Task1： Network Setup

主机 U ping VPN，可以正常通信：

```
root@839aa046cdc8:/# 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=0.134 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.072 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.072 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.073 ms
^C
--- 10.9.0.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.067/0.083/0.134/0.025 ms
root@839aa046cdc8:/#
```

VPN ping V 主机，可以正常通信：

```
root@d6b941997610:/# 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=0.105 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.079 ms
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.070/0.083/0.105/0.011 ms
```

主机 U ping 主机 V，不能正常通信：

```
root@4fec12169775:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

在 VPN 服务器上执行 tcpdump 命令，两个端口都可以正常嗅探到网络流量：

嗅探 eth0 端口：

```
root@d6b941997610:/# 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
21:12:54.344550 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, length
 64
21:12:54.344612 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length 6
4
21:12:54.737740 IP6 fe80::42:89ff:fe77:79.5353 > ff02::fb.5353: 0 [2q] PTR (QM)?
 _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
21:12:54.867644 IP6 fe80::547a:aeff:fe02:b7a6.5353 > ff02::fb.5353: 0 [2q] PTR (
QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
21:12:55.369552 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, length
 64
21:12:55.369624 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length 6
4
21:12:56.393504 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 3, length
 64
21:12:56.393573 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 3, length 6
4
21:12:57.417293 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 4, length
 64
21:12:57.417345 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 4, length 6
4
```

嗅探 eth1 端口：

```
root@d6b941997610:/# 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
21:11:16.009400 IP6 fe80::42:2dff:fe81:dcfe > ff02::2: ICMP6, router solicitatio
n, length 16
21:12:18.529906 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 1
, length 64
21:12:18.529952 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 1,
length 64
21:12:19.561957 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 2
, length 64
21:12:19.562025 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 2,
length 64
21:12:20.585751 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 3
, length 64
21:12:20.585801 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 3,
length 64
21:12:21.609665 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 32, seq 4
, length 64
21:12:21.609717 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 32, seq 4,
length 64
```

Task2： Create and Configure TUN Interface

Task2.A

修改程序如下所示，端口名修改为 liu0：

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

在主机 U 上运行程序，结果如下：

```
root@4fec12169775:/volumes# tun.py
Interface Name: liu0
```

另起终端查看主机 U 端口们可以观察到 liu0 端口：

```
[07/27/21]seed@VM:~/.../Labsetup$ docksh 4f
root@4fec12169775:/# ip address
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: liu0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
ult qlen 500
    link/none
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup 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
```

Task2.B:

添加以下代码为 liu0 端口自动分配 IP 地址：

```
25
26 os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
27 os.system("ip link set dev {} up".format(ifname))
```

再次运行程序，查看主机 U 端口，可以看到 liu0 被分配了 192.168.53.99 的地址：

```
root@4fec12169775:/# ip address
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
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
roup 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
7: liu0: <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 liu0
       valid_lft forever preferred_lft forever
```

Task2.C:

修改程序中 while 循环部分如下：

```
26 while True:
27         # Get a packet from the tun interface
28         packet = os.read(tun, 2048)
29         if packet:
30                 ip = IP(packet)
31                 print(ip.summary())
32
```

执行程序，同时 ping 192.168.53.1:

```
root@4fec12169775:/#
root@4fec12169775:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
```

```
root@4fec12169775:/volumes# tun.py
Interface Name: liu0
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
```

可以观察到 ping 不通，因为 ping 192.168.53.1 发送的请求报文要经过 liu0 端口而被捕获。

执行程序，同时 ping 192.168.60.1：

```
root@4fec12169775:/# 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=0.058 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.074 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.072 ms
64 bytes from 192.168.60.1: icmp_seq=6 ttl=64 time=0.072 ms
```

```
root@4fec12169775:/volumes# tun.py
Interface Name: liu0
```

可以观察到能正常 ping 通，因为 ping 192.168.60.1 发送的请求报文不经过 liu0 端口，不会被捕获。

Task2.d:

修改程序中的 while 循环如下来针对请求报文构造响应包：

```
26 while True:
27     # Get a packet from the tun interface
28     packet = os.read(tun, 2048)
29     if packet:
30         ip = IP(packet)
31         print(ip.summary())
32         if ICMP in ip:
33             if ip[ICMP].type == 8 and ip[ICMP].code == 0:
34                 newip = IP(src = ip.dst, dst = ip.src, ttl = 30)
35                 newicmp = ICMP(type = 0, id = ip[ICMP].id, seq = ip[ICMP].seq)
36                 data = ip[Raw].load
37                 newpkt = newip/newicmp/data
38                 os.write(tun, bytes(newpkt))
39
```

执行程序，同时 ping 192.168.53.1：

```
root@4fec12169775:/# 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=30 time=18.2 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=30 time=3.15 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=30 time=2.35 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=30 time=2.50 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=30 time=2.77 ms
^C
--- 192.168.53.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 2.350/5.797/18.218/6.216 ms
```

