

lab[7]-report

57118123 刘康辉

主机U的IP地址为10.9.0.5，VPN服务器的IP地址为10.9.0.11，主机V的IP地址为192.168.60.5。

Task 1: Network Setup

在主机U上ping服务器，得到结果如下，可知能够连接。

```
root@605f3bbdd218:/# 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=0.062 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.110 ms
^C
--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2043ms
rtt min/avg/max/mdev = 0.062/0.079/0.110/0.021 ms
```

在VPN服务器上利用tcpdump命令抓取数据包，得到结果如下。

```
root@8801e67cd65b:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
23:06:30.808332 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 1, length 64
23:06:30.808351 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 1, length 64
23:06:31.826025 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 2, length 64
23:06:31.826043 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 2, length 64
23:06:32.851733 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 3, length 64
23:06:32.851765 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 3, length 64
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
```

在VPN服务器上ping主机V，得到结果如下，可知能够连接。

```
root@8801e67cd65b:/# 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=0.099 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.105 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.170 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.099/0.124/0.170/0.032 ms
```

在VPN服务器上利用tcpdump命令抓取数据包，得到结果如下。

```

root@8801e67cd65b:/# tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
23:12:11.527276 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 33, seq 1, length 64
23:12:11.527341 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 33, seq 1, length 64
23:12:12.532049 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 33, seq 2, length 64
23:12:12.532112 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 33, seq 2, length 64
23:12:13.556014 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 33, seq 3, length 64
23:12:13.556084 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 33, seq 3, length 64
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel

```

在主机U上ping主机V，得到结果如下，可知无法连接。

```

root@605f3bbdd218:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2058ms

```

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

创建tun.py文件，代码如下。

```

#!/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

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%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))
while True:
    time.sleep(10)

```

在主机U上利用root权限运行该程序后，创建tun接口seu0。

```

root@605f3bbdd218:/volumes# chmod a+x tun.py
root@605f3bbdd218:/volumes# tun.py
Interface Name: seu0

```

利用ip address命令查看接口信息，可知接口seu0创建成功。

```

root@605f3bbdd218:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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: seu0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
18: eth0@if19: <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

```

Task 2.b: Set up the TUN Interface

利用ip命令配置接口seu0的ip地址并开启接口，可知接口seu0设置成功。

```
root@605f3bbdd218:/# ip addr add 192.168.53.99/24 dev seu0
root@605f3bbdd218:/# ip link set dev seu0 up
root@605f3bbdd218:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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: seu0: <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 seu0
        valid_lft forever preferred_lft forever
18: eth0@if19: <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
```

Task 2.c: Read from the TUN Interface

修改tun.py文件，代码如下。

```
#!/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

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d' % IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
while True:
    packet = os.read(tun, 2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())
```

在主机U上ping主机192.168.53.1，得到结果如下，可知无法连接。

```
root@605f3bbdd218:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2035ms
```

利用root权限运行该程序后，得到结果如下，可知icmp请求报文成功发送，但IP地址为192.168.53.1的主机不存在，导致ping无法连接。

```
root@605f3bbdd218:/volumes# chmod a+x tun.py
root@605f3bbdd218:/volumes# tun.py
Interface Name: seu0
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
```

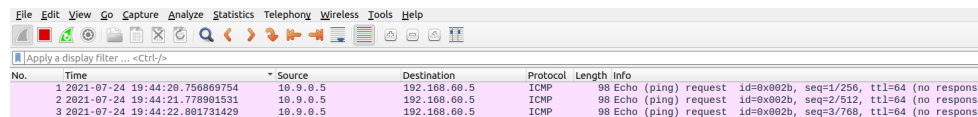
在主机U上ping主机V，得到结果如下，可知无法连接。

```
root@605f3bbdd218:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2058ms
```

利用ip route命令查看路由信息，可知192.168.60.0/24的路由经过接口eth0，而非接口seu0。

```
root@605f3bbdd218:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev seu0 proto kernel scope link src 192.168.53.99
```

利用wireshark抓包，得到结果如下，可知从10.9.0.5成功发送ICMP请求报文，但未收到ICMP响应报文。



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-24 19:44:20.756869754	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x002b, seq=1/256, ttl=64 (no response received)
2	2021-07-24 19:44:21.778901531	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x002b, seq=2/512, ttl=64 (no response received)
3	2021-07-24 19:44:22.801731429	10.9.0.5	192.168.60.5	ICMP	98	Echo (ping) request id=0x002b, seq=3/768, ttl=64 (no response received)

