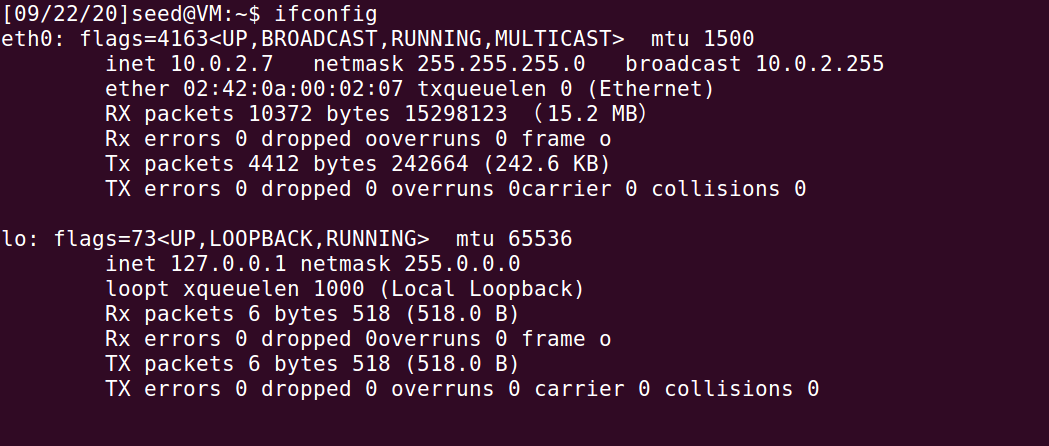
Lab7-report

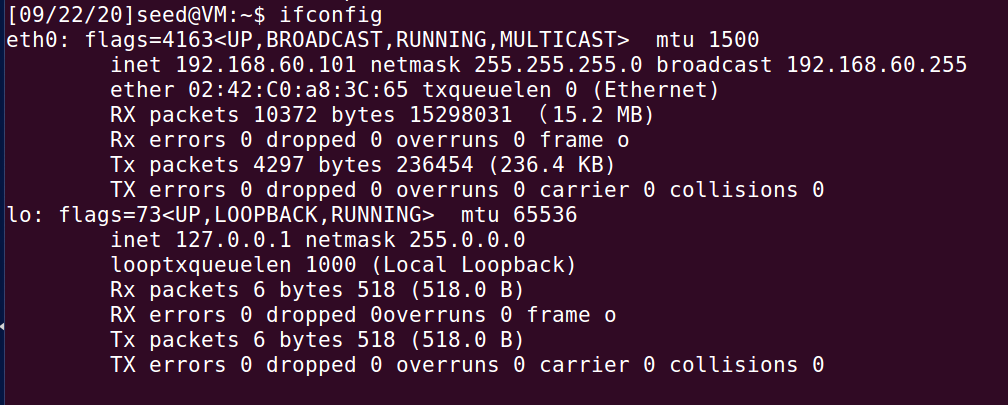
TASK1

创建主机U、VPN服务器、主机V

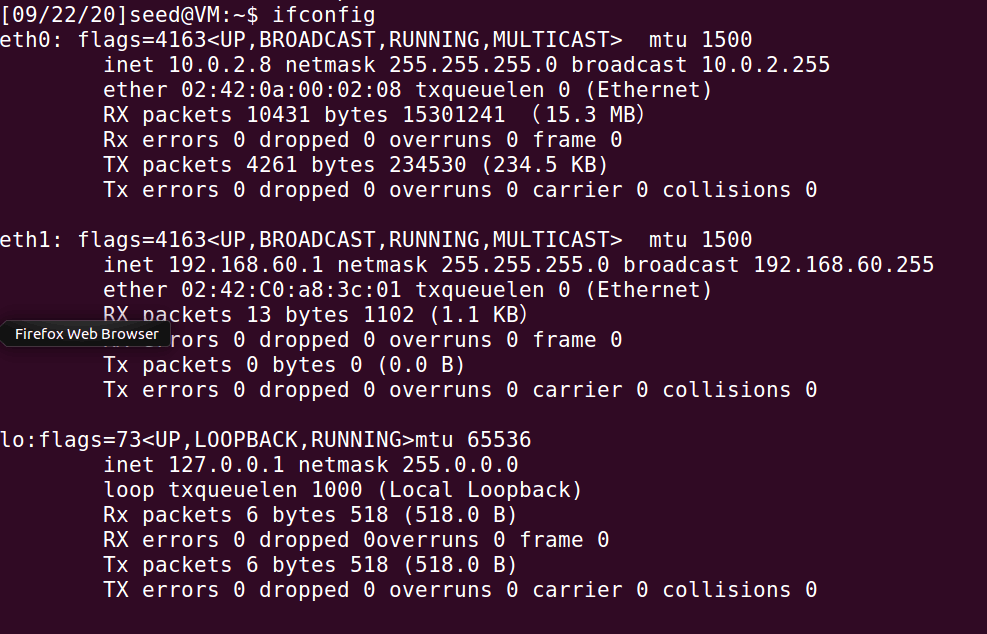
主机U



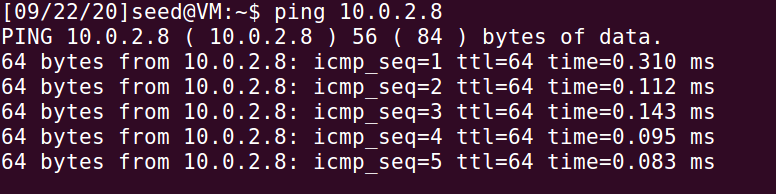
主机V



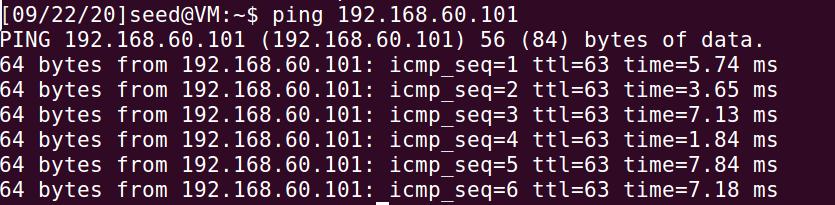
VPN服务器



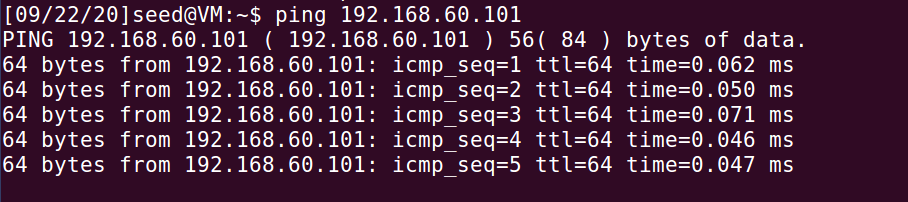
主机U连接VPN服务器



VPN服务器连接主机V

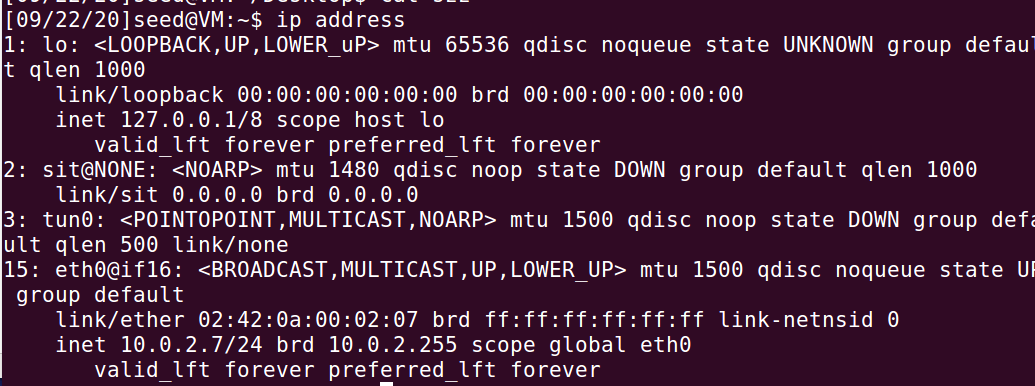


主机U连接主机V



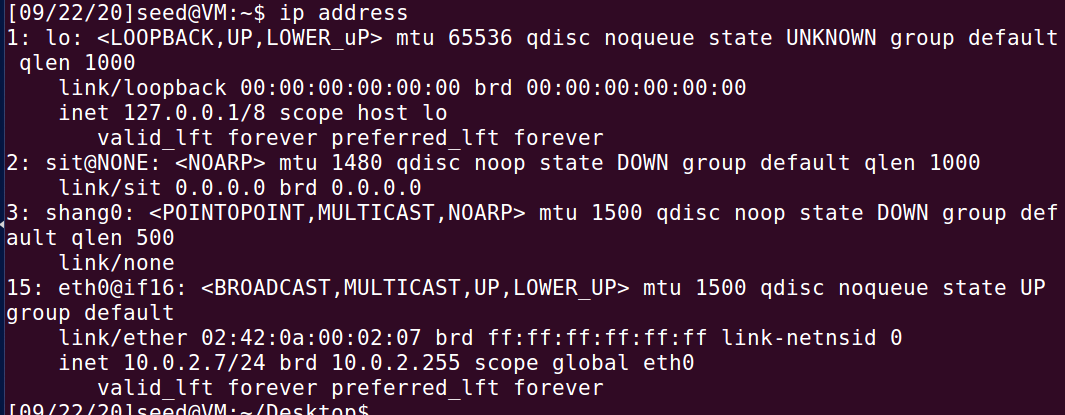
Task 2

查看所有⽹⼝信息

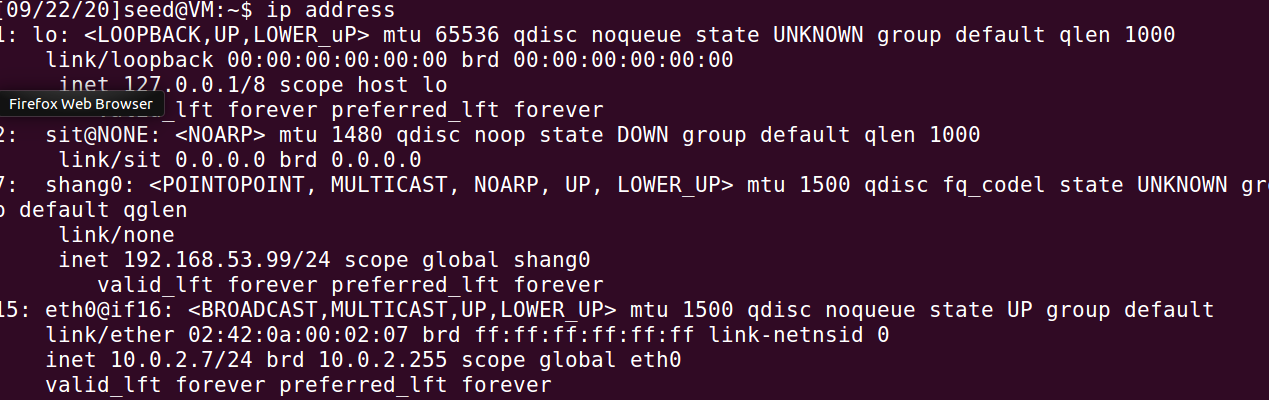


