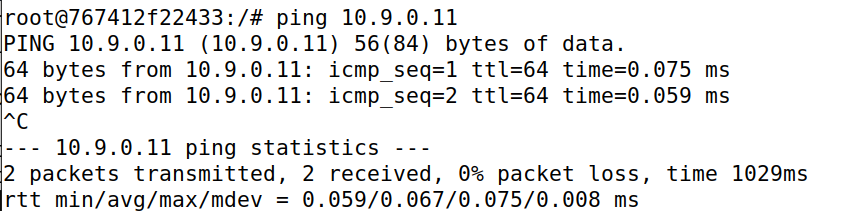
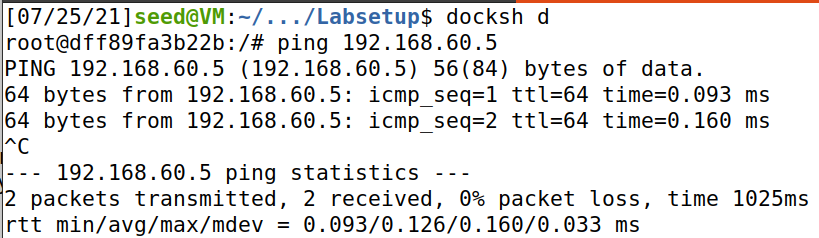
**TASK 1**