Task 2.d: Write to the TUN Interface

修改tun.py文件，代码如下。

```
#!/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

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d' % IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))

while True:
    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, 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))
```

在主机U上ping主机192.168.53.11，得到结果如下，可知能够连接。

```
root@b531d2abe50e:/# ping 192.168.53.11
PING 192.168.53.11 (192.168.53.11) 56(84) bytes of data.
64 bytes from 192.168.53.11: icmp_seq=1 ttl=99 time=3.22 ms
64 bytes from 192.168.53.11: icmp_seq=2 ttl=99 time=4.92 ms
64 bytes from 192.168.53.11: icmp_seq=3 ttl=99 time=3.90 ms
^C
--- 192.168.53.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 3.223/4.013/4.920/0.697 ms
```

利用root权限运行该程序后，得到结果如下。

```
root@b531d2abe50e:/volumes# chmod u+x tun.py
root@b531d2abe50e:/volumes# tun.py
Interface Name: seu0
IP / ICMP 192.168.53.99 > 192.168.53.11 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.11 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.11 echo-request 0 / Raw
```

修改tun.py文件，代码如下。

```
#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
while True:
    os.write(tun, bytes('Cyberspace'.encode('utf-8')))
    time.sleep(1)
```

在主机U上利用tcpdump命令抓取数据包，得到结果如下，可知内容与程序中一致。

```
root@b531d2abe50e:/# tcpdump -i seu0 -w dump
tcpdump: listening on seu0, link-type RAW (Raw IP), capture size 262144 bytes
^C2 packets captured
3 packets received by filter
0 packets dropped by kernel
root@b531d2abe50e:/# cat dump
000e00`zT

Cyberspace00`0Z

Cyberspaceroot@b531d2abe50e:/#
```

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

创建tun_server.py文件，代码如下。

```
#!/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))
```

创建tun_client.py文件，代码如下。

```
#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    packet = os.read(tun, 2048)
    if packet:
        sock.sendto(packet, ("10.9.0.11", 9090))
```

在主机U上利用root权限运行tun_client程序后，创建tun接口seu0。

```
root@9fea8514c7f2:/volumes# python3 tun_client.py
Interface Name: seu0
```

在主机U上ping主机192.168.53.1，得到结果如下，可知无法连接。

```
root@9fea8514c7f2:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2053ms
```


在VPN服务器上利用root权限运行tun_server程序后，得到结果如下，可知外部为10.9.0.5至0.0.0.0，内部为192.168.53.99至192.168.53.1，因为192.168.53.0/24的路由经过接口seu0。

```
root@4f0820f90916:/volumes# python3 tun_server.py
10.9.0.5:33011 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:33011 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:33011 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.53.1
```

在主机U上ping主机V，得到结果如下，可知无法连接。

```
root@9fea8514c7f2:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2043ms
```

在VPN服务器上利用root权限运行tun_server程序后，得到结果如下，可知未进入隧道，因为192.168.60.0/24的路由不经过接口seu0。

```
root@4f0820f90916:/volumes# python3 tun_server.py
```

在主机U上利用ip route命令设置192.168.60.0/24的路由经过接口seu0。

```
root@9fea8514c7f2:/# ip route add 192.168.60.0/24 dev seu0
root@9fea8514c7f2:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev seu0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev seu0 scope link
```

在主机U上ping主机V，得到结果如下，可知无法连接。

```
root@9fea8514c7f2:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2026ms
```

在VPN服务器上利用root权限运行tun_server程序后，得到结果如下，可知外部为10.9.0.5至0.0.0.0，内部为192.168.53.99至192.168.60.5。

```
root@4f0820f90916:/volumes# python3 tun_server.py
10.9.0.5:36415 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:36415 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:36415 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

Task 4: Set Up the VPN Server

修改tun_client.py文件，代码如下。

```
#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
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))
print("Interface Name: {}".format(ifname))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    packet = os.read(tun, 2048)
    if packet:
        sock.sendto(packet, ("10.9.0.11", 9090))

```

修改tun_server.py文件，代码如下。

```

#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.1/24 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))
    os.write(tun, bytes(pkt))

```

在主机U上利用root权限运行tun_client程序后，创建tun接口seu0，并且设置192.168.60.0/24的路由经过接口seu0。


