VPN Lab: The Container Version

57118225 宋雨帆

Task 1: Network Setup

1.主机 V ping VPN 服务器：

```
[07/27/21]seed@VM:~/.../Labsetup$ docksh 6f
root@6ff5d3ccf851:/# 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=0.070
ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.048
ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.049
ms
^C
--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, tim
e 2051ms
rtt min/avg/max/mdev = 0.048/0.055/0.070/0.010 ms
```

可以 ping 通

2.VPN 服务器 ping 主机 V:

```
root@c770cbcb3268:/# 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=0.0
49 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.0
53 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.0
51 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, tim
e 2027ms
rtt min/avg/max/mdev = 0.049/0.051/0.053/0.001 ms
```

可以 ping 通

3.主机 U 上 ping 主机 V:

```
[07/27/21]seed@VM:~/.../Labsetup$ docksh 91
root@91b09d1d9d0b:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

Ping 不通。

4. 路由器上运行 tcpdump：

嗅探接口 eth0:

```
root@c770cbcb3268:/# 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
20:21:44.758547 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
20:21:44.758588 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
20:21:44.758630 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 13, seq 1, length 64
20:21:44.758658 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 13, seq 1, length 64
20:21:45.778620 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 13, seq 2, length 64
20:21:45.778673 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 13, seq 2, length 64
20:21:46.802332 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 13, seq 3, length 64
20:21:46.802387 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 13, seq 3, length 64
20:21:49.780294 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
20:21:49.780502 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

嗅探接口 eth1:

```
root@c770cbcb3268:/# 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
20:19:25.462411 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 35, seq 1, length 64
20:19:25.462467 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 35, seq 1, length 64
20:19:26.484633 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 35, seq 2, length 64
20:19:26.484685 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 35, seq 2, length 64
20:19:27.506222 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 35, seq 3, length 64
20:19:27.506266 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 35, seq 3, length 64
20:19:30.514109 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
20:19:30.514365 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
```

网络流量都可以正常嗅探。

配置正常。

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

在代码中修改端口名为“song”：

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

在主机 U 上运行程序：

```
root@91b09d1d9d0b:/# cd volumes
root@91b09d1d9d0b:/volumes# tun.py
Interface Name: song0
```

打开另一个终端查看：

```
[07/27/21]seed@VM:~/.../Labsetup$ docksh 91
root@91b09d1d9d0b:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN grou
p 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: song0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN g
roup default qlen 500
    link/none
22: eth0@if23: <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
root@91b09d1d9d0b:/#
```

端口名成功修改为 song0。

Task 2.b: Set up the TUN Interface

程序中添加两行代码给端口 song0 自动分配 ip 地址：

```
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:
```

再次运行程序，并执行 ip address 命令：

```
root@91b09d1d9d0b:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault 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: song0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel s
tate UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global song0
       valid_lft forever preferred_lft forever
22: eth0@if23: <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
```

此时端口已经被成功分配了 ip 地址。

Task 2.c: Read from the TUN Interface

修改程序中的 while 循环：

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

再次执行程序。并 ping 192.168.53.0/24 网段中任一主机，这里我们 ping 192.168.53.2：

```
root@91b09d1d9d0b:/# ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
^C
--- 192.168.53.2 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5107ms
```

```
root@91b09d1d9d0b:/volumes# tun.py
Interface Name: song0
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
```

此时 ping 不通，从 run.py 程序的输出可以知道 ICMP 请求报文都被端口捕获了，因为发送给 192.168.53.0/24 的数据包是从 song0 端口发出。

Ping 192.168.60.1 时：

```
root@91b09d1d9d0b:/# 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=0.172 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.044 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.127 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.155 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 4095ms
rtt min/avg/max/mdev = 0.044/0.108/0.172/0.054 ms
```

此时能 ping 通，且此时程序没有输出。这是因为发送给 192.168.60.1 的报文不经过 song0 端口，所以没有捕获报文。

Task 2.d: Write to the TUN Interface

修改 while 循环如下：

```
while True:
    packet = os.read(tun,2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())
        if ICMP in ip:
          if ip[ICMP].type == 8 and ip[ICMP].code == 0:
            newip = IP(src=ip.dst, dst=ip.src,ttl = 64)
            newicmp = ICMP(type=0,id = ip[ICMP].id,seq =
ip[ICMP].seq)
            if ip[Raw].load:
              data = ip[Raw].load
              newpkt = newip/newicmp/data
            else:
              newpkt = newip/newicmp
            os.write(tun,bytes(newpkt))
```