为了将tun0改为shang0 ，将代码中初始化ifr 的代码改即可。

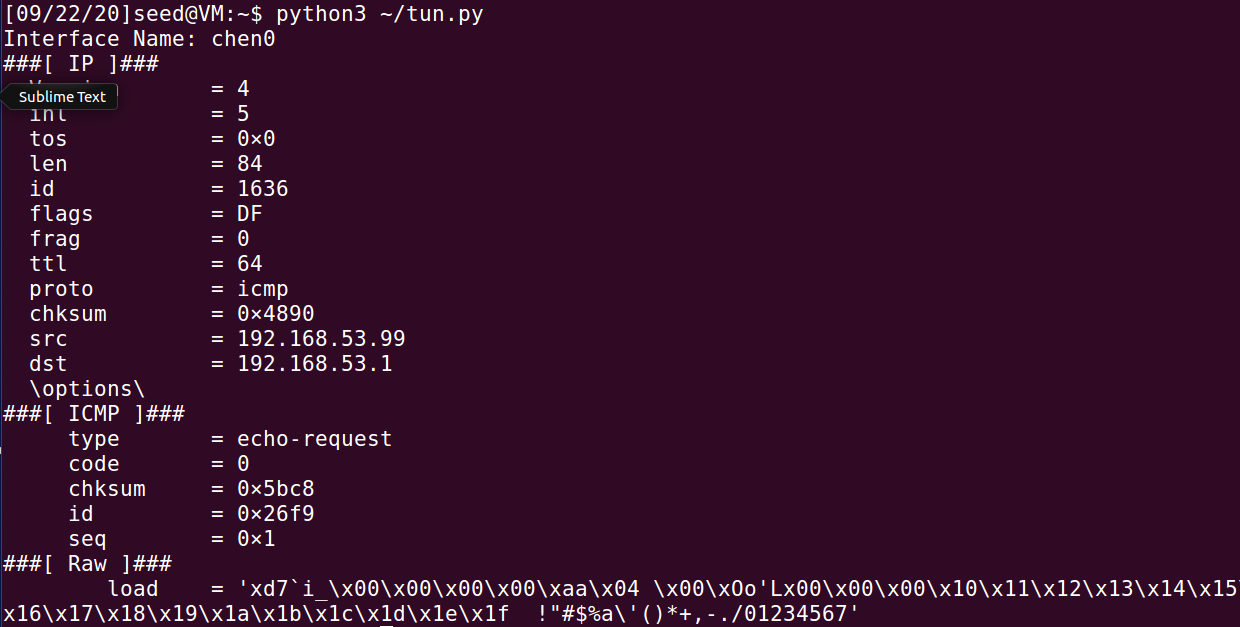
重新运⾏后查看：



按照题⽬中代码编写后，运⾏结果为

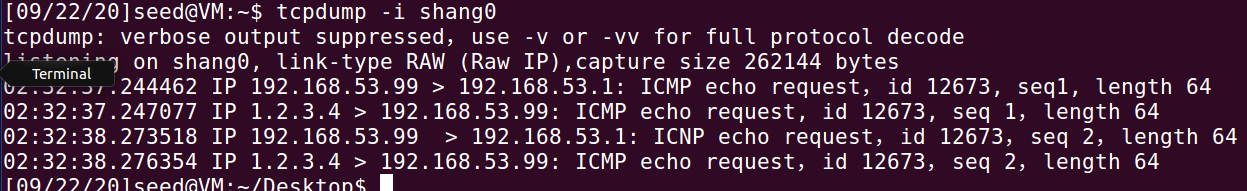


按照题⽬要求修改代码后运⾏，然后在主机U的另⼀个shell中输出为

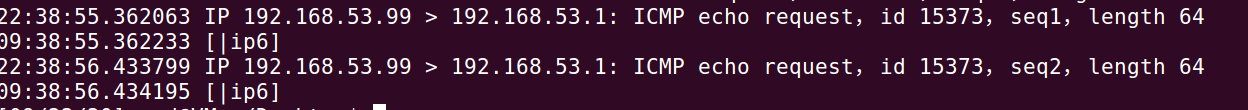


确实查看到了ICMP报⽂。

按照题⽬要求修改代码后运⾏，使⽤tcpdump 查看：

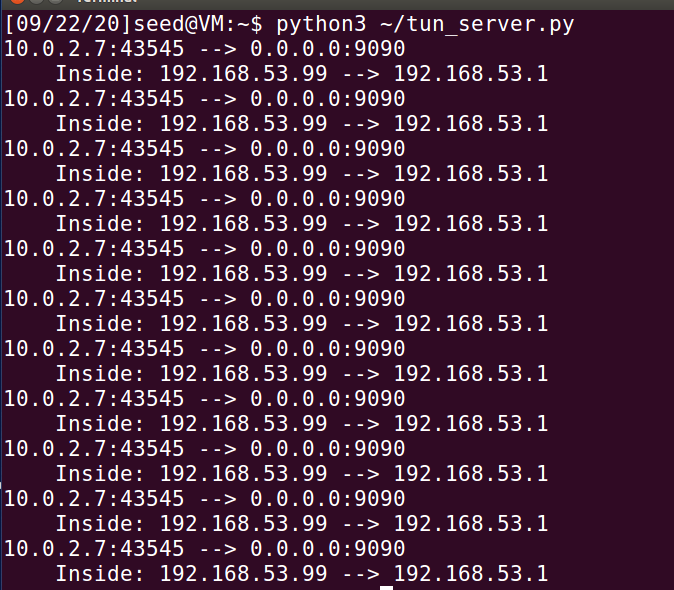


把代码修改成os.write(tun, b'evian')



Task 3

根据题⽬要求编写代码并运⾏后，在主机U中ping 192.168.53.1



Task 4

import fcntl

import struct import os import time

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002 IFF\_NO\_PI = 0x1000

