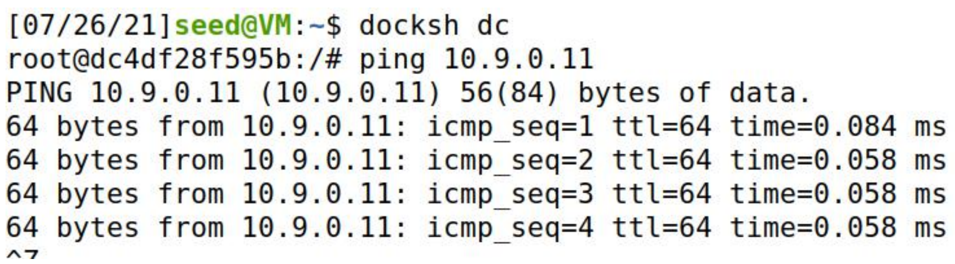
网络空间安全实训实验报告

纪盛谦 57118218 2021-7-26

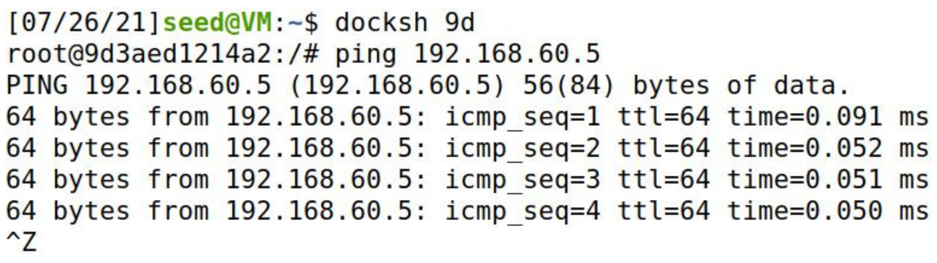
Task1 : Network Setup

Host U 可以连接到 VPN 服务器



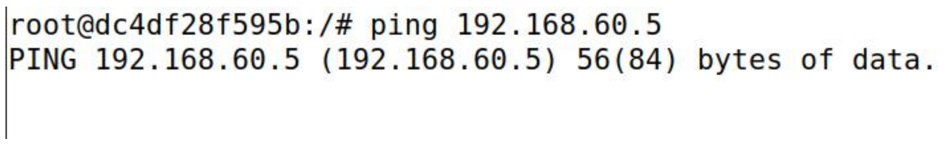
```
[07/26/21]seed@VM:~$ docksh dc
root@dc4df28f595b:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.084 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.058 ms
^Z
```

VPN 服务器可以连接到 Host V



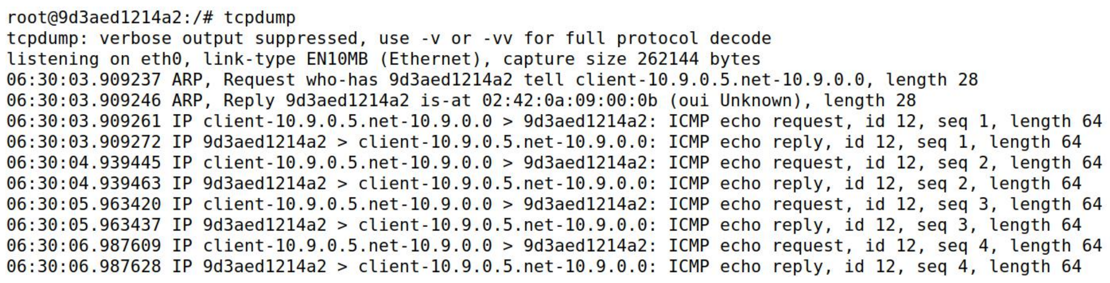
```
[07/26/21]seed@VM:~$ docksh 9d
root@9d3aed1214a2:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.050 ms
^Z
```

Host U 不能连接到 Host V



```
root@dc4df28f595b:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

路由中运行 tcpdump 可以抓包

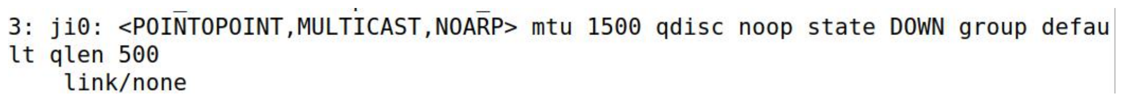


