

VPN Lab

57118107 任子悦

Task 1: Network Setup

查看 10.9.0.0/24 和 192.168.60.0/24 两个网段对应的网卡号。

```
5: br-dce06baa77cf: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
   link/ether 02:42:8f:fb:8f:e6 brd ff:ff:ff:ff:ff:ff
   inet 10.9.0.1/24 brd 10.9.0.255 scope global br-dce06baa77cf
       valid_lft forever preferred_lft forever
   inet6 fe80::42:8fff:fefb:8fe6/64 scope link
       valid_lft forever preferred_lft forever
6: br-ledbdea48b41: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
   link/ether 02:42:a6:52:9d:7e brd ff:ff:ff:ff:ff:ff
   inet 192.168.60.1/24 brd 192.168.60.255 scope global br-ledbdea48b41
       valid_lft forever preferred_lft forever
   inet6 fe80::42:a6ff:fe52:9d7e/64 scope link
       valid_lft forever preferred_lft forever
```

先嗅探 10.9.0.0/24 网段，在 VM 上输入命令 `sudo tcpdump -i br-dce06baa77cf -n`

```
[08/04/21]seed@VM:~/.../volumes$ sudo tcpdump -i br-dce06baa77cf -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on br-dce06baa77cf, link-type EN10MB (Ethernet), capture size 262144 bytes
```

VM 上可以看到

```
03:40:45.375715 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 1, length 64
03:40:45.375786 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 1, length 64
03:40:46.407389 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 2, length 64
03:40:46.407533 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 2, length 64
03:40:47.431558 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 3, length 64
03:40:47.431642 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 3, length 64
03:40:48.455923 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 4, length 64
03:40:48.456008 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 4, length 64
03:40:49.483119 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 5, length 64
03:40:49.483217 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 5, length 64
03:40:50.503413 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 12, seq 6, length 64
03:40:50.503518 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 12, seq 6, length 64
03:40:50.567550 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
03:40:50.567645 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

U(10.9.0.5)不可以 ping 通 V(192.168.60.5):

```
03:41:47.946602 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 1, length 64
03:41:48.967337 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 2, length 64
03:41:49.991759 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 3, length 64
03:41:51.016810 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 4, length 64
03:41:52.039329 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 5, length 64
03:41:53.067387 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 6, length 64
03:41:54.087868 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 7, length 64
03:41:55.111646 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 8, length 64
03:41:56.136051 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 9, length 64
03:41:57.160249 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 10, length 64
03:41:58.183211 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 11, length 64
03:41:59.207614 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 12, length 64
03:42:00.232511 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 13, length 64
03:42:01.255218 IP 10.9.0.5 > 192.168.60.5: ICMP echo request, id 13, seq 14, length 64
```

接着先嗅探 192.168.60.0/24 网段，在 VM 上输入命令 `sudo tcpdump -i br-6621fbb29c17 -n`

这时候如果在 U(10.9.0.5)上 ping V(192.168.60.5)，看不到任何结果，说明数据包没有通过 VNP Server。

VNP Server 可以 ping 通 V(192.168.60.5):

```

root@6ed42de28f87:/# 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=0.230 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.206 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.219 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.153 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=64 time=0.164 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=64 time=0.220 ms
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5126ms
rtt min/avg/max/mdev = 0.153/0.198/0.230/0.029 ms
root@6ed42de28f87:/#
03:45:51.623457 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 14, seq 2, length 64
03:45:52.653368 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 14, seq 3, length 64
03:45:52.653495 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 14, seq 3, length 64
03:45:53.672165 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 14, seq 4, length 64
03:45:53.672250 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 14, seq 4, length 64
03:45:54.696022 IP 192.168.60.11 > 192.168.60.5: ICMP echo request, id 14, seq 5, length 64
03:45:54.696112 IP 192.168.60.5 > 192.168.60.11: ICMP echo reply, id 14, seq 5, length 64
03:45:55.724174 ARP Request who-has 192.168.60.11 tell 192.168.60.5 length 28

```

Task 2: Create and Configure TUN Interface

Task 2.b: Set up the TUN Interface

修改程序

```

13
14 # Create the tun interface
15 tun = os.open("/dev/net/tun", os.O_RDWR)
16 ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
17 ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
18
19 # Get the interface name
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22
23 os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
24 os.system("ip link set dev {} up".format(ifname))
25
26 while True:|
27     time.sleep(10)
28

```

在 10.9.0.5 上运行脚本，再查看端口，已经给 tun0 分配了 IP 地址 192.168.53.99/24。

