VPN Tunneling Lab

刘熙达
57117232

Notes:
在实验中由于 VM 出现了一些问题，通过重新设置虚拟机网段才解决，所以在整个 Task1-Task9 中三台 VM 的 IP 地址发生过两次变化，特此说明。

Task 1: Network Setup

User:

```
[09/22/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:6e:e7:ce
          inet addr:192.168.206.128  Bcast:192.168.206.255  Mask:255.255.255.0
          inet6 addr: fe80::49f3:38b2:4f64:7294/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1785 errors:0 dropped:0 overruns:0 frame:0
          TX packets:885 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:146972 (146.9 KB)  TX bytes:86480 (86.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:553 errors:0 dropped:0 overruns:0 frame:0
          TX packets:553 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:100934 (100.9 KB)  TX bytes:100934 (100.9 KB)
```

Gateway:

```
[09/22/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:9f:32:13
          inet addr:192.168.206.130  Bcast:192.168.206.255  Mask:255.255.255.0
          inet6 addr: fe80::1aac:e89a:4eda:708f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1302 (1.3 KB)  TX bytes:6636 (6.6 KB)

ens37     Link encap:Ethernet  HWaddr 00:0c:29:9f:32:1d
          inet addr:192.168.99.4  Bcast:192.168.99.255  Mask:255.255.255.0
          inet6 addr: fe80::e24c:96e0:d1a3:3d83/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:373 (373.0 B)  TX bytes:6728 (6.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0
          TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:23408 (23.4 KB)  TX bytes:23408 (23.4 KB)
```

Host V:

```
[09/22/20]seed@VM:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:88:d3:fd
          inet addr:192.168.99.2  Bcast:192.168.99.255  Mask:255.255.255.0
          inet6 addr: fe80::f6a3:5e6f:40e0:1e6f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10400 (10.4 KB)  TX bytes:8921 (8.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:13097 (13.0 KB)  TX bytes:13097 (13.0 KB)
```

1.按照所给网络结构设置三台 VM,其中 gateway 有两张网卡，其中一个为外网地址，另外一个具有内网地址

Task 2: Create and Configure TUN Interface

a)

```python
#!/usr/bin/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'Liu', 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))
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
            ip = IP(packet)
            ip.show()
```

```
[09/22/20]seed@VM:~$ gedit tun.py
[09/22/20]seed@VM:~$ sudo python3 tun.py
Interface Name: Liu
```

1.使用上图中 python 程序创建 TUN 接口，并对新创建的接口进行配置

b)

```
[09/22/20]seed@VM:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
roup default qlen 1000
    link/ether 00:0c:29:6e:e7:ce brd ff:ff:ff:ff:ff:ff
    inet 192.168.206.128/24 brd 192.168.206.255 scope global dynamic ens33
       valid_lft 1752sec preferred_lft 1752sec
    inet6 fe80::49f3:38b2:4f64:7294/64 scope link
       valid_lft forever preferred_lft forever
3: Liu: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast stat
e UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global Liu
       valid_lft forever preferred_lft forever
    inet6 fe80::66b7:b4f3:3e69:aeb/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

2.另开启一个 terminal，使用 ip address 指令查看新接口的信息

c)

Ping 192.168.53.7:

```
[09/22/20]seed@VM:~$ ping 192.168.53.7
PING 192.168.53.7 (192.168.53.7) 56(84) bytes of data.
^Z
[1]+  Stopped                 ping 192.168.53.7
```

