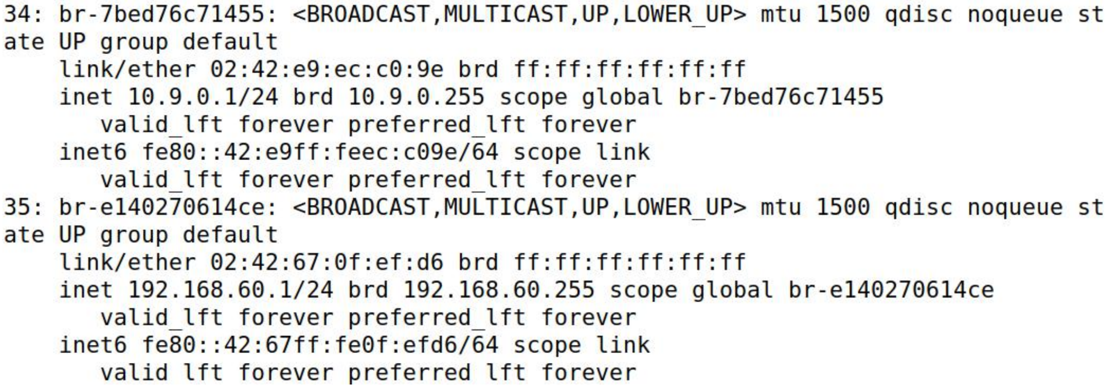
VPN Lab: The Container Version

57118203 陈萱妍

Task 1: Network Setup

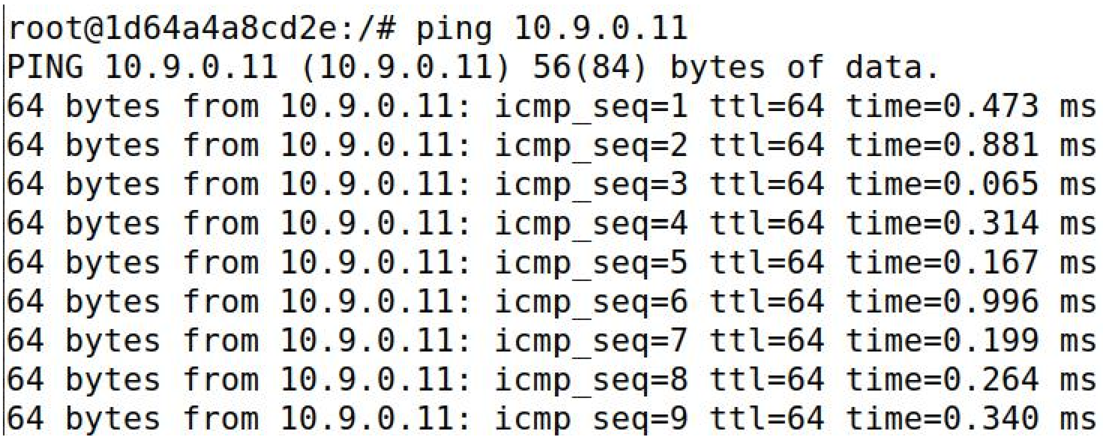
输入命令 ip addr 查看 10.9.0.0/24 和 192.168.60.0/24 两个网段对应的网卡号。



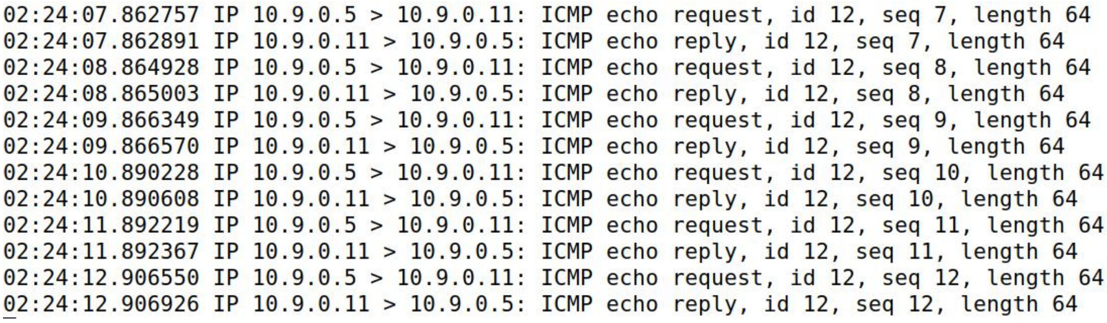
```
34: br-7bed76c71455: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue st
ate UP group default
    link/ether 02:42:e9:ec:c0:9e brd ff:ff:ff:ff:ff:ff
    inet 10.9.0.1/24 brd 10.9.0.255 scope global br-7bed76c71455
       valid_lft forever preferred_lft forever
    inet6 fe80::42:e9ff:feec:c09e/64 scope link
       valid_lft forever preferred_lft forever
35: br-e140270614ce: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue st
ate UP group default
    link/ether 02:42:67:0f:ef:d6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.60.1/24 brd 192.168.60.255 scope global br-e140270614ce
       valid_lft forever preferred_lft forever
    inet6 fe80::42:67ff:fe0f:efd6/64 scope link
       valid_lft forever preferred_lft forever
```

可观察到 10.9.0.1/24 对应的网卡号为 br-7bed76c71455，192.168.60.1/24 对应的网卡号为 br-e140270614ce。

输入命令 sudo tcpdump -i br-7bed76c71455 -n 对 10.9.0.0/24 网段进行嗅探。
在主机 U 上 ping VPN 服务器。



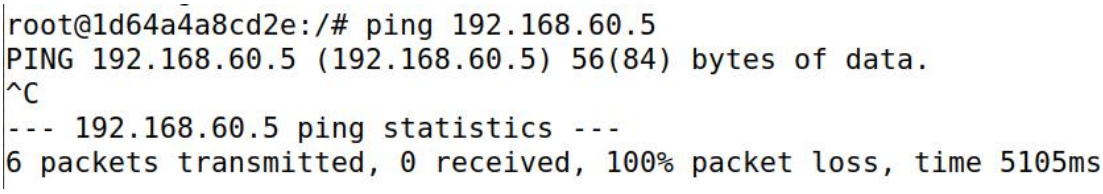
```
root@1d64a4a8cd2e:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.473 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.881 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.065 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.314 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.167 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.996 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.199 ms
64 bytes from 10.9.0.11: icmp_seq=8 ttl=64 time=0.264 ms
64 bytes from 10.9.0.11: icmp_seq=9 ttl=64 time=0.340 ms
```



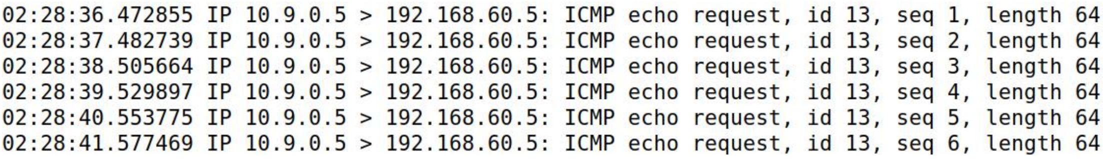
```
02:24:07.862757 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 7, length 64
02:24:07.862891 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 7, length 64
02:24:08.864928 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 8, length 64
02:24:08.865003 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 8, length 64
02:24:09.866349 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 9, length 64
02:24:09.866570 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 9, length 64
02:24:10.890228 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 10, length 64
02:24:10.890608 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 10, length 64
02:24:11.892219 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 11, length 64
02:24:11.892367 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 11, length 64
02:24:12.906550 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 12, length 64
02:24:12.906926 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 12, length 64
```