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)

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

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

os.system("ip addr add 192.168.53.98/24 dev {}".format(ifname))

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

IP\_A = "0.0.0.0"

PORT = 9090

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

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

while True:

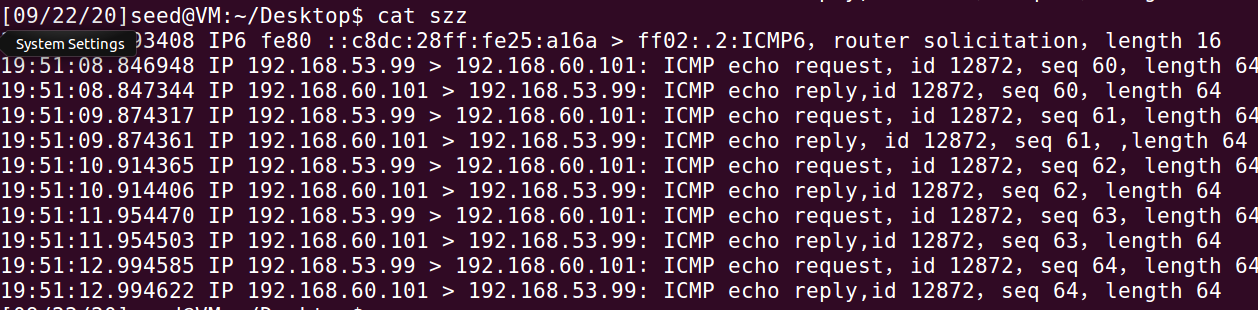
data, (ip, port) = sock.recvfrom(2048)

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

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

os.write(tun, data)

运⾏后，在主机U中ping 主机V： 192.168.60.101 ，在主机V中可以通过tcpdump 查看到



Task 5

import fcntl

import struct

import os

import time

import select

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

SERVER\_IP = "10.0.2.8"

SERVER\_PORT = 9090

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

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

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

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

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

os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))

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

os.system("route add -net 192.168.60.0/24 {}".format(ifname))

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

while True:

ready, \_, \_ = select.select([sock, tun], [], [])

for fd in ready:

if fd is sock:

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

pkt = IP(data)

print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))

os.write(tun, data)

if fd is tun:

packet = os.read(tun, 2048)

pkt = IP(packet)

print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

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

tun\_server.py :

import fcntl

import struct

import os

import time

import select

from scapy.all import \*

TUNSETIFF = 0x400454ca

IFF\_TUN = 0x0001

IFF\_TAP = 0x0002

IFF\_NO\_PI = 0x1000

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)

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

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

os.system("ip addr add 192.168.53.98/24 dev {}".format(ifname))

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

IP\_A = "0.0.0.0"

PORT = 9090

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

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

port

while True:

ready, \_, \_ = select.select([sock, tun], [], [])

for fd in ready:

if fd is sock:

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

pkt = IP(data)

print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))

os.write(tun, data)

if fd is tun:

packet = os.read(tun, 2048)

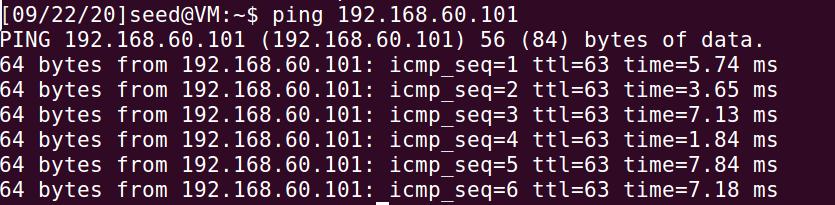
pkt = IP(packet)

print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))

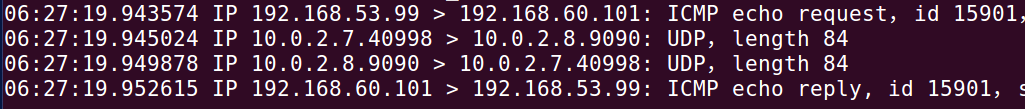
sock.sendto(packet, ("10.0.2.7", port))

然后在主机V中添加路由表项：ip route add 192.168.53.0/24 gw 192.168.60.1 eth0

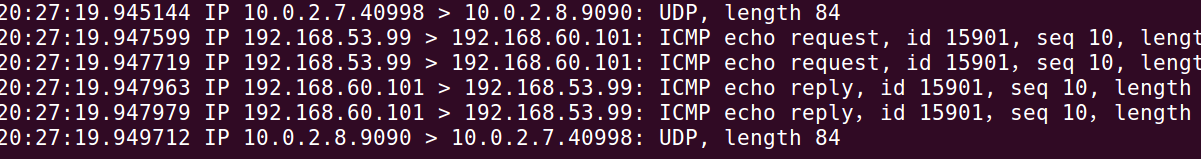
使ICMP响应报⽂都前往VPN服务器。设置完成后，在主机U上再次 ping 主机V：



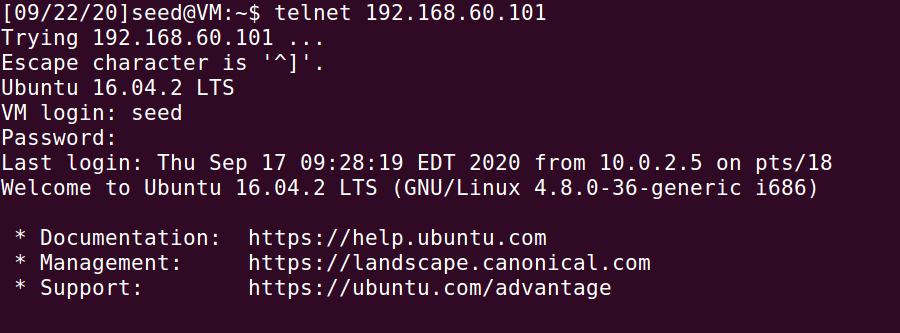
此时，主机U上的 tcpdump 为



VPN服务器上的 tcpdump 为



看到，收到了主机U封装成UDP报⽂的ICMP请求后，tun\_server.py 解析出ICMP请求，发送给主机V，主机V根据路由表将ICMP响应发送给VPN服务器，VPN服务器将ICMP响应封装在UDP报⽂中再发回给主机U。



