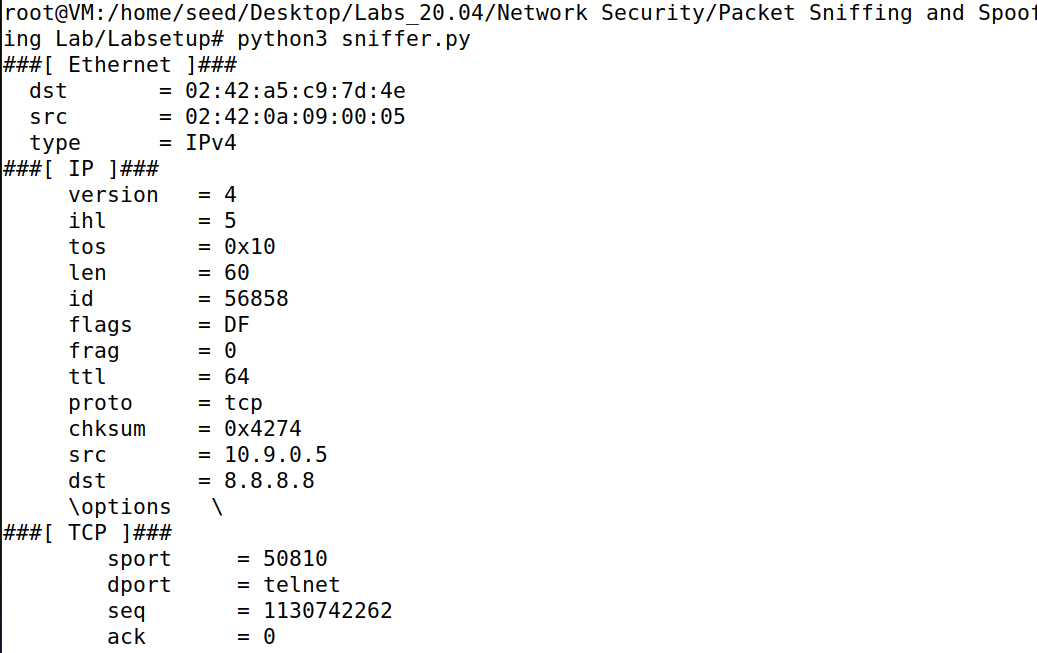


更改监听程序的filter如下所示



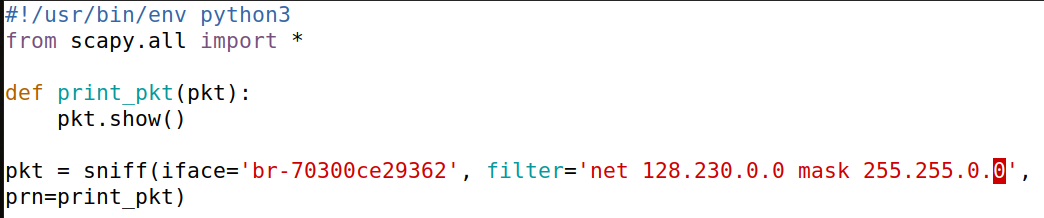
在利用docksh获取的shell中，telnet任意一个IP地址建立连接。

sniffer.py 捕获到的结果如下，其中 dport 端口为 telnet，默认为 23。

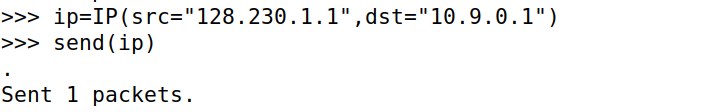




