LAB 7

57118116 陈煜

Task 1

```
[07/30/21]seed@VM:~/.../Labsetup$ dockps
fdd7c3f8d2f8  client-10.9.0.5
456bf17e580d  host-192.168.60.6
6d98530b31e1  server-router
1a4241e6df1e  host-192.168.60.5
```

在主机 U 上 ping 服务器：

```
root@fdd7c3f8d2f8:/# 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=1.51 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.172 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.206 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.167 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.169 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.210 ms
^C
--- 10.9.0.11 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5138ms
rtt min/avg/max/mdev = 0.167/0.405/1.511/0.494 ms
```

可以 ping 通。

在服务器上用 tcpdump 抓取数据包：

```
root@6d98530b31e1:/# 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
08:46:06.381862 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
08:46:06.381931 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
08:46:06.382019 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 1, length 64
08:46:06.382070 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 1, length 64
08:46:07.382862 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 2, length 64
08:46:07.382917 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 2, length 64
08:46:08.393955 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 3, length 64
08:46:08.394039 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 3, length 64
08:46:09.416867 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 4, length 64
08:46:09.416921 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 4, length 64
08:46:10.440864 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 5, length 64
08:46:10.440916 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 5, length 64
08:46:11.519171 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 6, length 64
08:46:11.519239 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 6, length 64
08:46:11.602138 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
08:46:11.602831 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
^C
16 packets captured
16 packets received by filter
0 packets dropped by kernel
```

在服务器上 ping 主机 V：

```
root@6d98530b31e1:/# 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=0.143 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.160 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.159 ms
^C
--- 192.168.60.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.143/0.154/0.160/0.007 ms
```

可以 ping 通。

在服务器上用 tcpdump 抓取数据包：

```
root@6d98530b31e1:/# 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
08:49:19.580457 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 28, seq 1, le
ngth 64
08:49:19.580550 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 28, seq 1, leng
th 64
08:49:20.587382 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 28, seq 2, le
ngth 64
08:49:20.587475 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 28, seq 2, leng
th 64
08:49:21.615616 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 28, seq 3, le
ngth 64
08:49:21.615709 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 28, seq 3, leng
th 64
08:49:24.585677 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
08:49:24.585923 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
08:49:24.585947 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
08:49:24.585972 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

在主机 U 上 ping 主机 V：

```
root@fdd7c3f8d2f8:/# 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 ---
12 packets transmitted, 0 received, 100% packet loss, time 11255ms
```

无法 ping 通。

Task 2.A

修改 tun.py 中的代码如下：

```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'ychen%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))

while True:
   time.sleep(10)

```

保存后，在主机 U 上运行：

```
root@fdd7c3f8d2f8:/volumes# chmod a+x tun.py
root@fdd7c3f8d2f8:/volumes# tun.py
Interface Name: ychen0
```

通过如下命令查看：

```
root@fdd7c3f8d2f8:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group 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: ychen0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
27: eth0@if28: <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
```

可以发现修改之后的接口。

Task 2.B

在 tun.py 中带入如下代码：

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

再次运行 tun.py 后查看：

```
root@fdd7c3f8d2f8:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group 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
3: ychen0: <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 ychen0
       valid_lft forever preferred_lft forever
27: eth0@if28: <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
```

接口被分配了 ip 地址。

Task 2.C

修改代码如下：

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

在主机 U 上运行上述代码，并 ping 192.168.53.0/24 网段内的 ip：

```
root@fdd7c3f8d2f8:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7155ms
```

无法 ping 通，并出现如下结果：

```
root@fdd7c3f8d2f8:/volumes# tun.py
Interface Name: ychen0
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
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
```

再次运行代码，并在主机 U 上 ping 主机 V：

```
root@fdd7c3f8d2f8:/# 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 ---
14 packets transmitted, 0 received, 100% packet loss, time 13309ms
```

发现 ping 不通。代码无响应：

