

57117208 傅靖邦

Task 1:

虚拟机 A: Host U 192.168.1.105

虚拟机 B: VPN server 192.168.1.107 192.168.10.107

虚拟机 C: Host V 192.168.1.108 192.168.10.108

使用虚拟机 B 作为 VPN server, 添加一个网卡, 使用内部网络模式
配置地址为 192.168.10.107

虚拟地址 C 配置地址为 192.168.10.108

host U ping VPN server

```
09/24/20]seed@VM:~$ ping 192.168.1.107
PING 192.168.1.107 (192.168.1.107) 56(84) bytes of data.
64 bytes from 192.168.1.107: icmp_seq=1 ttl=64 time=1.15 ms
64 bytes from 192.168.1.107: icmp_seq=2 ttl=64 time=0.749 ms
64 bytes from 192.168.1.107: icmp_seq=3 ttl=64 time=1.02 ms
64 bytes from 192.168.1.107: icmp_seq=4 ttl=64 time=0.795 ms
64 bytes from 192.168.1.107: icmp_seq=5 ttl=64 time=1.12 ms
64 bytes from 192.168.1.107: icmp_seq=6 ttl=64 time=0.814 ms
^C
```

```
09/24/20]seed@VM:~$ ping 192.168.10.108
PING 192.168.10.108 (192.168.10.108) 56(84) bytes of data.
^C
-- 192.168.10.108 ping statistics --
36 packets transmitted, 0 received, 100% packet loss, time 35818ms
```

```
09/24/20]seed@VM:~$ ping 192.168.10.107
PING 192.168.10.107 (192.168.10.107) 56(84) bytes of data.
64 bytes from 192.168.10.107: icmp_seq=1 ttl=64 time=0.909 ms
64 bytes from 192.168.10.107: icmp_seq=2 ttl=64 time=0.708 ms
64 bytes from 192.168.10.107: icmp_seq=3 ttl=64 time=0.575 ms
64 bytes from 192.168.10.107: icmp_seq=4 ttl=64 time=0.556 ms
64 bytes from 192.168.10.107: icmp_seq=5 ttl=64 time=0.611 ms
^C
192.168.10.107 ping statistics:
```

互相都可以 ping 通。

Task 2:

在 U 上运行 tun.py 程序添加一个 TUN 虚拟接口

```
#!/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'tun%d' % IFF_TUN | IFF_NO_PI)
fname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
# Get the interface name
fname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
while True:
```

Task2a:

```
valid lft forever preferred lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
lt qlen 500
    link/none
```

多出一个 tun0 接口

Task2b:

```

: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast sta
e UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global tun0
        valid_lft forever preferred_lft forever

```

Task2c

```
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()
```

```
###[ IP ]###
version    = 4
ihl        = 5
tos         = 0x0
len         = 84
id          = 24257
flags       = DF
frag        = 0
ttl         = 64
proto       = icmp
chksum      = 0xf01d
src          = 192.168.53.99
dst          = 192.168.53.22
\options    \
###[ ICMP ]###
type        = echo-request
code         = 0
chksum       = 0x50e9
id           = 0x2529
seq          = 0xa
###[ Raw ]###
load         = 'o\xbc1_\xb2\xc4\x08\x00\x08\t\n\x0b\x0c\r\x0e\x0f
1\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#%&\'()*
24567'
```

tun.py 打印出了发往 192.168.53.22 的数据包,数据包在路由选择时被发给了 tun0 虚拟接口。数据包的源地址被修改成了 tun0 端口的 IP 地址

ping 192.168.70.0/24 网内地址进行测试但是无法成功 ping 到，因为 192.168.70.0/24 这一网络还不在于 tun0 的路由之内。

Task2d

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

Source	Destination	Protocol	Length	Info
192.168.53.99	192.168.53.22	ICMP	84	Echo (ping) request
1.2.3.4	192.168.53.99	ICMP	84	Echo (ping) request
192.168.53.99	192.168.53.22	ICMP	84	Echo (ping) request
1.2.3.4	192.168.53.99	ICMP	84	Echo (ping) request
192.168.53.99	192.168.53.22	ICMP	84	Echo (ping) request
1.2.3.4	192.168.53.99	ICMP	84	Echo (ping) request
192.168.53.99	192.168.53.22	ICMP	84	Echo (ping) request
1.2.3.4	192.168.53.99	ICMP	84	Echo (ping) request
192.168.53.99	192.168.53.22	ICMP	84	Echo (ping) request