```
root@9d3aed1214a2:/# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
06:30:03.909237 ARP, Request who-has 9d3aed1214a2 tell client-10.9.0.5.net-10.9.0.0, length 28
06:30:03.909246 ARP, Reply 9d3aed1214a2 is-at 02:42:0a:09:00:0b (oui Unknown), length 28
06:30:03.909261 IP client-10.9.0.5.net-10.9.0.0 > 9d3aed1214a2: ICMP echo request, id 12, seq 1, length 64
06:30:03.909272 IP 9d3aed1214a2 > client-10.9.0.5.net-10.9.0.0: ICMP echo reply, id 12, seq 1, length 64
06:30:04.939445 IP client-10.9.0.5.net-10.9.0.0 > 9d3aed1214a2: ICMP echo request, id 12, seq 2, length 64
06:30:04.939463 IP 9d3aed1214a2 > client-10.9.0.5.net-10.9.0.0: ICMP echo reply, id 12, seq 2, length 64
06:30:05.963420 IP client-10.9.0.5.net-10.9.0.0 > 9d3aed1214a2: ICMP echo request, id 12, seq 3, length 64
06:30:05.963437 IP 9d3aed1214a2 > client-10.9.0.5.net-10.9.0.0: ICMP echo reply, id 12, seq 3, length 64
06:30:06.987609 IP client-10.9.0.5.net-10.9.0.0 > 9d3aed1214a2: ICMP echo request, id 12, seq 4, length 64
06:30:06.987628 IP 9d3aed1214a2 > client-10.9.0.5.net-10.9.0.0: ICMP echo reply, id 12, seq 4, length 64
```

Task2: Create and Confifigure TUN Interface

Task2.a : Name of the Interface

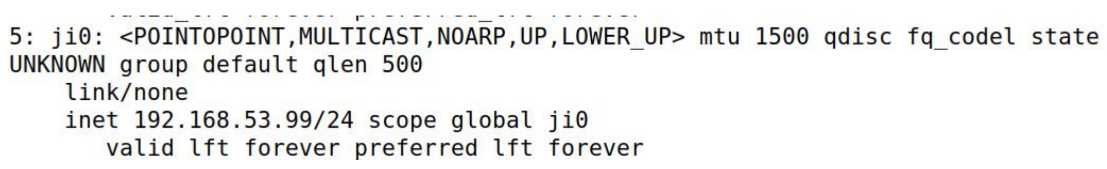
用我的姓氏 ji 作为端口名。



