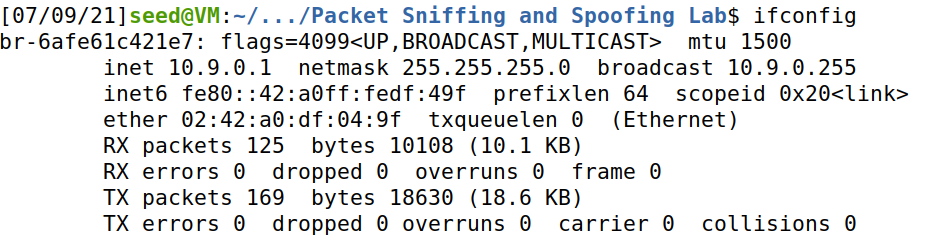
**Lab1**

57118111王雯

Task1.1A

查看网络接口：

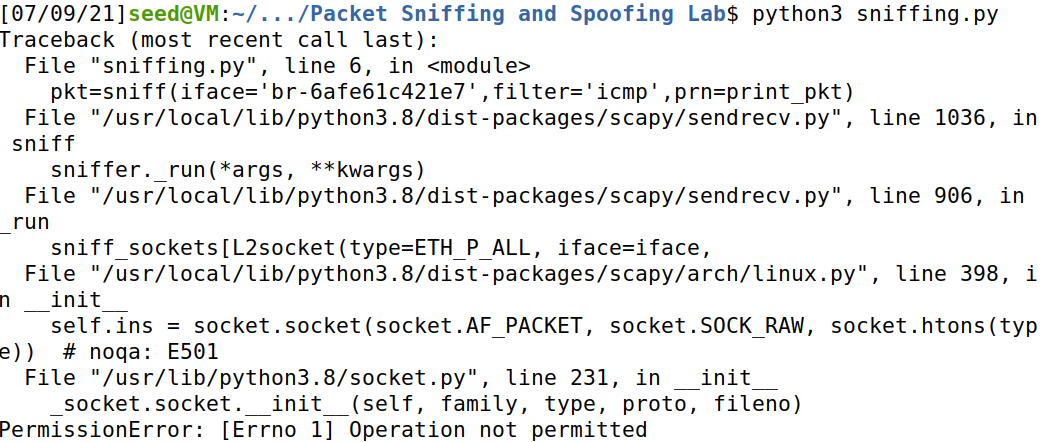


接口为br-6afe61c421e7

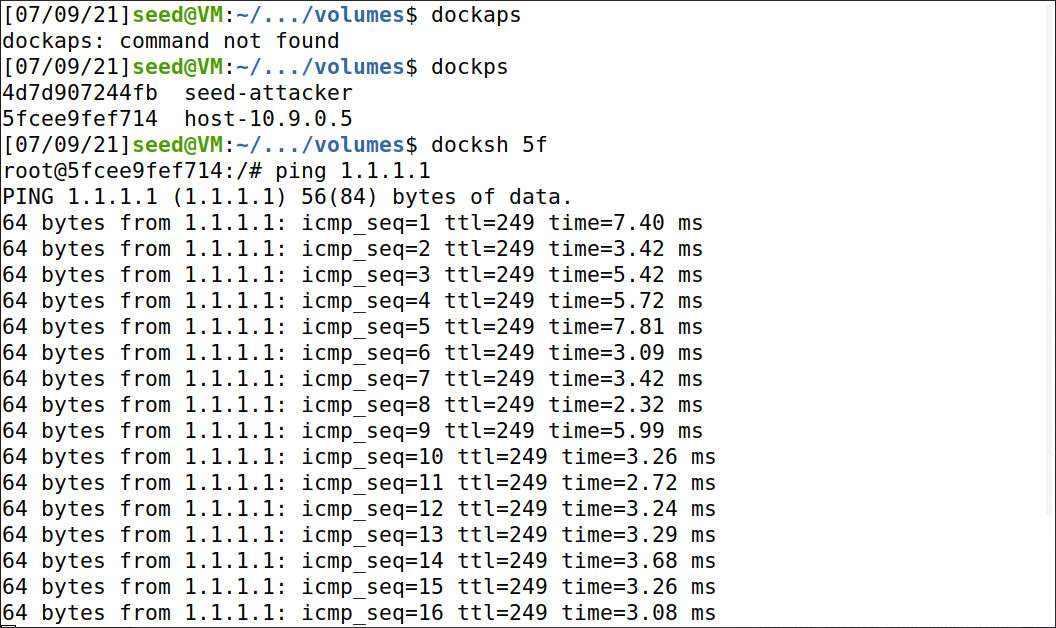
进入Lab目录下，新建sniffing.py文件，代码如下：