这里判断是否为 Echo request 包，然后将请求包源地址和目的地址交换，构造响应包，负载为原来数据包的负载。

运行程序，然后再次 ping 192.168.53.2：

```
root@91b09d1d9d0b:/# ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
64 bytes from 192.168.53.2: icmp_seq=1 ttl=64 time=5.18 ms
64 bytes from 192.168.53.2: icmp_seq=2 ttl=64 time=1.68 ms
64 bytes from 192.168.53.2: icmp_seq=3 ttl=64 time=7.27 ms
^C
--- 192.168.53.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 1.680/4.710/7.271/2.306 ms


root@91b09d1d9d0b:/volumes# tun.py
Interface Name: song0
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
```

此时能够 ping 通，说明我们伪造响应包成功。

修改 while 循环，不写入 ip 数据包而是任意数据：

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

运行程序，然后再次 ping 192.168.53.2：

```
root@91b09d1d9d0b:/# ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
^C
--- 192.168.53.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6137m
s
```

```
root@91b09d1d9d0b:/volumes# tun.py
Interface Name: song0
```

Ping 不通，且程序无输出，执行 tcpdump -i song0 -n 命令：

```
root@91b09d1d9d0b:/# tcpdump -i song0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on song0, link-type RAW (Raw IP), capture size 262144 bytes
04:02:00.005526 IP 192.168.53.99 > 192.168.53.2: ICMP echo request, id 85, seq 1, length 64
04:02:00.005902 [|ip6]
04:02:01.032813 IP 192.168.53.99 > 192.168.53.2: ICMP echo request, id 85, seq 2, length 64
04:02:01.032861 [|ip6]
04:02:02.057955 IP 192.168.53.99 > 192.168.53.2: ICMP echo request, id 85, seq 3, length 64
04:02:02.058005 [|ip6]
04:02:03.081812 IP 192.168.53.99 > 192.168.53.2: ICMP echo request, id 85, seq 4, length 64
04:02:03.081867 [|ip6]
```

可以看到发送的任意数据确实发送出去了，但是其不符合报文格式，没有什么用。

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

将 tun.py 程序的 while 循环修改为如下代码即为 tun_client.py：

```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'song%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))
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 上运行 tun_client.py，在 VPN 服务器上运行 tun_server.py。然后在主机 U 中 ping 192.168.53.5

```
root@91b09d1d9d0b:/# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
^C
--- 192.168.53.5 ping statistics ---
1072 packets transmitted, 0 received, 100% packet loss, time 109
6707ms
```

VPN 服务器输出如下：

```
root@c770cbcb3268:/volumes# tun_server.py
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:44874 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.5
```

此时 VPN 服务器成功捕获到了报文。这是因为 tun_client.py 程序将捕获的报文发给了 VPN 服务器的 9090 端口。

在主机 U 上 Ping 主机 V:

```
root@91b09d1d9d0b:/# 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 ---
43 packets transmitted, 0 received, 100% packet loss, time 43010
ms
```

```
root@c770cbcb3268:/volumes# tun_server.py
```

此时 VPN 服务器没有输出，这是因为此时主机 U 上没有去往 192.168.60.0/24 的路由，报文不会从 tun 端口发出。

在 tun_client.py 中添加如下代码用于自动配置路由：

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

重复操作：

```
root@c770cbcb3268:/volumes# tun_server.py
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

此时 VPN 服务器有输出，说明 tun_server.py 通过隧道接收到报文，实验成功。

Task 4: Set Up the VPN Server

修改 tun_server.py 代码使得建立一个 tun 接口将数据包路由到最终目的地，增加的代码类似于 task2：

```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'song%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))
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,data)
```

重复 task3 的操作，ping 192.168.60.5,tun_server.py 输出如下：

```
root@c770cbcb3268:/volumes# tun_server.py
Interface Name: song0
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:49644 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.5
```

我们可以在 VPN 服务器上嗅探 song0 端口：

```
root@91b09d1d9d0b:/# tcpdump -i song0 -n
tcpdump: verbose output suppressed, use -v or -vv for full proto
col decode
listening on song0, link-type RAW (Raw IP), capture size 262144
bytes
10:15:42.570447 IP 192.168.53.99 > 192.168.60.5: ICMP echo reque
st, id 126, seq 36077, length 64
10:15:43.596457 IP 192.168.53.99 > 192.168.60.5: ICMP echo reque
st, id 126, seq 36078, length 64
10:15:44.618286 IP 192.168.53.99 > 192.168.60.5: ICMP echo reque
st, id 126, seq 36079, length 64
10:15:45.644683 IP 192.168.53.99 > 192.168.60.5: ICMP echo reque
st, id 126, seq 36080, length 64
10:15:46.667055 IP 192.168.53.99 > 192.168.60.5: ICMP echo reque
st, id 126, seq 36081, length 64
```

