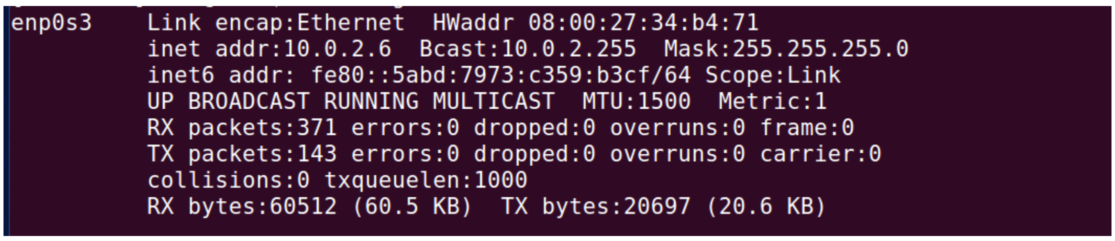
网安综合课程设计实验报告 7

VPN Tunneling Lab

Task 1: Network Setup

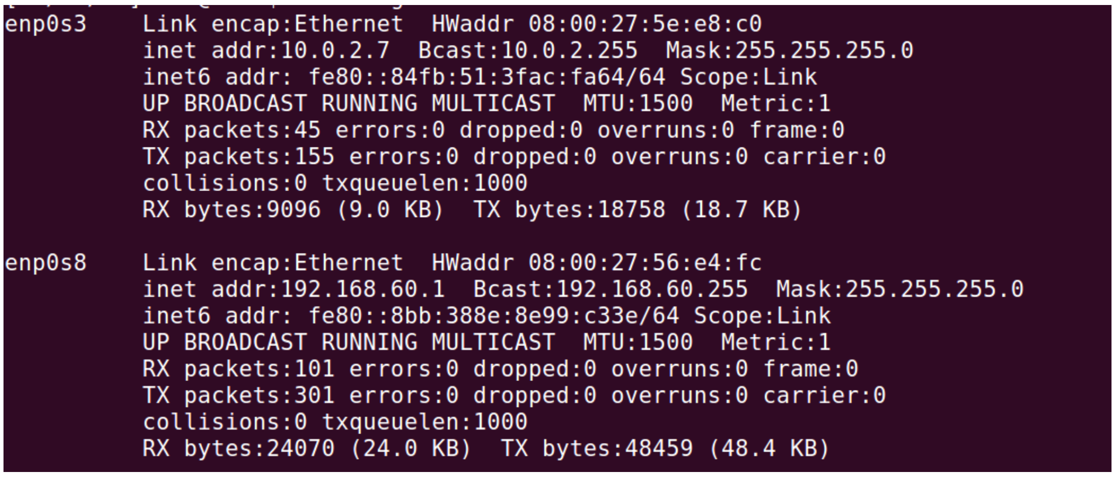
首先在 VPN 服务器增设一个网卡，设置为内网模式，将主机 V 加入到这一内网中，之后设置内网中这两个设备的 IP 地址，最终的结果如下：

主机 U：



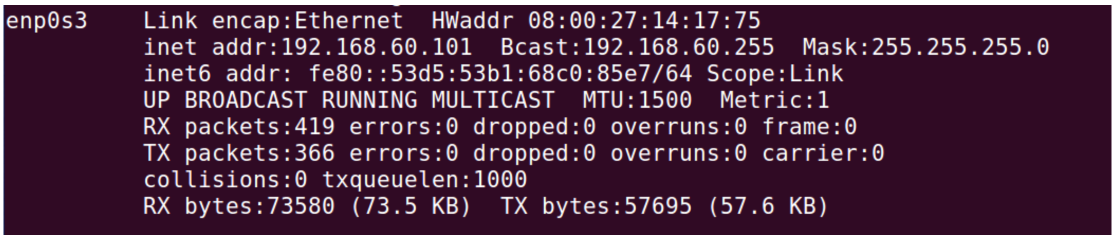
```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:34:b4:71
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5abd:7973:c359:b3cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:371 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:60512 (60.5 KB)  TX bytes:20697 (20.6 KB)
```

VPN 服务器：



```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:5e:e8:c0
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::84fb:51:3fac:fa64/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9096 (9.0 KB)  TX bytes:18758 (18.7 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:56:e4:fc
          inet addr:192.168.60.1  Bcast:192.168.60.255  Mask:255.255.255.0
          inet6 addr: fe80::8bb:388e:8e99:c33e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:0 dropped:0 overruns:0 frame:0
          TX packets:301 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24070 (24.0 KB)  TX bytes:48459 (48.4 KB)
```

主机 V：



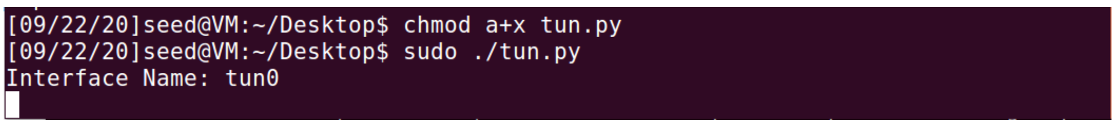
```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:14:17:75
          inet addr:192.168.60.101  Bcast:192.168.60.255  Mask:255.255.255.0
          inet6 addr: fe80::53d5:53b1:68c0:85e7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:419 errors:0 dropped:0 overruns:0 frame:0
          TX packets:366 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:73580 (73.5 KB)  TX bytes:57695 (57.6 KB)
```

其中，U 可以通过 NAT ping 通 VPN 服务器，无法 ping 主机 V。主机 V 可以 pingVPN 服务器。

Task2　Create and Configure TUN Interface

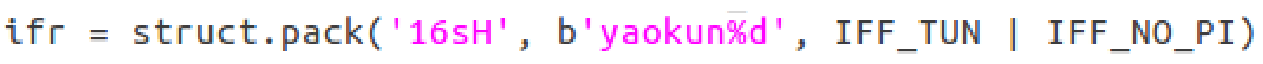
1> Name of the interface

编写 python 程序运行如下：



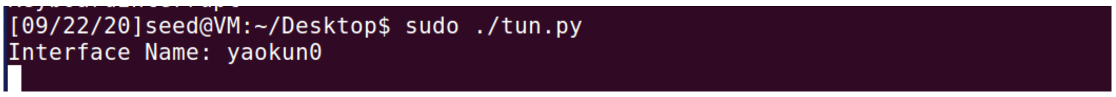
```
[09/22/20]seed@VM:~/Desktop$ chmod a+x tun.py
[09/22/20]seed@VM:~/Desktop$ sudo ./tun.py
Interface Name: tun0
```

另起终端发现增加了 tun0
修改代码为:

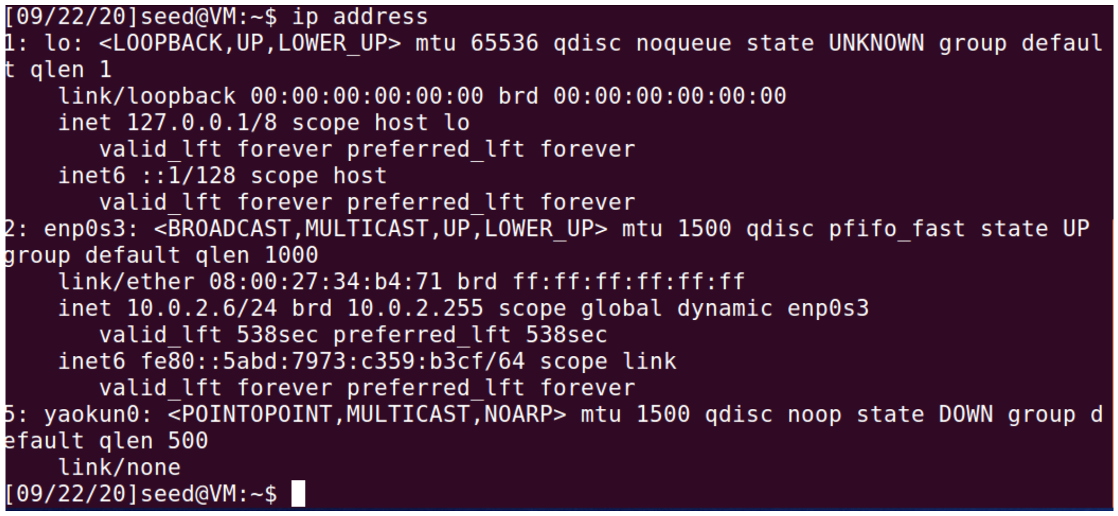


```
ifr = struct.pack('16sH', b'yaokun%d', IFF_TUN | IFF_NO_PI)
```

重新启动:



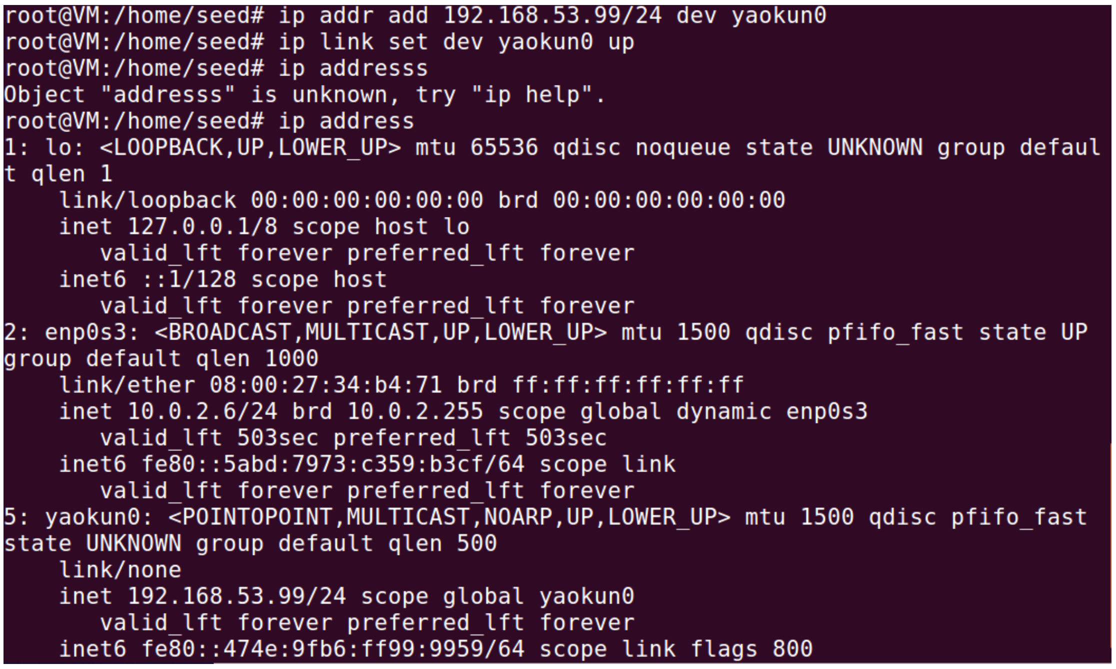
```
[09/22/20]seed@VM:~/Desktop$ sudo ./tun.py
Interface Name: yaokun0
```



```
[09/22/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:34:b4:71 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.6/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 538sec preferred_lft 538sec
    inet6 fe80::5abd:7973:c359:b3cf/64 scope link
       valid_lft forever preferred_lft forever
5: yaokun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group d
efault qlen 500
    link/none
[09/22/20]seed@VM:~$
```

改名成功。

2> Set up the TUN Interface

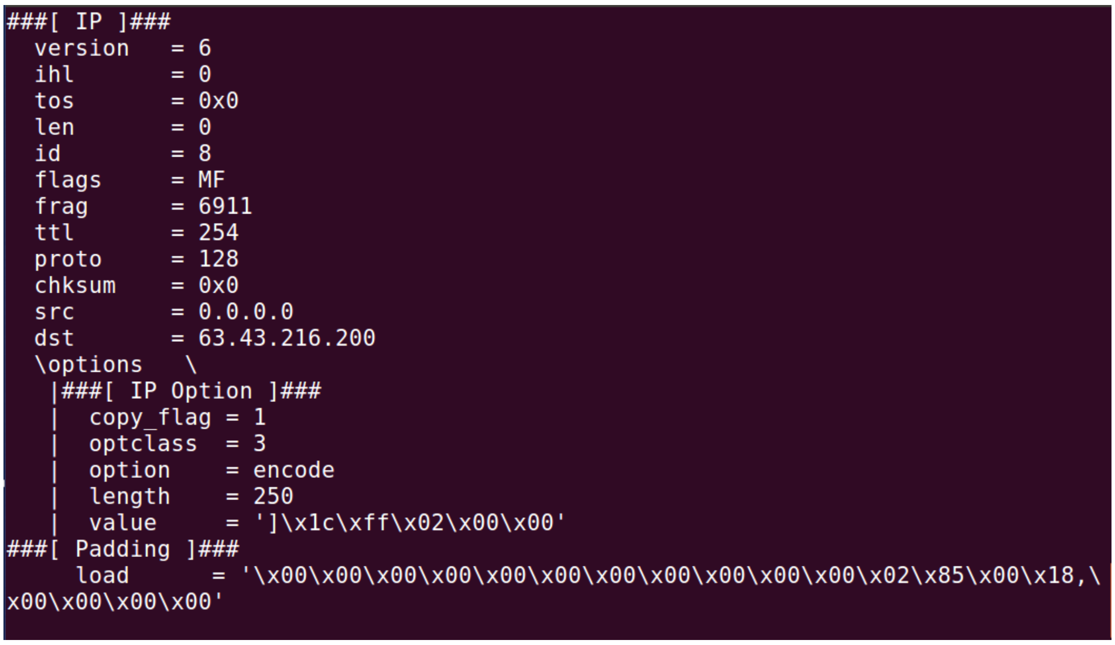


```
root@VM:/home/seed# ip addr add 192.168.53.99/24 dev yaokun0
root@VM:/home/seed# ip link set dev yaokun0 up
root@VM:/home/seed# ip addresss
Object "addresss" is unknown, try "ip help".
root@VM:/home/seed# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:34:b4:71 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.6/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 503sec preferred_lft 503sec
    inet6 fe80::5abd:7973:c359:b3cf/64 scope link
       valid_lft forever preferred_lft forever
5: yaokun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global yaokun0
       valid_lft forever preferred_lft forever
    inet6 fe80::474e:9fb6:ff99:9959/64 scope link flags 800
```

或者可以在程序中加入，使其自动化实现。

3> Read from the TUN Interface

改写代码后能够输出 IP 报文:



```
###[ IP ]###
  version   = 6
  ihl       = 0
  tos       = 0x0
  len       = 0
  id        = 8
  flags     = MF
  frag      = 6911
  ttl       = 254
  proto     = 128
  chksum    = 0x0
  src       = 0.0.0.0
  dst       = 63.43.216.200
  \options   \
   |###[ IP Option ]###
   |  copy_flag = 1
   |  optclass  = 3
   |  option    = encode
   |  length    = 250
   |  value     = ']\x1c\xff\x02\x00\x00'
###[ Padding ]###
     load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x85\x00\x18,\
x00\x00\x00\x00'
```

