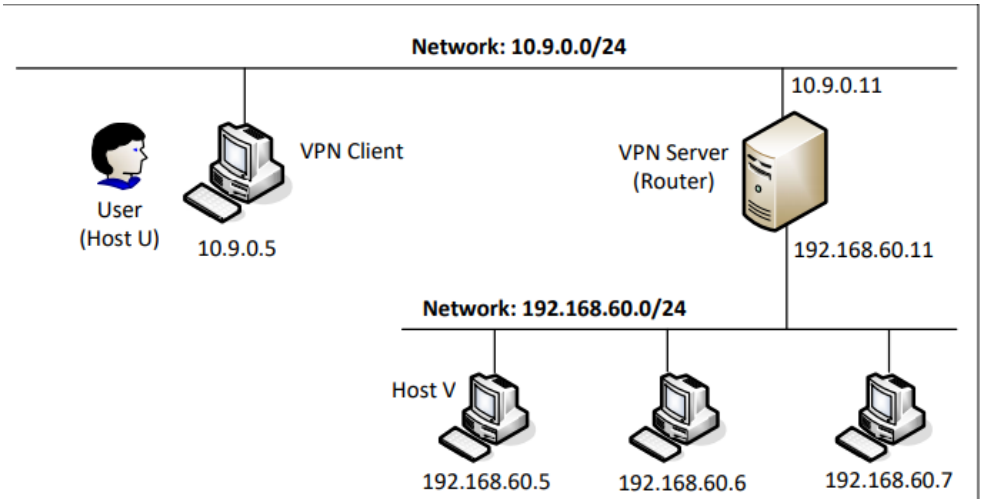
VPN Lab: The Container Version

57118204 陈盈

Task 1: Network Setup



查看各主机哈希值：

```
[07/30/21]seed@VM:~/.../volumes$ dockps
a8e789f7a114  host-192.168.60.6
f4b6288eee24  host-192.168.60.5
eb362125b187  client-10.9.0.5
fdc6973467eb  server-router
```

在主机 U 上 ping 服务器，可以 ping 通。

```
root@eb362125b187:/# 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=0.149 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.063 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.050 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.066 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.051 ms
64 bytes from 10.9.0.11: icmp_seq=7 ttl=64 time=0.050 ms
64 bytes from 10.9.0.11: icmp_seq=8 ttl=64 time=0.066 ms
64 bytes from 10.9.0.11: icmp_seq=9 ttl=64 time=0.080 ms
64 bytes from 10.9.0.11: icmp_seq=10 ttl=64 time=0.067 ms
64 bytes from 10.9.0.11: icmp_seq=11 ttl=64 time=0.063 ms
^C
--- 10.9.0.11 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10237m
s
rtt min/avg/max/mdev = 0.050/0.069/0.149/0.026 ms
```

在服务器上用 tcpdump 抓取数据包

```
root@fdc6973467eb:/#  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
22:48:45.948134 IP 10.9.0.1.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _ipps._tc
p.local. PTR (QM)? _ipp._tcp.local. (45)
22:48:48.849043 IP6 fe80::42:86ff:fea9:c71f.5353 > ff02::fb.5353: 0 [2q] PTR (Q
M)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
22:48:57.353764 IP6 fe80::a831:75ff:fe06:940a.5353 > ff02::fb.5353: 0 [2q] PTR
(QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
22:49:01.675787 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, lengt
h 64
22:49:01.675808 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length
64
22:49:02.680865 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, lengt
h 64
22:49:02.680887 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length
64
22:49:03.706006 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 3, lengt
h 64
22:49:03.706023 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 3, length
64
22:49:04.663957 IP6 fe80::a831:75ff:fe06:940a > ff02::2: ICMP6, router solicita
tion, length 16
22:49:04.663966 IP6 fe80::42:86ff:fea9:c71f > ff02::2: ICMP6, router solicitati
on, length 16
22:49:04.730843 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 4, lengt
```

在服务器上 ping 主机 V，可以 ping 通。

```
root@fdc6973467eb:/# 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=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=64 time=0.049 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=64 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=64 time=0.045 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=64 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=64 time=0.054 ms
^C
--- 192.168.60.5 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9217ms
rtt min/avg/max/mdev = 0.045/0.058/0.085/0.010 ms
```

在服务器上用 tcpdump 抓取数据包。

```
root@fdc6973467eb:/#  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
22:52:13.623763 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 24, length 64
22:52:13.623806 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 24, length 64
22:52:14.650188 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 25, length 64
22:52:14.650228 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 25, length 64
22:52:15.671572 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 26, length 64
22:52:15.671615 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 26, length 64
22:52:16.697663 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 27, length 64
22:52:16.697695 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 27, length 64
22:52:17.720130 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 28, length 64
22:52:17.720161 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 28, length 64
22:52:18.746085 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 29, length 64
22:52:18.746115 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 29, length 64
22:52:19.768356 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 24, seq 30, length 64
22:52:19.768399 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 24, seq 30, length 64
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
```