发现 ICMP 请求包成功通过隧道到达主机 V，且受到了主机 V 的 ICMP 响应包。但是此时还没有设置完成，此时隧道只有一个方向，故响应包无法到达主机 U。

Task 5: Handling Traffic in Both Directions

为建立另一个方向的隧道，我们修改代码中的 while 部分：

tun_client.py:

```
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,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))
```

如果数据包来自 tun 接口，则发给主机 U，如果数据包来自 socket 接口，则发给隧道。

Tun_server.py:

```
20 os.system("ip addr add 192.168.53.100/24 dev {}".format(ifname))
21 os.system("ip link set dev {} up".format(ifname))
22 IP_A = "0.0.0.0"
23 PORT = 9090
24 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
25 sock.bind((IP_A, PORT))
26 while True:
27    ready,_,_ = select.select([sock,tun],[],[])
28    for fd in ready:
29        if fd is sock:
30           data,(ip,port) = sock.recvfrom(2048)
31           pkt = IP(data)
32           print("From socket <==:{} --> {}".format('10.9.0.5',
   9090,IP_A,PORT))
33           os.write(tun,bytes(pkt))
34
35        if fd is tun:
36           packet = os.read(tun,2048)
37           pkt = IP(packet)
38           print("From tun ==>:{}-->{}".format(pkt.src,pkt.dst))
39           sock.sendto(packet,("10.9.0.5",9090))
```

如果数据包来自 tun 接口，则发给主机 V，如果数据包来自 socket 接口，则发给隧道。

重复之前的操作, ping 192.168.60.5:

```
root@91b09d1d9d0b:/# 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=2.73 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=1.69 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=1.69 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.71 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.61 ms
```

成功 ping 通。Tun_client.py 和 tun_server.py 输出如下：

```
root@91b09d1d9d0b:/volumes# tun_client.py
Interface Name: song0
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
```

```
root@c770cbcb3268:/volumes# tun_server.py
Interface Name: song0
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
```

建立 Telnet 连接：

```
root@91b09d1d9d0b:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6ff5d3ccf851 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu Jul 29 10:56:50 UTC 2021 on pts/2
seed@6ff5d3ccf851:~$
```

Telnet 连接也成功建立。

捕获 ping 过程中的数据包，查看 wireshark：

1	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
2	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
3	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=1/256, ttl=63 (no respons…
4	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=1/256, ttl=63 (reply in 5)
5	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=1/256, ttl=64 (request in…
6	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=1/256, ttl=64
7	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
8	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
9	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
10	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
11	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=2/512, ttl=63 (no respons…
12	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=2/512, ttl=63 (reply in 1…
13	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=2/512, ttl=64 (request in…
14	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=2/512, ttl=64
15	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
16	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
17	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
18	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
19	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=3/768, ttl=63 (no respons…
20	2021-07-29 07:0…	192.168.53.99	192.168.60.5	ICMP	100 Echo (ping) request id=0x00b8, seq=3/768, ttl=63 (reply in 2…
21	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=3/768, ttl=64 (request in…
22	2021-07-29 07:0…	192.168.60.5	192.168.53.99	ICMP	100 Echo (ping) reply id=0x00b8, seq=3/768, ttl=64
23	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
24	2021-07-29 07:0…	10.9.0.11	10.9.0.5	UDP	128 9090 → 9090 Len=84
25	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84
26	2021-07-29 07:0…	10.9.0.5	10.9.0.11	UDP	128 9090 → 9090 Len=84

数据报文从主机 U 发向主机 V，报文先通过 tun 到达 VPN 服务器，然后 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 服务器又通过 tun 将响应报文发给主机 U，从而完成主机 U 和主机 V 之间的通信。

Task 6: Tunnel-Breaking Experiment

主机 U 向主机 V 建立 Telnet 连接，然后终止程序，发现无法输入任何字符：

```
root@91b09d1d9d0b:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6ff5d3ccf851 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu Jul 29 10:56:50 UTC 2021 on pts/2
seed@6ff5d3ccf851:~$
```

这是因为停止程序后隧道中断，数据包无法到达。

短时间内再次执行程序：

```
seed@6ff5d3ccf851:~$ ls
seed@6ff5d3ccf851:~$ ls
seed@6ff5d3ccf851:~$ ls
seed@6ff5d3ccf851:~$ pwd
/home/seed
seed@6ff5d3ccf851:~$
```

如果此时很快地执行程序恢复隧道，会发现前面中断程序时没能显示的输入会再次显示，Telnet 连接恢复。因为断开程序时的输入会在缓存区中一直发送报文，如果恢复连接比较快速，前面的输入仍然会显示。但是较长时间还没再次执行程序就不能显示之前的输入了。