VPN Lab: The Container Version

目录

Task 1: Network Setup

```
[08/01/21]seed@VM:~/.../Labsetup$ dockps
eeb11a397638  host-192.168.60.8
7f1997463db7  host-192.168.60.7
d647f9154f80  client-10.9.0.5
38c86b079f60  server-router
```

10.9.0.5 ping 10.9.0.11，成功
10.9.0.5 ping 192.168.60.5，失败。

```
[08/01/21]seed@VM:~/.../Labsetup$ docksh d6
root@d647f9154f80:/# 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=0.114 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.091 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.153 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.159 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.091/0.129/0.159/0.028 ms
root@d647f9154f80:/# 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 ---
8 packets transmitted, 0 received, 100% packet loss, time 7174ms

root@d647f9154f80:/#
```

Server-router（10.9.0.11） ping 192.168.60.5，成功

```
[08/01/21]seed@VM:~/.../Labsetup$ docksh 38
root@38c86b079f60:/# 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=0.112 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.102 ms
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3082ms
rtt min/avg/max/mdev = 0.062/0.093/0.112/0.018 ms
root@38c86b079f60:/#
```

Server-router 监听 10.9.0.0/24 网段：

在 router 上监听，10.9.0.5 ping 10.9.0.11

```
root@38c86b079f60:/# 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
01:04:36.078899 IP6 fe80::42:feff:fe74:a7cb > ff02::2: ICMP6, router solicitation, length 16
01:05:12.883870 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 1, length 64
01:05:12.883886 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 1, length 64
01:05:13.901155 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 2, length 64
01:05:13.901184 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 2, length 64
01:05:14.925409 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 3, length 64
01:05:14.925468 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 3, length 64
01:05:15.949653 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 4, length 64
01:05:15.949706 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 4, length 64
01:05:16.973126 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 5, length 64
01:05:16.973188 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 5, length 64
01:05:18.060637 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
01:05:18.060909 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
01:05:18.060926 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
01:05:18.060935 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
01:05:31.890271 IP 10.9.0.1.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
01:05:34.851901 IP6 fe80::7486:ccff:fe9f:bc1d.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
01:05:35.324104 IP6 fe80::42:feff:fe74:a7cb.5353 > ff02::fb.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
^C
18 packets captured
18 packets received by filter
0 packets dropped by kernel
root@38c86b079f60:/#
```

Server-router 监听 10.9.0.0/24 网段：

在 router 上监听（注意修改命令中的端口号为 eth1），192.168.60.5 ping 192.168.60.11

```
root@38c86b079f60:/# 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
01:12:35.709005 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 1, length 64
01:12:35.709043 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 1, length 64
01:12:36.717397 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 2, length 64
01:12:36.717467 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 2, length 64
01:12:37.741933 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 3, length 64
01:12:37.741974 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 3, length 64
01:12:38.765033 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 4, length 64
01:12:38.765067 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 4, length 64
01:12:39.790827 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 5, length 64
01:12:39.790871 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 5, length 64
01:12:40.813384 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 33, seq 6, length 64
01:12:40.813406 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 33, seq 6, length 64
01:12:40.940540 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
01:12:40.940809 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@38c86b079f60:/#
```

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

```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'qmy%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)

```

在 10.9.0.5 上运行修改后的程序。端口号被修改为自己的名字。

```
root@d647f9154f80:/# cd volumes
root@d647f9154f80:/volumes# chmod a+x tun.py
root@d647f9154f80:/volumes# tun.py
Interface Name: qmy0
```

保持如上程序开启，新开一个窗口，查看 10.9.0.5 的所有端口，修改的端口生效。

```
[08/01/21]seed@VM:~/.../Labsetup$ docksh d6
root@d647f9154f80:/# 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: qmy0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
38: eth0@if39: <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@d647f9154f80:/#
```

Task 2.B：Set up the TUN Interface

增加如下两行代码

```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'qmy%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)

```

绑定了 ip 地址。

```
root@d647f9154f80:/# 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: qmy0: <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 qmy0
       valid_lft forever preferred_lft forever
38: eth0@if39: <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@d647f9154f80:/#
```

Task 2.c: Read from the TUN Interface

部分代码修改为:

```python
while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
        time.sleep(10)