在主机 U 上 ping 主机 V，ping 不通

```
root@eb362125b187:/# 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 ---
6 packets transmitted, 0 received, 100% packet loss, time 5120ms
```

Task 2: Create and Configure TUN Interface

Task 2.A: Name of the Interface

在代码此处将 tun 修改成自己名字简拼 cy 。

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

在主机 U(10.9.0.5)上运行 chmod a+x tun.py 和 tun.py 可以观察到修改接口成功。

```
root@eb362125b187:/volumes# chmod a+x tun.py
root@eb362125b187:/volumes# tun.py
Interface Name: cy0
```

然后在主机 U(10.9.0.5) 上运行 ip address 查看所有接口，可发现我们修改的 tun 接口，命名为 cy0。

```
root@eb362125b187:/# 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: cy0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
10: eth0@if11: <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
```

Task 2.B：Set up the TUN Interface

在 tun.py 文件中添加以下两行代码，编译运行后主机 U(10.9.0.5) 上运行 ifconfig 查看所有接口， 可观察到绑定 IP 地址。

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

```
root@eb362125b187:/# ifconfig
cy0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 192.168.53.99  netmask 255.255.255.0  destination 192.168.53.99
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Task 2.c: Read from the TUN Interface

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@eb362125b187:/# 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 ---
10 packets transmitted, 0 received, 100% packet loss, time 9194ms
```

无法 ping 通，并出现如下结果，可以看到程序有输出，但是请求无响应，因为实际主机不存在。

```
root@eb362125b187:/volumes# tun.py
Interface Name: cy0
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
```

再次运行代码，并在主机 U 上 ping 主机 V。

```
root@eb362125b187:/# 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 ---
6 packets transmitted, 0 received, 100% packet loss, time 5116ms
```

由于未添加路由，程序并无输出。

```
root@eb362125b187:/volumes# tun.py
Interface Name: cy0
c^CTraceback (most recent call last):
  File "./tun.py", line 26, in <module>
    packet = os.read(tun, 2048)
KeyboardInterrupt
```

Task 2.d: Write to the TUN Interface

修改代码如下：

```
while True:
# Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print(pkt.summary())
        if ICMP in pkt:
            newip = IP(src=pkt[IP].dst,dst=pkt[IP].src,
ihl=pkt[IP].ihl)
            newip.ttl = 99
            newicmp = ICMP(type = 0, id = pkt[ICMP].id,
seq = pkt[ICMP].seq)
            if pkt.haslayer(Raw):
                data = pkt[Raw].load
                newpkt = newip/newicmp/data
            else:
                newpkt = newip/newicmp
        os.write(tun, bytes(newpkt))
```

运行程序后，在主机 U 上 ping 192.168.53.0/24 网段

```
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=2.49 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=99 time=1.97 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=99 time=1.92 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=99 time=2.07 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=99 time=1.36 ms
64 bytes from 192.168.53.1: icmp_seq=6 ttl=99 time=1.74 ms
64 bytes from 192.168.53.1: icmp_seq=7 ttl=99 time=2.04 ms
64 bytes from 192.168.53.1: icmp_seq=8 ttl=99 time=2.53 ms
64 bytes from 192.168.53.1: icmp_seq=9 ttl=99 time=1.49 ms
64 bytes from 192.168.53.1: icmp_seq=10 ttl=99 time=1.27 ms
64 bytes from 192.168.53.1: icmp_seq=11 ttl=99 time=1.37 ms
^C
--- 192.168.53.1 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10033ms
rtt min/avg/max/mdev = 1.268/1.841/2.534/0.419 ms
```

可见返回的是程序构造的报文，因此仍然没有 ping 通。

```
root@eb362125b187:/volumes# tun.py
Interface Name: cy0
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
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

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

tun_client_task3.py

```python
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'cy%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))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP="10.9.0.11"
SERVER_PORT=9090
while True:
# Get a packet from the tun interface
   packet = os.read(tun, 2048)
   if packet:
     pkt = IP(packet)
     print(pkt.summary())
     sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

tun_server_task3.py

```python
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'cy%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))
    pkt = IP(data)
    print("Inside: {} --> {}".format(pkt.src, pkt.dst))
```

在主机 U 上运行 tun_client_task3.py，在服务器上运行 tun_server_task3.py，再 ping 192.168.53.0/24 网段下的 ip,发现 ping 不通。

```
root@eb362125b187:/# 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 ---
11 packets transmitted, 0 received, 100% packet loss, time 10232ms
```

tun_client_ask3.py 输出如下：

```
root@eb362125b187:/volumes# tun_client_task3.py
Interface Name: cy0
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
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

tun_server_task3.py 输出如下：

```
root@fdc6973467eb:/volumes# tun_server_task3.py
Interface Name: cy0
RTNETLINK answers: File exists
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.1
10.9.0.5:47577 --> 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: Set Up the VPN Server