可观察到通信正常。

在主机 U 上 ping 主机 V。



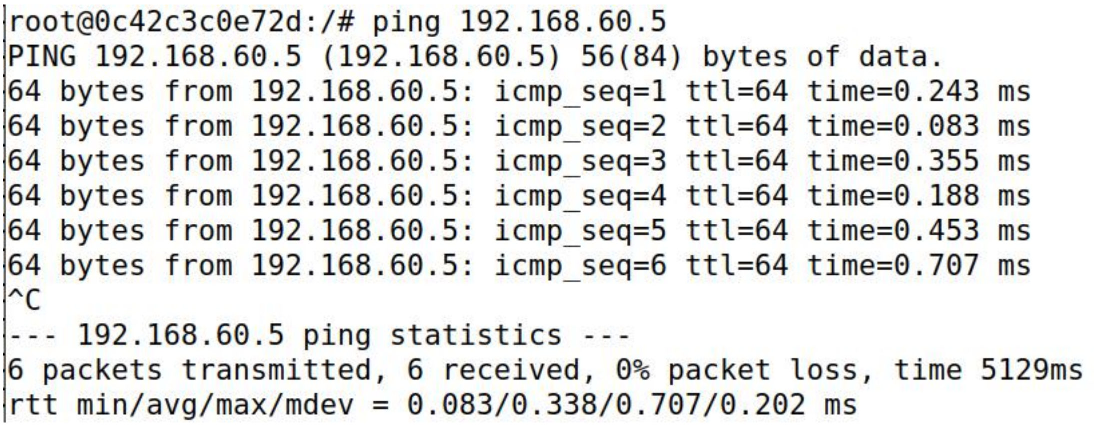
```
root@1d64a4a8cd2e:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5105ms
```



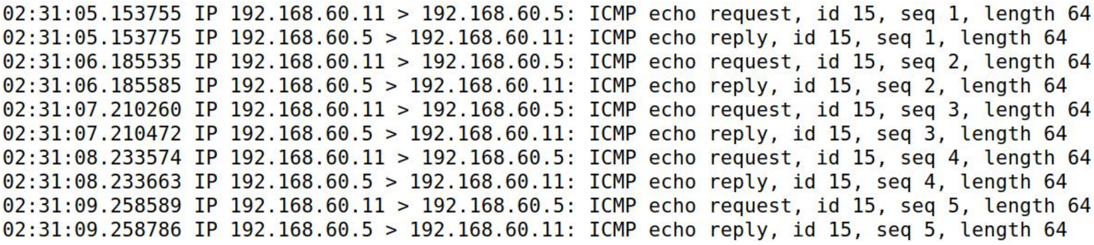
```
02:28:36.472855 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 1, length 64
02:28:37.482739 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 2, length 64
02:28:38.505664 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 3, length 64
02:28:39.529897 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 4, length 64
02:28:40.553775 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 5, length 64
02:28:41.577469 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 6, length 64
```

可观察到无法进行通信。

输入命令 sudo tcpdump -i br-e140270614ce -n 对 192.168.60.0/24 网段进行嗅探。在 VPN 服务器上 ping 主机 V。



```
root@0c42c3c0e72d:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.243 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.355 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.188 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.453 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=64 time=0.707 ms
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5129ms
rtt min/avg/max/mdev = 0.083/0.338/0.707/0.202 ms
```



```
02:31:05.153755 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 15, seq 1, length 64
02:31:05.153775 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 15, seq 1, length 64
02:31:06.185535 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 15, seq 2, length 64
02:31:06.185585 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 15, seq 2, length 64
02:31:07.210260 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 15, seq 3, length 64
02:31:07.210472 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 15, seq 3, length 64
02:31:08.233574 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 15, seq 4, length 64
02:31:08.233663 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 15, seq 4, length 64
02:31:09.258589 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 15, seq 5, length 64
02:31:09.258786 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 15, seq 5, length 64
```

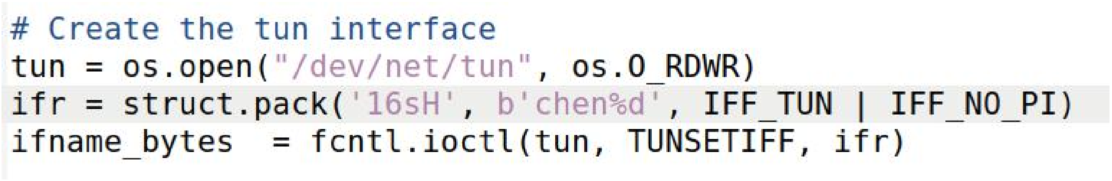
可观察到通信正常。

网络流量都可以正常嗅探，配置正常。

Task 2: Create and Confifigure TUN Interface

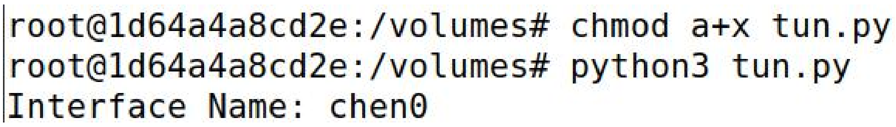
Task 2.a: Name of the Interface

修改代码 tun.py，将端口名修改为“chen”。



```
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'chen%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
```

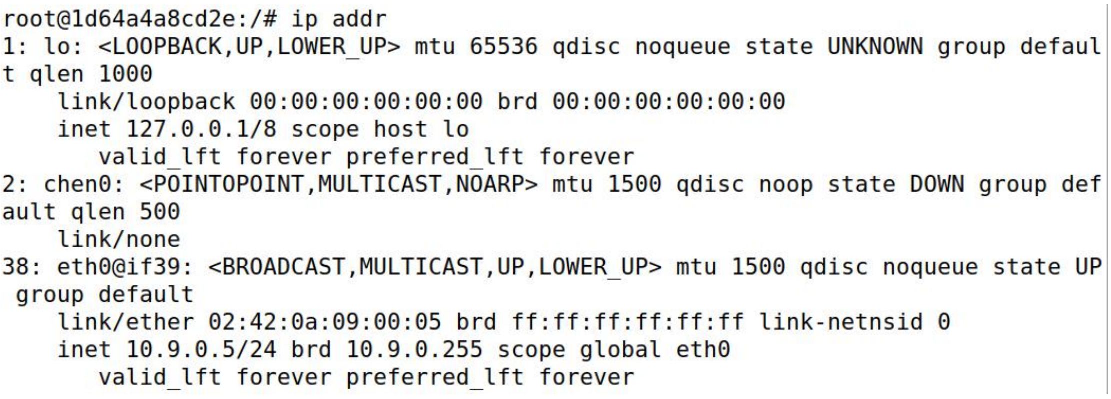
在主机 U 上运行程序。



```
root@1d64a4a8cd2e:/volumes# chmod a+x tun.py
root@1d64a4a8cd2e:/volumes# python3 tun.py
Interface Name:  chen0
```