```
root@4fec12169775:/volumes# tun.py
Interface Name: liu0
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
```

可以观察到能够正常 ping 通，证明成功构造了响应包。

再次修改 while 循环，写入任意数据：

```
25
26 while True:
27     # Get a packet from the tun interface
28     packet = os.read(tun, 2048)
29     if packet:
30         os.write(tun, b"hello")
31
```

运行程序，同时 ping 192.168.53.1：

```
root@4fec12169775:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
```

```
root@4fec12169775:/volumes# tun.py
Interface Name: liu0
```

查看 liu0 端口：

```
root@4fec12169775:/# tcpdump -i liu0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on liu0, link-type RAW (Raw IP), capture size 262144 bytes
22:09:53.833553 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 136, seq
22, length 64
22:09:53.835339 [|ip6]
22:09:54.860867 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 136, seq
23, length 64
22:09:54.860953 [|ip6]
22:09:55.881642 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 136, seq
24, length 64
22:09:55.881713 [|ip6]
```

可以观察到没有 ping 通，是因为发送的数据格式不正确，但数据仍然正常发送出去了。

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

编写 tun_server.py 程序如下：

```
tun_server.py
~/Desktop/Labs_20.04/Network Security/VPN Tunneling Lab/Labsetup/volumes

#!/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))
```

修改 tun.py 程序中以下部分并重新命名为 tun_client.py：

```
# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        # Send the packet via the tunnel
        sock.sendto(packet, ("10.9.0.11", 9090))
```

在主机 U 上运行 tun_client.py，在 VPN 服务器上运行 tun_server.py。同时在主机 U 中 ping 192.168.53.5，VPN 服务器输出如下所示：

```
root@c144a3694eeb:/volumes# python3 tun_server.py
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:33827 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
```

可以观察到，VPN 服务器成功捕获了主机 U 发送到 9090 端口的报文。

重复上述操作改为 ping 主机 V：

```
root@c144a3694eeb:/volumes# python3 tun_server.py
```

发现不能 ping 通，因为主机 U 上没有指向主机 V 子网的路由。

在 tun_client.py 中添加如下代码用于自动配置指向主机 V 子网的路由：

```
5 os.system("ip route add 192.168.60.0/24 dev {} via 192.168.53.99".format(ifname))
```

重新操作，可以观察到 VPN 端就收到了报文，路由配置成功：

```
root@c144a3694eeb:/volumes# python3 tun_server.py
10.9.0.5:49176 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49176 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49176 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49176 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

Task4： Set Up the VPN Server

修改 tun_server.py 代码，建立主机 U 到主机 V 的单向通道：

```
 8 TUNSETIFF = 0x400454ca
 9 IFF_TUN   = 0x0001
10 IFF_TAP   = 0x0002
11 IFF_NO_PI = 0x1000
12
13 # Create the tun interface
14 tun = os.open("/dev/net/tun", os.O_RDWR)
15 ifr = struct.pack('16sH', b'liu%d', IFF_TUN | IFF_NO_PI)
16 ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
17
18 # Get the interface name
19 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
20 print("Interface Name: {}".format(ifname))
21
22 os.system("ip addr add 192.168.53.100/24 dev {}".format(ifname))
23 os.system("ip link set dev {} up".format(ifname))
24
25 IP_A = "0.0.0.0"
26 PORT = 9090
27 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
28 sock.bind((IP_A, PORT))
29 while True:
30     data, (ip, port) = sock.recvfrom(2048)
31     print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
32     pkt = IP(data)
33     print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
34     os.write(tun, data)
35
```

在主机 U 上运行 tun_client.py，在 VPN 服务器上运行 tun_server.py。同时在主机 U 中 ping 192.168.53.5，VPN 服务器输出如下所示：

```
root@c144a3694eeb:/volumes# python3 tun_server.py
Interface Name: liu0
10.9.0.5:58489 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:58489 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:58489 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:58489 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:58489 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

在 VPN 服务器端嗅探 liu0 端口可以发现请求报文成功到达了主机 V，但是因为通道是单向的，响应报文不能到达主机 U：

```
root@c144a3694eeb:/# tcpdump -i liu0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on liu0, link-type RAW (Raw IP), capture size 262144 bytes
15:32:51.425320 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 122, seq
1, length 64
15:32:51.425524 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 122, seq 1,
 length 64
15:32:52.454579 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 122, seq
2, length 64
15:32:52.454656 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 122, seq 2,
 length 64
15:32:53.475184 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 122, seq
```

Task5:Handling Traffic in Both Directions

修改两个程序来建立反向通道:

tun_client.py 程序：

```
# Create UDP socket
IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))

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, bytes(pkt))

        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))
```

tun_server.py 程序:

```
24
25 IP_A = "0.0.0.0"
26 PORT = 9090
27 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
28 sock.bind((IP_A, PORT))
29 while True:
30     ready,_,_ = select.select([sock, tun],[],[])
31
32     for fd in ready:
33         if fd is sock:
34             data, (ip, port) = sock.recvfrom(2048)
35             pkt = IP(data)
36             print("From socket <==: {} --> {}".format('10.9.0.5', 9090, IP_A, PORT))
37             os.write(tun, bytes(pkt))
38
39         if fd is tun:
40             packet = os.read(tun, 2048)
41             pkt = IP(packet)
42             print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
43             sock.sendto(packet, ("10.9.0.5", 9090))
44
```

重复之前的测试，可以发现主机 U 成功 ping 通主机 V，证明通道搭建成功：

```
root@1f9fc7623d4b:/#
root@1f9fc7623d4b:/# 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=3.73 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=2.48 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=2.92 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=2.59 ms
```

```
root@1f9fc7623d4b:/volumes# python3 tun_client.py
Interface Name: liu0
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
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
```

```
root@c144a3694eeb:/volumes# python3 tun_server.py
Interface Name: liu0
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

使用主机 U Telnet 链接主机 V，可以成功连接:

```
root@1f9fc7623d4b:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
bb6011cc8dad login:
```

```
root@1f9fc7623d4b:/volumes# python3 tun_client.py
Interface Name: liu0
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
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
```

```
root@c144a3694eeb:/volumes# python3 tun_server.py
Interface Name: liu0
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 10.9.0.5 --> 9090
From socket <==: 10.9.0.5 --> 9090
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

在测试 ping 指令同时使用 wireshark 进行抓包：

No.	Time	Source	Destination	Protocol	Length	Info
22	2021-07-30 11:5…	192.168.60.5	192.168.53.99	ICMP	100	Echo (ping) reply id=0x00a2, seq=1/256, ttl=64
23	2021-07-30 11:5…	10.9.0.11	10.9.0.5	UDP	128	9090 → 9090 Len=84
24	2021-07-30 11:5…	10.9.0.11	10.9.0.5	UDP	128	9090 → 9090 Len=84
26	2021-07-30 11:5…	10.9.0.5	10.9.0.11	UDP	128	9090 → 9090 Len=84
27	2021-07-30 11:5…	10.9.0.5	10.9.0.11	UDP	128	9090 → 9090 Len=84
28	2021-07-30 11:5…	192.168.53.99	192.168.60.5	ICMP	100	Echo (ping) request id=0x00a2, seq=2/512, ttl=63 (no respons…
29	2021-07-30 11:5…	192.168.53.99	192.168.60.5	ICMP	100	Echo (ping) request id=0x00a2, seq=2/512, ttl=63 (reply in 3…
30	2021-07-30 11:5…	192.168.60.5	192.168.53.99	ICMP	100	Echo (ping) reply id=0x00a2, seq=2/512, ttl=64 (request in…
31	2021-07-30 11:5…	192.168.60.5	192.168.53.99	ICMP	100	Echo (ping) reply id=0x00a2, seq=2/512, ttl=64
32	2021-07-30 11:5…	10.9.0.11	10.9.0.5	UDP	128	9090 → 9090 Len=84
33	2021-07-30 11:5…	10.9.0.11	10.9.0.5	UDP	128	9090 → 9090 Len=84
35	2021-07-30 11:5…	10.9.0.5	10.9.0.11	UDP	128	9090 → 9090 Len=84
36	2021-07-30 11:5…	10.9.0.5	10.9.0.11	UDP	128	9090 → 9090 Len=84

```
Frame 7: 113 bytes on wire (904 bits), 113 bytes captured (904 bits) on interface any, id 0
```

从抓包结果可以看到，请求报文从主机 U 发向主机 V，报文先通过 tun 到达 VPN 服务器，VPN 服务器通过 tun 发往主机 V 报文；然后主机 V 返回响应报文通过 tun 达到 VPN 服务器，VPN 服务器通过 tun 将响应报文发给主机 U，从而完成主机 U 和主机 V 之间的一次通信。

Task6: Tunnel-Breaking Experiment

在主机 U 上 telnet 链接主机 V，正常可以执行指令，但终止建立通道的两个程序后，无法再输入字符：

```
seed@bb6011cc8dad:~$ touch test.txt
seed@bb6011cc8dad:~$ ls
test.txt
seed@bb6011cc8dad:~$
```

再次执行程序建立通道，可以发现在无法输入字符时键盘的敲击记录又显示了出来：

```
seed@bb6011cc8dad:~$ touch test.txt
seed@bb6011cc8dad:~$ ls
test.txt
seed@bb6011cc8dad:~$ asdkjfhasdbvasbvlkajsdbvkasdbvjasvdjasdvg
```

可以看到，在停止程序后，通道中断了但是 telnet 链接没有中断，期间输入的字符保存在缓存区中，此时恢复通道，缓存区中的报文发送到主机 V 端，就会显示出来，如果这个期间间隔较长时间就无法再显示中断期间键盘的输入。