Task3

在 VPN 服务器上运行 tun_server.py 程序，监听 9090 端口并打印出所有收到的数据

```
#!/usr/bin/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))
```

ping 192.168.53.0/24 网内任意 IP 地址

VPN 端输出：

```
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
192.168.1.105:58777 --> 0.0.0.0:9090
 Inside: 192.168.53.99 --> 192.168.53.22
```

VPN 服务器能够收到数据报，外层为 192.168.1.105 发往自身 9090 端口的 IP 数据报，内层为 192.168.53.99 发往 192.168.53.22 的 IP 数据报，可见内层数据报通过 IP 隧道在主机 U 与 VPN 服务器间实现了通信

ping 内部主机 V：

只能看到主机 U 以 0.0.0.0 的 IP 地址发往其他 IP 地址的数据报，没有看到发往内部主机 V 的数据报，原因是目的地地址为 V 的数据报没有被 tun 接口处理在主机 U 上配置路由信息，将 192.168.10.0/24 网段的出口设为虚拟接口 tun0


```

192.168.1.105:58777 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.10.108
192.168.1.105:58777 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.10.108
192.168.1.105:58777 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.10.108
192.168.1.105:58777 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.10.108

```

Task4

```

#!/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
IP_A = "0.0.0.0"
PORT = 9090
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun%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: tun0".format(ifname))

os.system("sudo ip addr add 192.168.53.1/24 dev tun0")
os.system("sudo ip link set dev tun0 up")

# 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)
    os.write(tun, data)
    print("Receive a packet\n")

```

主机 Uping 主机 V

Source	Destination	Protoc	Length	Info	Time
... VMware-CB:3b:c8	...	ARP	62	Who has 218.4.4.4? Tell 192.168.10.107	
... VMware-CB:3b:c8	...	ARP	62	Who has 202.12.27.33? Tell 192.168.10.107	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=1/256, ttl=63 (reply in 239)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=1/256, ttl=64 (request in 238)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=2/512, ttl=63 (reply in 242)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=2/512, ttl=64 (request in 241)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=3/768, ttl=63 (reply in 245)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=3/768, ttl=64 (request in 244)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=4/1024, ttl=63 (reply in 249)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=4/1024, ttl=64 (request in 248)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=5/1280, ttl=63 (reply in 253)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=5/1280, ttl=64 (request in 252)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=6/1536, ttl=63 (reply in 259)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=6/1536, ttl=64 (request in 258)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=7/1792, ttl=63 (reply in 263)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=7/1792, ttl=64 (request in 262)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=8/2048, ttl=63 (reply in 265)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=8/2048, ttl=64 (request in 264)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=9/2304, ttl=63 (reply in 267)	
... 192.168.10.108	192.168.53.99	ICMP	100	Echo (ping) reply id=0x2809, seq=9/2304, ttl=64 (request in 266)	
... 192.168.53.99	192.168.10.108	ICMP	100	Echo (ping) request id=0x2809, seq=10/2560, ttl=63 (reply in 269)	

Task5

客户主机

```

while True:
# this will block until at least one interface is ready
    ready, _, _ = 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, ('192.168.1.107', 9000))

```

服务器端:

```

while True:
# this will block until at least one interface is ready
    ready, _, _ = 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, ('192.168.1.105', 9000))

```

UpingV

```

[09/24/20]seed@VM:~$ ping 192.168.10.108
PING 192.168.10.108 (192.168.10.108) 56(84) bytes of data.
64 bytes from 192.168.10.108: icmp_seq=1 ttl=63 time=5.44 ms
64 bytes from 192.168.10.108: icmp_seq=2 ttl=63 time=4.87 ms
64 bytes from 192.168.10.108: icmp_seq=3 ttl=63 time=2.37 ms
64 bytes from 192.168.10.108: icmp_seq=4 ttl=63 time=4.93 ms
64 bytes from 192.168.10.108: icmp_seq=5 ttl=63 time=4.85 ms
64 bytes from 192.168.10.108: icmp_seq=6 ttl=63 time=4.65 ms
64 bytes from 192.168.10.108: icmp_seq=7 ttl=63 time=3.47 ms
64 bytes from 192.168.10.108: icmp_seq=8 ttl=63 time=4.90 ms
64 bytes from 192.168.10.108: icmp_seq=9 ttl=63 time=4.67 ms
64 bytes from 192.168.10.108: icmp_seq=10 ttl=63 time=4.87 ms
64 bytes from 192.168.10.108: icmp_seq=11 ttl=63 time=4.95 ms
64 bytes from 192.168.10.108: icmp_seq=12 ttl=63 time=4.73 ms
64 bytes from 192.168.10.108: icmp_seq=13 ttl=63 time=5.04 ms
64 bytes from 192.168.10.108: icmp_seq=14 ttl=63 time=4.85 ms
64 bytes from 192.168.10.108: icmp_seq=15 ttl=63 time=4.77 ms
64 bytes from 192.168.10.108: icmp_seq=16 ttl=63 time=4.84 ms
^C