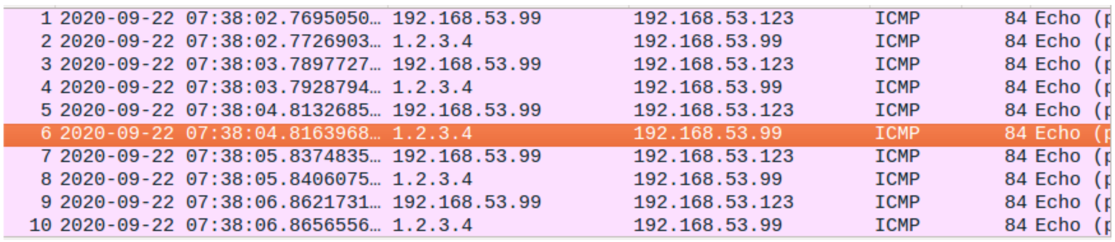
Ping 目的地址发现无正常反应，因为无法解析目的地址

4> Write to the TUN Interface

程序修改如下:

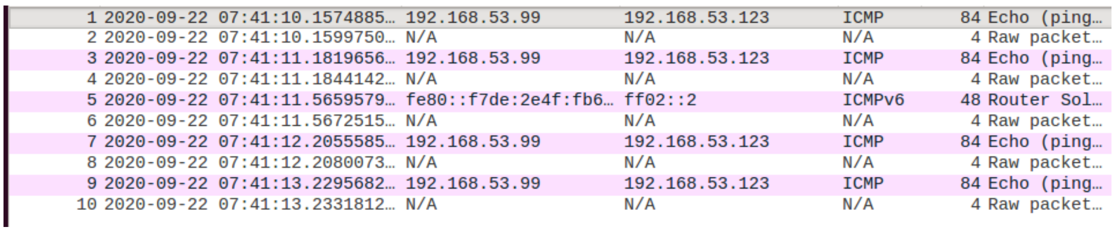
```
while True:
    packet=os.read(tun,2048)
    if True:
        ip=IP(packet)
        ip.show()

        newip=IP(src='1.2.3.4',dst=ip.src)
        newpkt=newip/ip.payload
        os.write(tun,bytes(newpkt))
```

继续 ping 目的地地，截取的报文如下:



```
 1 2020-09-22 07:38:02.7695050… 192.168.53.99    192.168.53.123   ICMP   84 Echo (
 2 2020-09-22 07:38:02.7726903… 1.2.3.4          192.168.53.99    ICMP   84 Echo (
 3 2020-09-22 07:38:03.7897727… 192.168.53.99    192.168.53.123   ICMP   84 Echo (
 4 2020-09-22 07:38:03.7928794… 1.2.3.4          192.168.53.99    ICMP   84 Echo (
 5 2020-09-22 07:38:04.8132685… 192.168.53.99    192.168.53.123   ICMP   84 Echo (
 6 2020-09-22 07:38:04.8163968… 1.2.3.4          192.168.53.99    ICMP   84 Echo (
 7 2020-09-22 07:38:05.8374835… 192.168.53.99    192.168.53.123   ICMP   84 Echo (
 8 2020-09-22 07:38:05.8406075… 1.2.3.4          192.168.53.99    ICMP   84 Echo (
 9 2020-09-22 07:38:06.8621731… 192.168.53.99    192.168.53.123   ICMP   84 Echo (
10 2020-09-22 07:38:06.8656556… 1.2.3.4          192.168.53.99    ICMP   84 Echo (
```

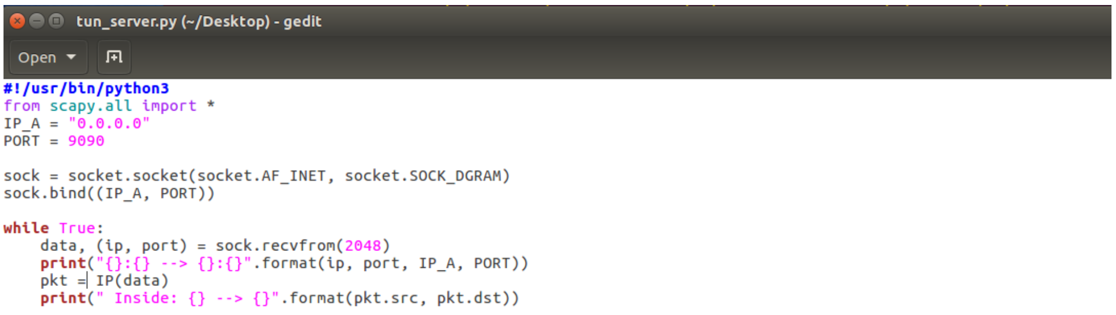
当发送无意义报文时，获得的结果:



```
  1 2020-09-22 07:41:10.1574885… 192.168.53.99           192.168.53.123          ICMP         84 Echo (ping…
  2 2020-09-22 07:41:10.1599750… N/A                     N/A                     N/A           4 Raw packet…
  3 2020-09-22 07:41:11.1819656… 192.168.53.99           192.168.53.123          ICMP         84 Echo (ping…
  4 2020-09-22 07:41:11.1844142… N/A                     N/A                     N/A           4 Raw packet…
  5 2020-09-22 07:41:11.5659579… fe80::f7de:2e4f:fb6… ff02::2                 ICMPv6       48 Router Sol…
  6 2020-09-22 07:41:11.5672515… N/A                     N/A                     N/A           4 Raw packet…
  7 2020-09-22 07:41:12.2055585… 192.168.53.99           192.168.53.123          ICMP         84 Echo (ping…
  8 2020-09-22 07:41:12.2080073… N/A                     N/A                     N/A           4 Raw packet…
  9 2020-09-22 07:41:13.2295682… 192.168.53.99           192.168.53.123          ICMP         84 Echo (ping…
 10 2020-09-22 07:41:13.2331812… N/A                     N/A                     N/A           4 Raw packet…
```