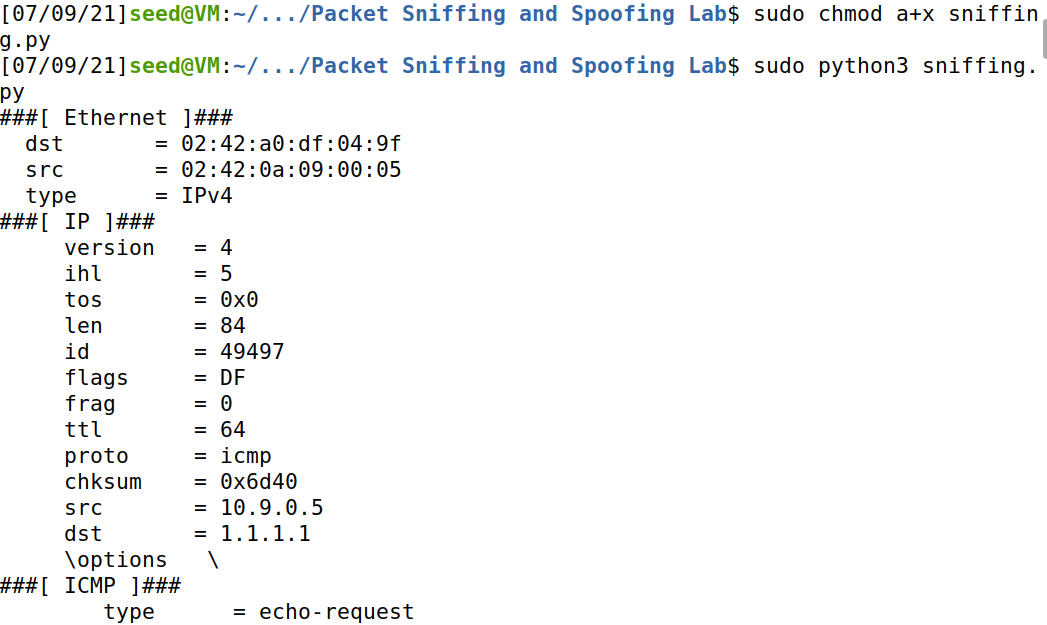


在普通用户下运行，可以发现无法运行：



当提权后，在root下运行程序，同时在docker上构造报文并发送，发现成功抓到报文：





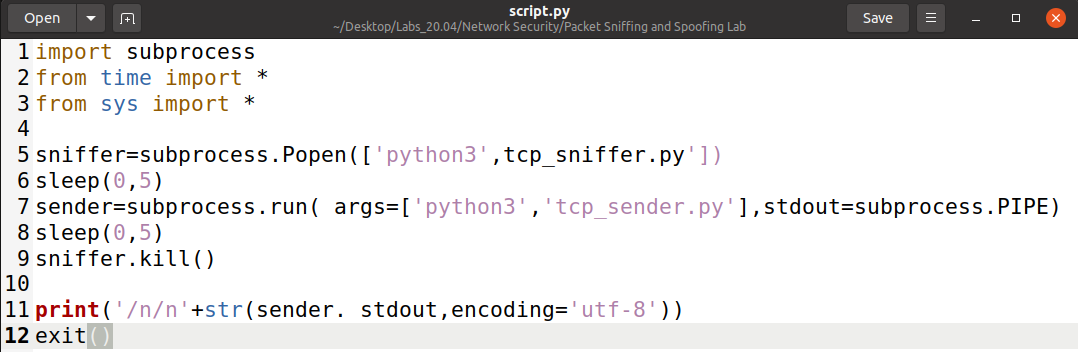
Task1.1B

第一问：捕获ICMP报文

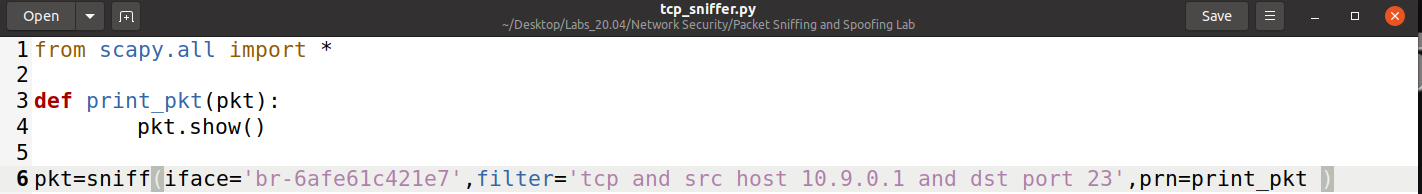