```

root@08cbef902277:/# ip addr
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: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/noprobe
    inet 192.168.53.99/24 scope global tun0
        valid_lft forever preferred_lft forever
9: eth0@if10: <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.c: Read from the TUN Interface

修改程序如下

```

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

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

while True:
    packet = os.read(tun,2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())

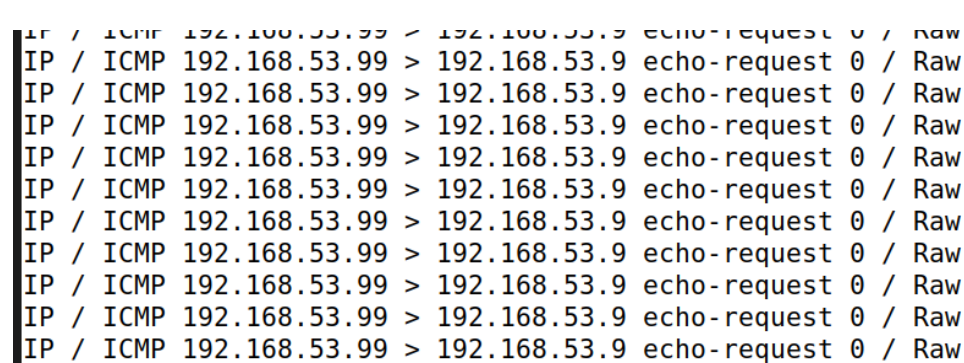
```

在 10.9.0.5 上 ping 192.168.53.9 结果如下

```

root@08cbef902277:/# ping 192.168.53.9
PING 192.168.53.9 (192.168.53.9) 56(84) bytes of data.

```



```

IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw

```

Task 2.d: Write to the TUN Interface

修改程序如下

```

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

while True:
    packet = os.read(tun,2048)
    if True:
        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))

```

运行脚本，再打开一个 U(10.9.0.5)的 shell 去 ping 192.168.53.9:

```
[08/04/21]seed@VM:~/../volumes$ docksh 08
root@08cbef902277:/# ping 192.168.53.9
PING 192.168.53.9 (192.168.53.9) 56(84) bytes of data.
64 bytes from 192.168.53.9: icmp_seq=1 ttl=99 time=4.31 ms
64 bytes from 192.168.53.9: icmp_seq=2 ttl=99 time=9.17 ms
64 bytes from 192.168.53.9: icmp_seq=3 ttl=99 time=9.64 ms
64 bytes from 192.168.53.9: icmp_seq=4 ttl=99 time=10.8 ms
64 bytes from 192.168.53.9: icmp_seq=5 ttl=99 time=9.91 ms
64 bytes from 192.168.53.9: icmp_seq=6 ttl=99 time=9.60 ms
^C
--- 192.168.53.9 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5013ms
rtt min/avg/max/mdev = 4.310/8.901/10.774/2.110 ms
root@08cbef902277:/#

root@08cbef902277:/volumes# python3 tun.py
Interface Name: tun0
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.9 echo-request 0 / Raw
```

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

修改脚本如下:

```
# 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: {}".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)

while True:
    packet = os.read(tun, 2048)
    if packet:
        sock.sendto(packet, ("10.9.0.11", 9090))
```

tun-server.py 程序如下:

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

在 10.9.0.5 上输入命令 `ip route add 192.168.60.0/24 dev tun0 via 192.168.53.99` 和 `ping 192.168.60.5`, 此时显示 ping 不通, 但是 server-router 结果如下, 说明已经通过隧道发到了 server-router:

```
root@6ed42de28f87:/volumes# python3 tun-server.py
10.9.0.5:41440 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.6
10.9.0.5:41440 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.6
10.9.0.5:41440 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.6
10.9.0.5:41440 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.6
10.9.0.5:41440 --> 0.0.0.0:9090
```

Task 4: Set Up the VPN Server

编写脚本如下

```
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000

tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'tun%d' % 0, IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.02/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
IP_A = "0.0.0.0"
PORT = 9090
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, data)
```