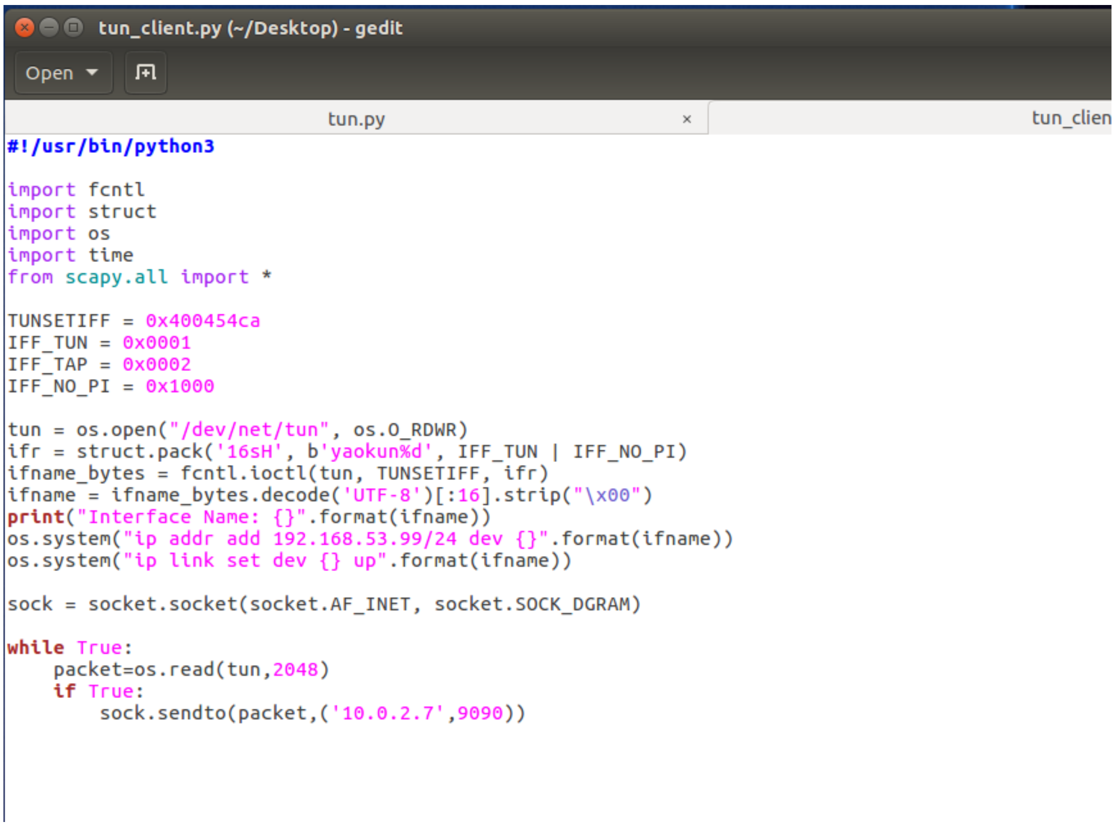
Task 3: send the IP packets to VPN server throught a tunnel

服务器端的代码如下：



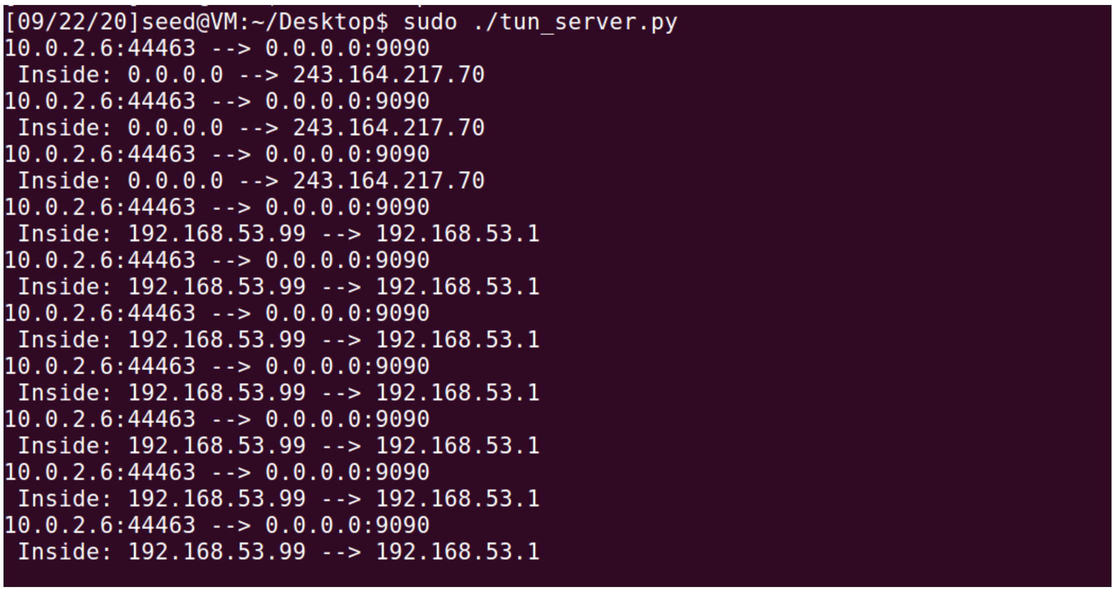
```
#!/usr/bin/python3
from scapy.all import *
IP_A = "0.0.0.0"
PORT = 9090

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))
```

客户端的代码如下：



```
#!/usr/bin/python3

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'yaokun%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))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    packet=os.read(tun,2048)
    if True:
        sock.sendto(packet,('10.0.2.7',9090))
```

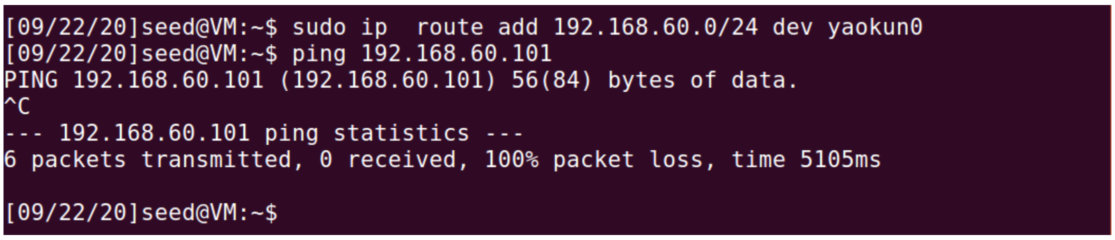
分别在服务器与客户端运行，并在客户端 ping 目的地址，在服务端的信息为：



```
[09/22/20]seed@VM:~/Desktop$ sudo ./tun_server.py
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 243.164.217.70
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 243.164.217.70
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 0.0.0.0 --> 243.164.217.70
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
```

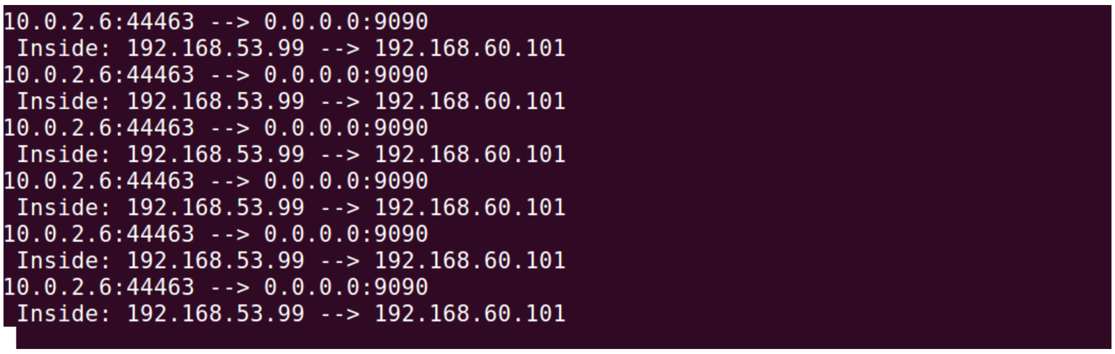
可以成功实现了隧道连接。

当 ping 主机 V 时无响应，做出如下的修改：



```
[09/22/20]seed@VM:~$ sudo ip  route add 192.168.60.0/24 dev yaokun0
[09/22/20]seed@VM:~$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
^C
--- 192.168.60.101 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5105ms

[09/22/20]seed@VM:~$
```

此时再 ping，服务器收到的内容：

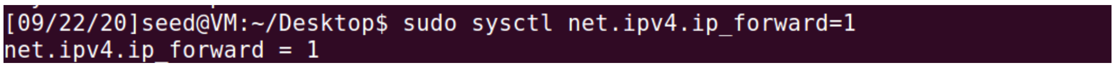


```
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.6:44463 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
```