过程与1.1A相同

第二问：捕获来自特定IP、目的端口23的数据包

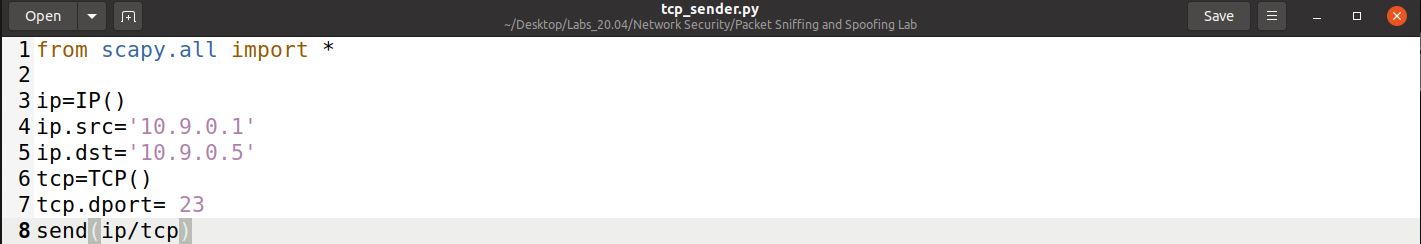
script.py的代码如下



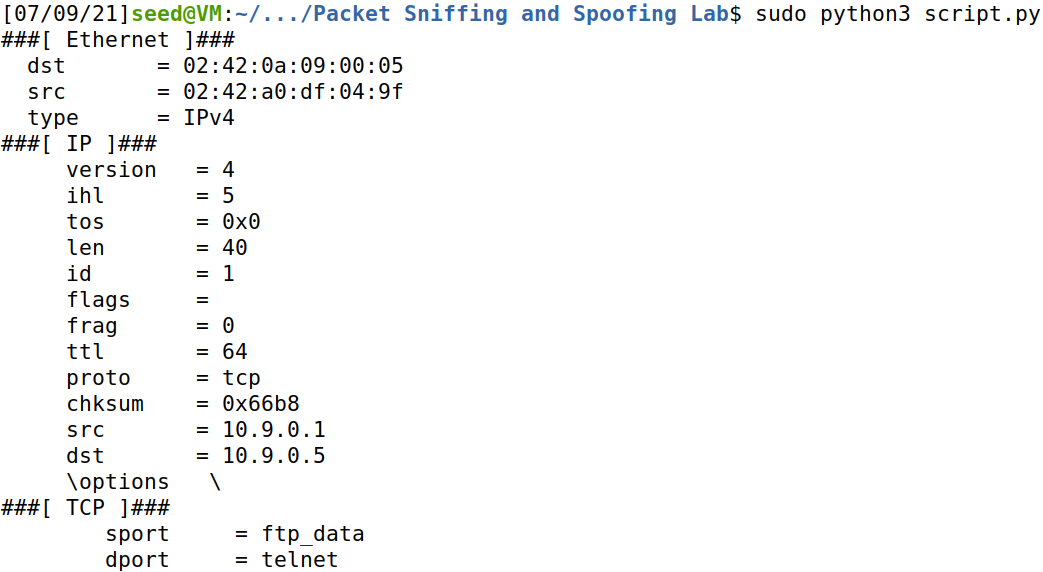
tcp\_sniffer.py代码如下：



tcp\_sender.py代码如下：

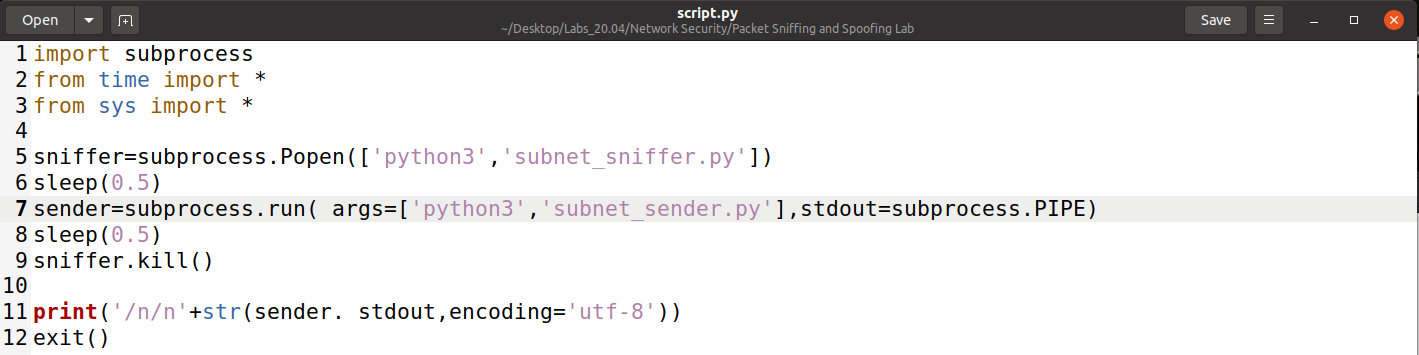