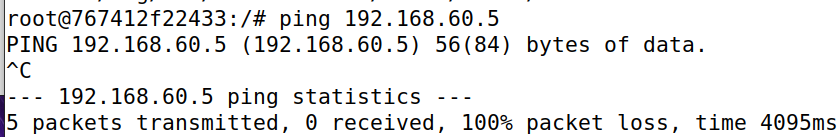
首先对环境进行测试

HOST U ping VPNServer，可以ping通

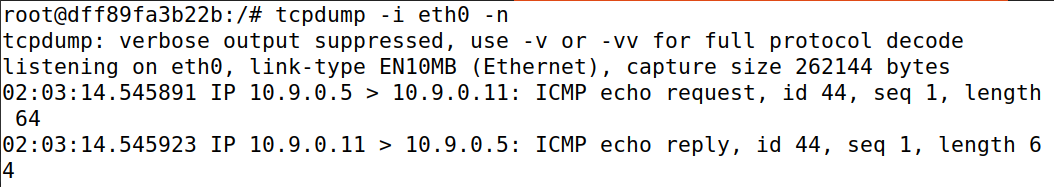


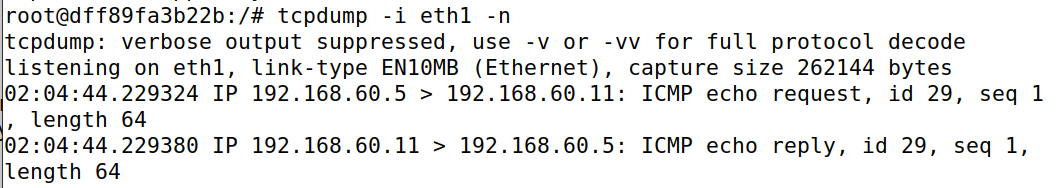
VPNServer ping HOST V，可以ping通



HOST U ping HOST V，连接失败

在router上运行TCP dump，结果如下，可以正常捕获报文

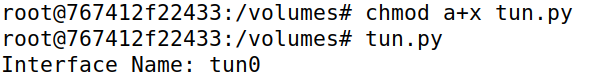


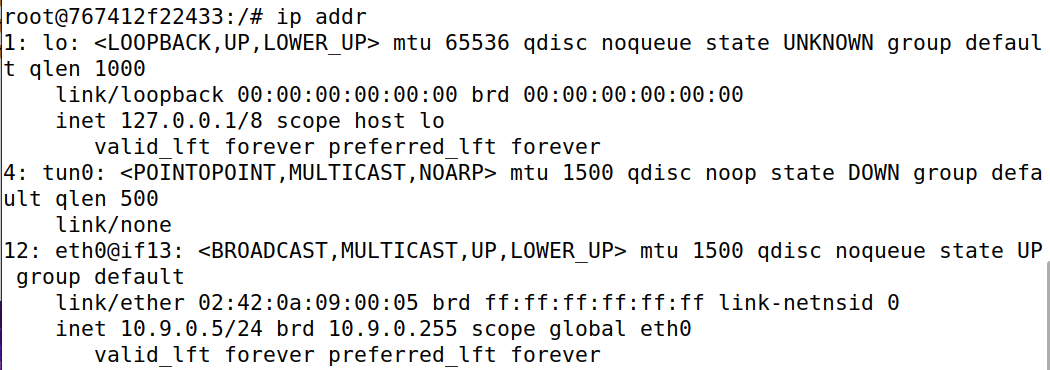


**TASK 2**

**2.A**

运行所给程序

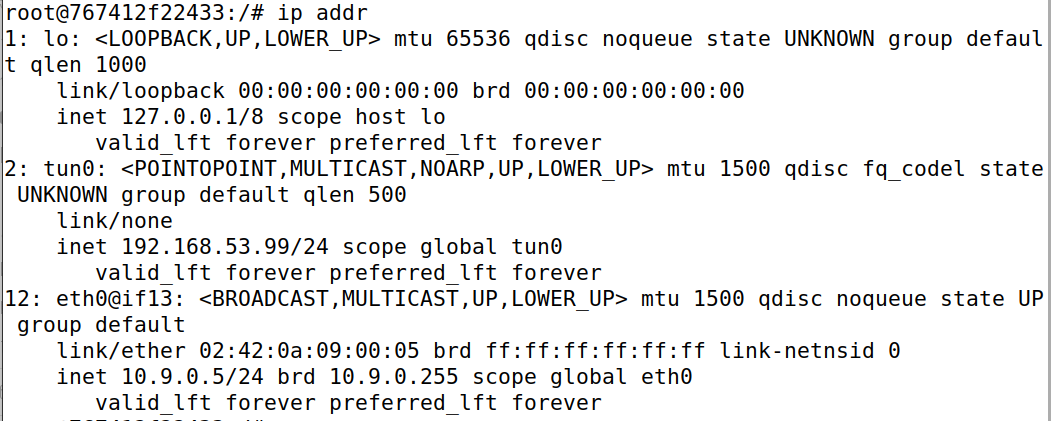


然后在新窗口中输入ip addr命令，结果如下，创建了接口但无地址信息等

**2.B**

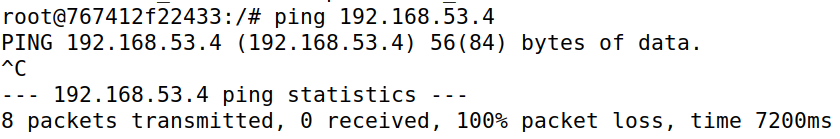
输入以下命令后，输入ip addr查看接口信息

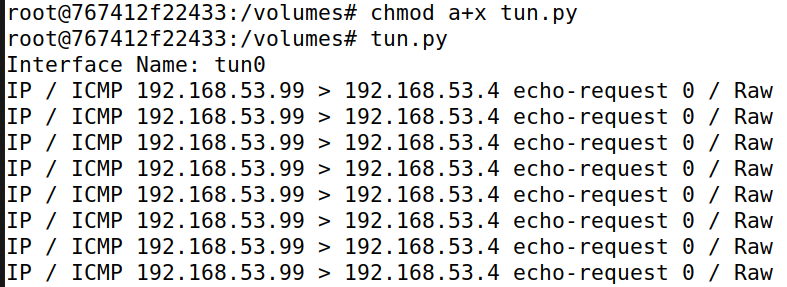




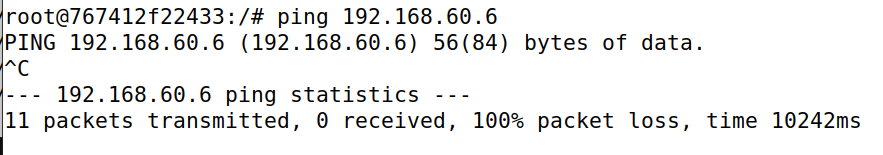
**2.C**

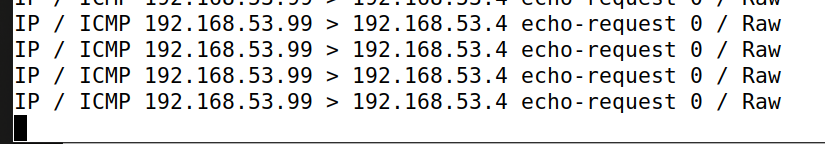