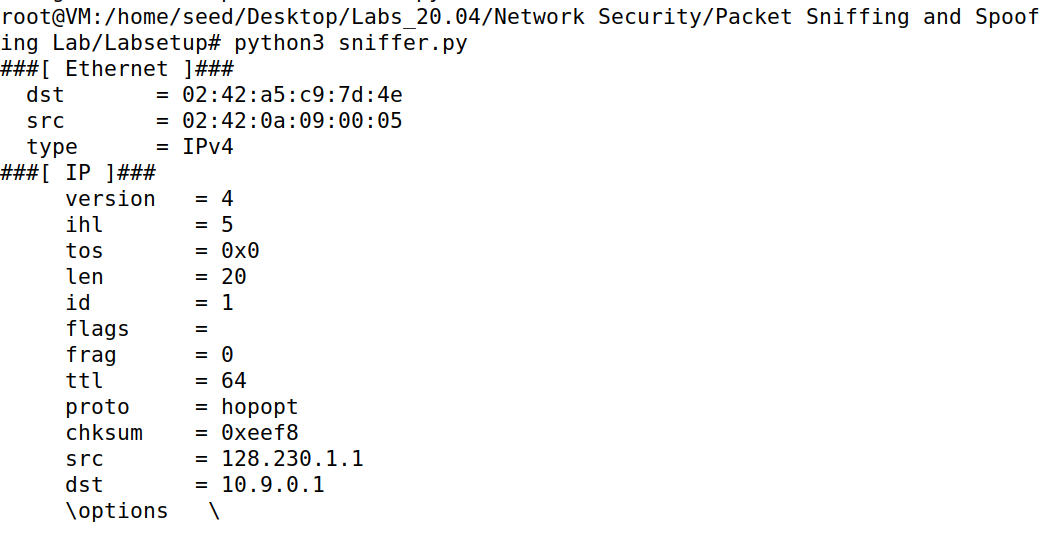
修改监听程序如下



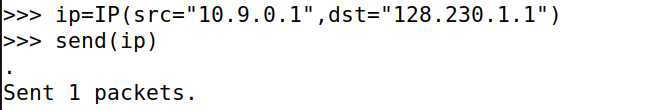
构造并发送一个源地址属于指定子网的数据包



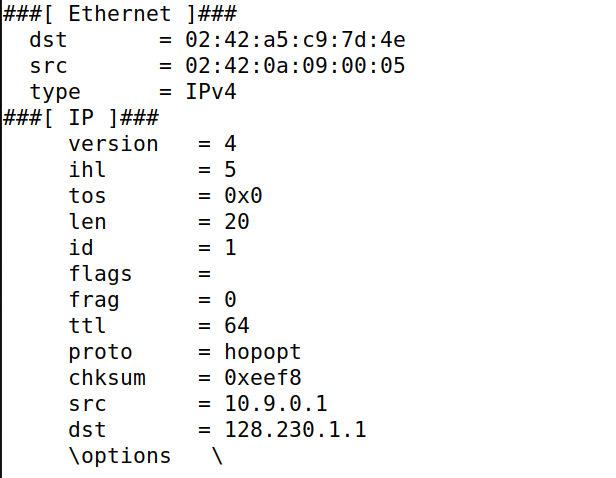
监听程序捕获成功



构造并发送一个宿地址属于指定子网的数据包

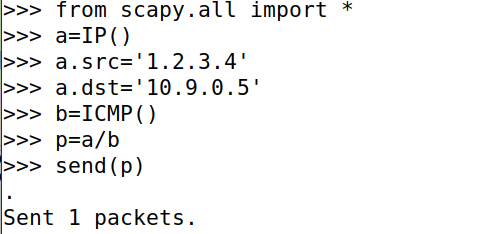


监听程序捕获成功

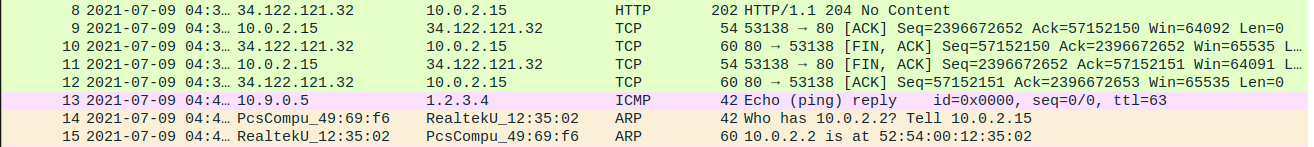