可观察到端口 chen0 被创建。

打开另一个终端，在主机 U 上输入命令 ip addr。

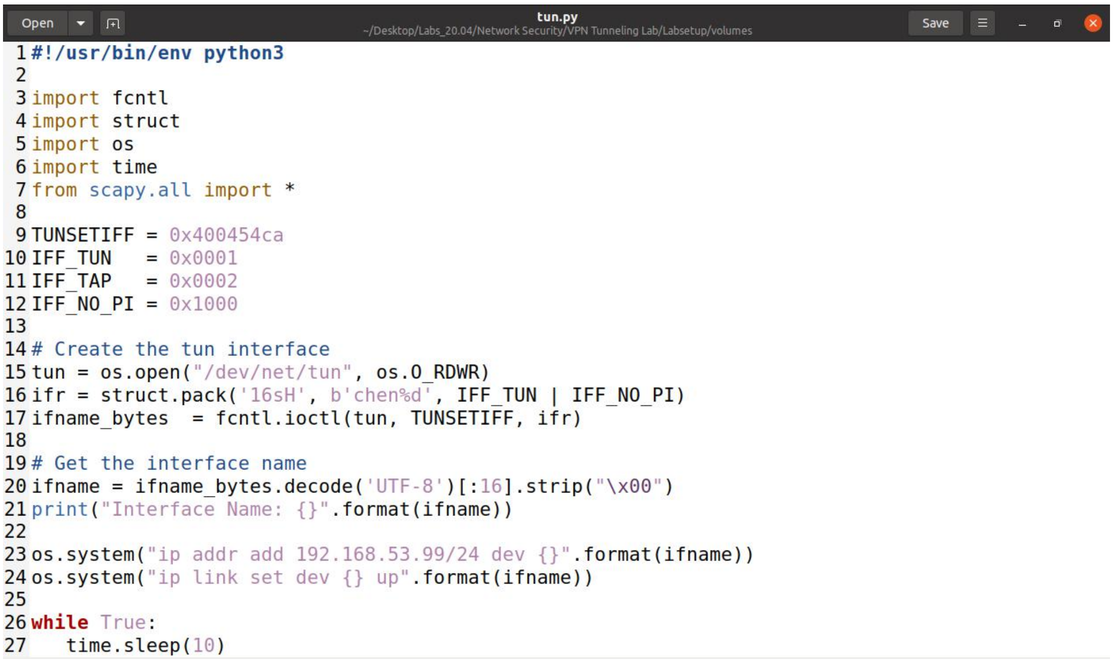


```
root@1d64a4a8cd2e:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: chen0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group def
ault qlen 500
    link/none
38: eth0@if39: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
```

可观察到虚拟网卡 chen0 已被添加。

Task 2.b: Set up the TUN Interface

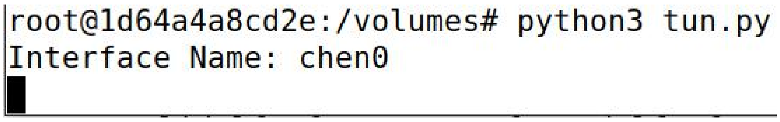
在程序中添加配置代码给端口 chen0 自动分配 ip 地址。



Open
tun.py
~/Desktop/Labs_20.04/Network Security/VPN Tunneling Lab/Labsetup/volumes
Save

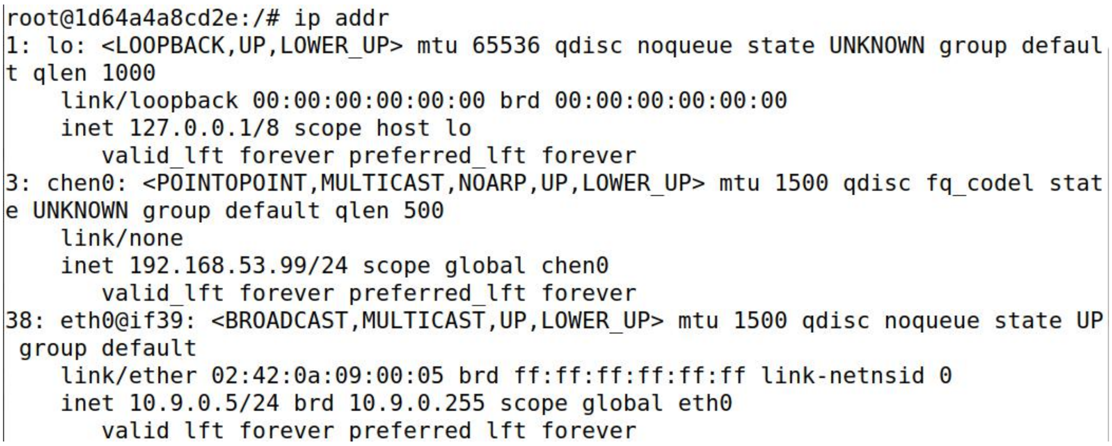
```python
#!/usr/bin/env python3

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

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

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'chen%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 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
   time.sleep(10)
```

在主机 U 上运行程序。



```
root@1d64a4a8cd2e:/volumes# python3 tun.py
Interface Name:  chen0
```

打开另一个终端，在主机 U 上输入命令 ip addr。

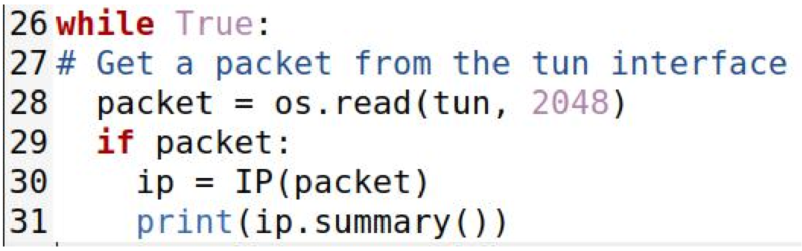


```
root@1d64a4a8cd2e:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: chen0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel stat
e UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global chen0
       valid_lft forever preferred_lft forever
38: eth0@if39: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid lft forever preferred lft forever
```

可观察到此时端口 chen0 已被成功分配 ip 地址。

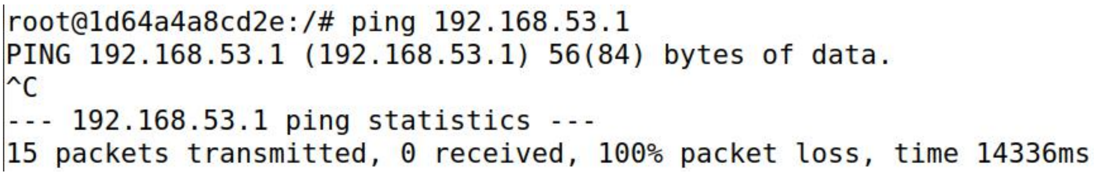
Task 2.c: Read from the TUN Interface

修改程序，添加 while 循环。

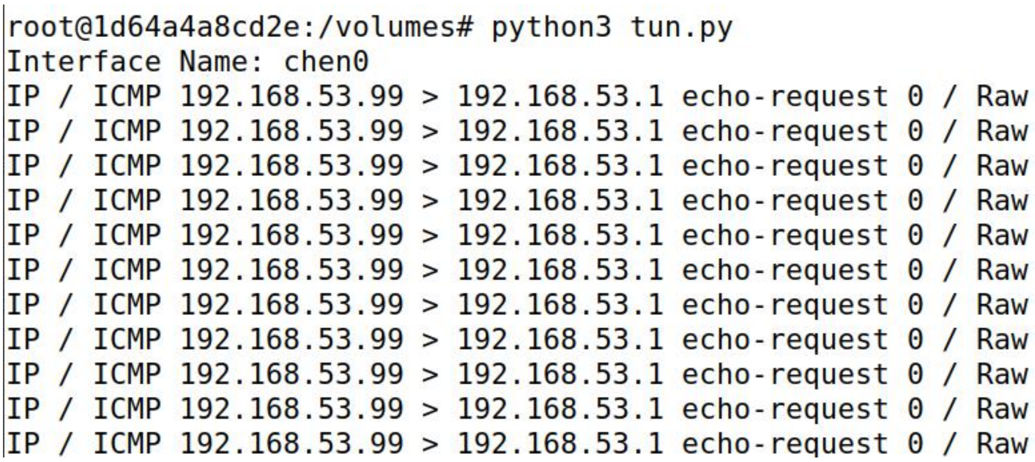


```
while True:
# Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
    ip = IP(packet)
    print(ip.summary())
```