```

```

From socket <==: 192.168.53.99 --> 192.168.10.108
From tun ==>: 192.168.10.108 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.10.108
From tun ==>: 192.168.10.108 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.10.108
From tun ==>: 192.168.10.108 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.10.108
From tun ==>: 192.168.10.108 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.10.108
From tun ==>: 192.168.10.108 --> 192.168.53.99

```

```

[09/24/20]seed@VM:~$ telnet 192.168.10.108
Trying 192.168.10.108...
Connected to 192.168.10.108.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
/M login:

```

Telnet 成功

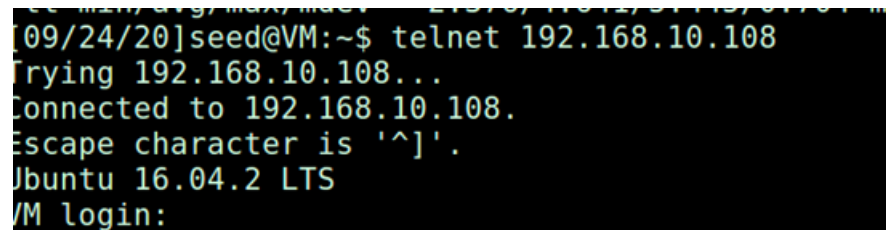
Task6:

使用 IP 隧道进行 telnet 通信过程中，关闭 tun_server.py。此时无法再输入命令，且客户端与服务端不再有报文传输。等待一段时间后重新连接，客户端和服务端传送大量报文，通过 wireshark，得知双方重新连接了 telnet 服务。

当 IP 隧道断开后，如果能在较短时间内重新连接上，仍可以继续断开前的业务。

Task7

主机 U 向主机 V 发起 telnet:

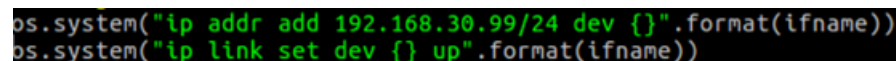


```
09/24/20]seed@VM:~$ telnet 192.168.10.108
Trying 192.168.10.108...
Connected to 192.168.10.108.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login:
```

可以通信

Task8

修改客户端 TUN 端口 IP 地址，使其与服务端 TUN 接口 IP 不位于同一网段



```
os.system("ip addr add 192.168.30.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
```

此时无法 ping 通

观察主机 U 上 wireshark: 主机 U 上既有从 ens33 发出的 192.168.1.105->192.168.1.107 的 UDP 数据报，也有从 tun0 发出的 192.168.30.99->192.168.10.108 的 ICMP 数据报。

观察 VPN 服务器上 wireshark: ens33 和 tun0 端口都分别有收到 UDP 和 ICMP 报文，但 ens38 端口却没有报文发出，即 VPN 服务器在内层的 ICMP 报文进行路由选择时，并未将其从 ens38 端口转发出去。

在 VPN 服务器上添加 192.168.30.0/24 网络与 tun0 端口关联的路由表项:

```
sudo ip route add 192.168.30.0/24 dev tun0
```

此时即可正常通信

Task9


```

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
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tap%d' % tap, IFF_TAP | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tap, TUNSETIFF, ifr)

#Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

#Configure the interface
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(tap, 2048)
    if True:
        ether = Ether(packet)
        ether.show()

```

ping 192.168.53.21 进行测试，发现无响应

```

ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?
ARP      42 Who has 192.168.53.21?

```

tap0 端口向外发送 ARP 请求，查询 192.168.53.21 的 MAC 地址，由于这是不存在的网络，所以 ARP 请求不能收到响应，导致 ICMP 数据包滞留在端口中不能发送出去。可以看出 tap 工作在 MAC 层。