VPN Lab：The Container Version

57118210 郑荔文

2 Task 1： Network Setup

Testing

```
[07/25/21]seed@VM:~/.../Labsetup$ dockps
d73420c809d7  host-192.168.60.5
18b2d5ed0c6f  host-192.168.60.6
096b4b55f6b4  server-router
11ea6b63138c  client-10.9.0.5
```

1.Host U 可以和 VPN 服务器连接

```
root@11ea6b63138c:/# 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=0.116 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.048 ms
^C
```

2.VPN 服务器可以和 Host V 连接

```
root@096b4b55f6b4:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.047 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.049 ms
```

3. Host U 不可以和 Host V 连接

```
root@11ea6b63138c:/# 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 ---
4 packets transmitted, 0 received, 100% packet loss, time 3082ms

root@11ea6b63138c:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
^C
--- 192.168.60.6 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3058ms
```

4.在路由器上运行 Tcpdump，可以捕捉到数据包

```
root@11ea6b63138c:/# 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=0.084 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.056 ms
64 bytes from 10.9.0.11: icmp_seq=5 ttl=64 time=0.052 ms
^C
--- 10.9.0.11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4077ms
rtt min/avg/max/mdev = 0.052/0.063/0.084/0.011 ms
```

```
root@096b4b55f6b4:/# 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
13:47:18.888275 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 19, seq 1, length 64
13:47:18.888291 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 19, seq 1, length 64
13:47:19.896026 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 19, seq 2, length 64
13:47:19.896042 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 19, seq 2, length 64
13:47:20.920500 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 19, seq 3, length 64
13:47:20.920517 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 19, seq 3, length 64
13:47:21.941656 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 19, seq 4, length 64
13:47:21.941673 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 19, seq 4, length 64
13:47:22.965273 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 19, seq 5, length 64
13:47:22.965289 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 19, seq 5, length 64
13:47:24.024344 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
13:47:24.024479 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
13:47:24.024484 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
13:47:24.024486 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

3 task 2：Create and Configure TUN Interface

3.1 Task 2.a: Name of the Interface

使用 root 权限在 Host U 上运行 tun.py

```
root@11ea6b63138c:/volumes# chmod a+x tun.py
root@11ea6b63138c:/volumes# tun.py
Interface Name: tun0
```

在 ip address 命令下查看各接口信息，发现 tun0 接口

```
[07/25/21]seed@VM:~/.../Labsetup$ docksh 11
root@11ea6b63138c:/# 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: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
24: eth0@if25: <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
root@11ea6b63138c:/#
```

修改 tun0 接口的名字

修改 tun.py 的代码，将 tun0 接口修改为 liwen0

```
#!/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'liwen%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)

```

重新运行该程序，则发现接口的名字发生改变

```
root@11ea6b63138c:/volumes# tun.py
Interface Name: liwen0
```

在 ip address 命令下也发现了更改后的名字

```
root@11ea6b63138c:/# 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: liwen0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
24: eth0@if25: <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
```

3.2 Task2.b：Set up the TUN interface

设置 liwen0 接口的信息，首先为其安排一个 ip 地址，并将其设为 up

```
root@11ea6b63138c:/# ip addr add 192.168.53.99/24 dev liwen0
root@11ea6b63138c:/# ip link set dev liwen0 up
```

在运行上述两条命令之后，再运行 ip address 命令，发现其中的 liwen0 接口（tun 接口）已经被 up，并且有对应分配的 ip 地址 192.168.53.99/24

```
root@11ea6b63138c:/# 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
4: liwen0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 50(
    link/none
    inet 192.168.53.99/24 scope global liwen0
       valid_lft forever preferred_lft forever
24: eth0@if25: <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
```

将这两条命令写入脚本中，也可以达到相同的效果

```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'liwen%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:
   time.sleep(10)

```

```
root@11ea6b63138c:/volumes# tun.py
Interface Name: liwen0
```

其结果如下:

```
root@11ea6b63138c:/# 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
5: liwen0: <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 liwen0
       valid_lft forever preferred_lft forever
24: eth0@if25: <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
```

在对 tun 接口进行前后，出现了区别，即 inet 192.168.53.99/24 scope global liwen0 Valid_lft forever preferred_lft forever，在设置之前 tun 接口无法实际使用，在设置之后，可以后续使用

3.3 Task 2.c: Read from the TUN Interface

为了从 TUN 接口读出信息，修改 tun.py 如下:

```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'liwen%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:
   packet=os.read(tun,2048)
   if packet:
        ip=IP(packet)
        print(ip.summary())
```

在 Host U 上 ping 192.168.53.0/24 网段内的主机，在 tun.py 程序下打印出如下信息:

```
root@11ea6b63138c:/# 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 ---
15 packets transmitted, 0 received, 100% packet loss, time 14336ms
```