在主机 U 上运行程序。
打开另一个终端，在主机 U 上 ping 192.168.53.1。



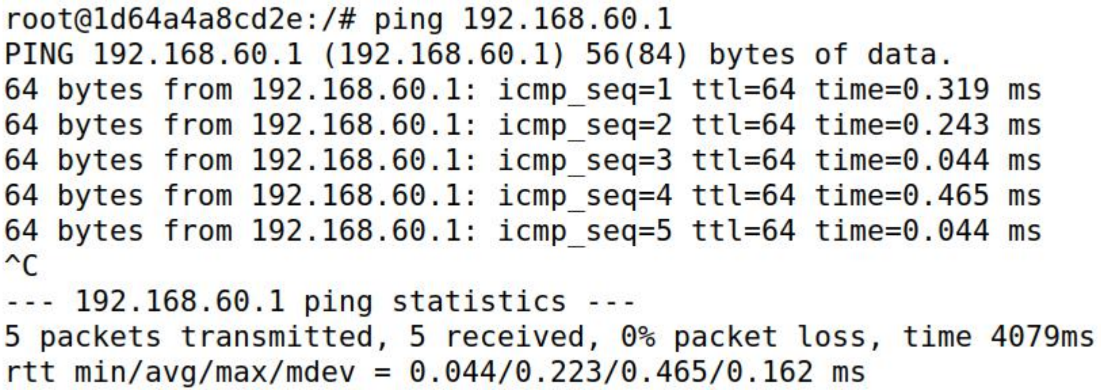
```
root@1d64a4a8cd2e:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
15 packets transmitted, 0 received, 100% packet loss, time 14336ms
```



```
root@1d64a4a8cd2e:/volumes# python3 tun.py
Interface Name: chen0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

可观察到无法 ping 通，目的地址与 chen0 接口 IP 地址位于同网段，通过 chen0 接口发出 ICMP 请求报文，但该网段没有目的主机，无法收到回复报文。

在主机 U 上 ping 192.168.60.1。

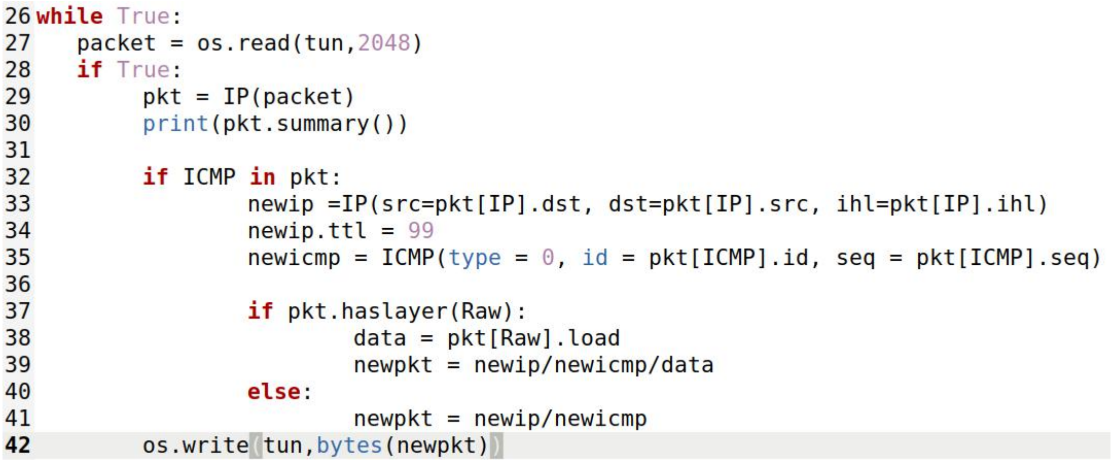


```
root@1d64a4a8cd2e:/# ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
64 bytes from 192.168.60.1: icmp_seq=1 ttl=64 time=0.319 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.243 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.044 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.465 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.044 ms
^C
--- 192.168.60.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.044/0.223/0.465/0.162 ms
```

可以 ping 通，但此时 tun.py 程序无输出。因为目的地址与 chen0 接口 ip 地址不同且无相应路由，ICMP 请求报文不会从该接口发出，所以没有捕获报文。

Task 2.d: Write to the TUN Interface

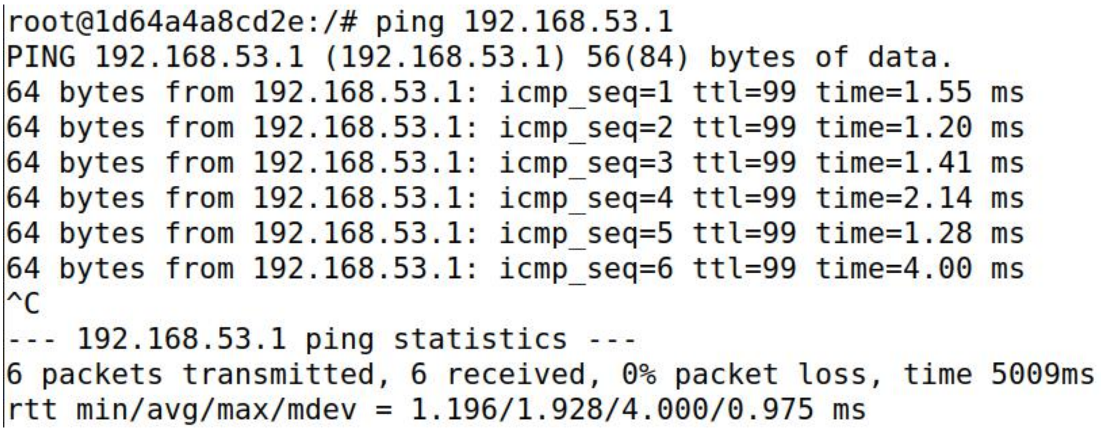
修改程序中的 while 循环。



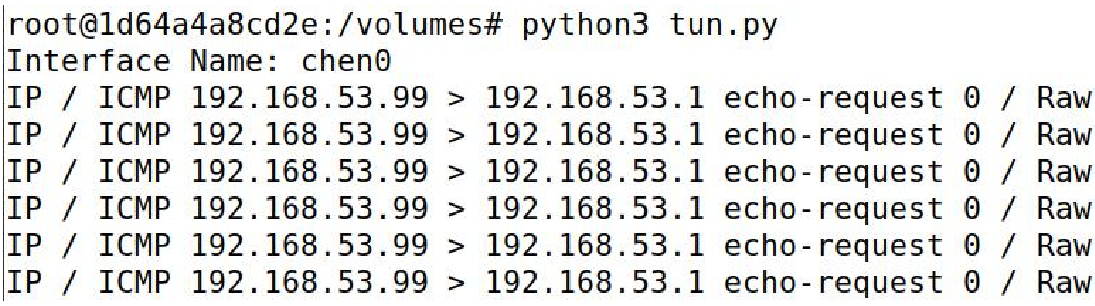
```
while True:
    packet = os.read(tun,2048)
    if True:
        pkt = IP(packet)
        print(pkt.summary())

        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))
```

根据收到的数据包构造新数据包写入接口。

在主机 U 上运行程序。
打开另一个终端，在主机 U 上 ping 192.168.53.1。



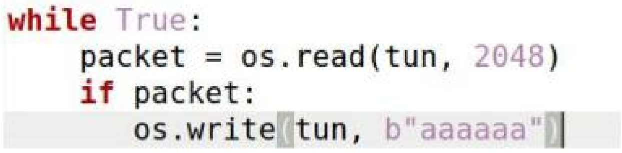
```
root@1d64a4a8cd2e:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.1: icmp_seq=1 ttl=99 time=1.55 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=99 time=1.20 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=99 time=1.41 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=99 time=2.14 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=99 time=1.28 ms
64 bytes from 192.168.53.1: icmp_seq=6 ttl=99 time=4.00 ms
^C
--- 192.168.53.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 1.196/1.928/4.000/0.975 ms
```