Task 6

在主机U与主机V连接之后，关闭 tun\_client.py ，发现⽆论输⼊什么，主机U的Telnet都没有显示。但TCP连接没有断。再次运⾏ tun\_client.py ，输⼊字符，等待⼀会⼉，发现之前输⼊的字符重新出现在Telnet界⾯中。

当关闭 tun\_client.py 后，之前建⽴的TCP连接会将内容缓存进缓冲区进⼊重连状态。如果恢复 tun\_client.py ，缓冲区的字符会重新通过TCP连接发送。

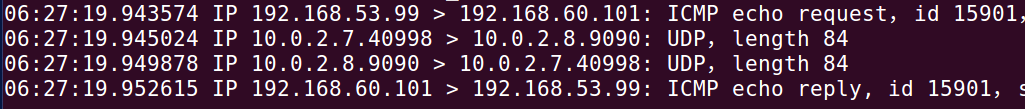
Task 7

使用ip route add 192.168.53.0/24 gw 192.168.60.1 eth0其结果是正常运⾏

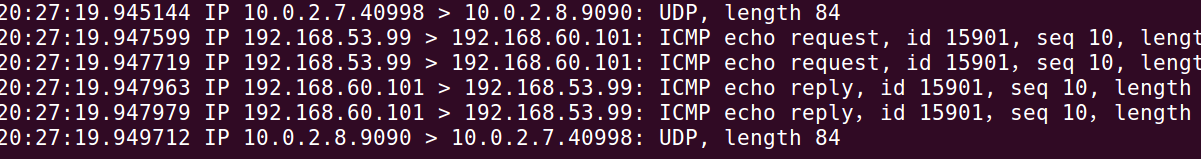
Task 8

将主机U上的TUN⽹⼝IP地址改为 192.168.30.99 后，⽆法 ping 通。

查看主机U上的 tcpdump 结果：

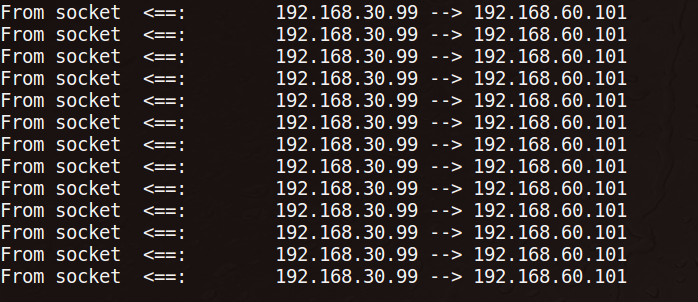


在VPN服务器上查看 tcpdump 结果



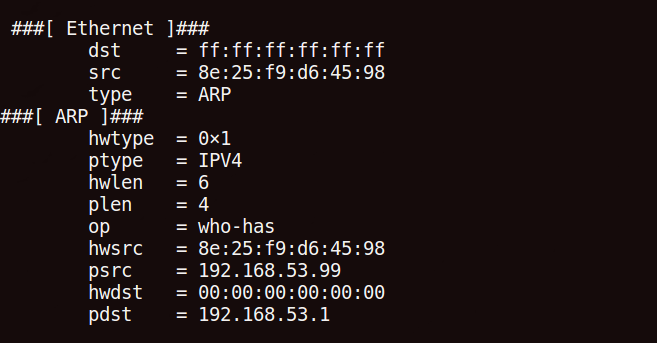
收到了UDP报⽂，并且将其发送给了主机V，也收到了主机V的ICMP响应报⽂，但并没有发出返回的UDP报⽂

查看 tun\_server.py 的输出可以发现



Task9

编写代码运⾏后，并在主机U上ping 192.168.53.1查看Python脚本输出



发现TAP⼝收到⼀个ARP请求