```
root@b531d2abe50e:/volumes# python3 tun_client.py
Interface Name: seu0
```

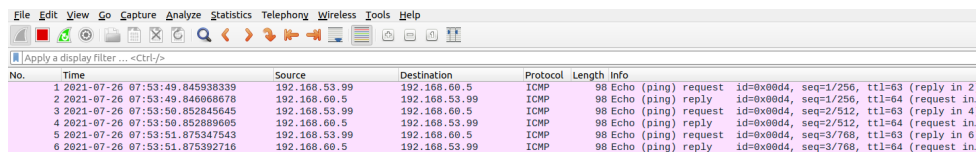
在主机U上ping主机V，得到结果如下，可知无法连接。

```
root@b531d2abe50e:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms
```

在VPN服务器上利用root权限运行tun_server程序后，得到结果如下，可知外部为10.9.0.5至0.0.0.0，内部为192.168.53.99至192.168.60.5。

```
root@386cb63eb798:/volumes# python3 tun_server.py
10.9.0.5:42160 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42160 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42160 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42160 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
```

利用wireshark抓包，得到结果如下，可知ICMP请求和响应报文都已经发送，但主机U未收到ICMP响应报文。



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-26 07:53:49.845938339	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x00d4, seq=1/256, ttl=63 (reply in 2)
2	2021-07-26 07:53:49.846068678	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x00d4, seq=1/256, ttl=64 (request in...)
3	2021-07-26 07:53:50.852845645	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x00d4, seq=2/512, ttl=63 (reply in 4)
4	2021-07-26 07:53:50.852889665	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x00d4, seq=2/512, ttl=64 (request in...)
5	2021-07-26 07:53:51.875347543	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x00d4, seq=3/768, ttl=63 (reply in 6)
6	2021-07-26 07:53:51.875392716	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x00d4, seq=3/768, ttl=64 (request in...)

Task 5: Handling Traffic in Both Directions

修改tun_client.py文件，代码如下。

```
#!/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

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d' % IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
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))
IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, 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, ("10.9.0.11", 9090))

```

修改tun_server.py文件，代码如下。

```

#!/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
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'seu%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
os.system("ip addr add 192.168.53.1/24 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:
    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, ("10.9.0.5", 9090))

```

在主机U上ping主机V，得到结果如下，可知能够连接。

```

root@b531d2abe50e:/# 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=3.36 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=3.79 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=4.26 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 3.358/3.802/4.262/0.369 ms

```

在主机U上利用root权限运行tun_client程序后，得到结果如下。

```

root@b531d2abe50e:/volumes# python3 tun_client.py
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99

```

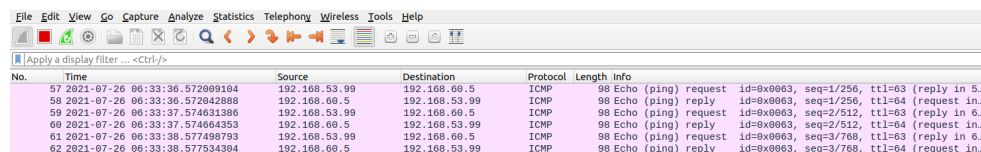
在VPN服务器上利用root权限运行tun_server程序后，得到结果如下。

```

root@386cb63eb798:/volumes# python3 tun_server.py
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99

```

利用wireshark抓包，得到结果如下，可知ICMP请求和响应报文传输成功。



No.	Time	Source	Destination	Protocol	Length	Info
57	2021-07-26 06:33:36.572009104	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x0063, seq=1/256, ttl=63 (reply in 5...
58	2021-07-26 06:33:36.572042888	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x0063, seq=1/256, ttl=64 (request in...
59	2021-07-26 06:33:37.574631386	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x0063, seq=2/512, ttl=63 (reply in 6...
60	2021-07-26 06:33:37.574664353	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x0063, seq=2/512, ttl=64 (request in...
61	2021-07-26 06:33:38.577498793	192.168.53.99	192.168.60.5	ICMP	98	Echo (ping) request id=0x0063, seq=3/768, ttl=63 (reply in 6...
62	2021-07-26 06:33:38.577534384	192.168.60.5	192.168.53.99	ICMP	98	Echo (ping) reply id=0x0063, seq=3/768, ttl=64 (request in...

在主机U上telnet远程连接主机V，得到结果如下，可知连接成功。

```

root@b531d2abe50e:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
26a91c7fea22 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

```

在主机U上利用root权限运行tun_client程序后，得到结果如下。