```
root@1d64a4a8cd2e:/volumes# python3 tun.py
Interface Name: chen0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

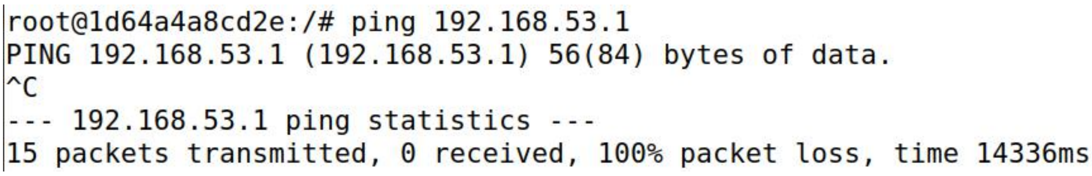
可观察到可以 ping 通，说明伪造相应包成功。

再次修改程序，将写入的 ip 数据包修改为 aaaa。



```
while True:
    packet = os.read(tun, 2048)
    if packet:
       os.write(tun, b"aaaaaa")
```

运行程序，在主机 U 上 ping 192.168.53.1。

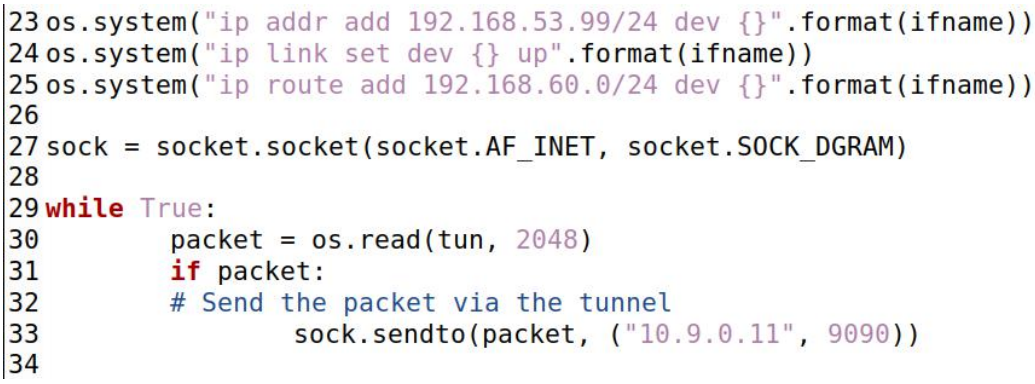


```
root@1d64a4a8cd2e:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
15 packets transmitted, 0 received, 100% packet loss, time 14336ms
```

可观察到无法 ping 通，程序也无输出。

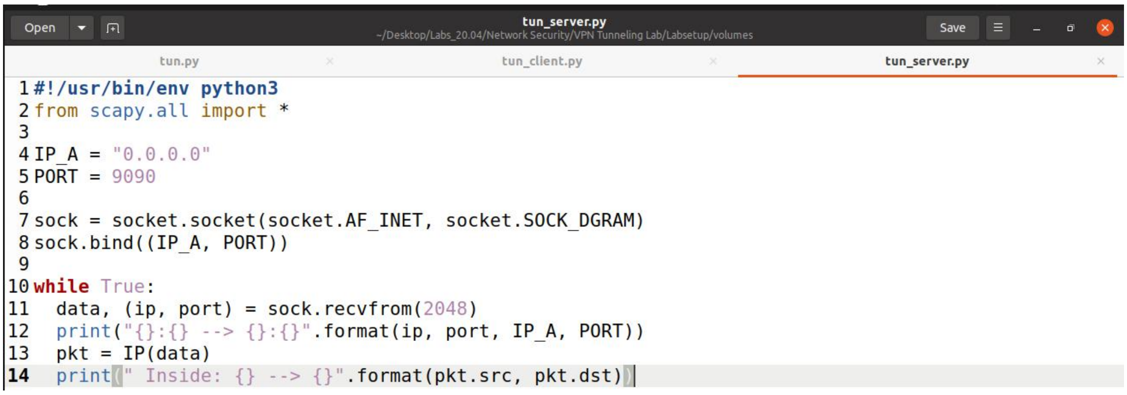
Task 3: Send the IP Packet to VPN Server Through a Tunnel

修改 tun.py 程序保存为 tun_client.py。
在主机 U 上运行 tun_client.py 程序。



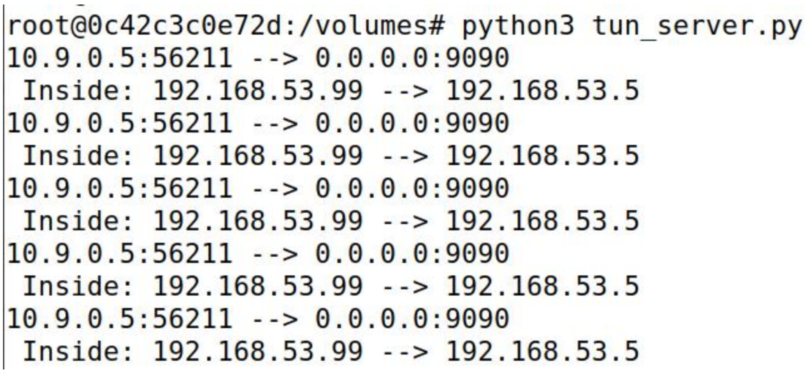
```
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

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

while True:
        packet = os.read(tun, 2048)
        if packet:
        # Send the packet via the tunnel
                sock.sendto(packet, ("10.9.0.11", 9090))

```

根据实验手册提供的代码创建 tun_server.py 程序。
在 VPN 服务器上运行 tun_server.py 程序。


Open
tun_server.py
~/Desktop/Labs_20.04/Network Security/VPN Tunneling Lab/Labsetup/volumes
Save
tun.py
tun_client.py
tun_server.py

```
#!/usr/bin/env 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))
```

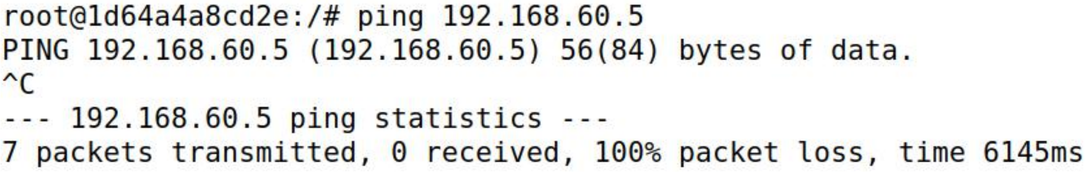
在主机 U 上 ping 192.168.53.5，VPN 服务器输出如下。



```
root@0c42c3c0e72d:/volumes# python3 tun_server.py
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
```

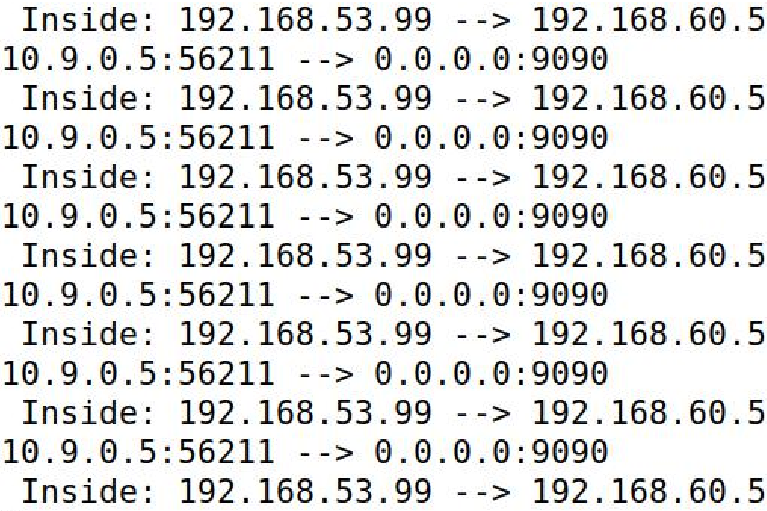
可观察到 VPN 服务器成功捕获到报文，通过 tun_server.py 程序将捕获的报文发给 VPN 服务器的 9090 端口。