```
root@11ea6b63138c:/volumes# tun.py
Interface Name: liwen0
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
```

在 Host U 上 ping 192.168.60.0/24 网段内的主机，在 tun.py 程序下并无信息

```
root@11ea6b63138c:/# 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 13304ms
```

```
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
```

3.4 Task 2.d：Write to the TUN Interface

修改 tun.py 程序如下：

对于从 TUN 端口读入的数据包，如果该报文为 ICMP 报文，将源地址和目的地址交换，构造 ICMP reply 报文，加上负载部分，将其写入 TUN 端口

```
26
27 while True:
28         packet=os.read(tun,2048)
29         if True:
30                 pkt=IP(packet)
31                 print(pkt.summary())
32                 if ICMP in pkt:
33                         newip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
34                         newip.ttl=64
35                         newicmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
36                         if pkt.haslayer(Raw):
37                                 data=pkt[Raw].load
38                                 newpkt=newip/newicmp/data
39                         else:
40                                 newpkt=newip/newicmp
41                 os.write(tun,bytes(newpkt))
```

运行上述代码，在 Host U 上 ping 192.168.53.0/24 网段下的主机（192.168.53.66），发现可以 ping 通（实际没有通，但是会返回 icmp reply 报文看起来就像通一样）

```
root@39da73e15698:/# ping 192.168.53.66
PING 192.168.53.66 (192.168.53.66) 56(84) bytes of data.
64 bytes from 192.168.53.66: icmp_seq=1 ttl=64 time=25.7 ms
64 bytes from 192.168.53.66: icmp_seq=2 ttl=64 time=1.36 ms
64 bytes from 192.168.53.66: icmp_seq=3 ttl=64 time=1.39 ms
64 bytes from 192.168.53.66: icmp_seq=4 ttl=64 time=1.22 ms
```

在 tun.py 处的输出如下：

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
IP / ICMP 192.168.53.99 > 192.168.53.66 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.66 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.66 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.66 echo-request 0 / Raw
```

在接口处随机输入字符，并无反应

```
root@16292e248858:/volumes# ./tun_spoof.py
Interface Name: liwen0
123513
```

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

在 router 上运行 tun_server.py

```
root@8af4c06cd0d6:/volumes# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 109  bytes 13710 (13.7 KB)
        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

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 108  bytes 13668 (13.6 KB)
        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
```

将提取负载部分的 ip 头，输出内层源地址和目的地址

Open a file ip_redirect.py docker-compose.yml tun_server.py tun_spoof.py

```
1 from scapy.all import *
2 IP_A="0.0.0.0"
3 PORT=9090
4
5 sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
6 sock.bind((IP_A,PORT))
7 while True:
8         data,(ip,port)=sock.recvfrom(2048)
9         print("{}:{}-->{}:{}".format(ip,port,IP_A,PORT))
10        pkt=IP(data)
11        print("         Inside:{}-->{}".format(pkt.src,pkt.dst))
```

在 Host U 上运行 tun_client.py 程序，将报文发往 10.9.0.11（VPN_SERCER）的 9090 端口

```
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 os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
26
27 sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
28
29 while True:
30         packet=os.read(tun,2048)
31         if packet:
32                 ip=IP(packet)
33                 print(ip.summary())
34                 sock.sendto(packet,("10.9.0.11",9090))
35
```

在 ping 192.168.53.0/24 的网段的主机时

```
root@39da73e15698:/# ping 192.168.53.67
PING 192.168.53.67 (192.168.53.67) 56(84) bytes of data.
```

```
root@39da73e15698:/# ping 192.168.53.67
PING 192.168.53.67 (192.168.53.67) 56(84) bytes of data.
^C
--- 192.168.53.67 ping statistics ---
48 packets transmitted, 0 received, 100% packet loss, time 48133ms
```

在 Host U 上有如下输出：

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.67 echo-request 0 / Raw
```

在 VPN server 上有如下输出：

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
10.9.0.5:53221-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.53.67
10.9.0.5:53221-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.53.67
10.9.0.5:53221-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.53.67
10.9.0.5:53221-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.53.67
10.9.0.5:53221-->0.0.0.0:9090
```

在未增加路由时，ping 1923168.60.0/24 网段的主机时

```
root@39da73e15698:/# 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 ---
34 packets transmitted, 0 received, 100% packet loss, time 33776ms
```

此时在 tun_client.py 和 tun_server.py 上均无输出

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
```

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
```

在增加 192.168.60.0/24 的路由后，

```
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 os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
26
```

ping 192.168.60.5

```
root@39da73e15698:/# 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 6136ms
```

在 tun_client.py 和 tun_server.py 上均有输出

在 VPN server 上有如下输出：

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
10.9.0.5:56112-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
10.9.0.5:56112-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
10.9.0.5:56112-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
10.9.0.5:56112-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
10.9.0.5:56112-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
```

在 Host U 上输出如下：

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
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
```