```
3: ji0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defau
lt qlen 500
    link/none
```

Task2.b : Set up the TUN Interface

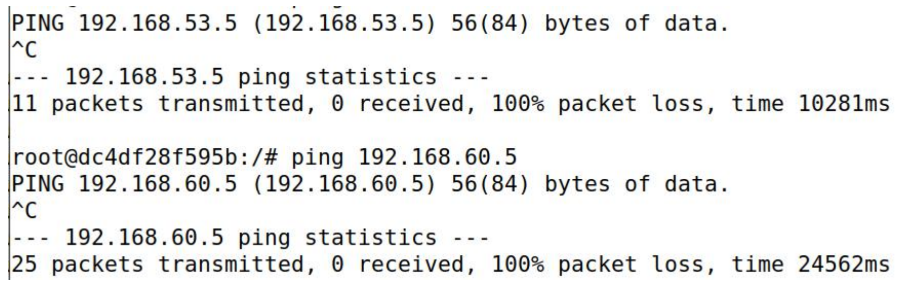
添加两条指令后再次运行程序，可以看到这个端口已经开启并且配置了 IP 地址。



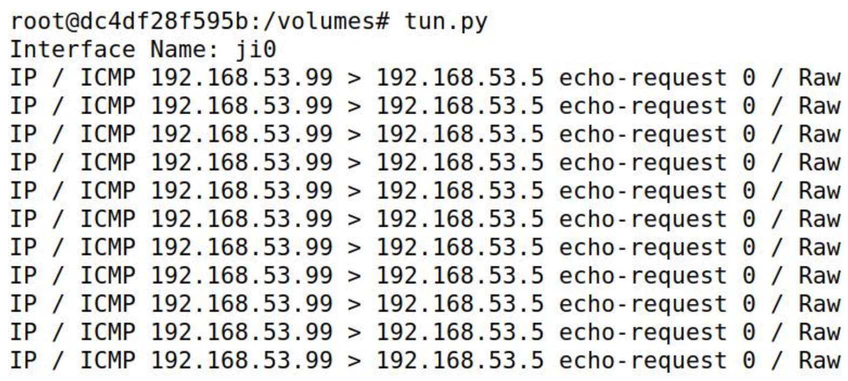
```
5: ji0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global ji0
       valid_lft forever preferred_lft forever
```

Task 2.c: Read from the TUN Interface

当 ping192.168.53.5 时，程序输出 ping 出去的 ICMP 报文，而当 ping192.168.60.5 时程序没有输出，因为代码中抓取的报文是抓取 ji 端口的报文，ji 端口 ip 是 192.168.53.99，ping192.168.53.5 的报文从 ji 端口发出可以被获取输出，而 ping192.168.60.5 的报文不经过 ji 端口，因此没有输出。



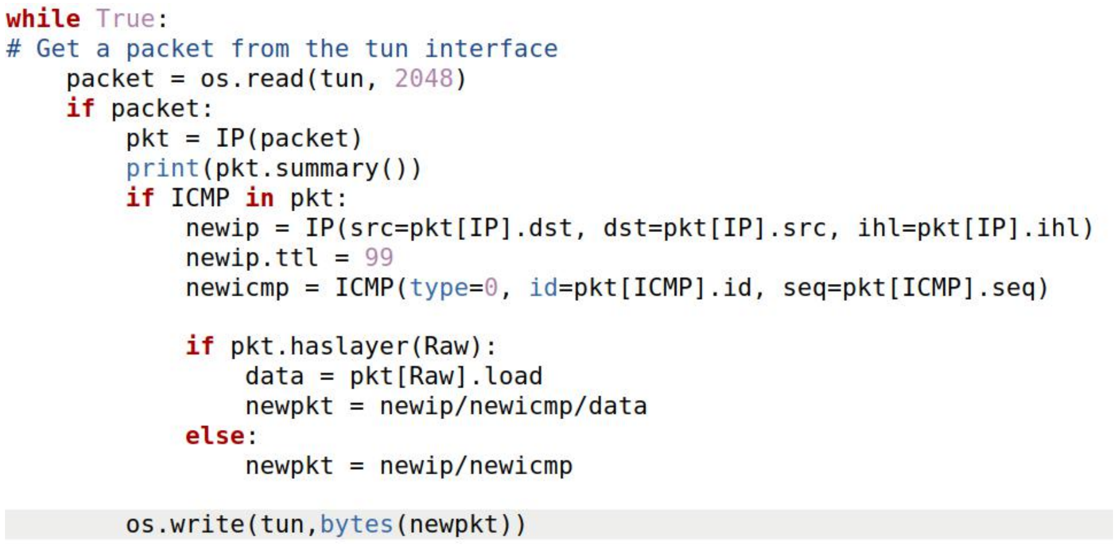
```
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
^C
--- 192.168.53.5 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss, time 10281ms

root@dc4df28f595b:/# 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 ---
25 packets transmitted, 0 received, 100% packet loss, time 24562ms
```



```
root@dc4df28f595b:/volumes# tun.py
Interface Name: ji0
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
```

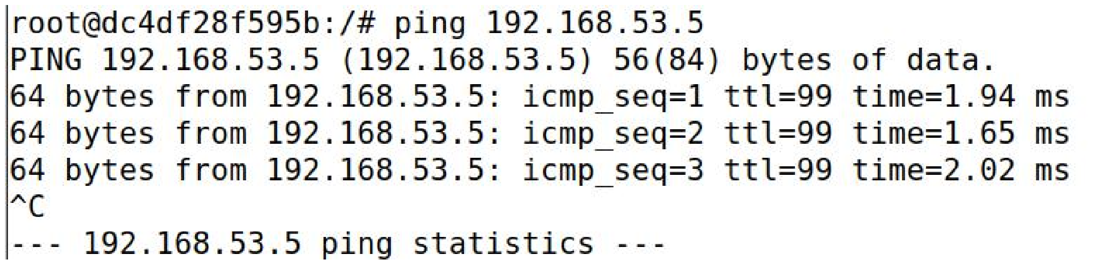
Task 2.d: Write to the TUN Interface

更部分代码，使其实现对 ICMP echo 报文构造响应报文功能，如下图

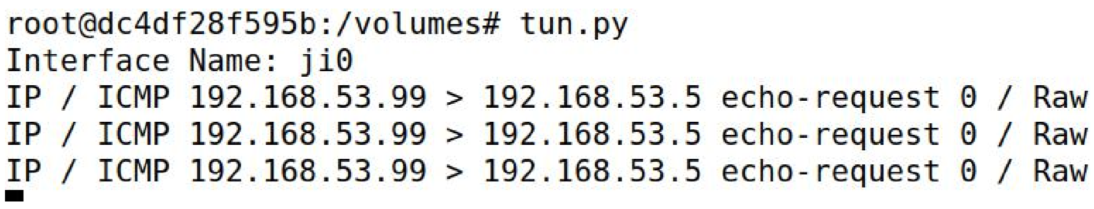