发现成功实现了。

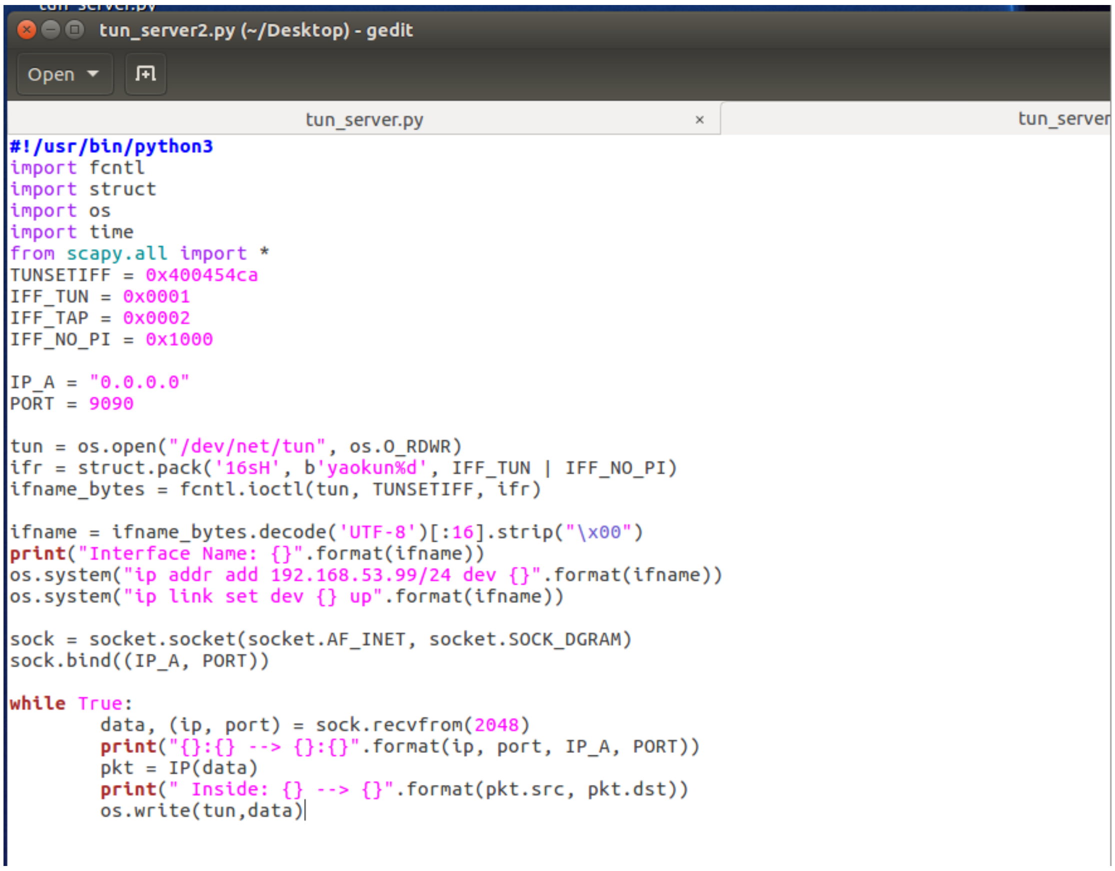
Task 4: set up the VPN server

首先开启端口转发：



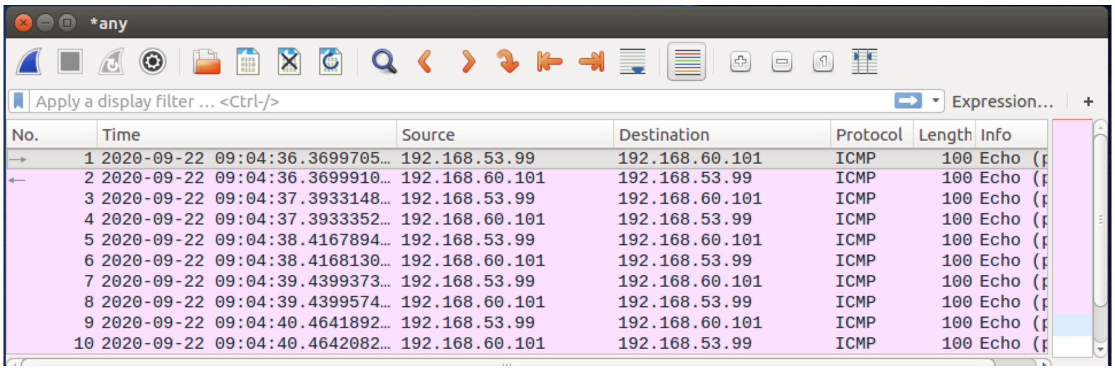
```
[09/22/20]seed@VM:~/Desktop$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

编写程序如下：



```
#!/usr/bin/python3
import fcntl
import struct
import os
import time
from scapy.all import *
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

IP_A = "0.0.0.0"
PORT = 9090

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'yaokun%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))