当HOST U ping 192.168.53.0/24时，如下图所示，tun.py打印出echo-request信息，因为去往53网段的流量经过tun0接口可以被捕捉到





当HOST U ping 192.168.60.0/24时，如下图所示，tun.py没有打印出任何信息，因为该流量并不经过tun0接口





**2.D**

将所给代码相关部分修改成如下所示

while True:

# Get a packet from the tun interface

packet = os.read(tun, 2048)

if True:

pkt = IP(packet)

print(pkt.summary())

if ICMP in pkt :

newip = IP(src=pkt[IP].dst,dst=pkt[IP].src, ihl=pkt[IP].ihl)

newip.ttl = 99

newicmp = ICMP(type = 0, id = pkt[ICMP].id,seq = pkt[ICMP].seq)

if pkt.haslayer(Raw) :

data = pkt[Raw].load

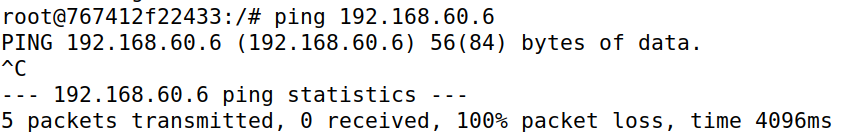
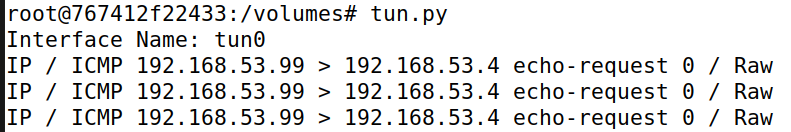
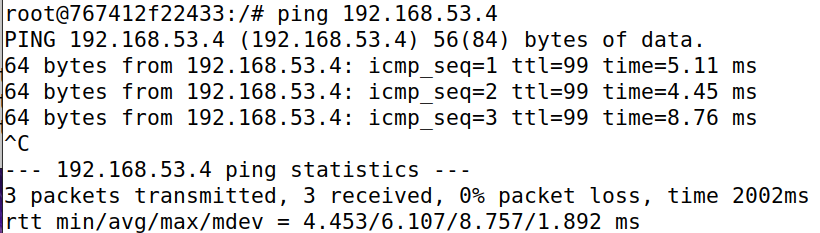
newpkt = newip/newicmp/data

else:

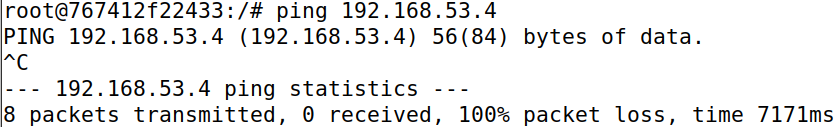
newpkt = newip/newicmp

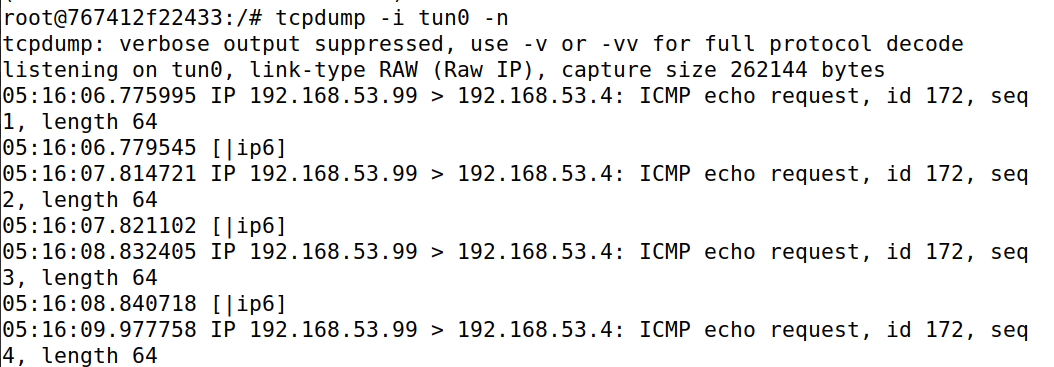
os.write(tun,bytes(newpkt))

运行后在用户主机上ping192.168.53.4(不存在地址)，结果如下，成功响应

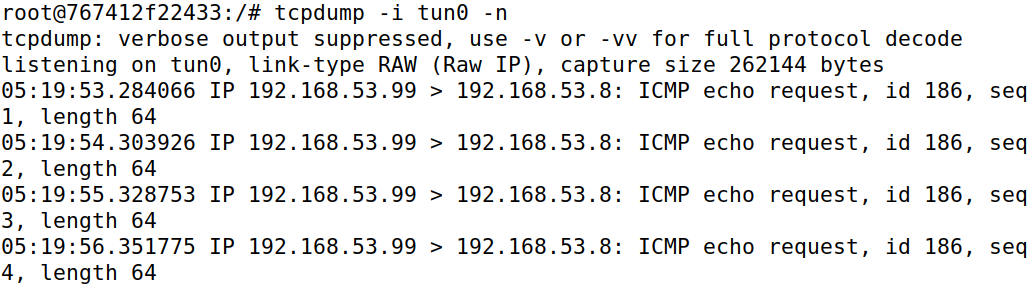
如果ping 60段的地址，仍旧无法ping通

倘若向接口随意写入字符串，结果如下



除了正常报文外还监听到了奇怪数据 [|ip6]，测试确认应为写入数据的变形

注释掉写入接口后监听结果如下



**TASK 3**

将所给代码修改成如下所示

**tun3.py(Client上的程序)**

os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

SERVER\_IP = "10.9.0.11"

SERVER\_PORT = 9090

while True:

# Get a packet from the tun interface

packet = os.read(tun, 2048)

if packet:

pkt = IP(packet)

print(pkt.summary())

sock.sendto(packet, (SERVER\_IP,SERVER\_PORT))

**编写Server上的监听程序**

**tun31.py（Server上的程序）**

#!/usr/bin/env python3

import fcntl

import struct

import os

import time

from scapy.all import \*

IP\_A = "0.0.0.0"

PORT = 9090

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

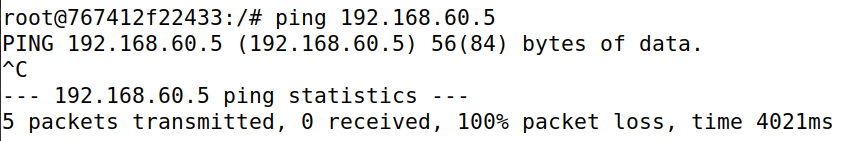
sock.bind((IP\_A, PORT))