在主机 U 上 ping 主机 V。



```
root@1d64a4a8cd2e:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6145ms
```

最初 VPN 服务器上无显示，在主机 U 上输入命令 ip route add 192.168.60.0/24 dev chen0 via 192.168.53.99 添加路由信息。



```
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:56211 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

可观察到 VPN 服务器输出内容，说明 tun_server.py 通过隧道接收到报文。

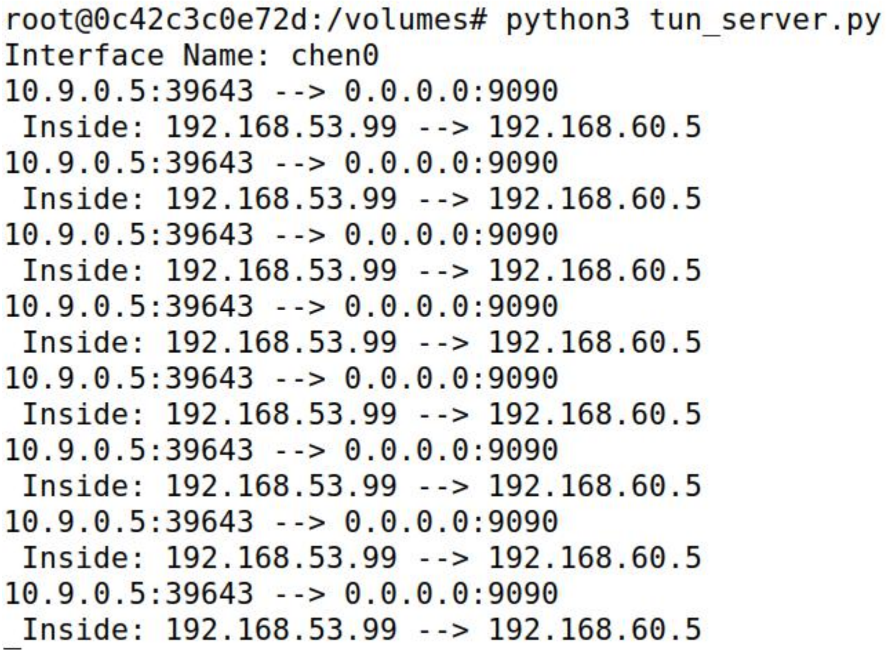
Task 4: Set Up the VPN Server

修改 tun_server.py 代码。

```
#!/usr/bin/env 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

# 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 192.168.53.02/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))


sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
IP_A = "0.0.0.0"
PORT = 9090
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)
```

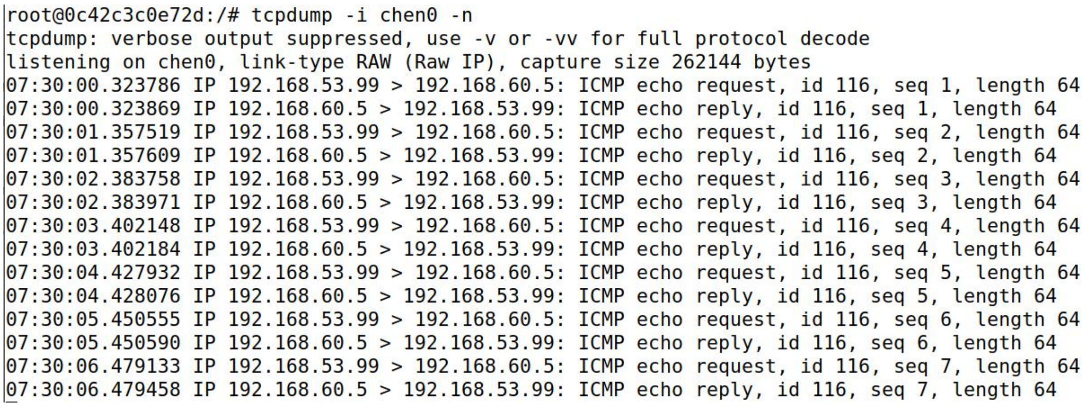
在 VPN 服务器上运行 tun_server.py 程序。
在主机 U 上运行 tun_client.py 程序。
打开另一个终端，在 VPN 服务器输入命令 tcpdump -i chen0 -n 嗅探 chen0 端口。
打开另一个终端，在主机 U 上 ping 192.168.60.5。

tun_server.py 输出如下。



```
root@0c42c3c0e72d:/volumes# python3 tun_server.py
Interface Name: chen0
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:39643 --> 0.0.0.0:9090
_Inside: 192.168.53.99 --> 192.168.60.5
```

嗅探端口显示如下。



```
root@0c42c3c0e72d:/# tcpdump -i chen0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on chen0, link-type RAW (Raw IP), capture size 262144 bytes
07:30:00.323786 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 1, length 64
07:30:00.323869 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 1, length 64
07:30:01.357519 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 2, length 64
07:30:01.357609 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 2, length 64
07:30:02.383758 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 3, length 64
07:30:02.383971 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 3, length 64
07:30:03.402148 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 4, length 64
07:30:03.402184 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 4, length 64
07:30:04.427932 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 5, length 64
07:30:04.428076 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 5, length 64
07:30:05.450555 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 6, length 64
07:30:05.450590 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 6, length 64
07:30:06.479133 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 116, seq 7, length 64
07:30:06.479458 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 116, seq 7, length 64
```

可观察到 VPN 服务器成功将 ICMP 请求包通过隧道发送到主机 V，且收到主机 V 的 ICMP 响应包，隧道一个方向的通信已成功建立。

Task 5: Handling Traffific in Both Directions

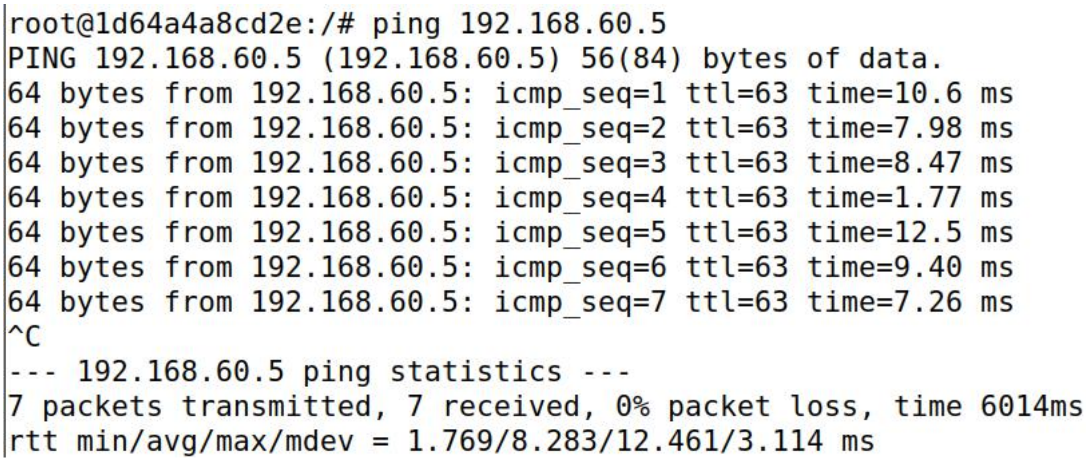
修改 tun_client 脚本如下。