**Task1.2**

设置伪装的ip地址为 1.2.3.4，dst 为目标地址，发送报文

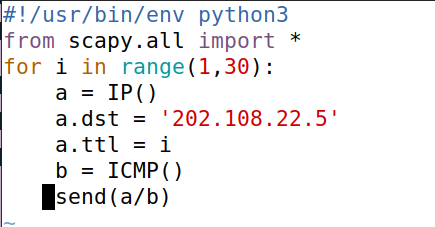


Wireshark成功捕获回复报文

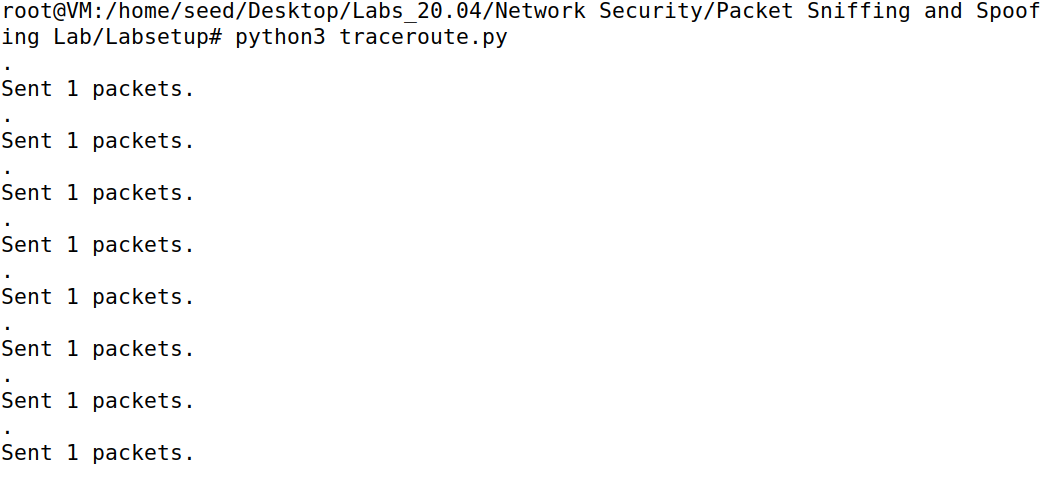


**Task1.3**

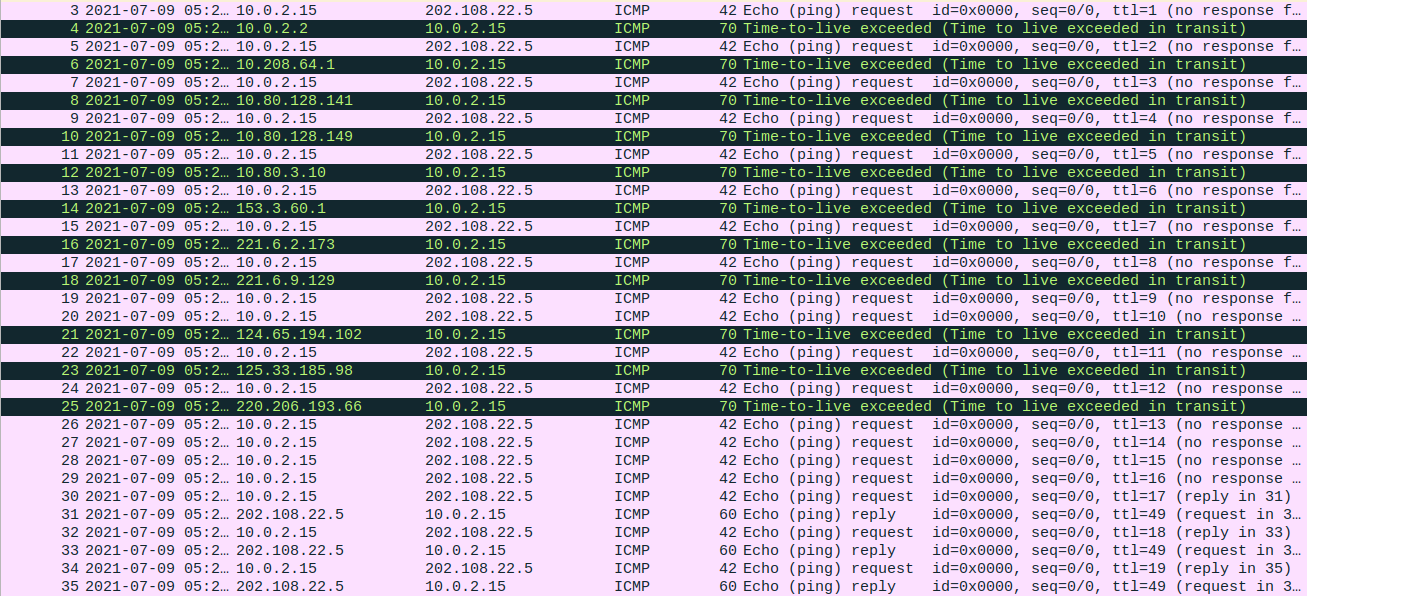
创建一个程序traceroute.py，向目标IP发送 ICMP 