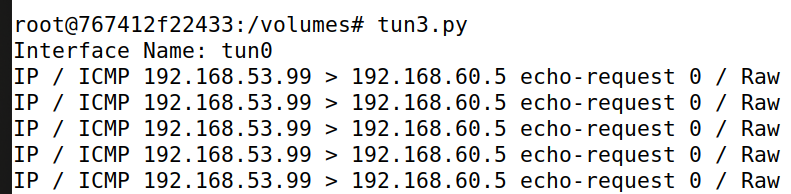
while True:

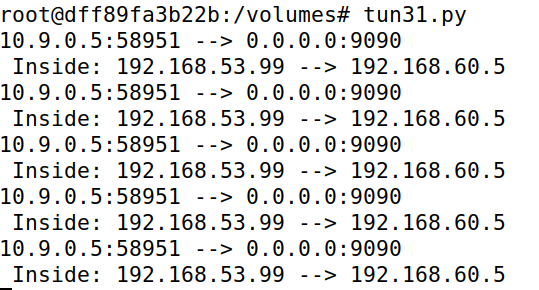
data, (ip, port) = sock.recvfrom(2048)

print("{}:{} --> {}:{}".format(ip, port, IP\_A, PORT))

pkt = IP(data)

 print(" Inside: {} --> {}".format(pkt.src, pkt.dst))

Client上截取到的报文数据如下

Server上监听到的数据如下

**TASK 4**

开启转发，重复TASK 3

在tun31.py中添加下述代码

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

# Create the tun interface

tun = os.open("/dev/net/tun", os.O\_RDWR)

ifr = struct.pack('16sH', b'tun%d', IFF\_TUN | IFF\_NO\_PI)

ifname\_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name

ifname = ifname\_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))

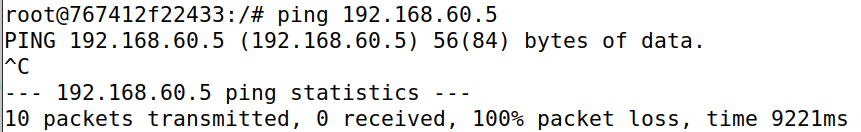
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))

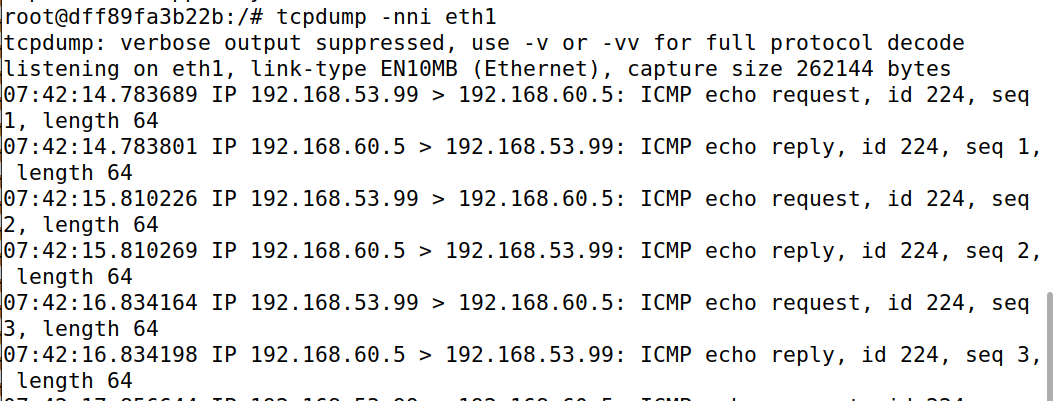
os.system("ip link set dev {} up".format(ifname))

在while True:循环语句中添加下述内容

os.write(tun,data)