程序监听到的报文如下：

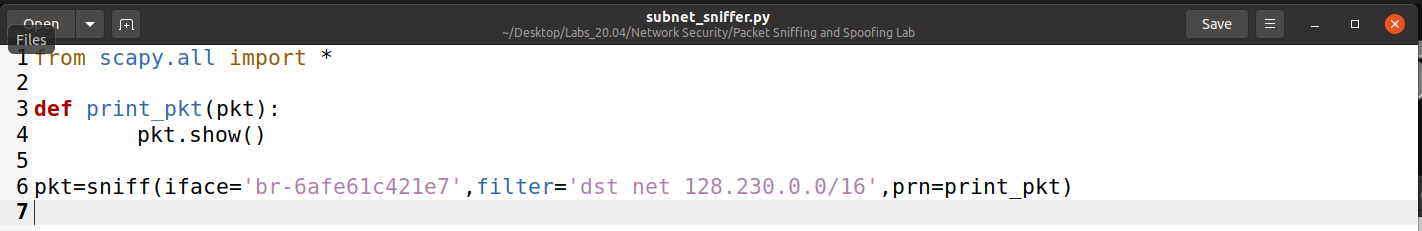


第三问：来自特定子网的报文

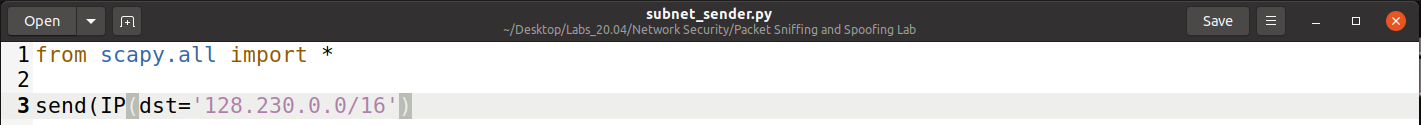
script.py代码如下：



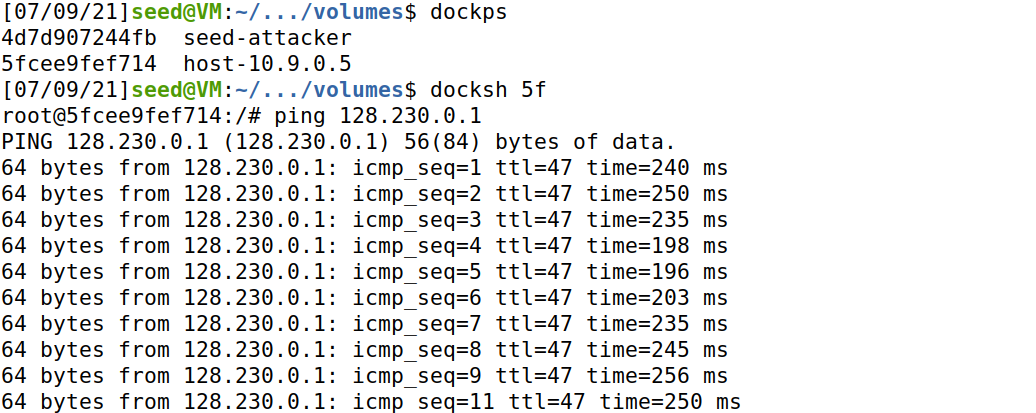
subnet\_sniffer.py代码如下：