```
#!/usr/bin/env 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

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'chen%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 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
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
fds = [sock,tun]

while True:
    ready,_,_=select.select(fds,[],[])
    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)
            if packet:
                pkt = IP(packet)
                print(pkt.summary())
                sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

修改 tun_server.py 脚本如下。

```
#!/usr/bin/env 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

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'chen%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))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP = "0.0.0.0"
SERVER_PORT = 9090
ip = '10.9.0.5'
port = 10000
sock.bind((SERVER_IP, SERVER_PORT))
fds = [sock,tun]

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

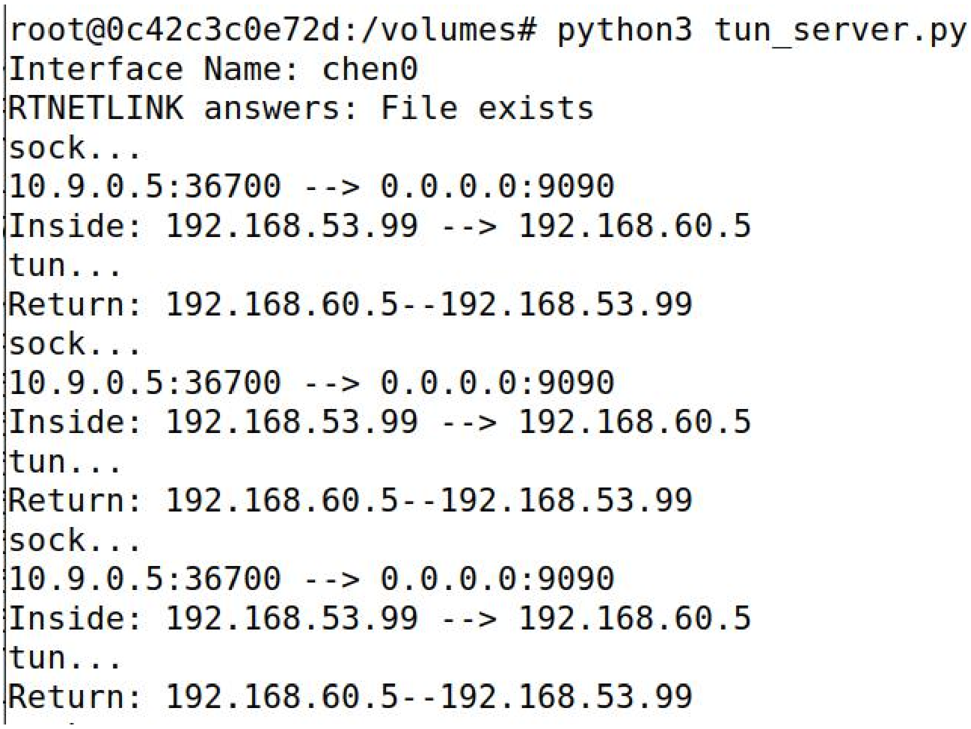
在 VPN 服务器上运行 tun_server.py 程序。
在主机 U 上运行 tun_client.py 程序。
打开另一个终端，在主机 U 上 ping 192.168.60.5。



```
root@1d64a4a8cd2e:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=10.6 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=7.98 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=8.47 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.77 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=12.5 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=9.40 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=7.26 ms
^C
--- 192.168.60.5 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6014ms
rtt min/avg/max/mdev = 1.769/8.283/12.461/3.114 ms
```

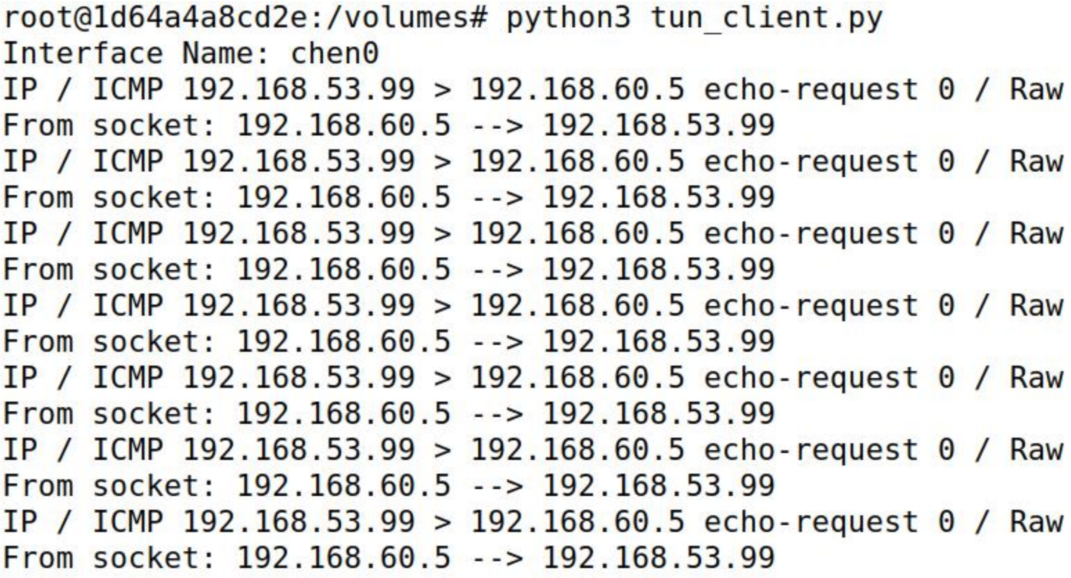
可观察到成功 ping 通。

tun_server.py 的输出如下。



```
root@0c42c3c0e72d:/volumes# python3 tun_server.py
Interface Name: chen0
RTNETLINK answers: File exists
sock...
10.9.0.5:36700 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
sock...
10.9.0.5:36700 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
sock...
10.9.0.5:36700 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
```

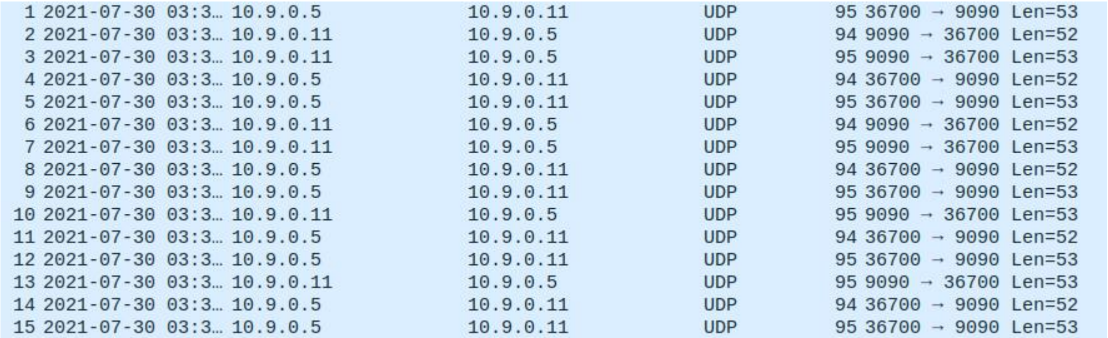
tun_client.py 的输出如下。