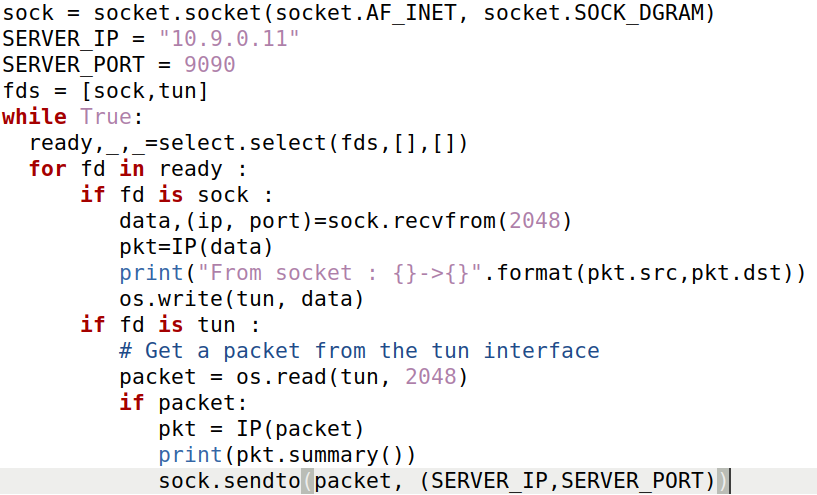
在Server上开启tcpdump截获报文查看，结果如下，HOST V进行了响应，但报文未被转发故HOST U端未收到响应报文



