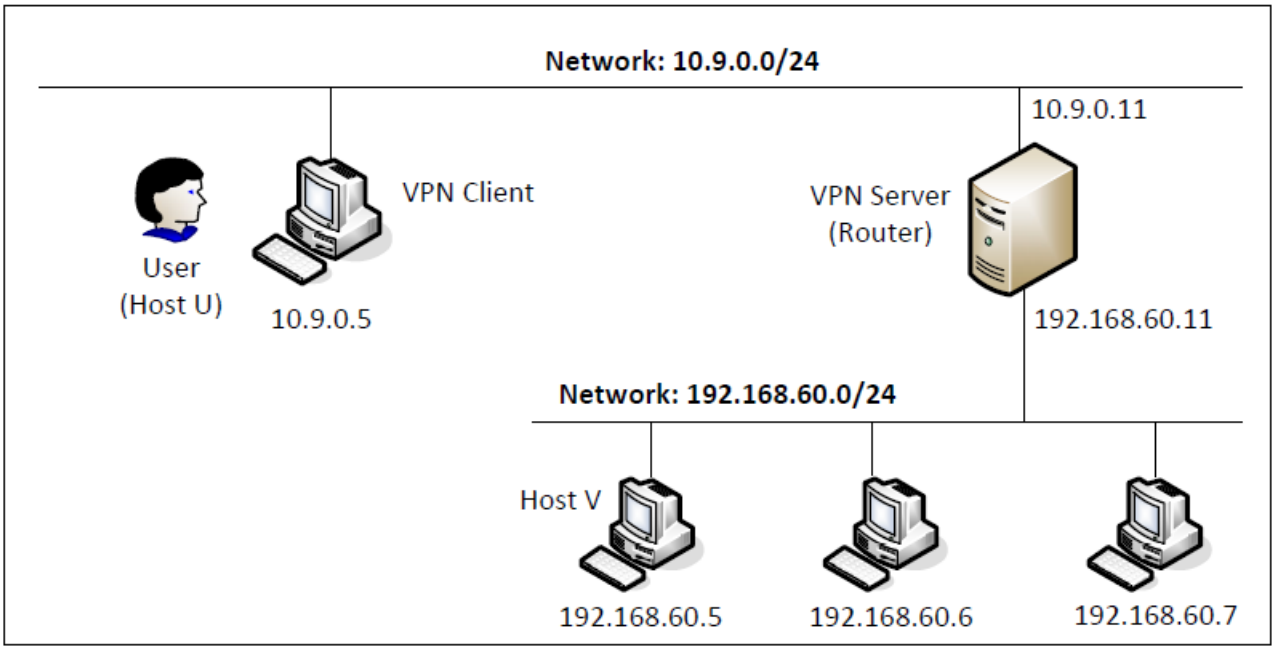
Chapter 6:VPN Lab: The Container Version

57118132 吴嘉琪
852809049@qq.com
Southeast University — 2021 年 7 月 27 日

网络拓扑



Task 1: Network Setup

验证Host U可以与VPN Server通信以及在路由器上tcpdump捕获的报文。

```
root@26e90a55e8a2:/# 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=0.108 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.103 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.077 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.113 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.103 ms
64 bytes from 10.9.0.11: icmp_seq=6 ttl=64 time=0.059 ms
```

```
root@bc5a6b24ba5c:/# 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
14:56:39.217666 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 1, length 64
14:56:39.217704 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 1, length 64
14:56:40.235788 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 2, length 64
14:56:40.235814 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 2, length 64
14:56:41.259382 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 3, length 64
14:56:41.259413 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 3, length 64
14:56:42.283422 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 4, length 64
14:56:42.283448 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 4, length 64
```

验证Host V可以与VPN Server通信以及在路由器上tcpdump捕获的报文。

```
root@8c229e3fb092:/# 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=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.144 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.084 ms
```

```
root@bc5a6b24ba5c:/# 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
15:02:22.133619 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 30, seq 1,
    length 64
15:02:22.133637 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 30, seq 1, length
    64
15:02:23.137716 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 30, seq 2,
    length 64
15:02:23.137755 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 30, seq 2, length
    64
15:02:24.141670 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 30, seq 3,
    length 64
15:02:24.141697 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 30, seq 3, length
    64
15:02:25.163152 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 30, seq 4,
    length 64
15:02:25.163177 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 30, seq 4, length
    64
15:02:26.190562 IP 192.168.60.5 > 10.9.0.11: ICMP echo request, id 30, seq 5,
    length 64
15:02:26.190584 IP 10.9.0.11 > 192.168.60.5: ICMP echo reply, id 30, seq 5, length
    64
```

验证Host U不可与Host V通信。

```
root@26e90a55e8a2:/# 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 ---
22 packets transmitted, 0 received, 100% packet loss, time 21482ms
```

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

修改代码，在代码此处将tun修改成自己名字简拼wjq。

```
ifr = struct.pack('16sH', b'wjq%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@26e90a55e8a2:/volumes# chmod a+x tun.py
root@26e90a55e8a2:/volumes# tun.py
Interface Name: wjq0
```