```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 32476
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xd011
  src       = 192.168.53.99
  dst       = 192.168.53.7
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x95fc
     id        = 0x132c
     seq       = 0x18
###[ Raw ]###
        load      = 'krj_\x88\xea\x05\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\
```

3.开启程序，使用 Ping 指令 Ping 192.168.53.7，Ping 指令无输出，但是 tun.py 可以抓取到包，程序输出 TUN 读取到的 IP 报文

Ping 192.168.99.4

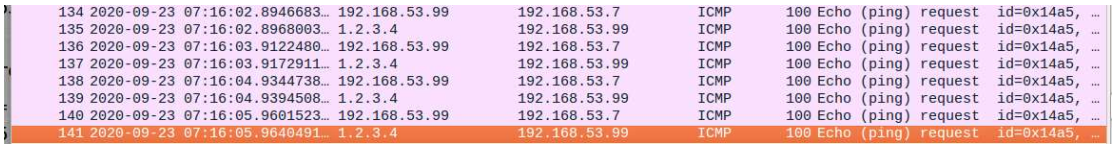
```
[09/22/20]seed@VM:~$ ping 192.168.99.4
PING 192.168.99.4 (192.168.99.4) 56(84) bytes of data.
64 bytes from 192.168.99.4: icmp_seq=1 ttl=128 time=1.20 ms
64 bytes from 192.168.99.4: icmp_seq=2 ttl=128 time=2.60 ms
64 bytes from 192.168.99.4: icmp_seq=3 ttl=128 time=1.87 ms
64 bytes from 192.168.99.4: icmp_seq=4 ttl=128 time=1.29 ms
64 bytes from 192.168.99.4: icmp_seq=5 ttl=128 time=2.13 ms
^Z
```

4.Ping 内网的 gateway，ping 指令有输出，但是 tun.py 没有任何输出

d)

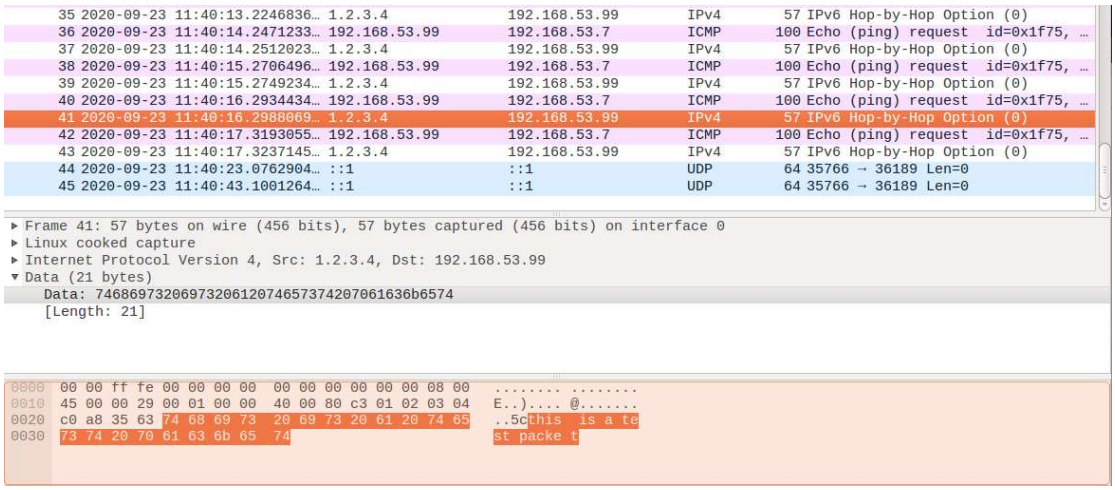
```
#!/usr/bin/python3
import fcntl
import struct
import os
import time
from scapy.all import *
TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'Liu', IFF_TUN | IFF_NO_PI)
ifname_bytes = 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))
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
            ip = IP(packet)
            ip.show()
            newip = IP(src='1.2.3.4', dst=ip.src)
            newpkt = newip/ip.payload
            os.write(tun, bytes(newpkt))
```

4.对上一步的程序做如图修改，再次运行



134	2020-09-23 07:16:02.8946683…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x14a5, …
135	2020-09-23 07:16:02.8968003…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x14a5, …
136	2020-09-23 07:16:03.9122480…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x14a5, …
137	2020-09-23 07:16:03.9172911…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x14a5, …
138	2020-09-23 07:16:04.9344738…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x14a5, …
139	2020-09-23 07:16:04.9394508…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x14a5, …
140	2020-09-23 07:16:05.9601523…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x14a5, …
141	2020-09-23 07:16:05.9640491…	1.2.3.4	192.168.53.99	ICMP	100	Echo (ping) request id=0x14a5, …

5.通过 Wireshark 可以看到发送往 TUN 接口的包



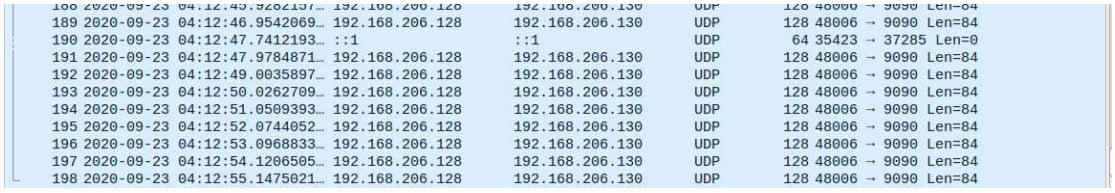
35	2020-09-23 11:40:13.2246836…	1.2.3.4	192.168.53.99	IPv4	57	IPv6 Hop-by-Hop Option (0)
36	2020-09-23 11:40:14.2471233…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x1f75, …
37	2020-09-23 11:40:14.2512023…	1.2.3.4	192.168.53.99	IPv4	57	IPv6 Hop-by-Hop Option (0)
38	2020-09-23 11:40:15.2706496…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x1f75, …
39	2020-09-23 11:40:15.2749234…	1.2.3.4	192.168.53.99	IPv4	57	IPv6 Hop-by-Hop Option (0)
40	2020-09-23 11:40:16.2934434…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x1f75, …
41	2020-09-23 11:40:16.2988069…	1.2.3.4	192.168.53.99	IPv4	57	IPv6 Hop-by-Hop Option (0)
42	2020-09-23 11:40:17.3193055…	192.168.53.99	192.168.53.7	ICMP	100	Echo (ping) request id=0x1f75, …
43	2020-09-23 11:40:17.3237145…	1.2.3.4	192.168.53.99	IPv4	57	IPv6 Hop-by-Hop Option (0)
44	2020-09-23 11:40:23.0762904…	::1	::1	UDP	64	35766 → 36189 Len=0
45	2020-09-23 11:40:43.1001264…	::1	::1	UDP	64	35766 → 36189 Len=0