```
while True:
# Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print(pkt.summary())
        if ICMP in pkt:
            newip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
            newip.ttl = 99
            newicmp = 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))
```

再次运行程序，可以看到此时 ping 192.168.53.5 能够 ping 通，并且在程序中出现 ping 的报文。



```
root@dc4df28f595b:/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
64 bytes from 192.168.53.5: icmp_seq=1 ttl=99 time=1.94 ms
64 bytes from 192.168.53.5: icmp_seq=2 ttl=99 time=1.65 ms
64 bytes from 192.168.53.5: icmp_seq=3 ttl=99 time=2.02 ms
^C
--- 192.168.53.5 ping statistics ---
```

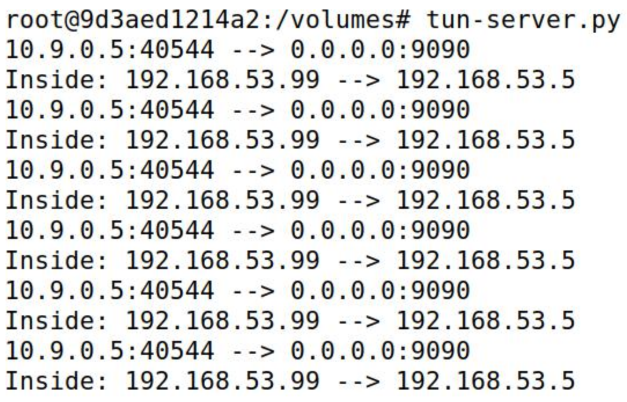