```

root@b531d2abe50e:/volumes# python3 tun_client.py
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99

```

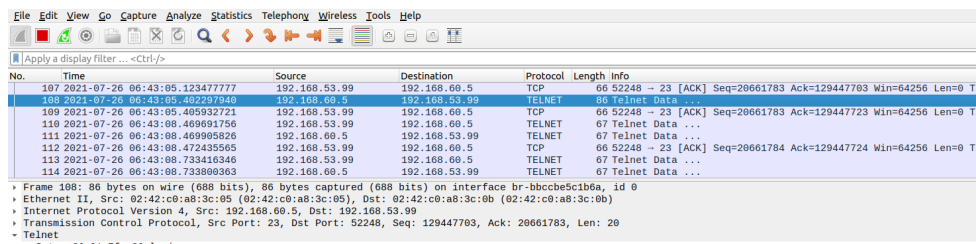
在VPN服务器上利用root权限运行tun_server程序后，得到结果如下。

```

root@386cb63eb798:/volumes# python3 tun_server.py
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99

```

利用wireshark抓包，得到结果如下，可知telnet报文传输成功。



No.	Time	Source	Destination	Protocol	Length	Info
107	2021-07-26 06:43:05.123477777	192.168.53.99	192.168.60.5	TCP	66	52248 → 23 [ACK] Seq=20661783 Ack=129447703 Win=64256 Len=0 T...
108	2021-07-26 06:43:05.402297949	192.168.60.5	192.168.53.99	TELNET	86	Telnet Data ...
109	2021-07-26 06:43:05.409932721	192.168.53.99	192.168.60.5	TCP	66	52248 → 23 [ACK] Seq=20661783 Ack=129447723 Win=64256 Len=0 T...
110	2021-07-26 06:43:08.469891756	192.168.53.99	192.168.60.5	TELNET	67	Telnet Data ...
111	2021-07-26 06:43:08.46995826	192.168.60.5	192.168.53.99	TELNET	67	Telnet Data ...
112	2021-07-26 06:43:08.472435565	192.168.53.99	192.168.60.5	TCP	66	52248 → 23 [ACK] Seq=20661784 Ack=129447724 Win=64256 Len=0 T...
113	2021-07-26 06:43:08.733416346	192.168.53.99	192.168.60.5	TELNET	67	Telnet Data ...
114	2021-07-26 06:43:08.733898363	192.168.60.5	192.168.53.99	TELNET	67	Telnet Data ...

▶ Frame 108: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface br-bbcb5c1b6a, id 0
 ▶ Ethernet II, Src: 02:42:c0:a8:3c:05 (02:42:c0:a8:3c:05), Dst: 02:42:c0:a8:3c:0b (02:42:c0:a8:3c:0b)
 ▶ Internet Protocol Version 4, Src: 192.168.60.5, Dst: 192.168.53.99
 ▶ Transmission Control Protocol, Src Port: 23, Dst Port: 52248, Seq: 129447783, Ack: 20661783, Len: 20
 - Telnet
 Data: 26a91c7fea22 login:

根据实验结果可知，流程是主机U将数据包从192.168.53.99的接口seu0进入隧道，然后利用socket封装发送到VPN服务器，再从VPN服务器的192.168.53.1的接口seu0离开隧道，经过解封装和转发使主机V得到数据包；而主机V将数据包发送到VPN服务器的192.168.53.1的接口seu0进入隧道，然后利用socket封装和转发到主机U，再从主机U的192.168.53.99的接口seu0离开隧道，经过解封装使主机U得到数据包。

Task 6: Tunnel-Breaking Experiment

在VPN服务器上利用Ctrl+C停止运行run_server程序如下。

```

^CTraceback (most recent call last):
  File "tun_server.py", line 23, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt

```

在telnet客户端输入qwertyuiop，得到结果如下，可知未显示字符。

```

seed@26a91c7fea22:~$ whoami
seed
seed@26a91c7fea22:~$

```

在VPN服务器上利用root权限重新运行tun_server程序后，得到结果如下，可知telnet重新连接。

```

root@386cb63eb798:/volumes# python3 tun_server.py
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5

```

在telnet客户端查看结果如下，可知之前输入的字符现在已经显示。

```

seed@26a91c7fea22:~$ whoami
seed
seed@26a91c7fea22:~$ qwertyuiop█

```

该现象的原因是tun_server程序停止后，无法传输报文但连接并未结束，因此重新运行tun_server程序后，未传输成功的报文会重新进行传输。