表示 ICMP 报文被 tun_server.py 接收

5 Task 4：Set Up the VPN Server

在 docker-compose.yml 中 net.ipv4.ip_forward 已经被设定为 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
```

通过 ifconfig 查看 router 各个接口的 ip 地址

```
root@8af4c06cd0d6:/volumes# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.11  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:0b  txqueuelen 0  (Ethernet)
        RX packets 109  bytes 13710 (13.7 KB)
        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

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.11  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:0b  txqueuelen 0  (Ethernet)
        RX packets 108  bytes 13668 (13.6 KB)
        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
```

在 ping 192.168.60.5

```
root@39da73e15698:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

在 router 上 tcpdump eth1 端口，暂无输出

```
root@8af4c06cd0d6:/# 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
```

修改 tun_server.py,将 data 再写入 tun 中

```
IP_A="0.0.0.0"
PORT=9090

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'liwen%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))

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))
        os.write(tun,data)
        print("write")
```

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
Interface Name: liwen0
10.9.0.5:41904-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
write
10.9.0.5:41904-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
write
10.9.0.5:41904-->0.0.0.0:9090
        Inside:192.168.53.99-->192.168.60.5
write
```

运行 tun_client.py,同时 ping 192.168.60.5,

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
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
```

此时再 router 的 tcpdump 中出现信息，说明 ICMP 报文到达 Host V（192.168.60.5）

```
root@8af4c06cd0d6:/# 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
05:39:01.703895 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 201, seq 10, length 64
05:39:01.703919 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 201, seq 10, length 64
05:39:02.725284 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 201, seq 11, length 64
05:39:02.725308 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 201, seq 11, length 64
05:39:03.749158 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 201, seq 12, length 64
05:39:03.749183 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 201, seq 12, length 64
05:39:04.774734 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 201, seq 13, length 64
```

但是由于此时 ICMP reply 没有被返回，所以在 Host U ping 的时候，看似没有 ping 通

```
root@39da73e15698:/# 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 ---
16 packets transmitted, 0 received, 100% packet loss, time 15351ms
```

6 Task 5：Handling Traffic in Both Directions

为了使得回复的报文可以通过 VPN tun 返回，需要修改 tun_client.py 报文如下：

```
ifr = struct.pack('16sH', b'liwen%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)
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("From tun==>:{}-->{}".format(pkt.src,pkt.dst))
                                sock.sendto(packet,("10.9.0.11",9090))

```

修改 tun_server.py 报文如下：

```python
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))

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


ip="10.9.0.5"
port=10000
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,"0.0.0.0",9090))
                        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))

```

此时从 Host U ping Host V，可以 ping 通，且之间存在报文交互

```
root@39da73e15698:/# 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=3.15 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=1.66 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=1.52 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.86 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.50 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=1.33 ms
^C
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5013ms
rtt min/avg/max/mdev = 1.333/1.837/3.151/0.608 ms
```

在 tun_server.py 的输出如下：

外层 ip 为从 10.9.0.5 到 0.0.0.0，内层 ip 为 192.168.53.99 到 192.168.60.5，又从 192.168.60.5 返回 192.168.53.99

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
Interface Name: liwen0
sock...
10.9.0.5:47520-->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...
10.9.0.5:47520-->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...
10.9.0.5:47520-->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...
10.9.0.5:47520-->0.0.0.0:9090
```

在 tun_client.py 的输出如下：

从 tun 取得的报文从 192.168.53.99 到达 192.168.60.5.，再将从 192.168.60.5 到 192.168.53.99 的返回报文送给 socket

```
root@39da73e15698:/volumes# ./tun_client.py
Interface Name: liwen0
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
```

从 Host U telnet Host V 也可以登录上

```
root@39da73e15698:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
ed2434d55547 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

7 Task 6：Tunnel-Breaking Experiment

在 telnet 连接之后，停止 tun_server.py

```
^CTraceback (most recent call last):
  File "tun_server.py", line 37, in <module>
    ready,_,_=select.select(fds,[],[])
KeyboardInterrupt

root@8af4c06cd0d6:/volumes#
```

此时在远程登陆时无法输入

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

seed@ed2434d55547:~$
```

ping 192.168.60.5 时，也得不到 reply 的数据包

```
root@39da73e15698:/# 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 ---
3 packets transmitted, 0 received, 100% packet loss, time 2053ms
```

在 tun_client.py 处只有一边的输出

```
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
From tun==>:192.168.53.99-->192.168.60.5
```

重新运行 tun_server.py 后，出现返回报文

```
root@8af4c06cd0d6:/volumes# python3 tun_server.py
Interface Name: liwen0
sock...
10.9.0.5:47520-->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...
```

此时在远程登录时，可以输入信息（存在一定延时）

```
seed@ed2434d55547:~$ 1231232
-bash: 1231232: command not found
seed@ed2434d55547:~$ 122333exit
```