```
root@dc4df28f595b:/volumes# tun.py
Interface Name: ji0
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
```

同时该程序除了原宿 ip 调换位置，还更改了 ttl 属性值，其 ttl 设定为 99，而从接受到的数据报文中也可以看到 ttl 仍为 99。

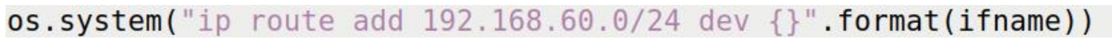
Task3: Send the IP Packet to VPN Server Through a Tunnel

在 Host U 上运行 tun-client、在 VPN 服务器上运行 tun-server，从 Host U ping 192.168.53.5,此时在 VPN 服务器上可以看到有从 Host U 发来的报文，其中封装了 192.168.53.99 到 192.168.53.5 的报文，因为在 ping 的时候经过 tun-client，该程序将这个 ping 报文封装并转发给了 10.9.0.11 的 9090 端口，而 tun-server 就在监听这个端口，因此获得该报文。



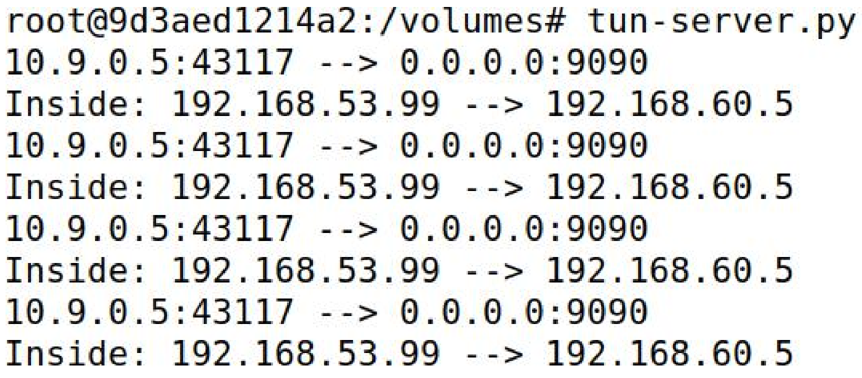
```
root@9d3aed1214a2:/volumes# tun-server.py
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:40544 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
```

在 tun-client 的代码中添加这一行，实现将发往 192.168.60.0 的报文经过 tun 端口发出。



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

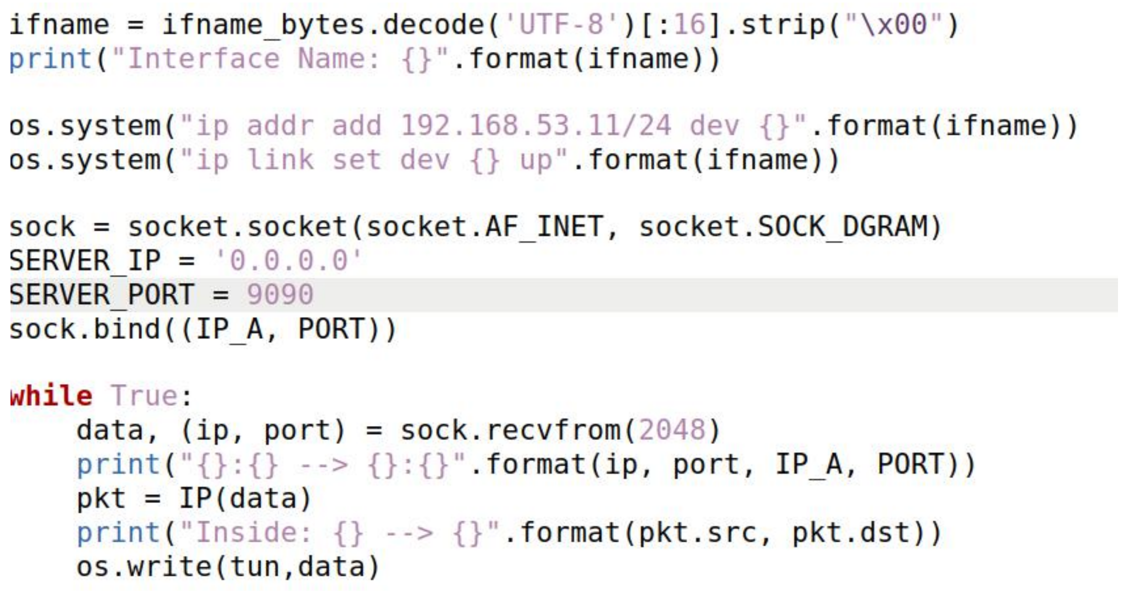
此时再次从 Host U ping 192.168.60.5，在 VPN 服务器上可以接受到经过封装后的报文，但由于没有转发功能，因此 ping 不通。



```
root@9d3aed1214a2:/volumes# tun-server.py
10.9.0.5:43117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:43117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:43117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:43117 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```

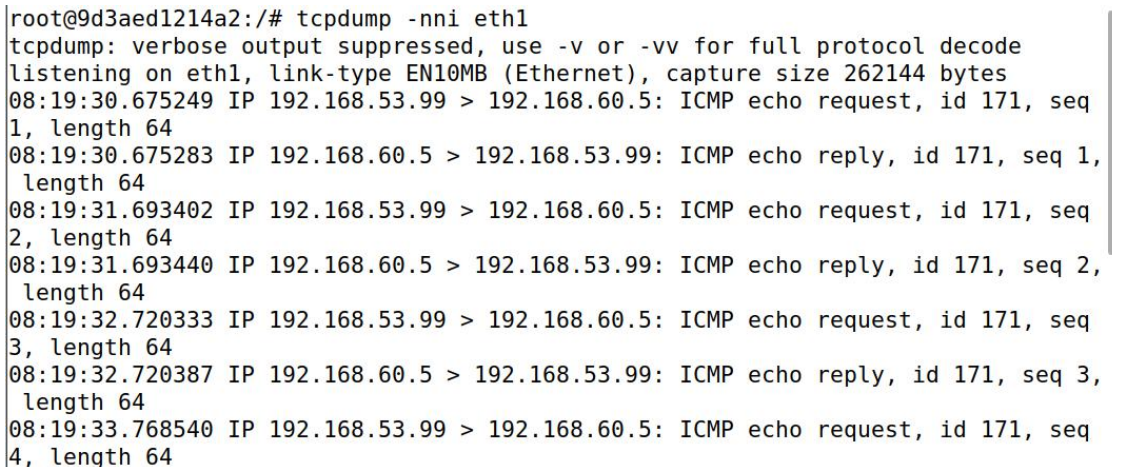
Task4: Set Up the VPN Server

经过更改 tun-server 代码如下图



```
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))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP = '0.0.0.0'
SERVER_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)
```