```
▸ Frame 41: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 1.2.3.4, Dst: 192.168.53.99
▾ Data (21 bytes)
    Data: 746869732069732061207465737420706163b6574
    [Length: 21]
```

```
0000  00 00 ff fe 00 00 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 29 00 01 00 00  40 00 80 c3 01 02 03 04   E..).... @.......
0020  c0 a8 35 63 74 68 69 73  20 69 73 20 61 20 74 65   ..5cthis  is a te
0030  73 74 20 70 61 63 6b 65  74                       st packe t
```

6.修改 IP 包的数据部分，实现向 TUN 接口写入任意数据

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



189	2020-09-23 04:12:46.9542069…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
190	2020-09-23 04:12:47.7412193…	::1	::1	UDP	64	35423 → 37285 Len=0
191	2020-09-23 04:12:47.9784871…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
192	2020-09-23 04:12:49.0035897…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
193	2020-09-23 04:12:50.0262709…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
194	2020-09-23 04:12:51.0509393…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
195	2020-09-23 04:12:52.0744052…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
196	2020-09-23 04:12:53.0968833…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
197	2020-09-23 04:12:54.1206505…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84
198	2020-09-23 04:12:55.1475021…	192.168.206.128	192.168.206.130	UDP	128	48006 → 9090 Len=84

```
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
```

1.运行 tun_server.py 和 tun_client.py，ping 192.168.53.0/24 网段内的 192.168.53.1，tun_server.py 的输出和 Wireshark 抓包结果如上图