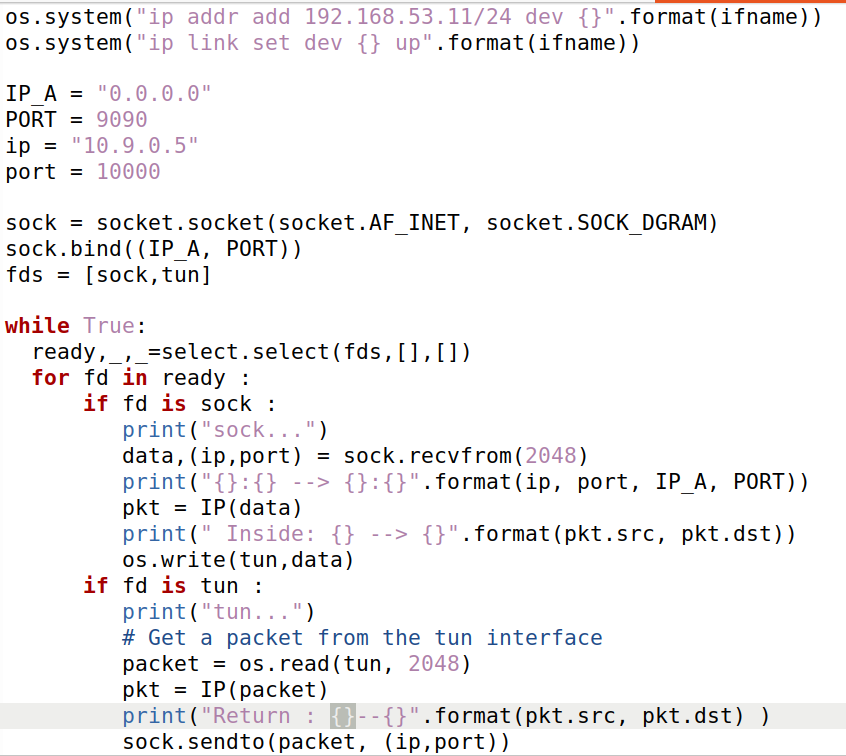


**TASK 5**

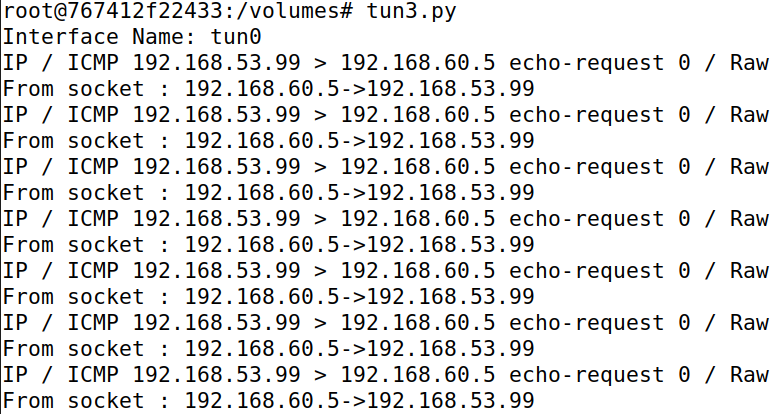
将tun3.py的结尾部分修改成如下所示

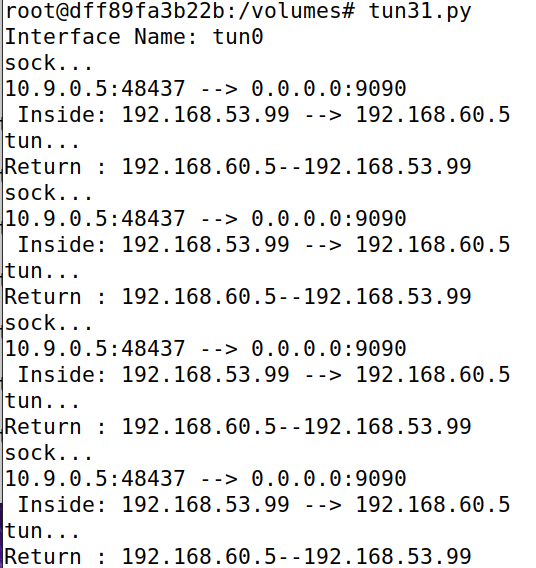


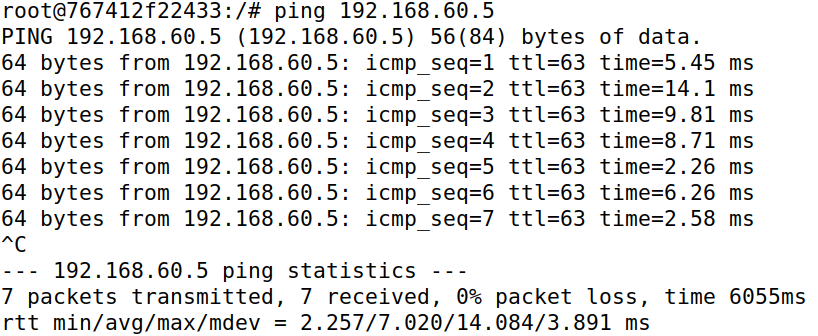
tun31.py如下

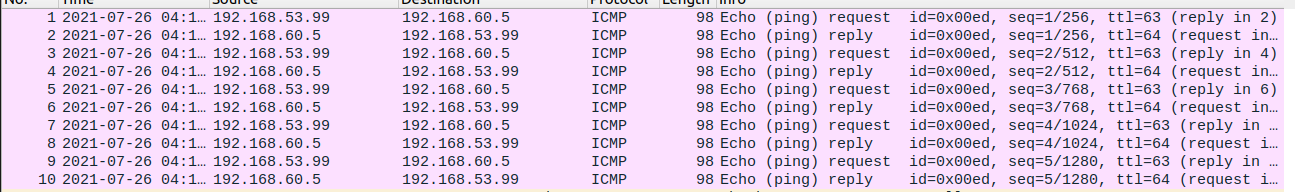


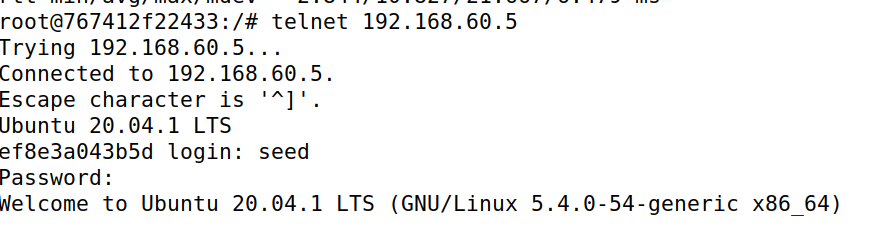
分别在用户主机和server上运行程序，在用户主机上ping内网主机，结果如下