确保路由器上打开了 IP 转发。

```yaml
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
```

tun_client_task4.py 和 tun_client_task3.py 一致

tun_server_task4.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'cy%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))
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")
```

在服务器上运行 tun_server_task4.py，在主机 U 上运行 tun_client_task4.py，然后在 U 上 ping 192.168.60.5。

```
root@eb362125b187:/# 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 ---
13 packets transmitted, 0 received, 100% packet loss, time 12285ms
```

程序输出如下：

```
root@eb362125b187:/volumes# tun_client_task4.py
Interface Name: cy0
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
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
```

```
root@fdc6973467eb:/volumes# tun_server_task4.py
Interface Name: cy0
RTNETLINK answers: File exists
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
write
10.9.0.5:50795 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```

在服务器上通过 tcpdump 抓取报文。

```
root@fdc6973467eb:/# 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
23:50:19.391030 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 142, seq 1, length 64
23:50:19.391152 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 142, seq 1, length 64
23:50:20.411612 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 142, seq 2, length 64
23:50:20.411640 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 142, seq 2, length 64
23:50:21.433848 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 142, seq 3, length 64
23:50:21.433934 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 142, seq 3, length 64
23:50:22.459477 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 142, seq 4, length 64
23:50:22.459501 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 142, seq 4, length 64
23:50:23.482570 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 142, seq 5, length 64
23:50:23.482599 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 142, seq 5, length 64
23:50:24.409763 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
23:50:24.409817 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
23:50:24.409832 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
23:50:24.409845 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
```

说明 ICMP 报文到达目的主机，但是没有响应。

Task 5: Handling Traffic in Both Directions

tun_client_task5.py

```python
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'cy%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))
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))
```

tun_server_task5.py

```
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'cy%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)
         print("Return: {}--{}".format(pkt.src,pkt.dst))
         sock.sendto(packet,(ip,port))
```

在服务器上运行 tun_server_task5.py，在主机 U 上运行 tun_client_task5.py，在 U 上 ping 192.168.60.5， 发现此时可以 ping 通。

```
root@eb362125b187:/# 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=2.72 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=2.66 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=2.48 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=2.28 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.53 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=2.47 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=2.26 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=2.26 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=1.75 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=2.13 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=1.79 ms
^C
--- 192.168.60.5 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10034ms
rtt min/avg/max/mdev = 1.528/2.212/2.718/0.365 ms
```

程序输出信息如下：

```
root@eb362125b187:/volumes# tun_client_task5.py
Interface Name: cy0
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
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
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@fdc6973467eb:/volumes# tun_server_task5.py
Interface Name: cy0
RTNETLINK answers: File exists
sock...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 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...
10.9.0.5:35326 --> 0.0.0.0:9090
```

telnet 192.168.60.5 ，结果同理。

```
root@eb362125b187:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
f4b6288eee24 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.
```

程序输出如下：

```
root@eb362125b187:/volumes# tun_client_task5.py
Interface Name: cy0
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet S
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
From socket: 192.168.60.5 --> 192.168.53.99
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet A
IP / TCP 192.168.53.99:38798 > 192.168.60.5:telnet PA / Raw
From socket: 192.168.60.5 --> 192.168.53.99
```

```
root@fdc6973467eb:/volumes# tun_server_task5
Interface Name: cy0
RTNETLINK answers: File exists
sock...
10.9.0.5:33857 --> 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...
10.9.0.5:33857 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
sock...
10.9.0.5:33857 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
tun...
Return: 192.168.60.5--192.168.53.99
tun...
Return: 192.168.60.5--192.168.53.99
sock...
10.9.0.5:33857 --> 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...
10.9.0.5:33857 --> 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...
10.9.0.5:33857 --> 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...
10.9.0.5:33857 --> 0.0.0.0:9090
```

Task 6: Tunnel-Breaking Experiment

在 task5telnet 连接建立成功的基础上，停止运行 tun_server_task5.py

```
Inside: 192.168.53.99 --> 192.168.60.5
^CTraceback (most recent call last):
  File "./tun_server_task5.py", line 29, in <module>
    ready,_,_=select.select(fds,[],[])
KeyboardInterrupt
```

发现无法在 U 中输入信息，所有的敲击结果都在缓冲区不停地重发。

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

seed@f4b6288eee24:~$
```

再次运行 tun_server_task5.py， VPN 又建立起来，敲击结果就会显示在终端，可以输入信息。

```
seed@f4b6288eee24:~$ d^Cssszsaxsxascdcsasdscvcc dsccascxexrgegld

-bash: ssszsaxsxascdcsasdscvcc: command not found
seed@f4b6288eee24:~$ ls
```

停止并恢复运行 tun_client_task5.py，结果同理。

```
c^CTraceback (most recent call last):
  File "./tun_client_task5.py", line 26, in <module>
    ready,_,_=select.select(fds,[],[])
KeyboardInterrupt

seed@f4b6288eee24:~$ aaadfsdggfdanjraenelrng
-bash: aaadfsdggfdanjraenelrng: command not found
```