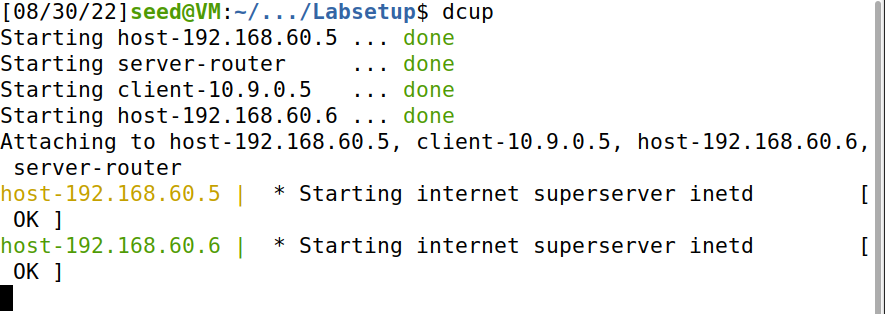
**Task 4 - VPN**

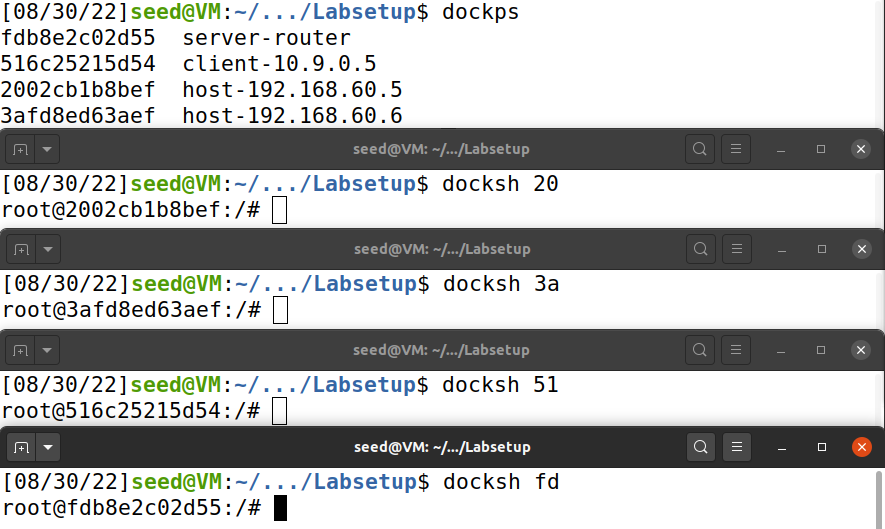
**环境配置**

进入实验目录/home/seed/Desktop/Labs\_20.04/Network Security/VPN Tunneling Lab/Labsetup

输入dcup命令开启容器：

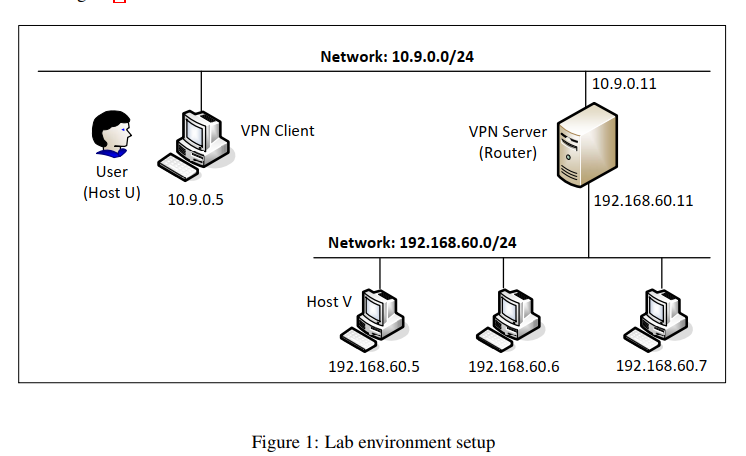


另外开启多个终端，输入docksh xx来连接到对应容器：



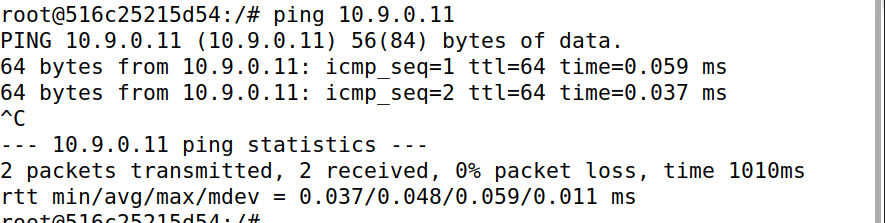
从上到下依次是host-192.168.60.6、host-192.168.60.5、client-10.9.0.5、server-router。

实验网络拓扑如下：



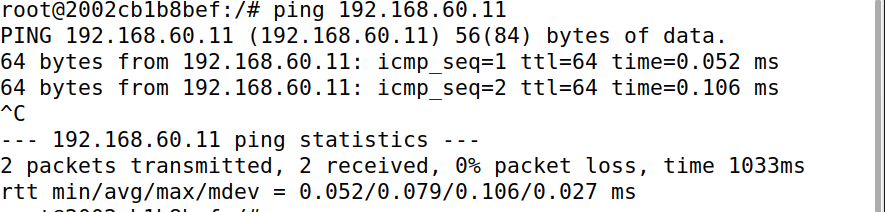
1. **Testing Network**

在client（10.9.0.5）上ping VPN server（10.9.0.11），测试两者通信：



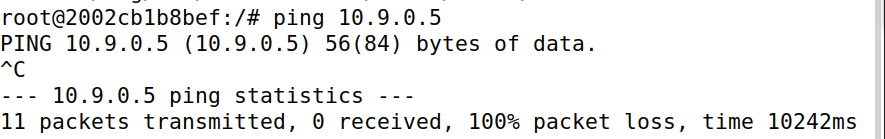
成功ping通。

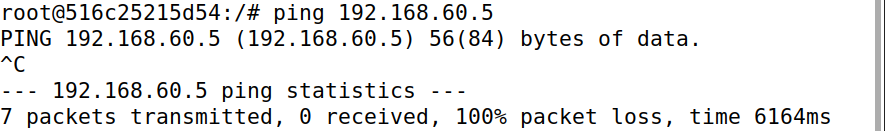
在host-192.168.60.5上ping VPN Server，测试两者通信：