```
[09/26/20]seed@VM:~$ sudo ip route add 192.168.60.0/24 dev Liu vi
a 192.168.53.99
```

2.使用 IP route 指令将内网网段的路由 entry 添加到配置中

```
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.1
```

3. 运行 tun_server.py 和 tun_client.py，ping 内网网段内的 192.168.80.1，tun_server.py 的输出如上图

Task 4: Set Up the VPN Server

```python
import struct
import os
import time
import socket
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'Liu', 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 192.168.53.100/24 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(data))
```

1.修改 tun_server.py 的程序，使其具有转发 IP 包的功能。代码如上图。

```
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
10.0.2.4:60367 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.60.101
```

1	2020-09-26 10:05:09.5520121…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
2	2020-09-26 10:05:09.5520308…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply
3	2020-09-26 10:05:10.5727403…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
4	2020-09-26 10:05:10.5727810…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply
5	2020-09-26 10:05:11.5942898…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
6	2020-09-26 10:05:11.5943077…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply
7	2020-09-26 10:05:12.6196130…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
8	2020-09-26 10:05:12.6196498…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply
9	2020-09-26 10:05:13.6435901…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
10	2020-09-26 10:05:13.6436307…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply
11	2020-09-26 10:05:14.6649953…	192.168.53.99	192.168.60.101	ICMP	100	Echo (ping) request
12	2020-09-26 10:05:14.6650117…	192.168.60.101	192.168.53.99	ICMP	100	Echo (ping) reply

2.运行程序，tun_server.py 输出如上，在 host V 使用 Wireshark 抓包结果如图。可以看到，在 host V 接收到来自 TUN 的 ICMP request 报文，同时发送一个 ICMP reply。但是由于 TUN_server 的配置问题，该 ICMP reply 无法返回到 host U 上。

Task 5: Handling Traffic in Both Directions

```
[09/26/20]seed@VM:~$ ping 192.168.255.101
PING 192.168.255.101 (192.168.255.101) 56(84) bytes of data.
64 bytes from 192.168.255.101: icmp_seq=1 ttl=63 time=4.88 ms
64 bytes from 192.168.255.101: icmp_seq=2 ttl=63 time=12.3 ms
64 bytes from 192.168.255.101: icmp_seq=3 ttl=63 time=11.9 ms
64 bytes from 192.168.255.101: icmp_seq=4 ttl=63 time=11.7 ms
64 bytes from 192.168.255.101: icmp_seq=5 ttl=63 time=11.6 ms
64 bytes from 192.168.255.101: icmp_seq=6 ttl=63 time=11.7 ms
64 bytes from 192.168.255.101: icmp_seq=7 ttl=63 time=12.1 ms
64 bytes from 192.168.255.101: icmp_seq=8 ttl=63 time=12.9 ms
64 bytes from 192.168.255.101: icmp_seq=9 ttl=63 time=11.8 ms
64 bytes from 192.168.255.101: icmp_seq=10 ttl=63 time=13.5 ms
64 bytes from 192.168.255.101: icmp_seq=11 ttl=63 time=15.6 ms
64 bytes from 192.168.255.101: icmp_seq=12 ttl=63 time=11.1 ms
64 bytes from 192.168.255.101: icmp_seq=13 ttl=63 time=11.7 ms
64 bytes from 192.168.255.101: icmp_seq=14 ttl=63 time=3.64 ms
```

