VPN Lab

57118138 李嘉怡

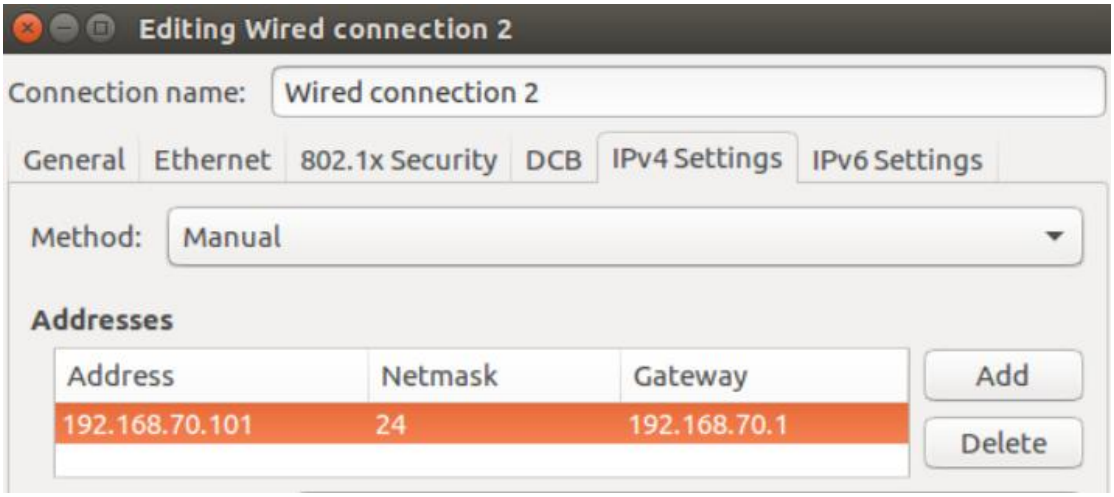
Task 1: Network Setup

Host U（VPN 客户端）的 IP 地址：10.0.2.4
Gateway（VPN 服务器）的 IP 地址：10.0.2.5 和 192.168.70.1
Host V（专用网络中的主机）的 IP 地址：192.168.70.101

配置 Host V（专用网络中的主机）：

网络连接方式设置为“内部网络”：

手动配置 IP 地址：



配置 Gateway（VPN 服务器）：

新建网卡 2，连接方式设置为“内部网络”：

给网卡 2 手动配置 IP 地址：

测试连通性：

Host U 的连通性：可与 VPN 服务器相互通信

```
[09/22/20]seed@VM:~$ ping 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.756 ms
```

与 Host V 不连通：

```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
^C
--- 192.168.70.101 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4100ms
```

VPN Server 的连通性：

与 Host U 相互通信：

```
[09/22/20]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.453 ms
```

与 Host V 相互通信：

```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
64 bytes from 192.168.70.101: icmp_seq=1 ttl=64 time=0.782 ms
```

Host V 的连通性：

可与 VPN 服务器通信：

```
[09/22/20]seed@VM:~$ ping 192.168.70.1
PING 192.168.70.1 (192.168.70.1) 56(84) bytes of data.
64 bytes from 192.168.70.1: icmp_seq=1 ttl=64 time=0.581 ms
```

与 Host U 不连通：

```
[09/22/20]seed@VM:~$ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
^C
--- 10.0.2.4 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9212ms
```

Task 2：Create and Configure TUN Interface

2.A:

1、在 Host U 上修改程序使程序创建的新接口名为 lina0，运行 tun.py：

```
[09/22/20]seed@VM:~/lab7$ sudo ./tun.py
Interface Name: lina0
```

2、在新终端查看，该程序创建了一个名为 lina0 的接口：

```
[09/22/20]seed@VM:~/lab7$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N group default qlen 1
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_
fast state UP group default qlen 1000
    link/ether 08:00:27:bf:a7:39 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic enp0s3
       valid_lft 520sec preferred_lft 520sec
    inet6 fe80::9995:be34:6272:2266/64 scope link
       valid_lft forever preferred_lft forever
6: lina0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state
DOWN group default qlen 500
    link/none
```

2.B:

1、在 tun.py 文件中添加下面两行代码，配置 lina0 接口：

```
#configure the tun interface
os.system("ip addr add 192.168.37.41/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
```