```
root@1d64a4a8cd2e:/volumes# python3 tun_client.py
Interface Name: chen0
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
```

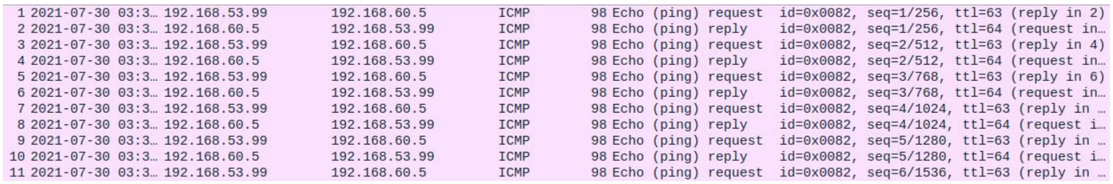
使用 wireshark 抓包上述过程。

10.9.0.0/24 一侧的报文如下。



```
 1 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        95 36700 → 9090 Len=53
 2 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        94 9090 → 36700 Len=52
 3 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        95 9090 → 36700 Len=53
 4 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        94 36700 → 9090 Len=52
 5 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        95 36700 → 9090 Len=53
 6 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        94 9090 → 36700 Len=52
 7 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        95 9090 → 36700 Len=53
 8 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        94 36700 → 9090 Len=52
 9 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        95 36700 → 9090 Len=53
10 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        95 9090 → 36700 Len=53
11 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        94 36700 → 9090 Len=52
12 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        95 36700 → 9090 Len=53
13 2021-07-30 03:3… 10.9.0.11         10.9.0.5         UDP        95 9090 → 36700 Len=53
14 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        94 36700 → 9090 Len=52
15 2021-07-30 03:3… 10.9.0.5          10.9.0.11        UDP        95 36700 → 9090 Len=53
```

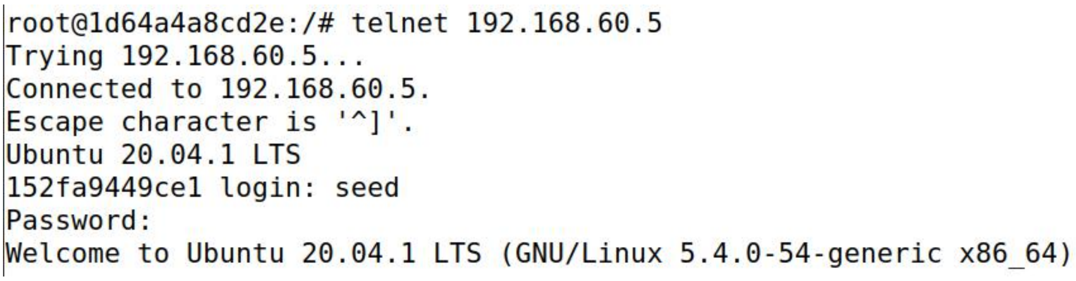
192.168.60.0/24 一侧的报文如下。



```
 1 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=1/256, ttl=63 (reply in 2)
 2 2021-07-30 03:3… 192.168.60.5    192.168.53.99   ICMP    98 Echo (ping) reply    id=0x0082, seq=1/256, ttl=64 (request in…
 3 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=2/512, ttl=63 (reply in 4)
 4 2021-07-30 03:3… 192.168.60.5    192.168.53.99   ICMP    98 Echo (ping) reply    id=0x0082, seq=2/512, ttl=64 (request in…
 5 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=3/768, ttl=63 (reply in 6)
 6 2021-07-30 03:3… 192.168.60.5    192.168.53.99   ICMP    98 Echo (ping) reply    id=0x0082, seq=3/768, ttl=64 (request in…
 7 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=4/1024, ttl=63 (reply in …
 8 2021-07-30 03:3… 192.168.60.5    192.168.53.99   ICMP    98 Echo (ping) reply    id=0x0082, seq=4/1024, ttl=64 (request i…
 9 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=5/1280, ttl=63 (reply in …
10 2021-07-30 03:3… 192.168.60.5    192.168.53.99   ICMP    98 Echo (ping) reply    id=0x0082, seq=5/1280, ttl=64 (request i…
11 2021-07-30 03:3… 192.168.53.99   192.168.60.5    ICMP    98 Echo (ping) request  id=0x0082, seq=6/1536, ttl=63 (reply in …
```

可观察到主机 U 将 ICMP 请求报文包装为 UDP 报文发给 VPN 服务器，随后 VPN 服务器解析 UDP 报文，将其中的 ICMP 请求报文发给目的地址即主机 V，随后主机 V 发送 ICMP 响应报文到 VPN 服务器,VPN 服务器将其包装为 UDP 发给主机 U，主机 U 解析 UDP 报文，收到了来自主机 V 的 ICMP 回应报文，从而完成主机 U 和主机 V 之间的通信。

在主机 U 上 telnet 主机 V。经过短暂等待后结果如下

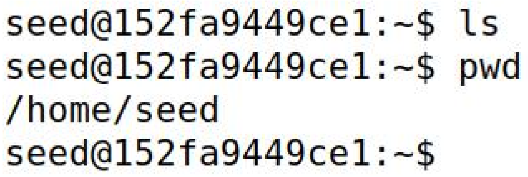


```
root@1d64a4a8cd2e:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
152fa9449ce1 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

可观察到 telnet 连接成功建立。

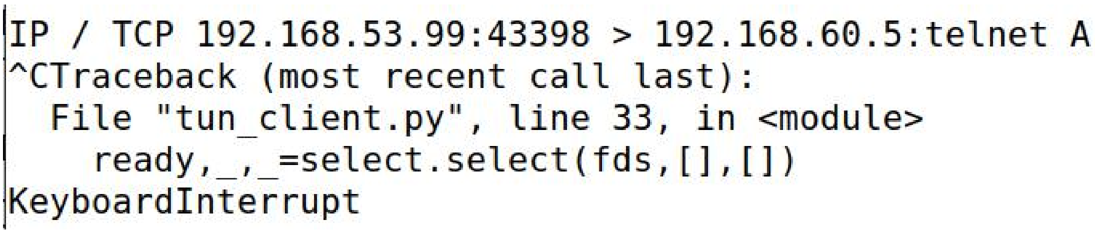
Task 6: Tunnel-Breaking Experiment

在主机 U 上 telnet 主机 V。



```
seed@152fa9449ce1:~$ ls
seed@152fa9449ce1:~$ pwd
/home/seed
seed@152fa9449ce1:~$
```

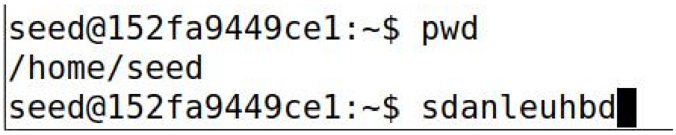
停止运行 tun_client.py 脚本，在 telnet 窗口无法输入任何字符。



```
IP / TCP 192.168.53.99:43398 > 192.168.60.5:telnet A
^CTraceback (most recent call last):
  File "tun_client.py", line 33, in <module>
    ready,_,_=select.select(fds,[],[])
KeyboardInterrupt
```

因为停止程序后隧道中断，数据包无法到达。

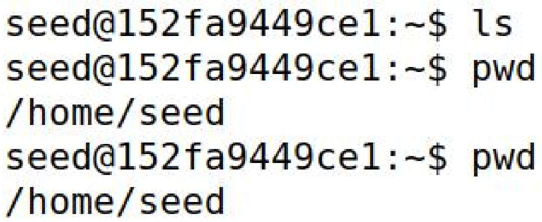
再次运行 tun_client.py 脚本。



```
seed@152fa9449ce1:~$ pwd
/home/seed
seed@152fa9449ce1:~$ sdanleuhbd
```

可观察到在停止程序后输入的内容会显示出来。

恢复隧道后，telnet 连接恢复。



```
seed@152fa9449ce1:~$ ls
seed@152fa9449ce1:~$ pwd
/home/seed
seed@152fa9449ce1:~$ pwd
/home/seed
```

再次运行程序后，隧道立即恢复，中断时的 tcp 报文可再次通过隧道发送。