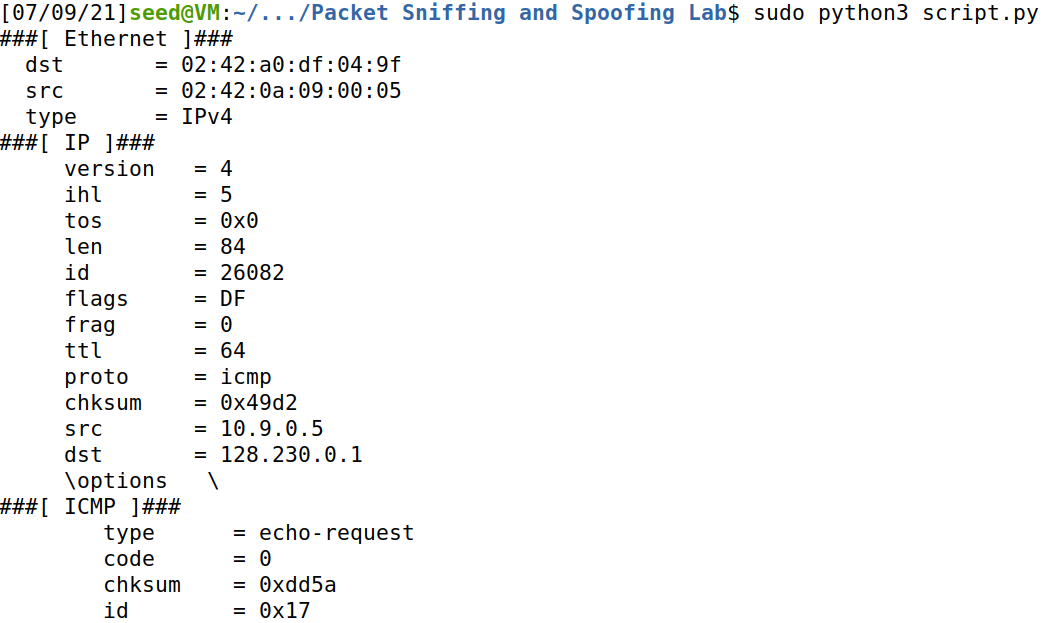
subnet\_sender.py代码如下：



登录10.9.0.5 ping128.230.0.1



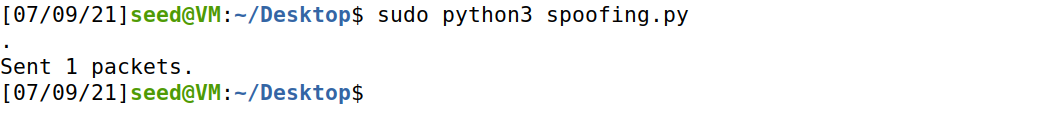
程序监听到的报文如下：



Task1.2

使用spoofing.py发包



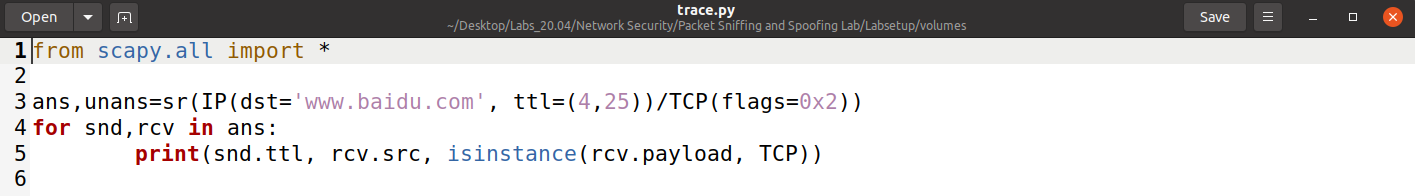