2、在 Host U 上用 ip address 命令查看该接口信息，新增了 IP 地址且网卡处于 UP 状态：

```
10: lina0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdis
c pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet 192.168.37.41/24 scope global lina0
       valid_lft forever preferred_lft forever
    inet6 fe80::3f98:6300:3b12:2cbe/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

2.C:

1、添加代码，使程序可以捕获报文并打印出报文内容：

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

2、Host U 向与 lina0 接口同网段的主机发送 ping 报文：

```
[09/22/20]seed@VM:~/lab7$ ping 192.168.37.5
PING 192.168.37.5 (192.168.37.5) 56(84) bytes of data.
^C
--- 192.168.37.5 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8178ms
```

3、tun.py 程序打印出如下内容，即本机发送的 ping 报文：

```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 6736
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0x54da
  src       = 192.168.37.41
  dst       = 192.168.37.5
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x17c7
     id        = 0x168d
     seq       = 0x9
###[ Raw ]###
        load      = '\x8d\xc3i_\xe2|\x05\x00\x08\t\n\x0b\x0c\r\x0e
\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x
1f !"#$%&\'()*+,-./01234567'
```

Host U 有网段不同的两个接口，主机发送 ping 报文时，发现 lina0 和报文的网段相同，则从该接口发送出去，即该报文的源地址为 lina0 接口的 IP 地址，则程序可以捕获到该报文。

4、向 192.168.70 网段的 IP 地址发送 ping 报文：

```
[09/22/20]seed@VM:~/lab7$ ping 192.168.70.1
PING 192.168.70.1 (192.168.70.1) 56(84) bytes of data.
^C
--- 192.168.70.1 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5114ms
```

程序什么都没有打印，查看 wireshark：

```
11… 2020-… 10.0.2.4        192.168.70.1   ICMP      98 Echo (ping) request
11… 2020-… 10.0.2.4        192.168.70.1   ICMP      98 Echo (ping) request
11… 2020-… 10.0.2.4        192.168.70.1   ICMP      98 Echo (ping) request
```

可以看到，Host U 发送的 ping 报文，以 10.0.2.4 为源地址，即报文从 enp0s3 接口发送出去，而程序创建的是 lina0 接口，故未能捕获该报文。

2.D:

1、编写程序，将从 lina0 接口捕获的包加上 IP 头之后发出去：

```
# Send out a spoof packet using the tun interface
newip = IP(src='1.2.3.4', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

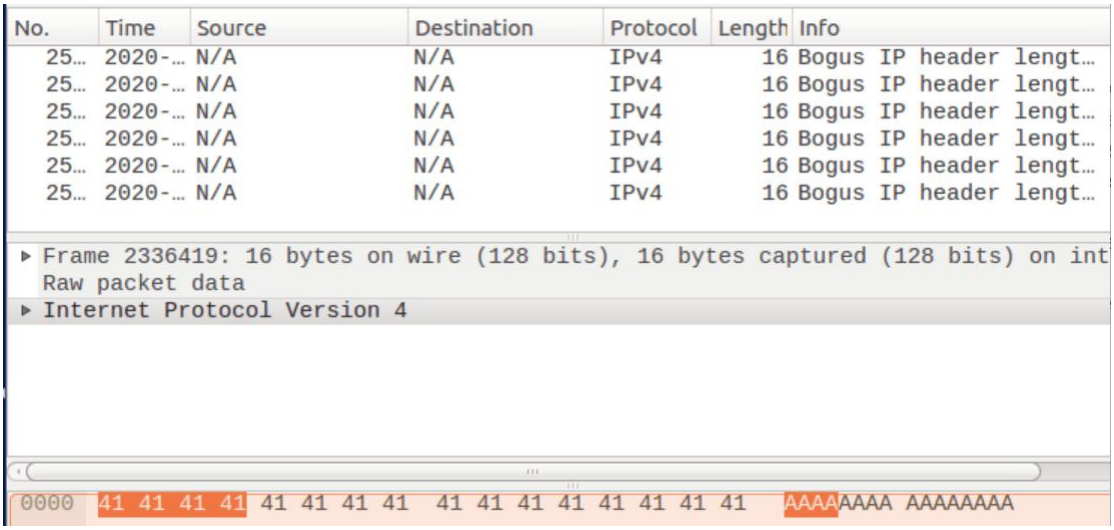
2、打开 wireshark，可以看到本机发送的 ICMP 报文换上新的 IP 头后的报文：

```
 1 2020-… 192.168.37.41   192.168.37.56  ICMP      84 Echo (ping) request
 2 2020-… 1.2.3.4         192.168.37.41  ICMP      84 Echo (ping) request
 3 2020-… 192.168.37.41   192.168.37.56  ICMP      84 Echo (ping) request
 4 2020-… 1.2.3.4         192.168.37.41  ICMP      84 Echo (ping) request
```