成功ping通。

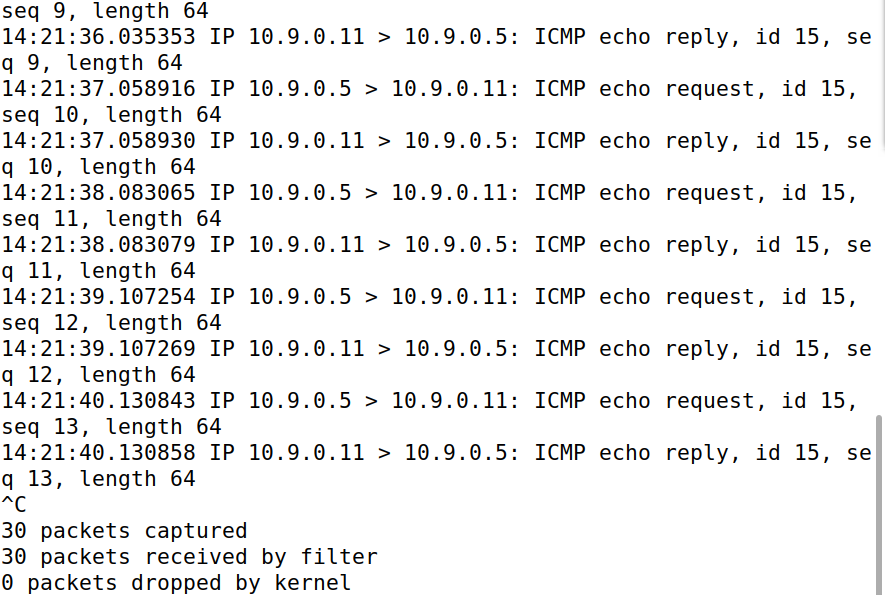
测试192.168.60.5和10.9.0.5之间的连通性：



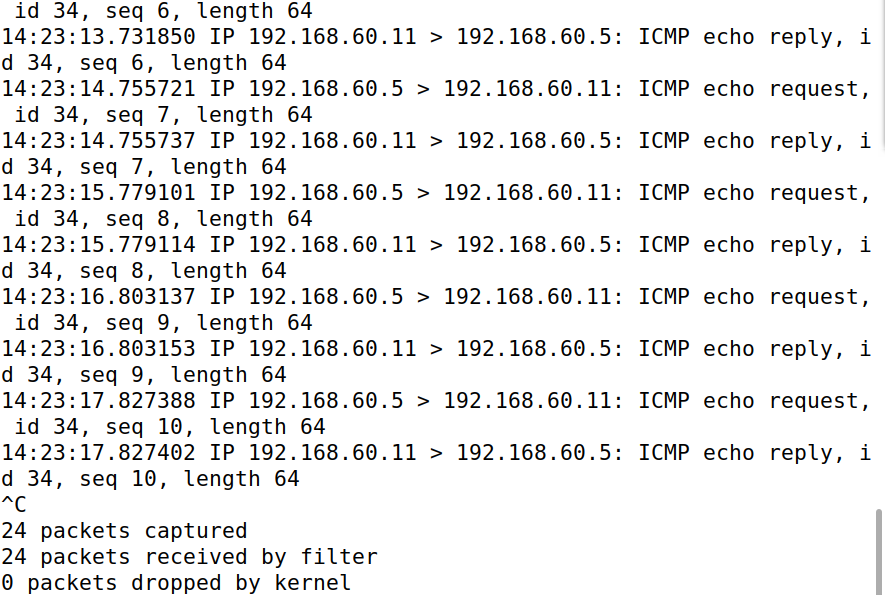


两边均是不通的。

在Server上输入tcpdump -i eth0 -n，运行tcpdump，监听eth0网卡，然后在10.9.0.5上ping Server（10.9.0.11），结果如下：



在Server上输入tcpdump -i eth1 -n，运行tcpdump，监听eth1网卡，然后在192.168.60.5上ping Server（192.168.60.11），结果如下：

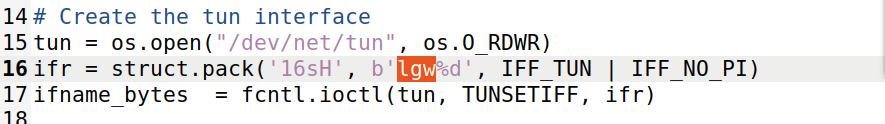


连通性测试完成。

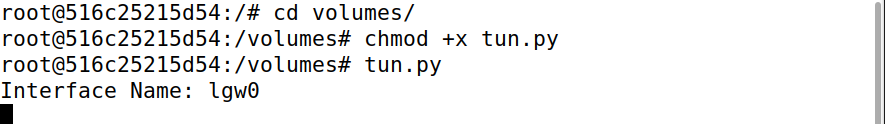
1. **Create and Configure TUN Interface**