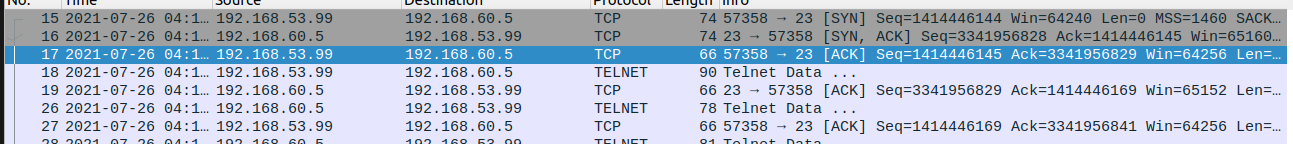






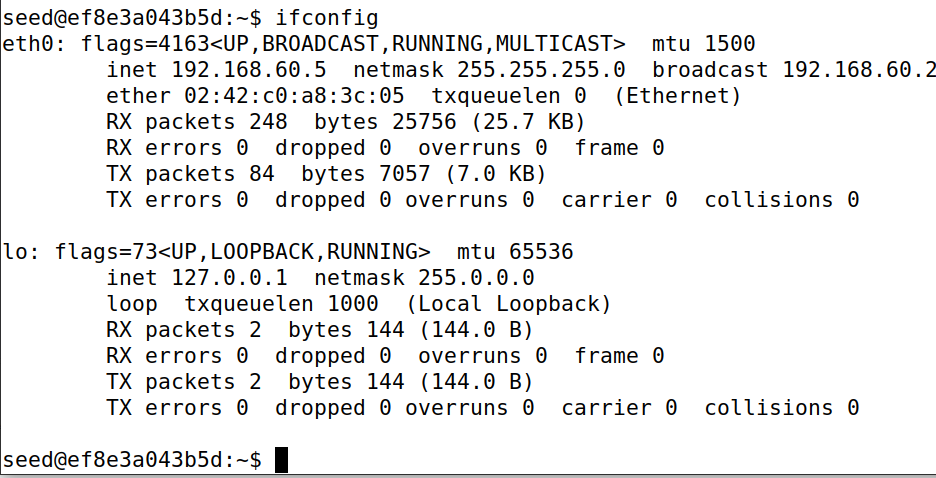
成功ping通内网主机，wireshark流量如下

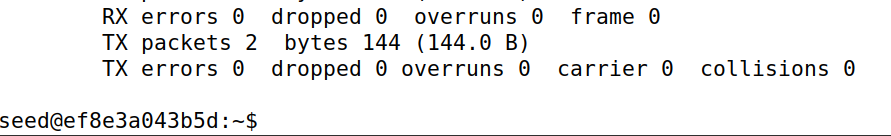
进行telnet尝试，同样成功



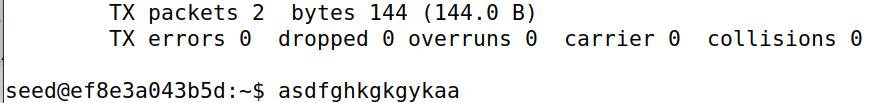
**TASK 6**

首先测试telnet连接

中止一方VPN程序，尝试输入，发现没有回显



重新连接后等待一下显示出来



在中止程序的一定时间内，telnet连接仍旧保留，客户端输入的数据一直在缓冲状态故没有显示，当恢复连接后，数据成功发送出去并收到回应显示出来