3、编写程序将任意数据写入接口：

```
while True:
        os.write(tun,bytes('AAAAAAAAAAAAAAAA'.encode('utf-8')))
```

4、打开 wireshark，可以看到本机发送的只包含数据的报文：



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

1、tun_client.py 的地址填上 VPN 服务器的地址和端口号，在 Host U 上运行：

```
# Send the packet via the tunnel
sock.sendto(packet, ('10.0.2.5', 9090))
```

即 TUN 接口捕获的报文，将发往地址 10.0.2.5:9090。

2、在 VPN 服务器上运行 tun_server.py：

```
[09/22/20]seed@VM:~/lab7$ sudo ./tun_server.py
```

3、在主机 U 上向 192.168.70.0/24 网段的主机发送 ping 报文：

```
[09/22/20]seed@VM:~$ ping 192.168.70.23
PING 192.168.70.23 (192.168.70.23) 56(84) bytes of data.
```

VPN 服务器未打印任何信息。因为未设置路由，ping 报文会直接从网卡发出去，而不会传给 TUN 接口，这样就无法建立隧道，服务器也不会收到该报文。

4、设置路由，将目的地址是 192.168.70.0/24 网段的报文从 lina0 接口转发：

```
[09/22/20]seed@VM:~$ sudo route add -net 192.168.70.0/24 lina0
```

```
[09/22/20]seed@VM:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
  Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0
    0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0
    0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0
    0 enp0s3
192.168.37.0    0.0.0.0         255.255.255.0   U     0      0
    0 lina0
192.168.70.0    0.0.0.0         255.255.255.0   U     0      0
    0 lina0
```

5、再对 Host V 发送 ping 报文：

```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
```

6、VPN 服务器打印内容如下：

```
10.0.2.4:37372 --> 0.0.0.0:9090
 Inside: 192.168.37.41 --> 192.168.70.101
10.0.2.4:37372 --> 0.0.0.0:9090
 Inside: 192.168.37.41 --> 192.168.70.101
```

客户端的 TUN 接口捕获到 ping 报文后，将报文作为数据部分封装在新的 IP 头里，即源地址为 10.0.2.4，宿地址为 10.0.2.5，将新的报文发送到内核，内核通过路由设置选择合适的接口发送给服务器。服务器收到后，将 IP 头去除，取出原始报文，得到原始报文的源地址 192.168.37.41，宿地址为 192.168.70.101。服务器收到报文但未进行转发。

Task4: Set Up the VPN Server

1、修改 tun_server.py，在 VPN 服务器上运行：

```python
#!/usr/bin/python3
import fcntl
import struct
import os
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
IP_A = "10.0.2.5"
PORT = 9090

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lina%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")
#configure the tun interface
os.system("ip addr add 192.168.37.23/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
while True:
        data, (ip, port) = sock.recvfrom(2048)
        pkt = IP(data)
        os.write(tun, bytes(pkt))
```

该程序创建并配置一个 TUN 接口，创建 socket 并绑定地址和端口，将从 socket 接收到的数据写入 TUN 接口。

2、VPN 服务器开启 IP 转发，让其充当网关：

```
[09/22/20]seed@VM:~$  sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

3、Host U 运行 tun_client.py 并向 Host V 发送 ping 报文：

```
[09/22/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
```

4、打开 Host V 上的 wireshark：

→	2	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
←	3	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	4	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	5	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	6	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	7	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	9	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	10	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	11	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	12	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	15	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	16	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	19	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	20	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply
	21	2020-…	192.168.37.41	192.168.70.101	ICMP	100 Echo (ping) request
	22	2020-…	192.168.70.101	192.168.37.41	ICMP	100 Echo (ping) reply

可以看到 Host V 收到了来自 Host U 的 ping 报文，并作出回应。VPN 服务器收到 Host U 的报文后，将 IP 头去除，将原始的 IP 报文发送到内核，由于服务器开启了 IP 转发，内核查找路由，从合适的接口发送出去，到达 Host V。

Task5: Handling Traffic in Both Directions

1、修改 Host U 的程序，将 while 循环内的代码替换如下：