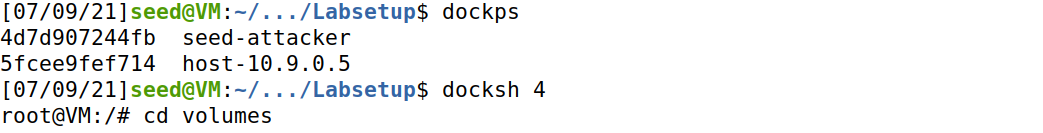
Wireshark 捕获到的来回的数据包显示如下：

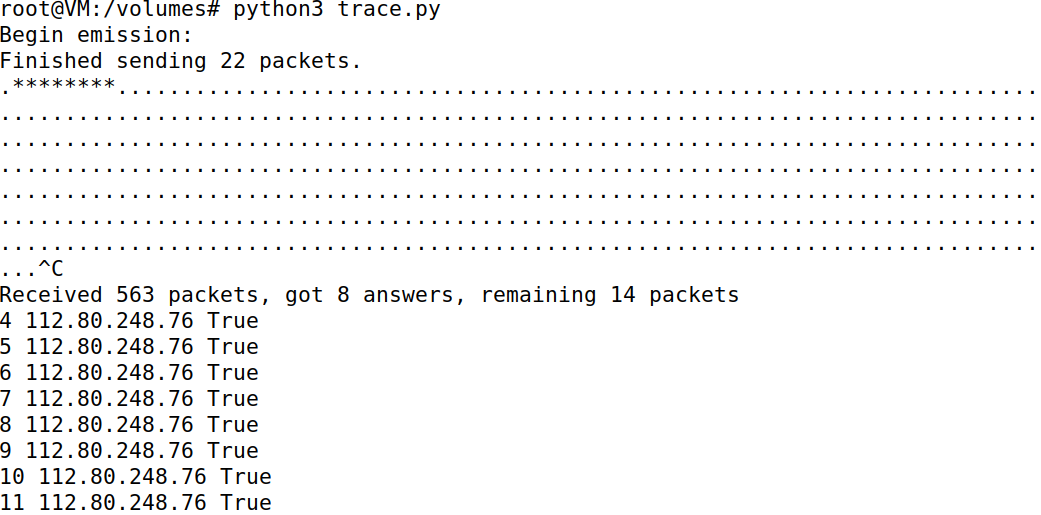
C:\Users\dengtong\AppData\Roaming\Tencent\Users\1977988055\QQ\WinTemp\RichOle\}IDU~1YJ7{]O6XRAHT4BW%R.png