然后在Host 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wjq0。

```
oot@26e90a55e8a2:/volumes# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
    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: wjq0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group defa
    link/none
9: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    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Host U(10.9.0.5)上运行ifconfig查看所有接口，可观察到绑定IP地址。

```
wjq0: 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
```

Task 2.c: Read from the TUN Interface

ping 192.168.53.5 ,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@5fc7a6b64b73:/volumes# ping 192.168.53.5
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
^C
--- 192.168.53.5 ping statistics ---
53 packets transmitted, 0 received, 100% packet loss, time 53256ms
```

在ping 192.168.60.5 时，由于未添加路由，程序并无输出。

Task 2.d: Write to the TUN Interface

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'zhl%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))

while True:
        # Get a packet from the tun interface
        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))
```

此时我们ping 192.168.53.5 可以观察到返回的是我们构造的报文(ttl=99)，在接口处我们可以看到完整的IP/ICMP/Raw 三层报文。

```
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
```

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

在 Host U容器上 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'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 {} via 192.168.53.99".format(ifname
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    time.sleep(5)
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
```

```
    if packet:
        ip = IP(packet)
        print(ip.summary())
        SERVER_IP = '10.9.0.11'
        sock.sendto(packet, (SERVER_IP, 9090))
```

在 Router容器上 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'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 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket
SERVER_IP = '0.0.0.0'
SERVER_PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((SERVER_IP, SERVER_PORT))

while True:
    # Get a packet from socket
    data, (ip, port) = sock.recvfrom(2048)
    packet = IP(data)
```

```python
    if packet:
        print("{}:{} --> {}:{}".format(ip, port, SERVER_IP, SERVER_PORT))
        print("   Inside: {} --> {}".format(packet.src, packet.dst))
        # Send the packet via the tunnel
        os.write(tun, data)
```

ping 192.168.60.12
在Router端启动程序

```
root@579c7416bec7:/volumes# python3 tun_server.py
Interface Name: tun0
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
   Inside: 192.168.53.99 --> 192.168.60.12
10.9.0.5:46834 --> 0.0.0.0:9090
```

在Host U端启动程序

```
root@cffa1bf13172:/volumes# python3 tun_client.py
Interface Name: tun0
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
```

```
IP / ICMP 192.168.53.99 > 192.168.60.12 echo-request 0 / Raw
```

VPN Client 和 VPN Server通过 tun0接发报文，之后服务端通过 TUN把数据包发送到虚拟接口进行解封装，和自动转发。

Task 4: Set Up the VPN Server

在 Router容器上 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'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 192.168.53.1/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket
SERVER_IP = '0.0.0.0'
SERVER_PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((SERVER_IP, SERVER_PORT))

while True:
    # Get a packet from socket
    data, (ip, port) = sock.recvfrom(2048)
    packet = IP(data)
    if packet:
```

```
        print("{}:{} --> {}:{}".format(ip, port, SERVER_IP, SERVER_PORT))
        print("    Inside: {} --> {}".format(packet.src, packet.dst))
        # Send the packet via the tunnel
        os.write(tun, data)
```

Task 5: Handling Traffic in Both Directions

编写 tun_server_select.py在 Router容器,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'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 192.168.53.1/24 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))
sock_1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)


while True:
    # this will block until at least one interface is ready
    ready, _, _ = select.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))
            if pkt:
                os.write(tun, data)
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            if pkt:
                SERVER_IP = '10.9.0.5'
                sock_1.sendto(packet, (SERVER_IP, 9090))
```

编写 tun_client_select.py在 Host U容器,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'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 {} via 192.168.53.99".format(ifname

IP_A = "0.0.0.0"
PORT = 9090
```

```python
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
sock_1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    # this will block until at least one interface is ready
    ready, _, _ = select.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))
            if pkt:
                os.write(tun, data)
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
            if pkt:
                SERVER_IP = '10.9.0.11'
                sock_1.sendto(packet, (SERVER_IP, 9090))
```

在Host U容器上执行 ping 192.168.60.5命令，结果如所示
router结果

```
root@3251fd42ecda:/volumes# python3 tun_server_select.py
Interface Name: tun0
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

Host U结果

```
root@ca06e6ac4ab9:/volumes# python3 tun_client_select.py
Interface Name: tun0
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
```

执行telnet 192.168.60.5之后：
router结果

```
root@3251fd42ecda:/volumes# python3 tun_server_select.py
Interface Name: tun0
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.53.99 --> 192.168.60.5
```

Host U结果

```
root@ca06e6ac4ab9:/volumes# python3 tun_client_select.py
```

```
Interface Name: tun0
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
From tun ==>: 192.168.53.99 --> 192.168.60.5
From socket <==: 192.168.60.5 --> 192.168.53.99
```

发现能攻击成功

```
root@ca06e6ac4ab9:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
11eb6614bd71 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: Tunnel-Breaking Experiment

telnet连接后终止server，会无法输入内容，这时候重新启动server仍能显示刚才键入内容，

```
seed@11eb6614bd71:~$
```

```
seed@11eb6614bd71:~$ aaaaaasasdasfas
```