```
while True:
    # this will block until at least one interface is ready
    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, ('10.0.2.5', 9090))
```

select 函数用于同时监听两个接口，当 socket 有报文时，去除 IP 报头将它写到 TUN 接口中，相当于隧道的收方将报文还原；当 TUN 接口有报文时，加上 IP 报头通过 socket 发出去，相当于隧道的发送方将报文封装在隧道中。

Host U 发送报文时，报文经过协议栈到达 TUN 接口，TUN 将报文封装到隧道中再经过路由从合适网卡发送出去；Host V 接收报文时，socket 获取报文，去除 IP 头得到原始报文。

2、修改 VPN 服务器的程序，将 while 循环内的代码替换如下：

```
while True:
    # this will block until at least one interface is ready
    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, ('10.0.2.4', 60122))
```

VPN 服务器转发封装在隧道的报文时，去除 IP 报头得到原始报文，将原始报文写入 TUN 接口，找到合适的路由转发出去；VPN 服务器将非隧道报文封装在 IP 隧道中发送出去时，将原始报文封装在新的 IP 报文中发送出去。

3、在 Host U 上向 Host V 发送 ping 报文：

```
[09/23/20]seed@VM:~$ ping 192.168.70.101
PING 192.168.70.101 (192.168.70.101) 56(84) bytes of data.
64 bytes from 192.168.70.101: icmp_seq=1 ttl=63 time=5.58 ms
64 bytes from 192.168.70.101: icmp_seq=2 ttl=63 time=6.46 ms
64 bytes from 192.168.70.101: icmp_seq=3 ttl=63 time=6.49 ms
```

Host U 收到 Host V 的回应，说明隧道搭建完成，Host U 可访问 Host V。Host V 向 Host U 发送 ping 报文也可收到回应。

服务器端可以看到两者的交互过程：

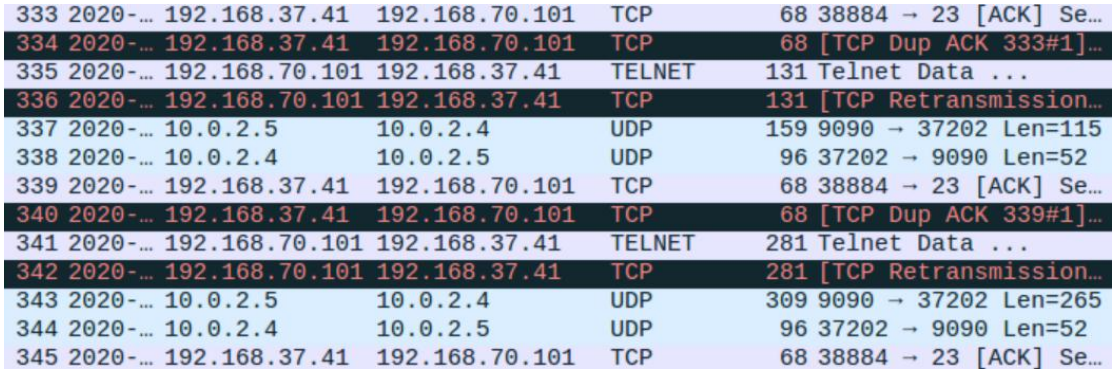
```
From tun ==>: 192.168.70.101 --> 192.168.37.41
From socket <==: 192.168.37.41 --> 192.168.70.101
From socket <==: 192.168.37.41 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.37.41
```

4、在 Host U 上向 Host V 发起 telnet 连接：