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))
        os.write(tun,data)
```

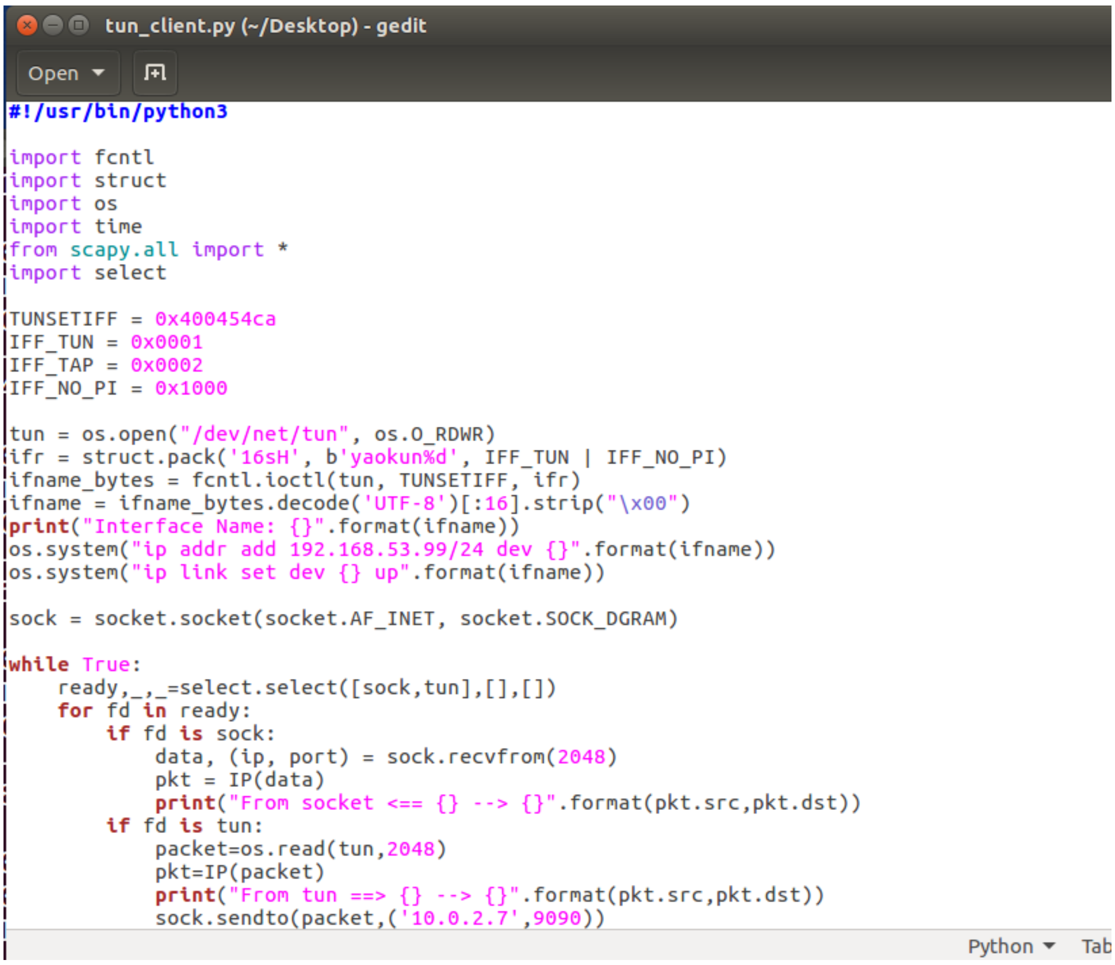
运行并在客户端 ping 主机 V，可以在主机 V 中抓到相关的包信息：



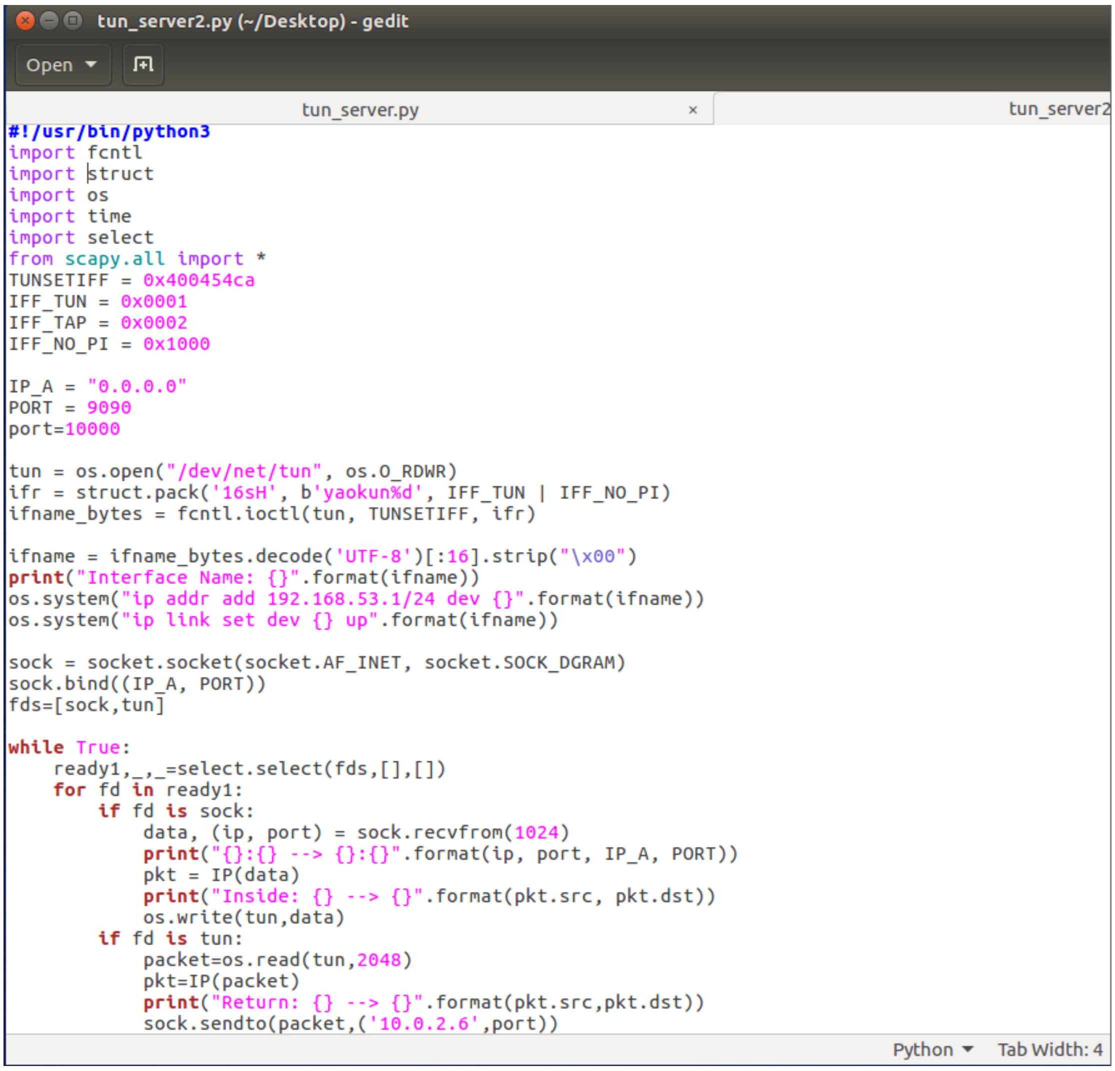
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-22 09:04:36.3699705…	192.168.53.99	192.168.60.101	ICMP	100	Echo (
2	2020-09-22 09:04:36.3699910…	192.168.60.101	192.168.53.99	ICMP	100	Echo (
3	2020-09-22 09:04:37.3933148…	192.168.53.99	192.168.60.101	ICMP	100	Echo (
4	2020-09-22 09:04:37.3933352…	192.168.60.101	192.168.53.99	ICMP	100	Echo (
5	2020-09-22 09:04:38.4167894…	192.168.53.99	192.168.60.101	ICMP	100	Echo (
6	2020-09-22 09:04:38.4168130…	192.168.60.101	192.168.53.99	ICMP	100	Echo (
7	2020-09-22 09:04:39.4399373…	192.168.53.99	192.168.60.101	ICMP	100	Echo (
8	2020-09-22 09:04:39.4399574…	192.168.60.101	192.168.53.99	ICMP	100	Echo (
9	2020-09-22 09:04:40.4641892…	192.168.53.99	192.168.60.101	ICMP	100	Echo (
10	2020-09-22 09:04:40.4642082…	192.168.60.101	192.168.53.99	ICMP	100	Echo (

Task 5: Handling Traffic in both directions

分别修改程序如下：

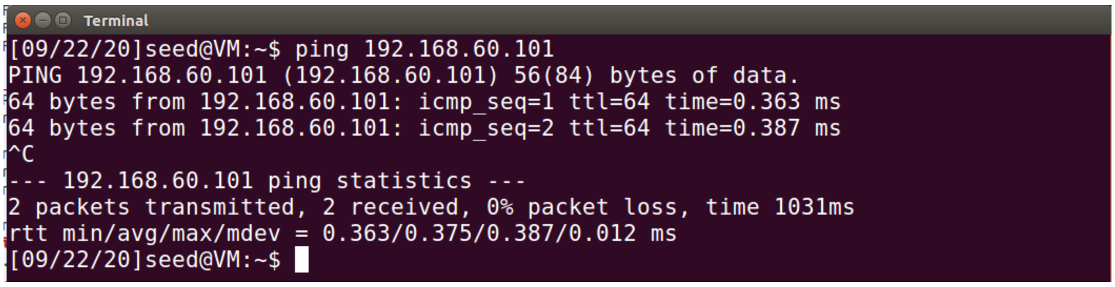


```
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *
import select

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'yaokun%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))

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))
        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',9090))
```