**2A Name of the Interface**

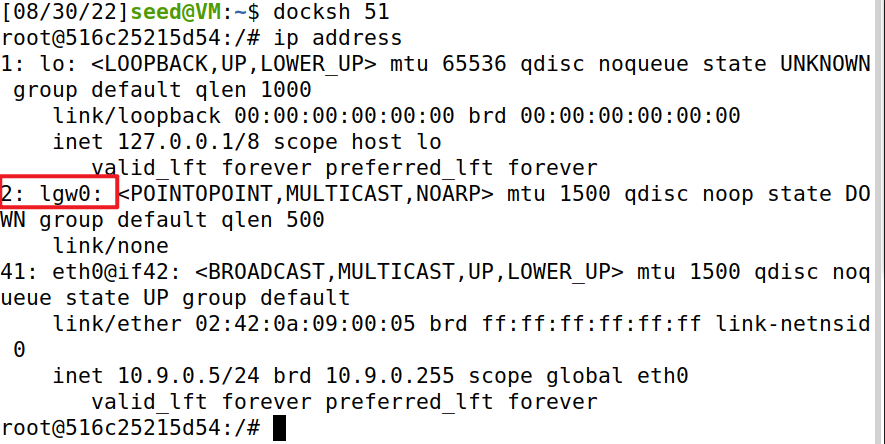
将tun.py中的网卡名称tun修改为lgw：



在10.9.0.5中输入chmod +x tun.py添加运行权限后执行：



另开一个shell进入10.9.0.5容器，输入ip address查看容器内的网卡：

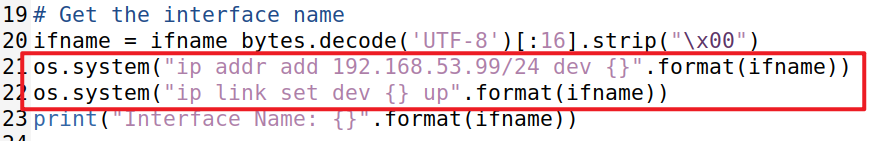


可以看到有前面新创建的网卡。

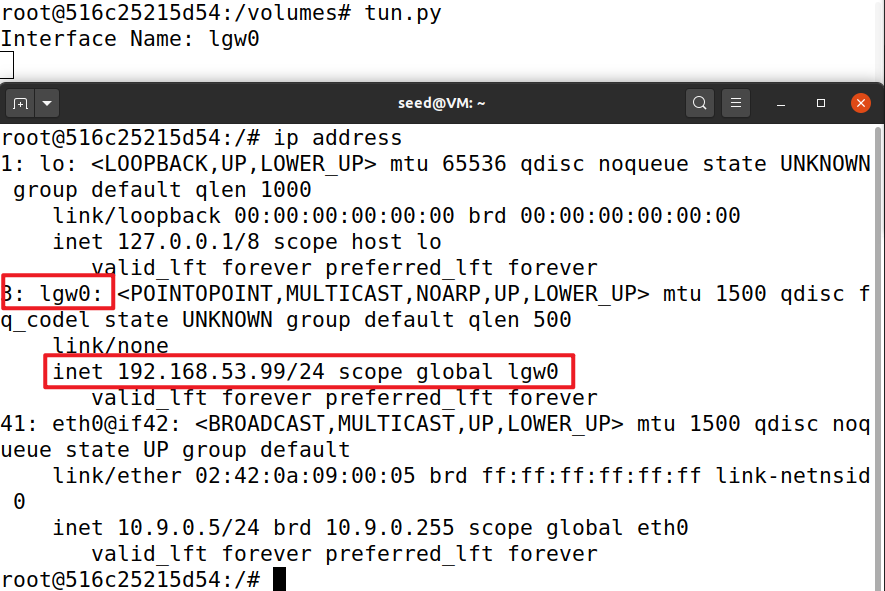
**2B Set up the TUN Interface**

在网卡可用之前，我们需要做两件事，一是为其分配地址，而是开启网卡。

在tun.py中添加以下内容为网卡分配IP地址：



再次在10.9.0.5上运行，在前面所开启的另一个终端中输入ip address查看网卡信息：

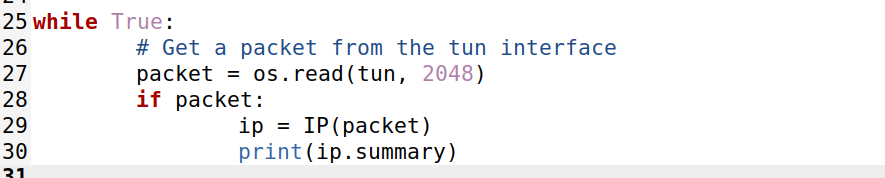


可以看到已经成功分配了ip地址。

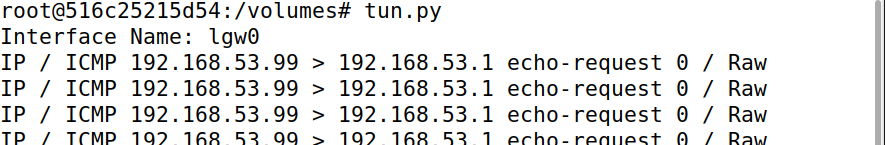
**2C Read from the TUN Interface**

这个实验将会从TUN interface中读取数据。

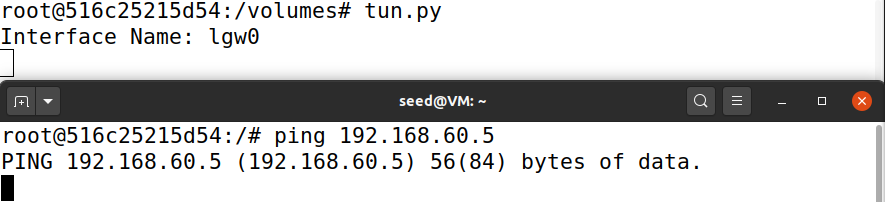
修改tun.py文件，修改while循环如下：



在10.9.0.5上运行tun.py文件，在另一个终端上ping 192.168.53.0/24子网的地址，以192.168.53.1为例，查看tun.py的输出：



继续ping 192.168.60.0/24的地址，以192.168.60.5为例，查看tun.py的输出：

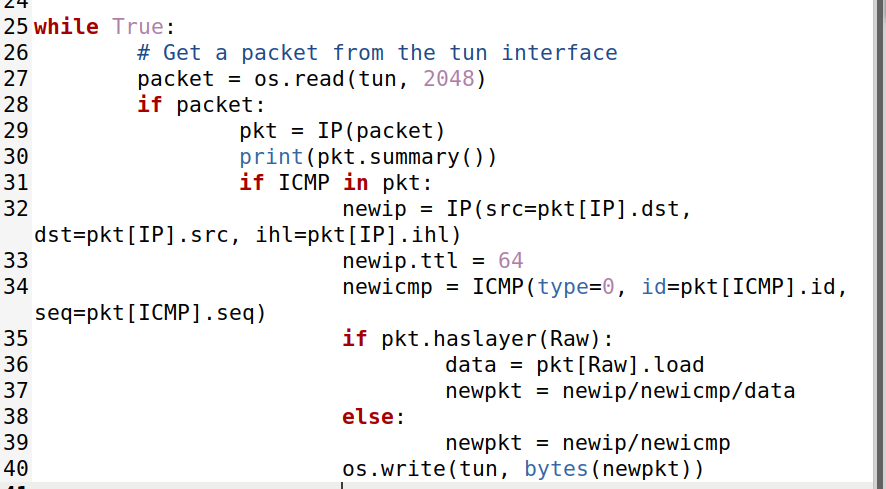


是没有输出的。

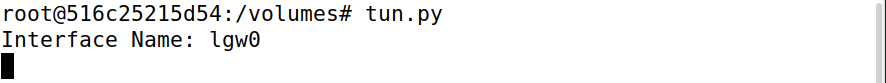
**2D Write to the TUN Interface**

向TUN interface写入数据。

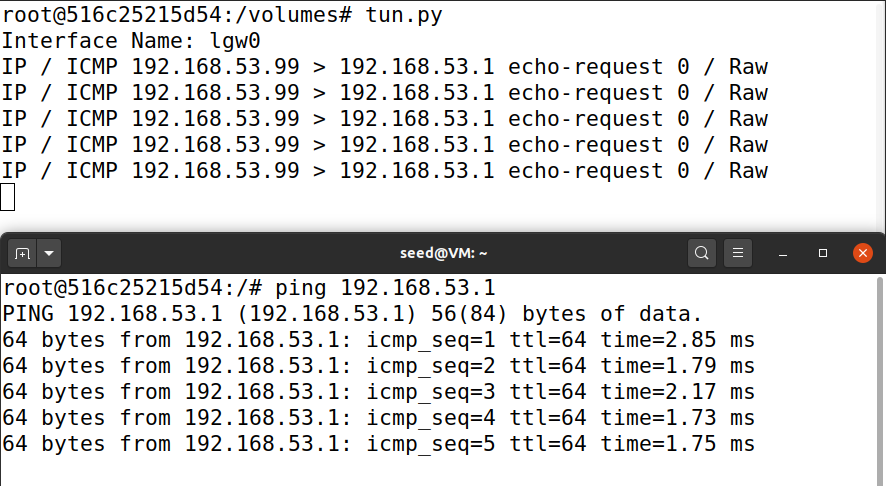
修改tun.py while循环内容如下，收到ICMP request后写入echo reply packet：