```
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From socket <==: 0.0.0.0 --> 157.147.109.66
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
From socket <==: 192.168.255.101 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.255.101
```

43	2020-09-26 21:45:27.2508499…	192.168.53.99	192.168.255.101	ICMP	100	Echo (ping) request …
44	2020-09-26 21:45:27.2508978…	192.168.255.101	192.168.53.99	ICMP	100	Echo (ping) reply …
45	2020-09-26 21:45:27.9069778…	PcsCompu_77:9b:bf		ARP	62	Who has 10.80.128.28?…
46	2020-09-26 21:45:28.2516190…	192.168.53.99	192.168.255.101	ICMP	100	Echo (ping) request …
47	2020-09-26 21:45:28.2516658…	192.168.255.101	192.168.53.99	ICMP	100	Echo (ping) reply …
48	2020-09-26 21:45:29.2534670…	192.168.53.99	192.168.255.101	ICMP	100	Echo (ping) request …
49	2020-09-26 21:45:29.2535134…	192.168.255.101	192.168.53.99	ICMP	100	Echo (ping) reply …
50	2020-09-26 21:45:29.2539126…	::1	::1	UDP	64	46264 → 40375 Len=0
51	2020-09-26 21:45:30.2563236…	192.168.53.99	192.168.255.101	ICMP	100	Echo (ping) request …
52	2020-09-26 21:45:30.2563651…	192.168.255.101	192.168.53.99	ICMP	100	Echo (ping) reply …

1.修改程序并运行，发现通过 Host U 的终端可以 ping 通位于内网的 Host V，说明 VPN 搭建成功。Tun_server.py 和 Wireshark 的抓包结果如图所示。

```
[09/26/20]seed@VM:~$ telnet 192.168.255.101
Trying 192.168.255.101...
Connected to 192.168.255.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Sep 26 21:49:05 EDT 2020 from 192.168.53.99 on pt
s/0
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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


[09/26/20]seed@VM:~$
```

180	2020-09-26 21:49:05.8012542…	192.168.53.99	192.168.255.101	TELNET	70	Telnet Data ...
181	2020-09-26 21:49:05.8012699…	192.168.255.101	192.168.53.99	TCP	68	23 → 37044 [ACK] Seq=…
182	2020-09-26 21:49:05.8045832…	192.168.255.101	192.168.53.99	TELNET	70	Telnet Data ...
183	2020-09-26 21:49:05.8080389…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…
184	2020-09-26 21:49:05.8261397…	192.168.255.101	192.168.53.99	TELNET	139	Telnet Data ...
185	2020-09-26 21:49:05.8317008…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…
186	2020-09-26 21:49:05.8317147…	192.168.255.101	192.168.53.99	TELNET	70	Telnet Data ...
187	2020-09-26 21:49:05.8352541…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…
188	2020-09-26 21:49:06.1016291…	192.168.255.101	192.168.53.99	TELNET	168	Telnet Data ...
189	2020-09-26 21:49:06.1050066…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…
190	2020-09-26 21:49:06.1050178…	192.168.255.101	192.168.53.99	TELNET	70	Telnet Data ...
191	2020-09-26 21:49:06.1092729…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…
192	2020-09-26 21:49:06.1134868…	192.168.255.101	192.168.53.99	TELNET	131	Telnet Data ...
193	2020-09-26 21:49:06.1205986…	192.168.53.99	192.168.255.101	TCP	68	37044 → 23 [ACK] Seq=…

2.再次使用 telnet 进行测试，发现可以成功登录。

Task 6: Tunnel-Breaking Experiment

```
From socket <==: 192.168.53.99 --> 192.168.255.101
From tun ==>: 192.168.255.101 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.255.101
From tun ==>: 192.168.255.101 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.255.101
From tun ==>: 192.168.255.101 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.255.101
^Z
[2]+  Stopped                 sudo python3 tun_server.py
[09/26/20]seed@VM:~$ sudo python3 tun_server.py
Traceback (most recent call last):
  File "tun_server.py", line 16, in <module>
    ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
OSError: [Errno 16] Device or resource busy
[09/26/20]seed@VM:~$ sudo lsof -i:9090
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
python3 2936 root    4u  IPv4  38340      0t0  UDP *:9090
[09/26/20]seed@VM:~$ sudo kill -9 2936
[09/26/20]seed@VM:~$ sudo python3 tun_server.py
Interface Name: Liu
From tun ==>: 0.0.0.0 --> 48.127.162.116
From socket <==: 192.168.53.99 --> 192.168.255.101
From tun ==>: 192.168.255.101 --> 192.168.53.99
```

1.切断 tun,在 telnet 中看不到自己的输入，短时间内重新连接 VPN,控制台内出现了之前输入的字符，且 telnet 连接依然保持。

Task 7: Routing Experiment on Host V