```
#!/usr/bin/python3
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

IP_A = "0.0.0.0"
PORT = 9090
port=10000

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'yaokun%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 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
fds=[sock,tun]

while True:
    ready1,_,_=select.select(fds,[],[])
    for fd in ready1:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(1024)
            print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
            pkt = IP(data)
            print("Inside: {} --> {}".format(pkt.src, pkt.dst))
            os.write(tun,data)
        if fd is tun:
            packet=os.read(tun,2048)
            pkt=IP(packet)
            print("Return: {} --> {}".format(pkt.src,pkt.dst))
            sock.sendto(packet,('10.0.2.6',port))
```

重新运行并 ping 目的主机，发现返回了报文：



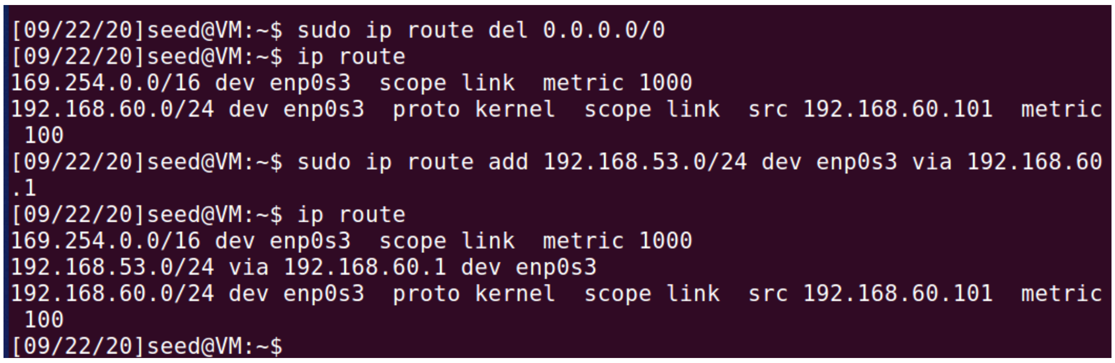
```
[09/22/20]seed@VM:~$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=64 time=0.363 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=64 time=0.387 ms
^C
--- 192.168.60.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.363/0.375/0.387/0.012 ms
[09/22/20]seed@VM:~$ 
```

Task 6: Tunnel -Breaking Experiment

当正在使用虚拟隧道进行连接的 U 与 V 通信时，断开 tun 后，连接也中断。但当连接恢复后，又能够继续 ping 通。

Task 7: Routing Experiment on Host V

操作如下：

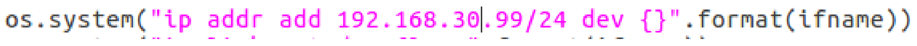


```
[09/22/20]seed@VM:~$ sudo ip route del 0.0.0.0/0
[09/22/20]seed@VM:~$ ip route
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.60.0/24 dev enp0s3  proto kernel  scope link  src 192.168.60.101  metric
 100
[09/22/20]seed@VM:~$ sudo ip route add 192.168.53.0/24 dev enp0s3 via 192.168.60
.1
[09/22/20]seed@VM:~$ ip route
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.53.0/24 via 192.168.60.1 dev enp0s3
192.168.60.0/24 dev enp0s3  proto kernel  scope link  src 192.168.60.101  metric
 100
[09/22/20]seed@VM:~$
```

可以在 U 中 ping 通 V

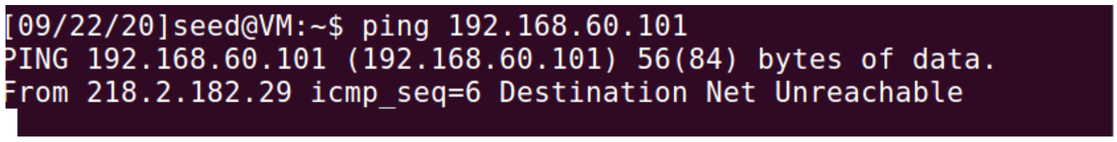
Task 8: Experiment with the TUN IP Address

在 U 中改变 IP：



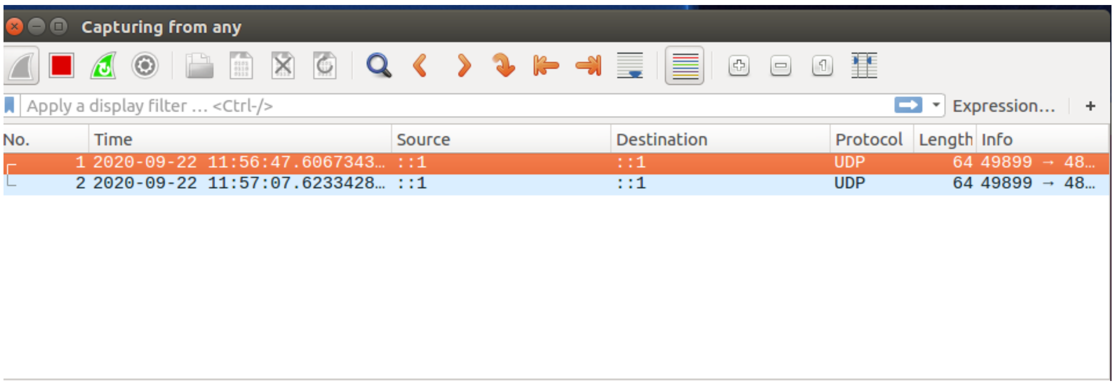
```
os.system("ip addr add 192.168.30.99/24 dev {}".format(ifname))
```

重新运行程序，并 ping：



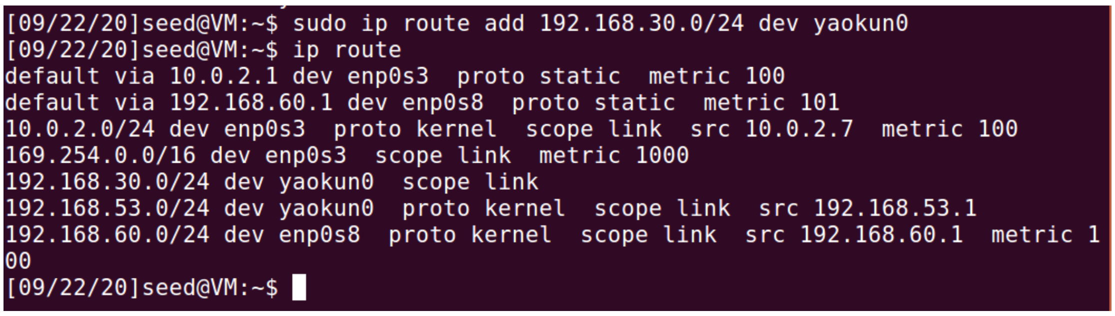
```
[09/22/20]seed@VM:~$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
From 218.2.182.29 icmp_seq=6 Destination Net Unreachable
```

在 V 中没有收到 ICMP 报文：



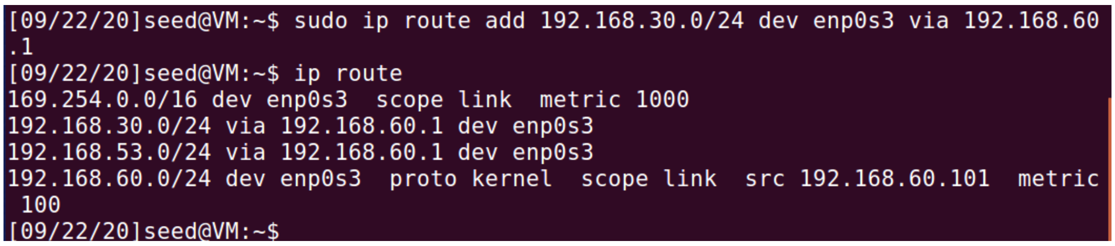
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-22 11:56:47.6067343…	::1	::1	UDP	64	49899 → 48…
2	2020-09-22 11:57:07.6233428…	::1	::1	UDP	64	49899 → 48…

手动添加反向路由，在 VPN server 中：