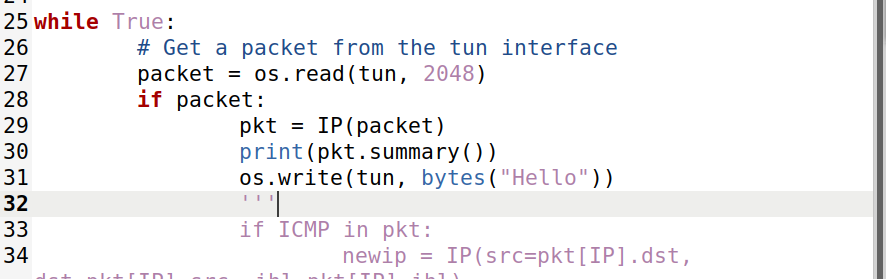
在10.9.0.5上运行：



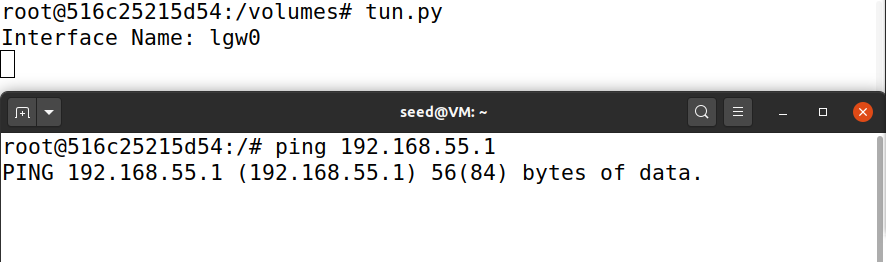
在另一个终端上ping 192.168.53.1，结果如下：



修改tun.py的循环部分如下：



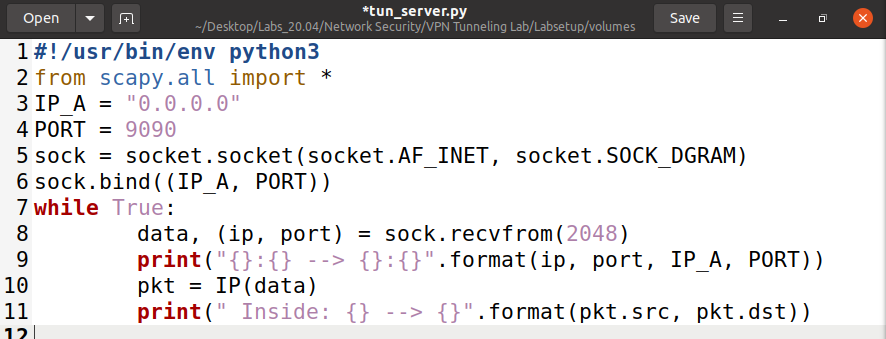
再次运行，ping 192.168.55.1，结果如下：



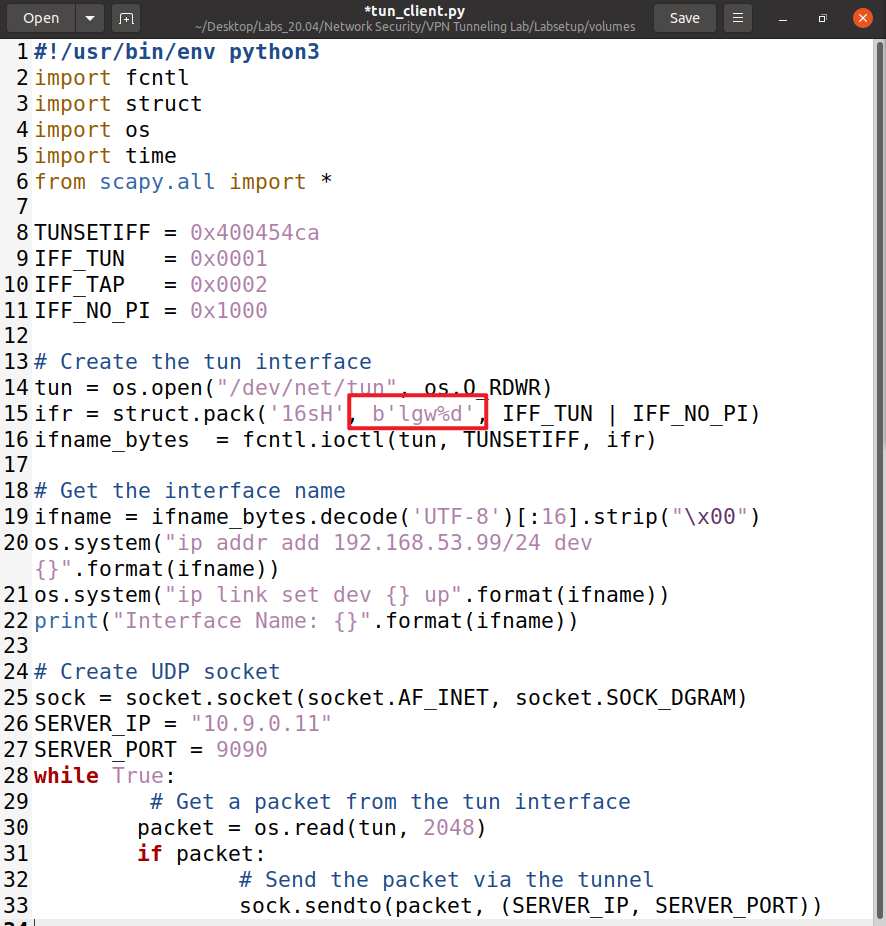
发现不通。

1. **Send the IP Packet to VPN Server Through a Tunnel**

在server router中创建tun\_server.py文件，内容如下:

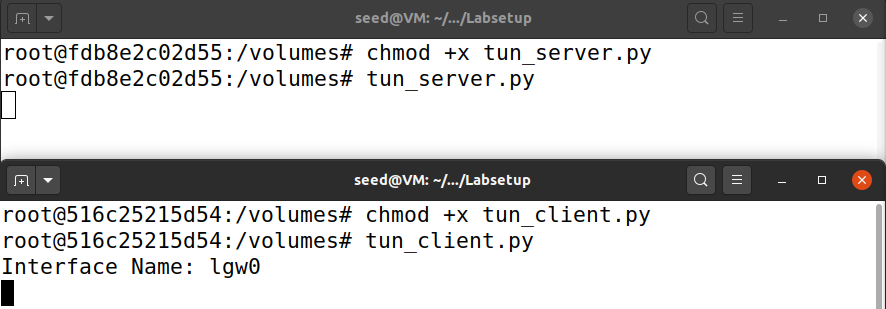


在10.9.0.5上创建tun\_client.py文件，内容如下：

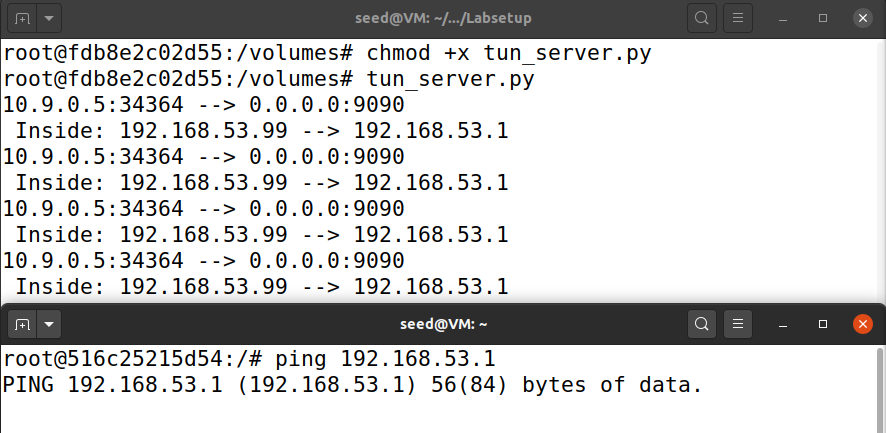


注意修改网卡名称为“lgw”。

在VPN Server(10.9.0.11)上运行tun\_server.py，在Host U(10.9.0.5)上运行tun\_client.py：

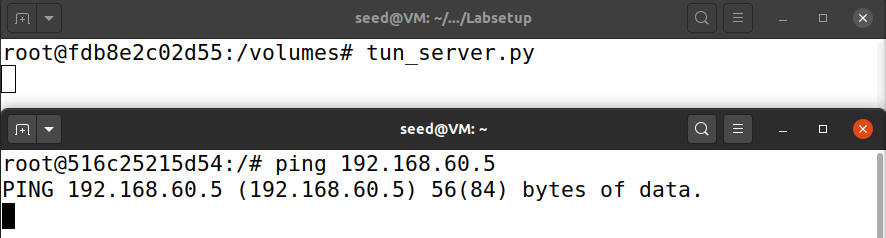


然后另开shell，在Host U上Ping 192.168.53.0/24，以192.168.53.1为例，然后查看VPN server的输出，如下所示:



发现10.9.0.5到192.168.53.0/24网段是不通的，但server router上可以看到相应输出的内容，原因是10.9.0.5发出的ping的流量进入了lgw0网卡后被tun\_client.py程序读取，发送到了tun\_server，但是没有收取返回的包，所以显示为不通。

在Host U上继续ping Host V(192.168.60.5)，结果如下：

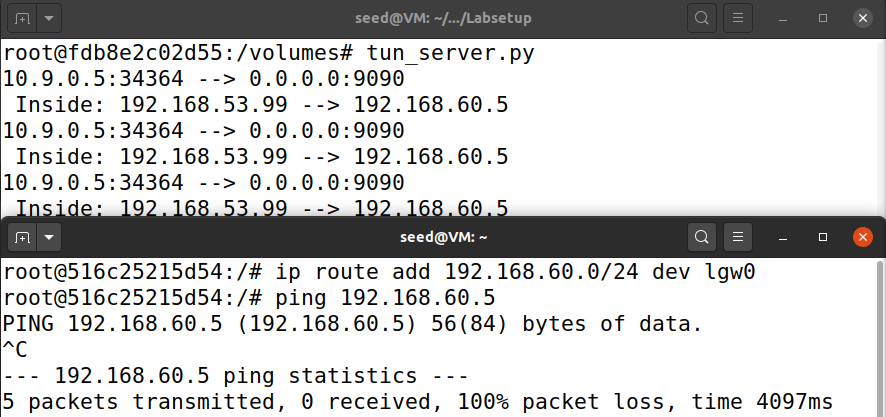


发现VPN Server没有输出了，因为流量没有被lgw0网卡接收，也就送不到VPN server了。

既然无法被网卡lgw0接收，我们就自己设定路由，使流量转到lgw0网卡上，在Host U上运行命令ip route add 192.168.60.0/24 dev lgw0，来手动添加路由转发，使其经过lgw0网：



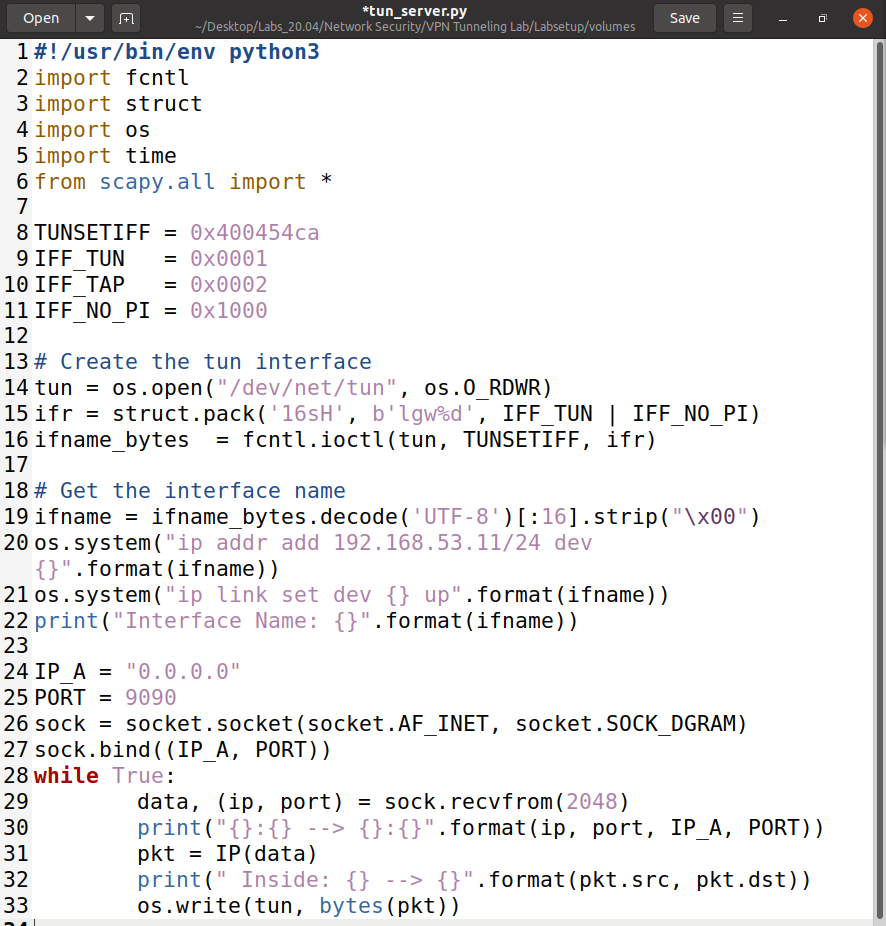
然后再ping 192.168.60.5，查看tun\_server.py的输出:



可以看到在server上有了192.168.53.99到192.168.60.5的流量。因为此时的流量经过了网卡lgw0，可以被捕获到。

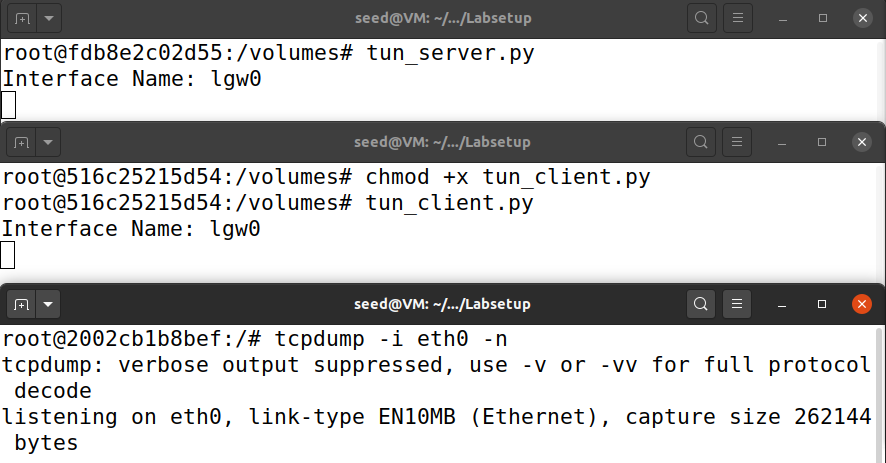
1. **Spoofing NS Records for Another Domain**

修改tun\_server.py文件，如下所示:

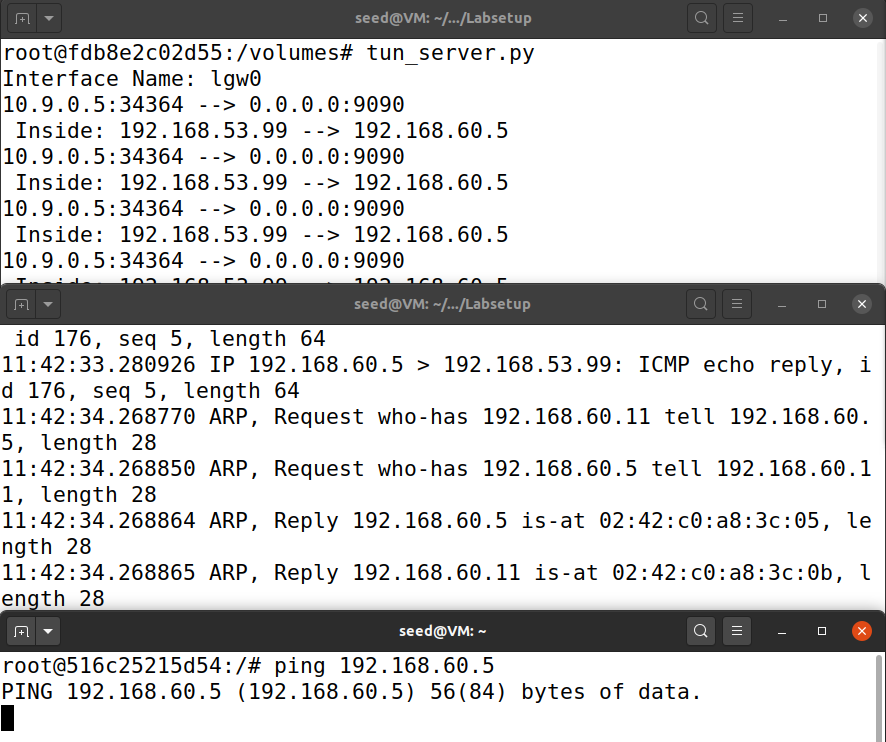


仍是注意修改网卡名称。

在VPN Server上运行tun\_server.py文件，在Host U上运行tun\_client.py文件，在Host V(192.168.60.5)上输入tcpdump -i eth0 -n开启监听：