Task1.3

编写如下程序







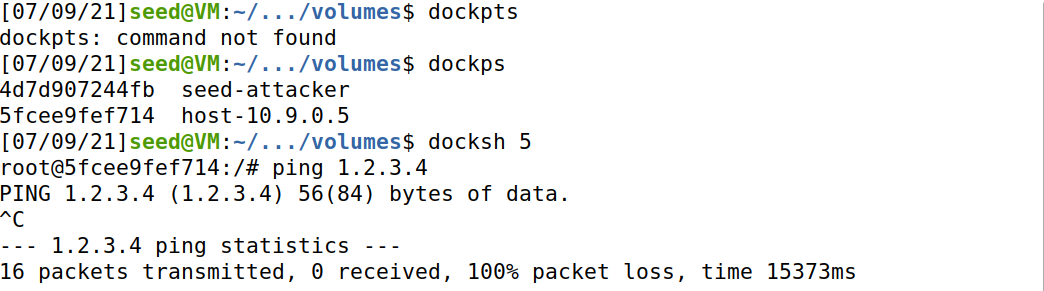
Task1.4

编写的程序如下：

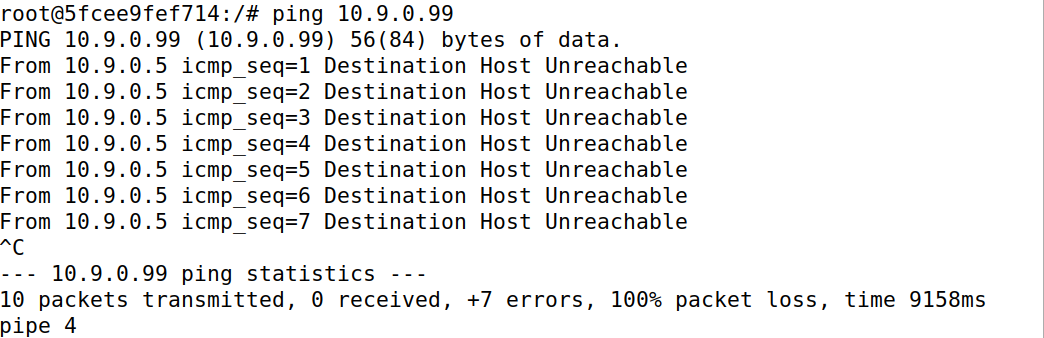


在运行前，分别ping三个地址

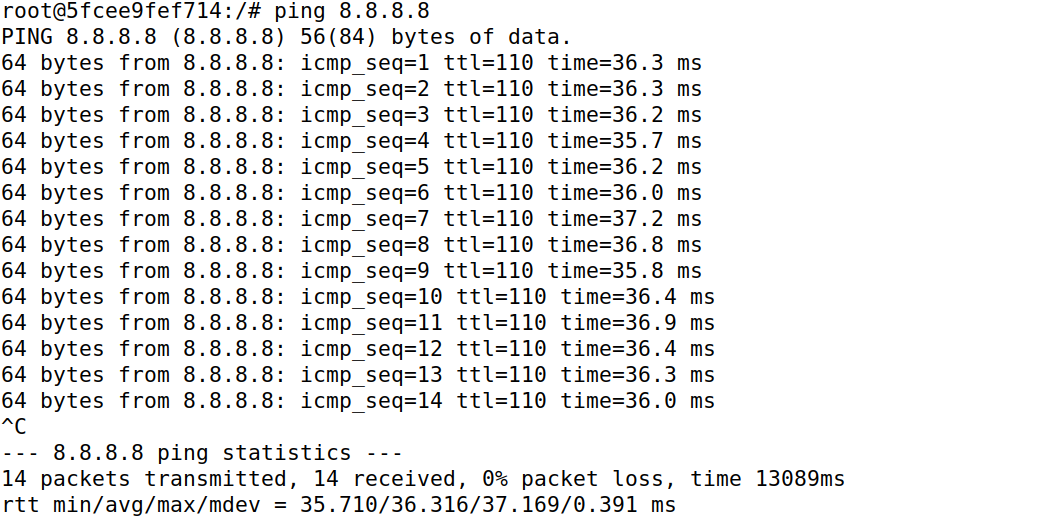
ping1.2.3.4



ping10.9.0.99



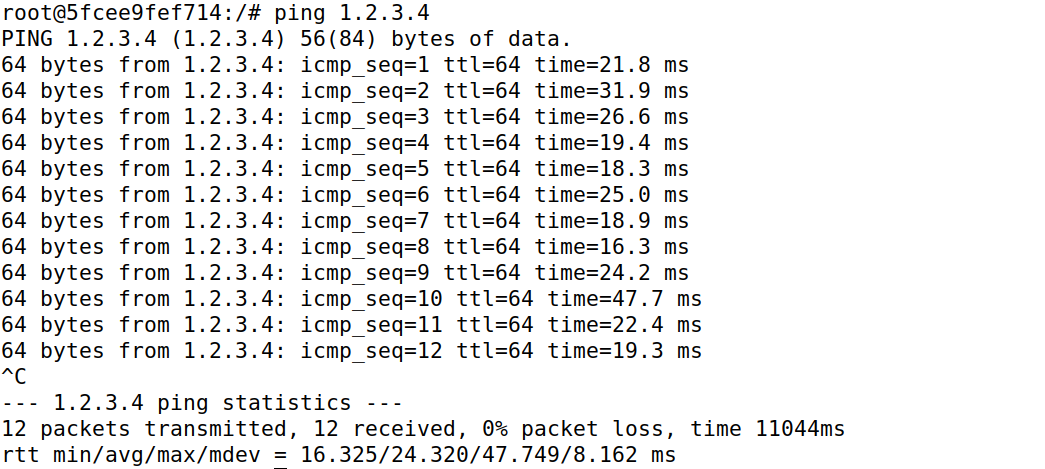
ping8.8.8.8



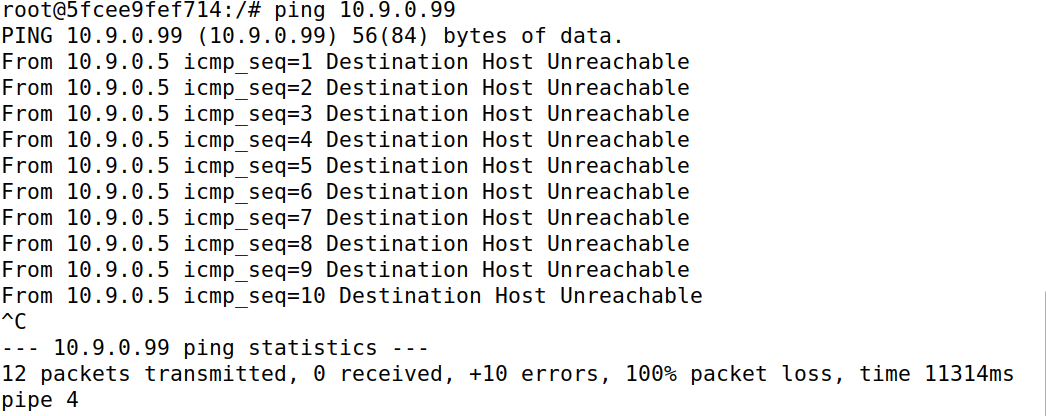
可以发现，在运行sniff\_spoof.py前，无法ping通1.2.3.4和10.9.0.99，但能ping通8.8.8.8

运行sniff\_spoof.py，再次ping三个地址：