```
[09/26/20]seed@VM:~$ sudo ip route del 0.0.0.0/0
[09/26/20]seed@VM:~$ sudo ip route
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.255.0/24 dev enp0s3  proto kernel  scope link  src 192.168.255.101  metr
ic 100
[09/26/20]seed@VM:~$ sudo ip route add 10.0.2.0/24 dev enp0s3 via 192.168.255.1
[09/26/20]seed@VM:~$ sudo ip route
10.0.2.0/24 via 192.168.255.1 dev enp0s3
169.254.0.0/16 dev enp0s3  scope link  metric 1000
192.168.255.0/24 dev enp0s3  proto kernel  scope link  src 192.168.255.101  metr
ic 100
```

1.重新配置 host V 的路由信息，使用指令如上图

Task 8: Experiment with the TUN IP Address

On host U:

```
     8 2020-09-26 22:10:52.8296171… 10.0.2.4                192.168.255.101
     9 2020-09-26 22:10:53.8545528… 10.0.2.4                192.168.255.101
    10 2020-09-26 22:10:54.8787166… 10.0.2.4                192.168.255.101
    11 2020-09-26 22:10:55.9014660… 10.0.2.4                192.168.255.101
    12 2020-09-26 22:10:56.9255924… 10.0.2.4                192.168.255.101
    13 2020-09-26 22:10:57.9503715… 10.0.2.4                192.168.255.101
    14 2020-09-26 22:10:58.9736245… 10.0.2.4                192.168.255.101
    15 2020-09-26 22:10:59.9979334… 10.0.2.4                192.168.255.101
    16 2020-09-26 22:11:01.0215130… 10.0.2.4                192.168.255.101
    17 2020-09-26 22:11:02.0465453… 10.0.2.4                192.168.255.101
    18 2020-09-26 22:11:03.0701754… 10.0.2.4                192.168.255.101
    19 2020-09-26 22:11:04.0941506… 10.0.2.4                192.168.255.101
    20 2020-09-26 22:11:05.1175344… 10.0.2.4                192.168.255.101
    21 2020-09-26 22:11:06.1424520… 10.0.2.4                192.168.255.101
```

On host gateway:

```
No.   Time                          Source      Destination     Pr
    1 2020-09-26 22:10:56.8330964… ::1         ::1             UD
    2 2020-09-26 22:11:16.8482463… ::1         ::1             UD
```

1. 通过 Wireshark 抓包可以看到，发送的 IP 报文在 host gateway 被丢弃

Task 9: Experiment with the TAP Interface

```
[09/26/20]seed@VM:~$ ping 192.168.53.2
PING 192.168.53.2 (192.168.53.2) 56(84) bytes of data.
From 192.168.53.99 icmp_seq=1 Destination Host Unreachable
From 192.168.53.99 icmp_seq=2 Destination Host Unreachable
From 192.168.53.99 icmp_seq=3 Destination Host Unreachable
From 192.168.53.99 icmp_seq=4 Destination Host Unreachable
From 192.168.53.99 icmp_seq=5 Destination Host Unreachable
From 192.168.53.99 icmp_seq=6 Destination Host Unreachable
From 192.168.53.99 icmp_seq=7 Destination Host Unreachable
From 192.168.53.99 icmp_seq=8 Destination Host Unreachable
```

```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 6a:93:07:de:6c:14
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 6a:93:07:de:6c:14
     psrc      = 192.168.53.99
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.53.2
```

1.使用 TAP 接口重复 task2 中的实验，host U 的 terminal 中 ping 指令显示 destination host unreachable，程序打印出 ARP 报文信息。