数据包，TTL初始值设置为1, 那么发出 的 ICMP 数据包在经历一个路由结点后, 就会失活被抛弃,利用循环, 不断增加TTL的值，最终使得数据包到达目的地，即得到第一个reply。



运行程序traceroute.py



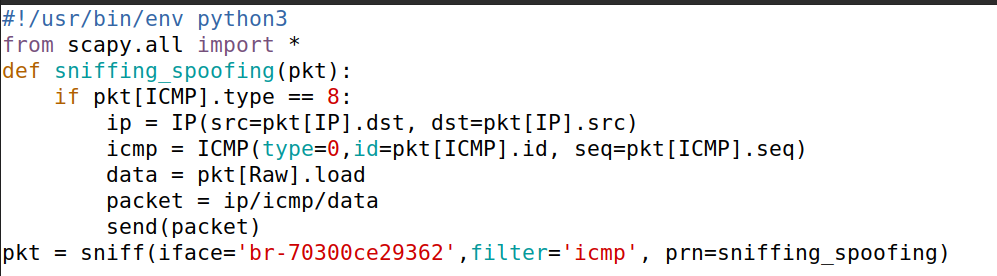
利用wireshark抓包结果如下



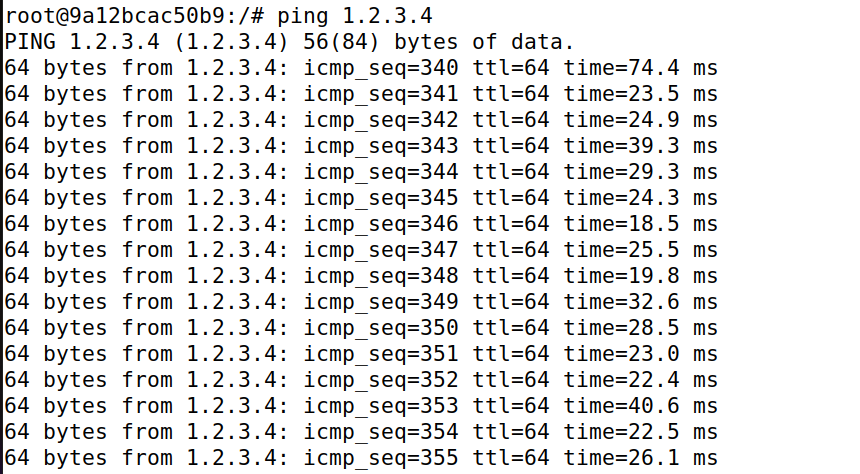
可以看到，在TTL=17时，得到第一个reply报文，说明VM和目的地址间隔约为17跳.