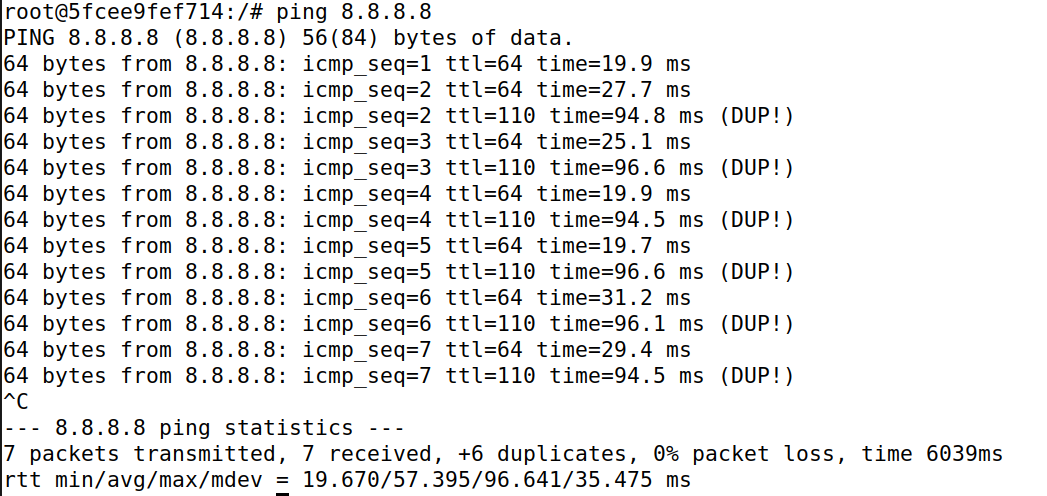
ping 1.2.3.4



ping10.9.0.99



ping8.8.8.8



**结论：**可以发现，在运行 sniff\_spoof.py 之后， 10.9.0.99 仍然无法 ping 通，但1.2.3.4 能 ping 通。因为在运行程序之前，网关 10.9.0.5 告知主机无法通过 ARP 协议找到1.2.3.4 和 10.0.9.99 对应的 MAC 地址，因此无法 ping 通；而 8.8.8.8 在互联网上存在，因此可以 ping 通。

在运行程序之后， ping 1.2.3.4 需要经过网关 10.9.0.5，网关拦截 ICMP 报文并欺骗主机可以 ping 通 1.2.3.4。而 10.9.0.99 和主机在同一个局域网内，通过广播 ARP 寻找相应的 MAC 地址，不需要经过网关，因此网关无法欺骗主机，10.9.0.99 仍然 ping 不通。