```
root@fdd7c3f8d2f8:/volumes# tun.py
Interface Name: ychen0
^CTraceback (most recent call last):
  File "./tun.py", line 27, in <module>
    packet = os.read(tun, 2048)
KeyboardInterrupt
```

Task 2.D

修改代码如下：

```
25 while True:
26 # Get a packet from the tun interface
27         packet = os.read(tun, 2048)
28         if packet:
29                 pkt=IP(packet)
30                 print(pkt.summary())
31                 if ICMP in pkt:
32                         newip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
33                         newip.ttl=99
34                         newicmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
35                         if pkt.haslayer(Raw):
36                                 data=pkt[Raw].load
37                                 newpkt=newip/newicmp/data
38                         else:
39                                 newpkt=newip/newicmp
40                 os.write(tun,bytes(newpkt))
```

运行程序后，在主机 U 上 ping 192.168.53.0/24 网段下的 ip：

```
root@fdd7c3f8d2f8:/# 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=99 time=347 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=99 time=1.24 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=99 time=2.86 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=99 time=1.57 ms
^C
--- 192.168.53.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 1.240/88.056/346.558/149.247 ms
```

程序输出如下：

```
root@fdd7c3f8d2f8:/volumes# tun.py
Interface Name: ychen0
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
```

可见返回的是程序构造的报文，因此仍然没有 ping 通。

Task 3

编写代码 tun_server.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'ychen%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))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
server.bind((SERVER_IP,SERVER_PORT))

while True:
        data,(ip,port)=server.recvfrom(2048)
        print("{}:{} --> {}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
```

编写代码 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'ychen%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))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
```

```python
sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="10.9.0.11"
SERVER_PORT=9090

while True:
        packet=os.read(tun,2048)
        if packet:
                pkt=IP(packet)
                print(pkt.summary())
                sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

在主机 U 上运行 tun_client.py，在服务器上运行 tun_server.py，再 ping 192.168.53.0/24 网段下的 ip：

```
root@fdd7c3f8d2f8:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4098ms
```

发现 ping 不通。

tun_client.py 输出如下：

```
root@fdd7c3f8d2f8:/volumes# python3 tun_client.py
Interface Name: ychen0
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
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
```

tun_server.py 输出如下：

```
root@6d98530b31e1:/volumes# python3 tun_server.py
Interface Name: ychen0
RTNETLINK answers: File exists
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:51135 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
```

可知，管道外部是 10.9.0.5 → 0.0.0.0，管道内部是 192.168.53.99 → 192.168.53.1。

Task 4

查看 docker-compose.yml，让 ip_forward 为 1：

```
Router:
    image: handsonsecurity/seed-ubuntu:large
    container_name: server-router
    tty: true
    cap_add:
            - ALL
    devices:
            - "/dev/net/tun:/dev/net/tun"
    sysctls:
            - net.ipv4.ip_forward=1
    volumes:
            - ./volumes:/volumes
    networks:
        net-10.9.0.0:
            ipv4_address: 10.9.0.11
        net-192.168.60.0:
            ipv4_address: 192.168.60.11
    command: bash -c "
                  ip route del default  &&
                  ip route add default via 10.9.0.1 &&
                  tail -f /dev/null
             "
```

修改 tun_server.py 代码如下：

```
#!/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'ychen%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 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

```

```python
server=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
server.bind((SERVER_IP,SERVER_PORT))

while True:
        data,(ip,port)=server.recvfrom(2048)
        print("{}:{} --> {}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
        pkt=IP(data)
        print("Inside: {} --> {}".format(pkt.src,pkt.dst))
        os.write(tun,data)
        print("write")
        pkt=IP(data)
        print("Inside: {} --> {}".format(pkt.src, pkt.dst))
```