**Task1.4**

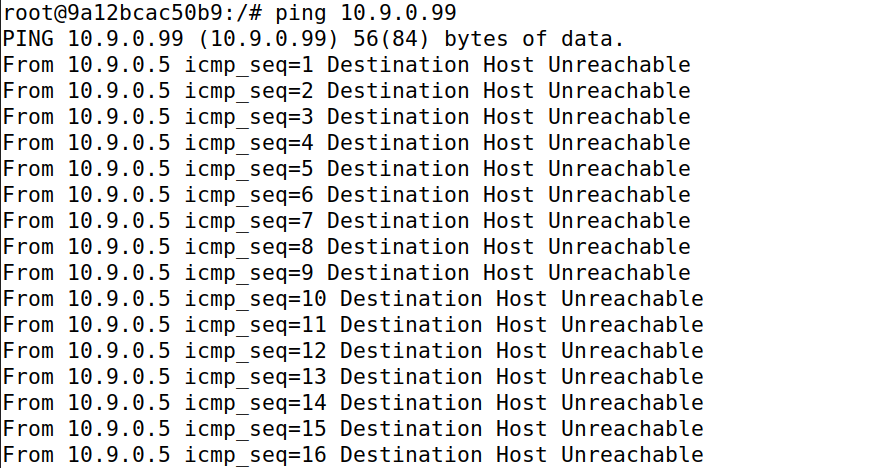
构造监听+欺骗程序如下



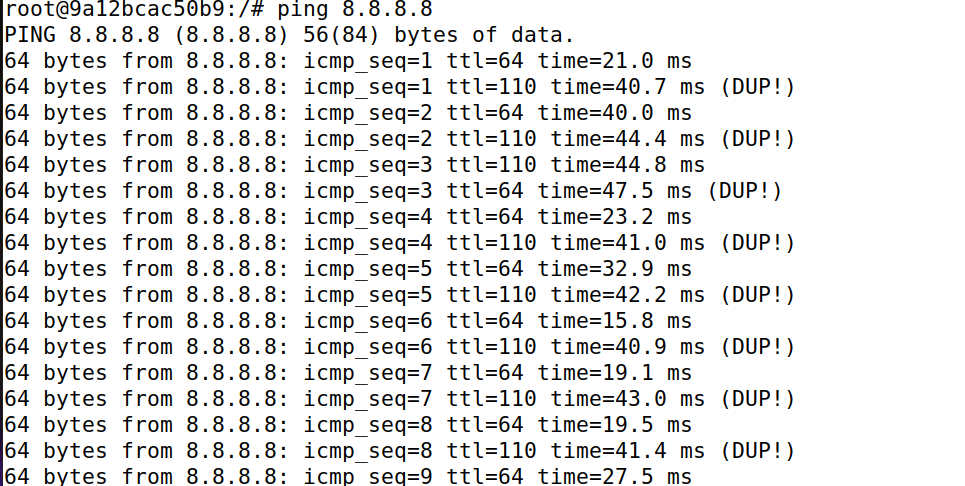
当ping一个网络上不存在的IP时，由于伪造报文，仍可以接收到响应。



当ping一个局域网内不存在的IP时，输出结果显示目标主机不可达，且没有发送伪造 reply 数据包。



当ping一个网络上存在的IP时，可以发现能 ping 通，而且出现 DUP！字样，说明收到多个 reply 报文，一个是正常 ping 通的 reply，一个是我们伪造的 reply。



外网的两个地址能 ping 通是因为，这两个的报文要经过 attacker 机出去，所以被attacker 检测到，并伪造了返回报文；但是 ping 内网的地址时，不需要经过 attacker，所以 attacker 没有返回伪造报文，且10.9.0.99 这个内网地址不存在，所以ping不通。