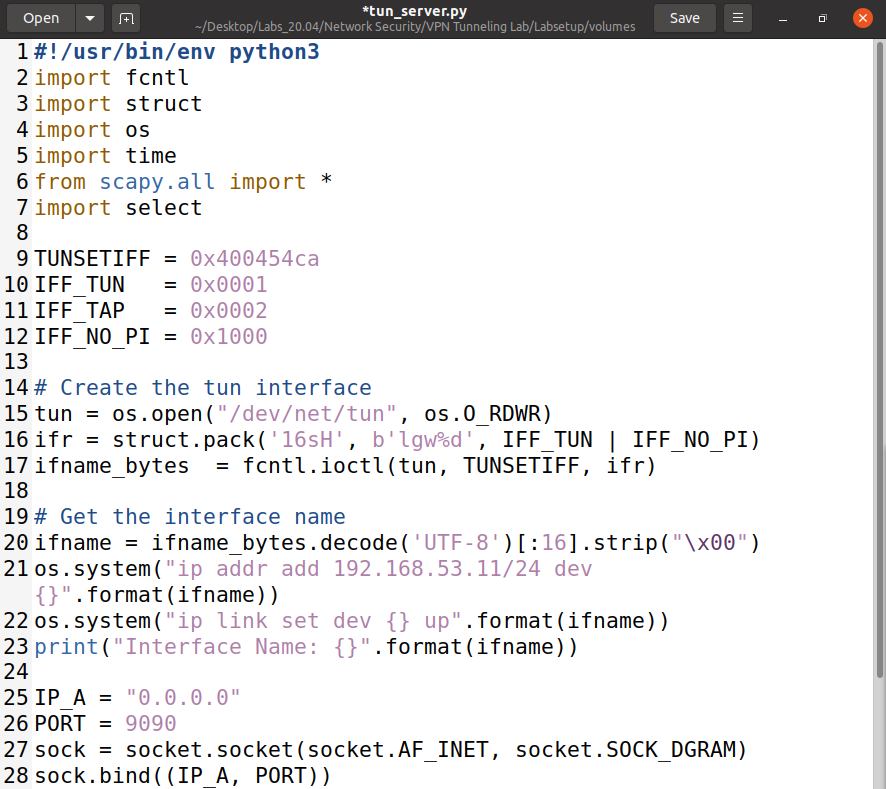
在Host U上ping 192.168.60.5，结果如下：

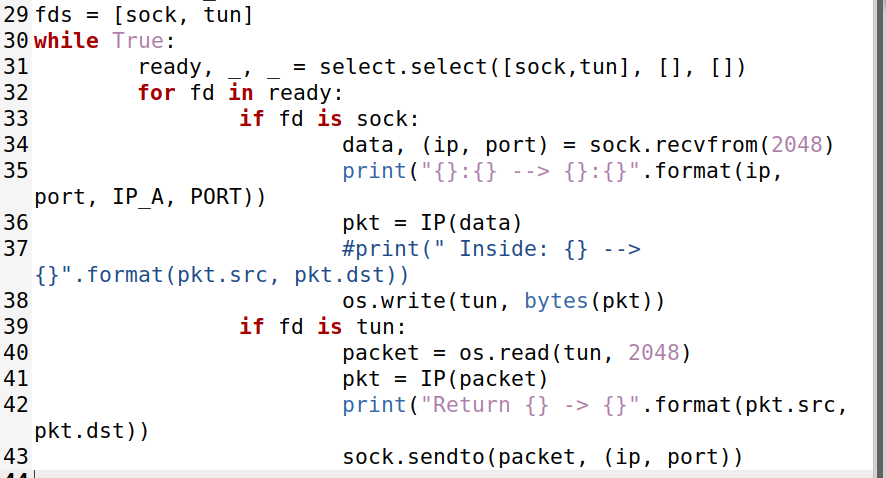


可以看到在server上和host V上均有输出。

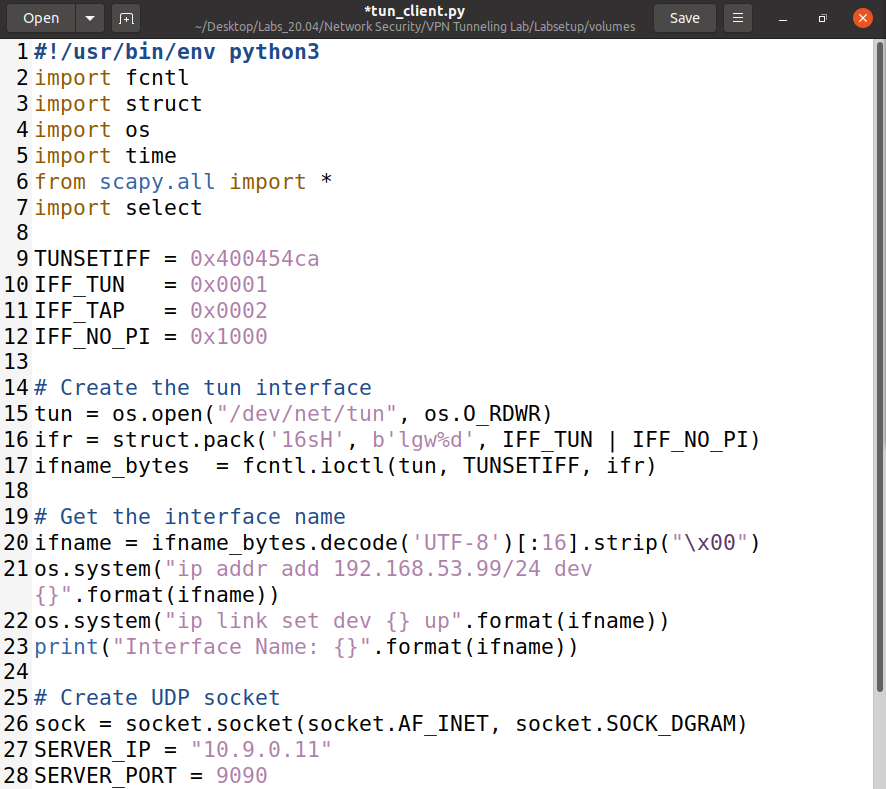
1. **Handling Traffic in Both Directions**

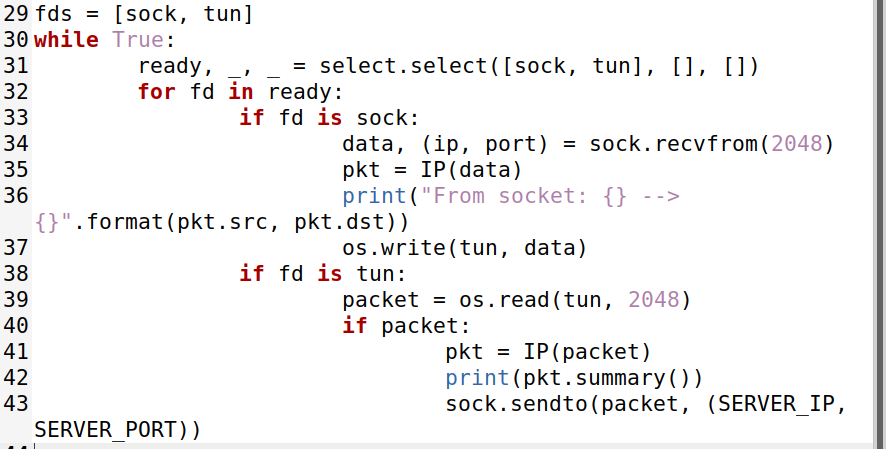
修改tun\_server.py，内容如下:



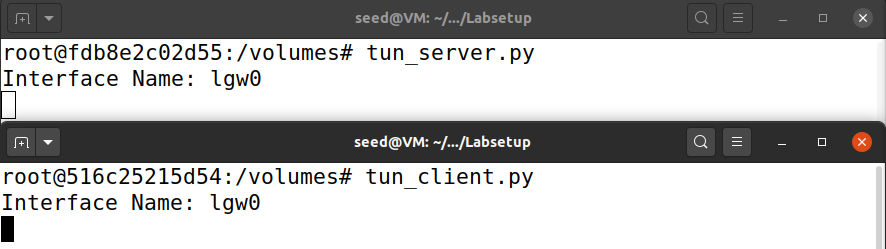


修改tun\_client.py，内容如下：

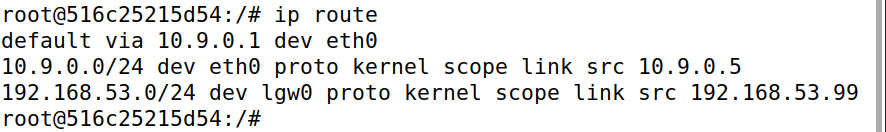




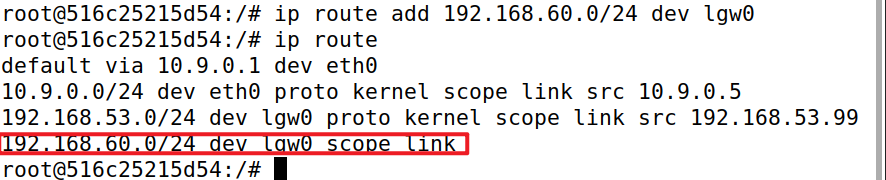
在VPN Server上运行tun\_server.py，在Host U上运行tun\_client.py：