在服务器上运行 tun_server.py，在主机 U 上运行 tun_client.py，在 U 上 ping 192.168.60.5，在服务器上通过 tcpdump 查看报文：

```
root@fdd7c3f8d2f8:/# 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 ---
7 packets transmitted, 0 received, 100% packet loss, time 6126ms
```

```
root@6d98530b31e1:/volumes# python3 tun_server.py
Interface Name: ychen0
RTNETLINK answers: File exists
10.9.0.5:51133 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:51133 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:51133 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
```

```
root@fdd7c3f8d2f8:/volumes# python3 tun_client.py
Interface Name: ychen0
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
```

tcpdump 抓取到报文：

```
root@6d98530b31e1:/# tcpdump -nni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:35:06.161199 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 106, seq 1, length 64
11:35:06.204452 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 106, seq 1, length 64
11:35:06.210340 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 106, seq 1, length 64
11:35:06.221059 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 106, seq 1, length 64
11:35:07.054832 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 106, seq 2, length 64
11:35:07.263272 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 106, seq 2, length 64
11:35:07.264427 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 106, seq 2, length 64
11:35:07.264491 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 106, seq 2, length 64
11:35:08.078610 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 106, seq 3, length 64
11:35:08.078730 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 106, seq 3, length 64
```

说明 ICMP 报文到达目的主机，但是没有响应。

Task 5

修改 tun_server.py 代码如下：

```
#!/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'ychen%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 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
```

```
sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
ip="10.9.0.5"
port=10000
sock.bind((SERVER_IP,SERVER_PORT))
fds=[sock,tun]

while True:
        ready,_,_=select.select(fds,[],[])
        for fd in ready:
                if fd is sock:
                        print("sock...")
                        data,(ip,port)=sock.recvfrom(2048)
                        print("{}:{} --> {}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
                        pkt=IP(data)
                        print("Inside: {} --> {}".format(pkt.src,pkt.dst))
                        os.write(tun,data)
                if fd is tun:
                        print("tun...")
                        packet=os.read(tun,2048)
                        pkt=IP(packet)
```

修改 tun_client.py 代码如下：

```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'ychen%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))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
```

```python
sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="10.9.0.11"
SERVER_PORT=9090
fds=[sock,tun]

while True:
        ready,_,_=select.select(fds,[],[])
        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)
                        if packet:
                                pkt=IP(packet)
                                print(pkt.summary())
                                sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

在服务器上运行 tun_server.py，在主机 U 上运行 tun_client.py，在 U 上 ping 192.168.60.5，发现此时可以 ping 通：

```
root@fdd7c3f8d2f8:/# 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=3.50 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=6.34 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=6.34 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=2.28 ms
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 2.275/4.615/6.343/1.781 ms
```

程序输出信息如下：

```
root@fdd7c3f8d2f8:/volumes# python3 tun_client.py
Interface Name: ychen0
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket: 192.168.60.5 --> 192.168.53.99
```

```
root@6d98530b31e1:/volumes# python3 tun_server.py
Interface Name: ychen0
RTNETLINK answers: File exists
sock...
10.9.0.5:60756 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
sock...
10.9.0.5:60756 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
sock...
10.9.0.5:60756 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
```

telnet 也成功：

```
root@fdd7c3f8d2f8:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1a4241e6df1e 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 are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Task 6

在主机 U 上运行 tun_client.py，在服务器上运行 tun_server.py 后，在主机 U 上 telnet 192.168.60.5：

```
root@fdd7c3f8d2f8:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1a4241e6df1e 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 are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Fri Jul 30 12:03:56 UTC 2021 from 192.168.53.99 on pts/1
seed@1a4241e6df1e:~$
```

停止运行 tun_server.py，发现无法在 U 中输入信息。

再次运行 tun_server.py，发现此时可以输入信息：

```
To restore this content, you can run the 'unminimize' command.
Last login: Fri Jul 30 12:05:27 UTC 2021 from 192.168.53.99 on pts/1
seed@1a4241e6df1e:~$ abc
-bash: abc: command not found
seed@1a4241e6df1e:~$
```