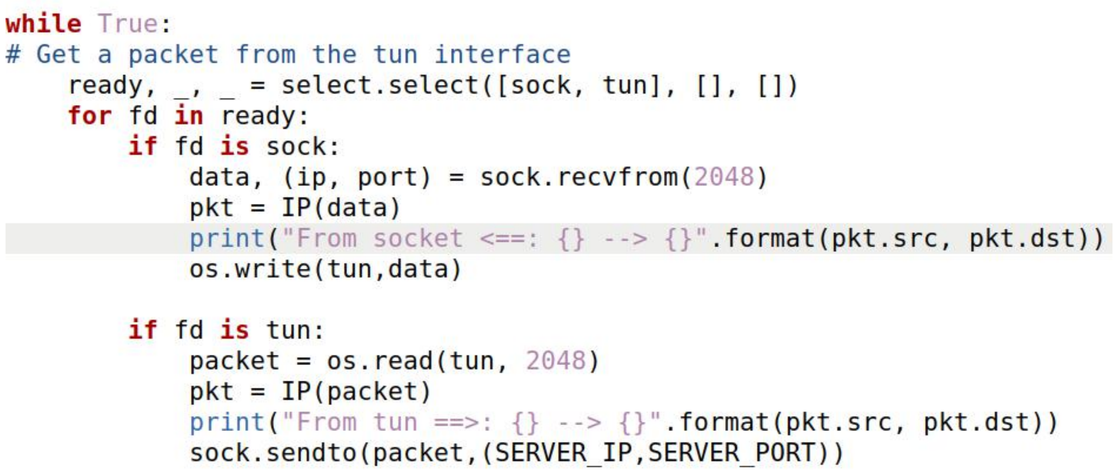
再次从 Host U ping Host V，此时 VPN 服务器在接收到 ICMP 报文后会将其转发给 Host V，通过 tcpdump 指令监听端口 eth1，可以看到抓取的转发的报文是从 192.168.53.99 向 192.168.60.5 的 ICMP echo 报文。



```
root@9d3aed1214a2:/# tcpdump -nni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
08:19:30.675249 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 171, seq
1, length 64
08:19:30.675283 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 171, seq 1,
 length 64
08:19:31.693402 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 171, seq
2, length 64
08:19:31.693440 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 171, seq 2,
 length 64
08:19:32.720333 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 171, seq
3, length 64
08:19:32.720387 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 171, seq 3,
 length 64
08:19:33.768540 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 171, seq
4, length 64
```

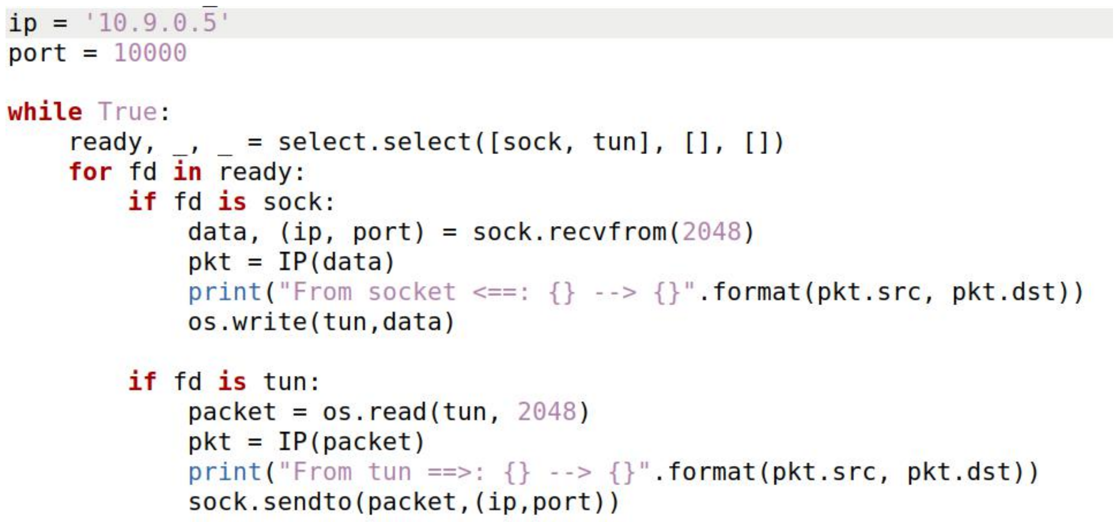
Task5: Handling Traffific in Both Directions