```
[09/23/20]seed@VM:~$ telnet 192.168.70.101
Trying 192.168.70.101...
Connected to 192.168.70.101.
```

连接成功。Host V 向 Host U 发起 telnet 连接也可成功。

5、打开服务器的 wireshark：



333	2020-…	192.168.37.41	192.168.70.101	TCP	68	38884 → 23 [ACK] Se…
334	2020-…	192.168.37.41	192.168.70.101	TCP	68	[TCP Dup ACK 333#1]…
335	2020-…	192.168.70.101	192.168.37.41	TELNET	131	Telnet Data ...
336	2020-…	192.168.70.101	192.168.37.41	TCP	131	[TCP Retransmission…
337	2020-…	10.0.2.5	10.0.2.4	UDP	159	9090 → 37202 Len=115
338	2020-…	10.0.2.4	10.0.2.5	UDP	96	37202 → 9090 Len=52
339	2020-…	192.168.37.41	192.168.70.101	TCP	68	38884 → 23 [ACK] Se…
340	2020-…	192.168.37.41	192.168.70.101	TCP	68	[TCP Dup ACK 339#1]…
341	2020-…	192.168.70.101	192.168.37.41	TELNET	281	Telnet Data ...
342	2020-…	192.168.70.101	192.168.37.41	TCP	281	[TCP Retransmission…
343	2020-…	10.0.2.5	10.0.2.4	UDP	309	9090 → 37202 Len=265
344	2020-…	10.0.2.4	10.0.2.5	UDP	96	37202 → 9090 Len=52
345	2020-…	192.168.37.41	192.168.70.101	TCP	68	38884 → 23 [ACK] Se…

可以看到 telnet 通信的过程，U 给 V 发送封装在 UDP 报文中的 telnet 报文，服务器还原后将其发给 V，V 给 U 发送原始 telnet 报文，服务器将其封装在 UDP 报文后将其发给 U。

Task6: Tunnel-Breaking Experiment

1、Host U 通过 telnet 登录 Host V：

```
[09/23/20]seed@VM:~$ telnet 192.168.70.101
Trying 192.168.70.101...
Connected to 192.168.70.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Sep 23 22:53:41 EDT 2020 from 192.168.37.41 on pts
/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

2、将 tun_server.py 程序断开：

```
  File "./tun_server.py", line 32, in <module>
    ready, _, _ = select.select([sock, tun], [], [])
KeyboardInterrupt
```

3、在 Host U 的终端输入指令，无法显示：

```
[09/23/20]seed@VM:~$
```

4、重新开启 tun_server.py 程序：

```
[09/23/20]seed@VM:~/lab7$ sudo ./tun_server.py
From tun ==>: 0.0.0.0 --> 163.217.161.8
From socket <==: 192.168.37.41 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.37.41
From socket <==: 192.168.37.41 --> 192.168.70.101
From socket <==: 192.168.37.41 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.37.41
From socket <==: 192.168.37.41 --> 192.168.70.101
From socket <==: 192.168.37.41 --> 192.168.70.101
From tun ==>: 192.168.70.101 --> 192.168.37.41
From socket <==: 192.168.37.41 --> 192.168.70.101
```

服务器端显示 Host U 和 V 传递了大量报文。

30	2020-…	10.0.2.4	10.0.2.5	UDP	98	37202 → 9090 Len=54
31	2020-…	192.168.37.41	192.168.70.101	TELNET	70	Telnet Data ...
32	2020-…	192.168.37.41	192.168.70.101	TCP	70	[TCP Retransmission…
33	2020-…	192.168.70.101	192.168.37.41	TELNET	70	Telnet Data ...
34	2020-…	192.168.70.101	192.168.37.41	TCP	70	[TCP Retransmission…
35	2020-…	10.0.2.5	10.0.2.4	UDP	98	9090 → 37202 Len=54
36	2020-…	10.0.2.4	10.0.2.5	UDP	99	37202 → 9090 Len=55
37	2020-…	192.168.37.41	192.168.70.101	TELNET	71	Telnet Data ...
38	2020-…	192.168.37.41	192.168.70.101	TCP	71	[TCP Retransmission…
39	2020-…	192.168.70.101	192.168.37.41	TELNET	70	Telnet Data ...
40	2020-…	192.168.70.101	192.168.37.41	TCP	70	[TCP Retransmission…
41	2020-…	10.0.2.5	10.0.2.4	UDP	98	9090 → 37202 Len=54
42	2020-…	10.0.2.4	10.0.2.5	UDP	97	37202 → 9090 Len=53
43	2020-…	192.168.37.41	192.168.70.101	TELNET	69	Telnet Data ...
44	2020-…	192.168.37.41	192.168.70.101	TCP	69	[TCP Keep-Alive] 38…
45	2020-…	192.168.70.101	192.168.37.41	TELNET	1088	Telnet Data ...
46	2020-…	192.168.70.101	192.168.37.41	TCP	1088	[TCP Retransmission…
47	2020-…	10.0.2.5	10.0.2.4	UDP	1116	9090 → 37202 Len=10…

wireshark 显示了两个主机中断后建立通信的过程。