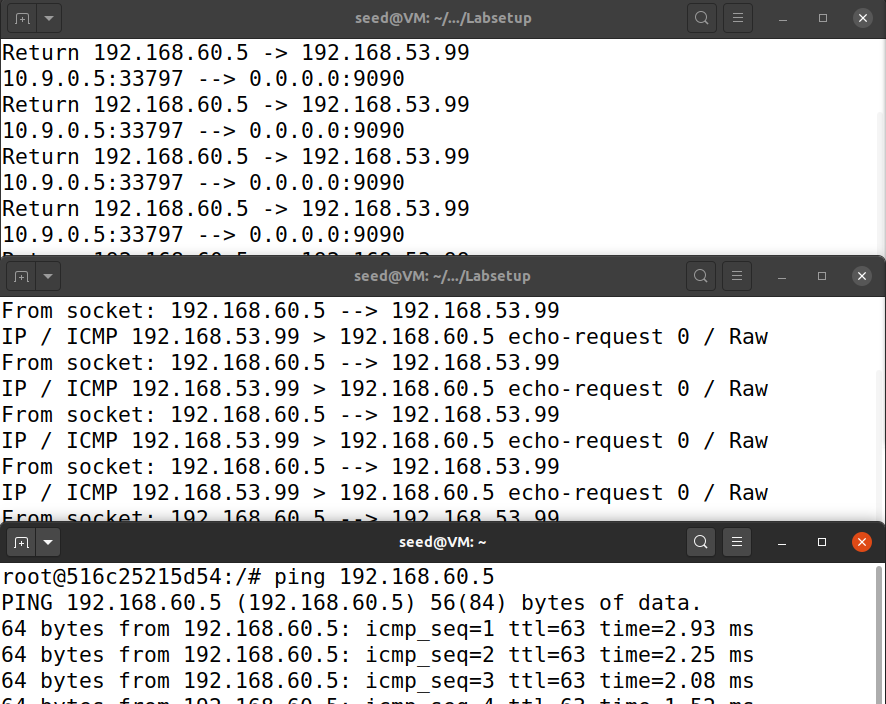
在host U中输入ip route查看路由表记录：



没有192.168.60.0网段的记录，手动添加：

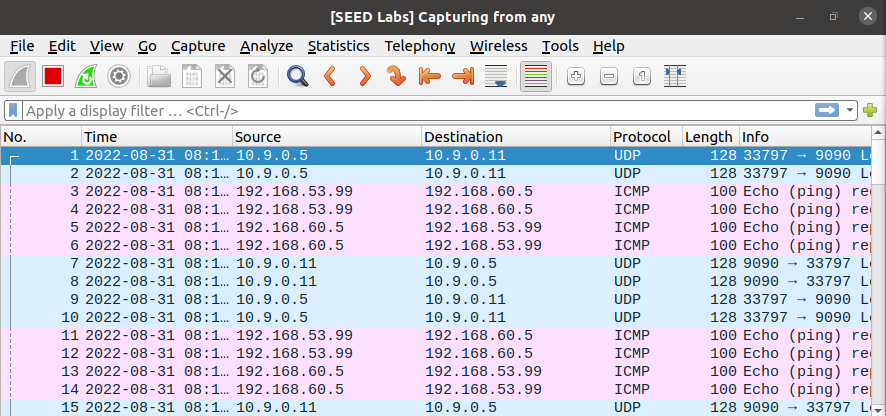


在Host U上ping 192.168.60.5，结果如下：



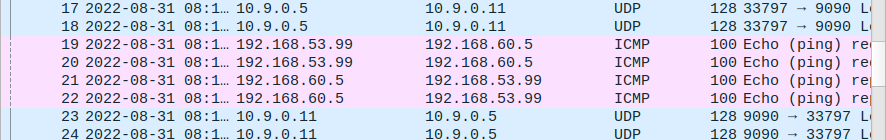
可以看到有回复了。

在wireshark中监听所有网卡，再次ping 192.168.60.5，在wireshark中看到结果如下：

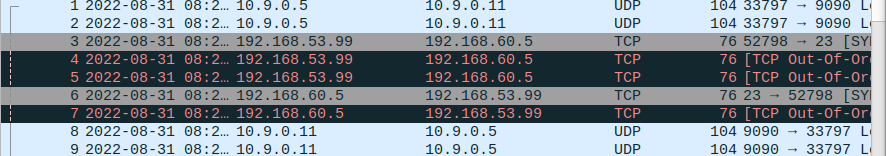


对以上捕获到的流量进行分析：

可以看到是从Host U(10.9.0.5)发送到Router(10.9.0.11/192.168.60.11)，再到Host V(192.168.60.5)，然后回送的包是相反的顺序：



在使用telnet连接host V时也是同样的路径：

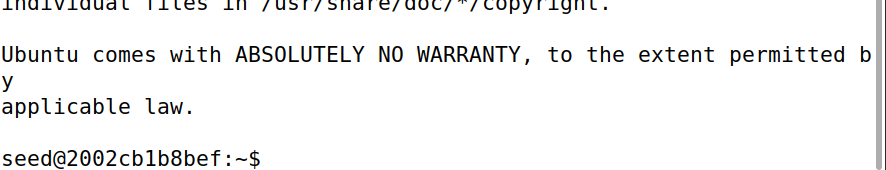


1. **Tunnel-Breaking Experiment**

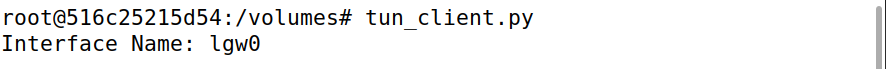
在Host U上发起对Host V的telnet连接：

终止tun\_client.py程序的运行。

在Host U上输入一些内容，发现无论输入什么都没有显示：



再次运行起tun\_client.py：



再次在Host U中运行tun\_client.py，再尝试在telnet中输入，发现连接情况正常了：