在 server-router 上运行上述程序, 在 U 上再次运行 Task 3 中的程序; 再打开一个 server-router 的 shell, 输入命令 `tcpdump -nni eth1`; 打开一个 U(10.9.0.5) 的 shell, 输入命令 `ping 192.168.60.5`。可在运行 python 程序的 server-router 的 shell 上看到:

```
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:59103 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```

可在输入 `tcpdump -nni eth1` 命令的 server-router 的 shell 上看到:

```
15:19:43.702938 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 186, seq 1, length 64
15:19:43.703045 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 186, seq 1, length 64
15:19:44.708431 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 186, seq 2, length 64
15:19:44.708460 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 186, seq 2, length 64
15:19:45.733132 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 186, seq 3, length 64
15:19:45.733165 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 186, seq 3, length 64
15:19:46.756627 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 186, seq 4, length 64
15:19:46.756658 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 186, seq 4, length 64
15:19:47.780816 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 186, seq 5, length 64
15:19:47.780893 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 186, seq 5, length 64
```

可见, server-router 已经将通过隧道收到的报文发给目的主机, 目的主机也给出了回应。到此, 隧道一个方向的通信已经建立了。

Task 5: Handling Traffic in Both Directions

在 10.9.0.5 上运行如下脚本

```

1#!/usr/bin/env python3
2import fcntl
3import struct
4import os
5import time
6from scapy.all import *
7
8TUNSETIFF = 0x400454ca
9IFF_TUN = 0x0001
10IFF_TAP = 0x0002
11IFF_NO_PI = 0x1000
12
13# Create the tun interface
14tun = os.open("/dev/net/tun", os.O_RDWR)
15ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
16ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
17
18# Get the interface name
19ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
20print("Interface Name: {}".format(ifname))
21os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
22os.system("ip link set dev {} up".format(ifname))
23os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
24
25sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
26SERVER_IP="10.9.0.11"
27SERVER_PORT=9090
28fds = [sock,tun]
29
30while True:
31    ready,_,_ = select.select(fds,[],[])
32    for fd in ready:
33        if fd is sock:
34            data,(ip,port)=sock.recvfrom(2048)
35            pkt = IP(data)
36            print("From socket: {} --> {}".format(pkt.src,pkt.dst))
37            os.write(tun,data)
38        if fd is tun:
39            packet = os.read(tun,2048)
40            if packet:
41                pkt = IP(packet)
42                print(pkt.summary())
43                sock.sendto(packet,(SERVER_IP,SERVER_PORT))

```

在 server-router 上运行如下脚本：

```

1#!/usr/bin/env python3
2import fcntl
3import struct
4import os
5import time
6from scapy.all import *
7TUNSETIFF = 0x400454ca
8IFF_TUN = 0x0001
9IFF_TAP = 0x0002
10IFF_NO_PI = 0x1000
11
12# Create the tun interface
13tun = os.open("/dev/net/tun", os.O_RDWR)
14ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
15ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
16
17# Get the interface name
18ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
19print("Interface Name: {}".format(ifname))
20os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
21os.system("ip link set dev {} up".format(ifname))
22os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
23
24sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
25SERVER_IP = "0.0.0.0"
26SERVER_PORT = 9090
27ip = '10.9.0.5'
28port = 10000
29sock.bind((SERVER_IP, SERVER_PORT))
30fds = [sock,tun]
31
32while True:
33    ready,_,_ = select.select(fds,[],[])
34    for fd in ready:
35        if fd is sock:
36            print("sock...")
37            data,(ip, port) = sock.recvfrom(2048)
38            print("{}: {} --> {}: {}".format(ip, port, SERVER_IP,
SERVER_PORT))
39            pkt = IP(data)
40            print("Inside: {} --> {}".format(pkt.src, pkt.dst))
41            os.write(tun, data)
42        if fd is tun:
43            print("tun...")
44            packet = os.read(tun,2048)
45            pkt = IP(packet)
46            print("Return: {}--{}".format(pkt.src,pkt.dst))
47            sock.sendto(packet,(ip,port))

```

10.9.0.5 能够 ping 通 192.168.60.5

```

root@08cbef902277:/# 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=3.74 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=10.3 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=14.2 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=10.5 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=2.22 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=13.5 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=6.47 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=10.6 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=13.7 ms

```


Task 6: Tunnel-Breaking Experiment

保持 Task 5 中的两个 python 程序继续运行，在 10.9.0.5 上 telnet 192.168.60.5，登陆成功：

```
root@08cbef902277:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5a2556993865 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
```