```
[09/22/20]seed@VM:~$ sudo ip route add 192.168.30.0/24 dev yaokun0
[09/22/20]seed@VM:~$ ip route
default via 10.0.2.1 dev enp0s3  proto static  metric 100
default via 192.168.60.1 dev enp0s8  proto static  metric 101
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.7  metric 100
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.30.0/24 dev yaokun0  scope link
192.168.53.0/24 dev yaokun0  proto kernel  scope link  src 192.168.53.1
192.168.60.0/24 dev enp0s8  proto kernel  scope link  src 192.168.60.1  metric 1
00
[09/22/20]seed@VM:~$ 
```

在 V 中：

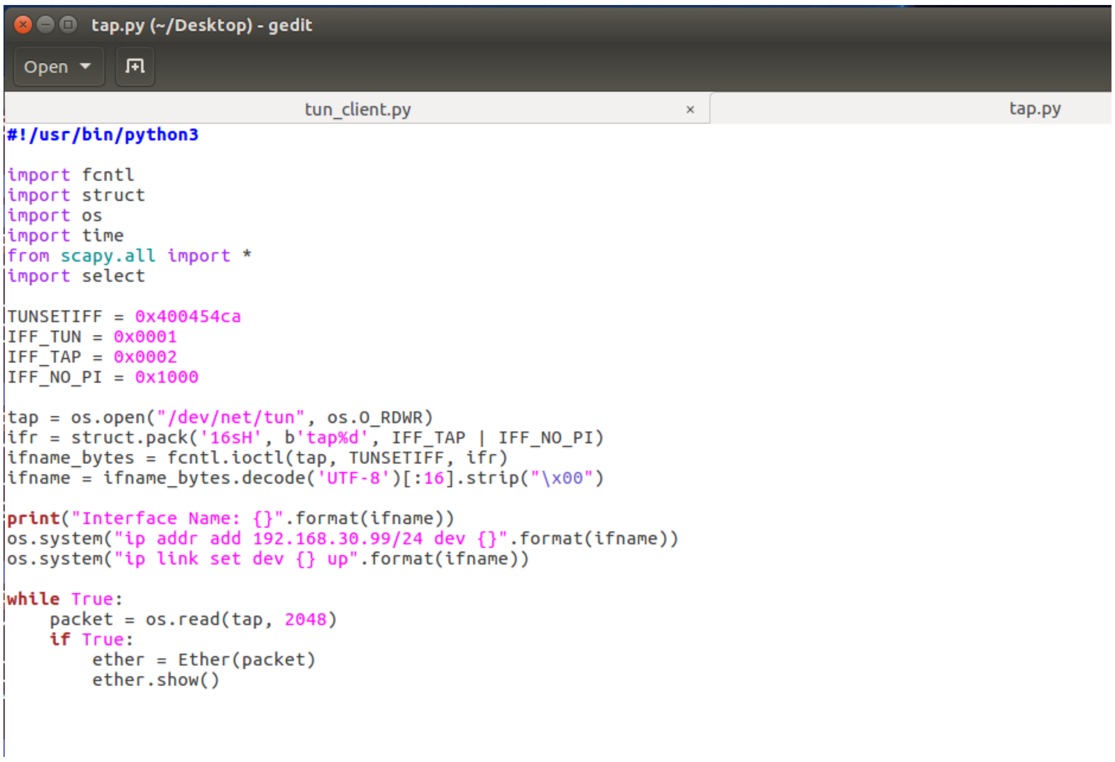


```
[09/22/20]seed@VM:~$ sudo ip route add 192.168.30.0/24 dev enp0s3 via 192.168.60
.1
[09/22/20]seed@VM:~$ ip route
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.30.0/24 via 192.168.60.1 dev enp0s3
192.168.53.0/24 via 192.168.60.1 dev enp0s3
192.168.60.0/24 dev enp0s3  proto kernel  scope link  src 192.168.60.101  metric
 100
[09/22/20]seed@VM:~$
```

此时 U 可以 ping 通 V

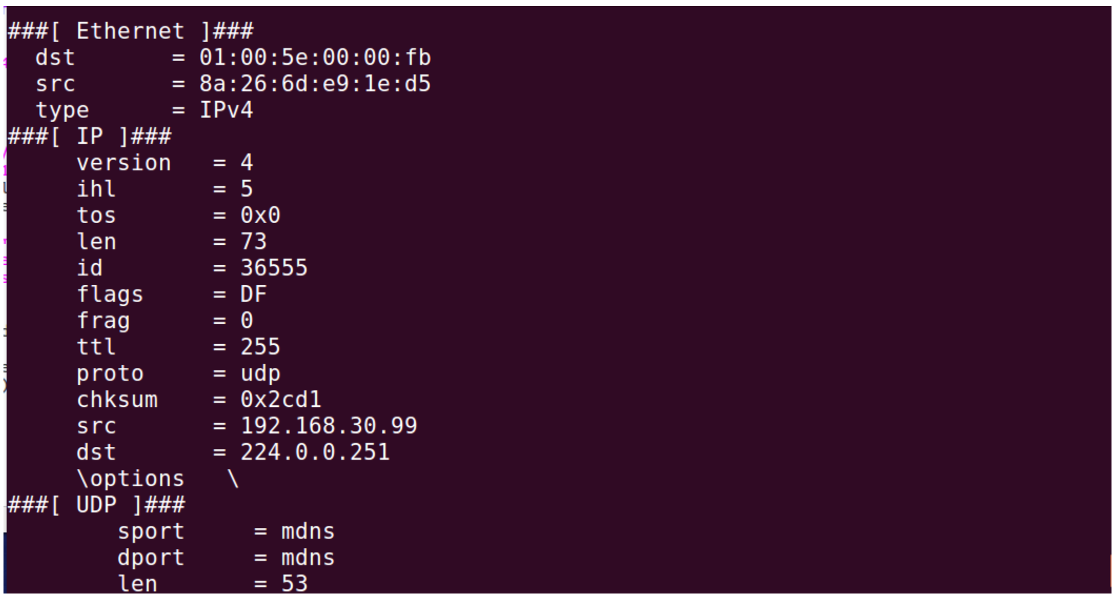
Task 9: Experiment with the TAP interface

在主机 U 中编写程序如下:



```
#!/usr/bin/python3

import fcntl
import struct
import os
import time
from scapy.all import *
import select

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tap%d', IFF_TAP | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tap, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")

print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.30.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
    packet = os.read(tap, 2048)
    if True:
        ether = Ether(packet)
        ether.show()
```

运行之后 ping:



```
###[ Ethernet ]###
  dst       = 01:00:5e:00:00:fb
  src       = 8a:26:6d:e9:1e:d5
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 73
     id        = 36555
     flags     = DF
     frag      = 0
     ttl       = 255
     proto     = udp
     chksum    = 0x2cd1
     src       = 192.168.30.99
     dst       = 224.0.0.251
     \options   \
###[ UDP ]###
        sport     = mdns
        dport     = mdns
        len       = 53
```

证明了 TAP 与 MAC 地址绑定