更改 tun-server 部分代码如下：



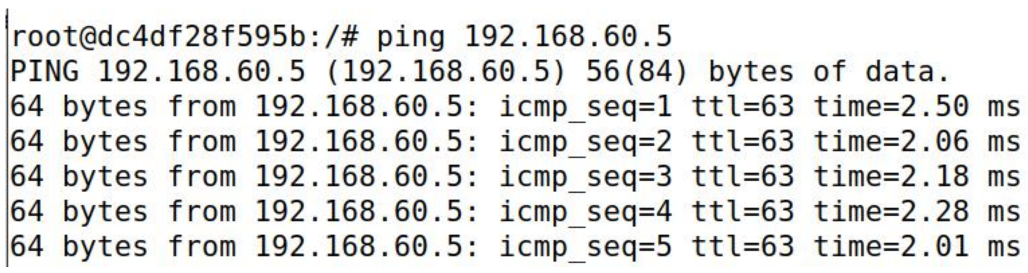
```
while True:
# Get a packet from the tun interface
    ready, _, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            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))
```

更改 tun-client 部分代码如下：



```
ip = '10.9.0.5'
port = 10000

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,(ip,port))
```

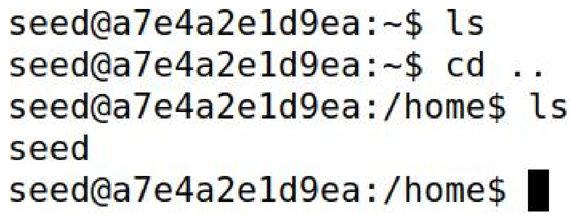
增加了转发功能后，此时从 Host U ping Host V，可以 ping 通



```
root@dc4df28f595b:/# 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=2.50 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=2.06 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=2.18 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=2.28 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=2.01 ms
```

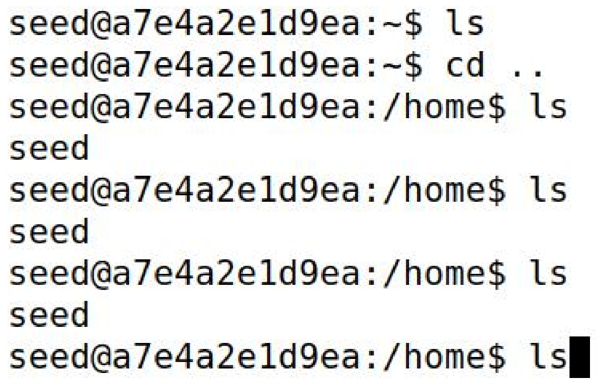
Task6: Tunnel-Breaking Experiment

从 Host U 向 Host V 进行 telnet，当终止 VPN 程序后，再次输入则输入不进去了，因为连接断开，当 Host U 收不到对方响应的时候不会显示所输入的内容。



```
seed@a7e4a2e1d9ea:~$ ls
seed@a7e4a2e1d9ea:~$ cd ..
seed@a7e4a2e1d9ea:/home$ ls
seed
seed@a7e4a2e1d9ea:/home$ 
```

当再次运行 VPN 程序，此时之前终止程序后输入的内容会再次显示，因为 Host U 将断开时候的内容存入到缓冲区一直发送直到收到 Host V 的响应，因此再次建立连接后，缓冲区的内容发送后能够接收到响应，因此会显示终止 VPN 程序后输入 telnet 的内容。



```
seed@a7e4a2e1d9ea:~$ ls
seed@a7e4a2e1d9ea:~$ cd ..
seed@a7e4a2e1d9ea:/home$ ls
seed
seed@a7e4a2e1d9ea:/home$ ls
seed
seed@a7e4a2e1d9ea:/home$ ls
seed
seed@a7e4a2e1d9ea:/home$ ls
```