```

10.9.0.5 ping 192.168.53.1，ping 不通。但是本程序有输出。

10.9.0.5 ping 192.168.60.1，ping 通。但是本程序没有输出。

```
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
27 packets transmitted, 0 received, 100% packet loss, time 26662ms

root@d647f9154f80:/# ping 192.168.60.1
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
64 bytes from 192.168.60.1: icmp_seq=1 ttl=64 time=0.107 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.132 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.143 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.144 ms
64 bytes from 192.168.60.1: icmp_seq=6 ttl=64 time=0.145 ms
64 bytes from 192.168.60.1: icmp_seq=7 ttl=64 time=0.141 ms
64 bytes from 192.168.60.1: icmp_seq=8 ttl=64 time=0.107 ms
64 bytes from 192.168.60.1: icmp_seq=9 ttl=64 time=0.127 ms
^C
--- 192.168.60.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8200ms
rtt min/avg/max/mdev = 0.088/0.126/0.145/0.019 ms
root@d647f9154f80:/#
```

```
root@d647f9154f80:/volumes# tun.py
Interface Name: qmy0
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
^CTraceback (most recent call last):
  File "./tun.py", line 32, in <module>
    time.sleep(10)
KeyboardInterrupt

root@d647f9154f80:/volumes# tun.py
Interface Name: qmy0
```

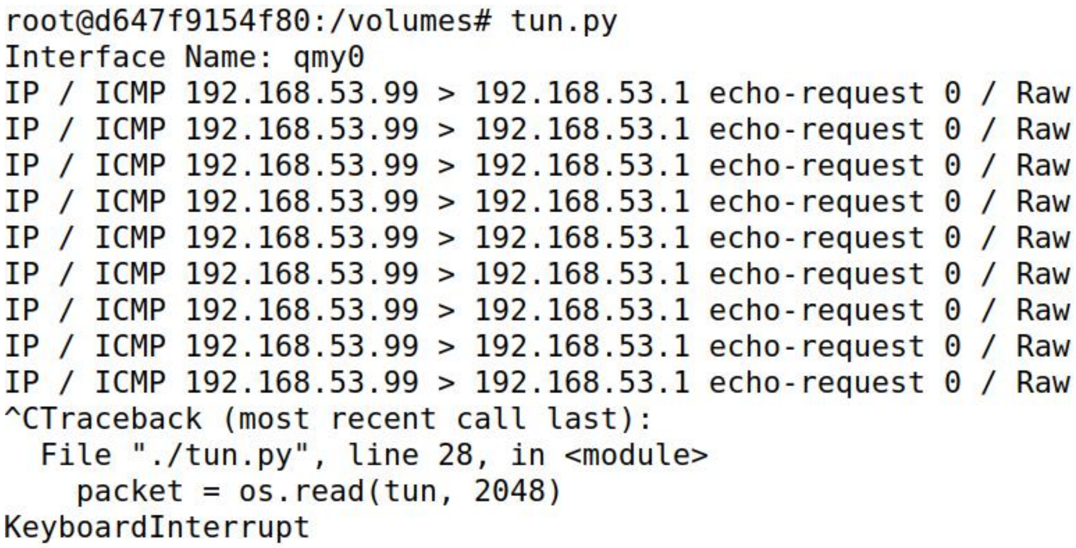
Task 2.d: Write to the TUN Interface

部分代码修改如下:

```python
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))
```

ping 192.168.53.1, 观察到返回的是代码构造的报文（ttl=99），在接口处收到了 IP/ICMP/Raw 三层报文的回复。

```
root@d647f9154f80:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.1: icmp_seq=1 ttl=99 time=4.52 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=99 time=7.10 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=99 time=8.14 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=99 time=11.3 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=99 time=11.0 ms
64 bytes from 192.168.53.1: icmp_seq=6 ttl=99 time=7.25 ms
64 bytes from 192.168.53.1: icmp_seq=7 ttl=99 time=8.58 ms
64 bytes from 192.168.53.1: icmp_seq=8 ttl=99 time=4.57 ms
64 bytes from 192.168.53.1: icmp_seq=9 ttl=99 time=9.74 ms
^C
--- 192.168.53.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8054ms
rtt min/avg/max/mdev = 4.517/8.031/11.342/2.334 ms
root@d647f9154f80:/#
```



```
root@d647f9154f80:/volumes# tun.py
Interface Name: qmy0
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
^CTraceback (most recent call last):
  File "./tun.py", line 28, in <module>
    packet = os.read(tun, 2048)
KeyboardInterrupt
```

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

tunclient.py 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'qmy%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)
SERVER_IP = "10.9.0.11"
SERVER_PORT = 9090


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

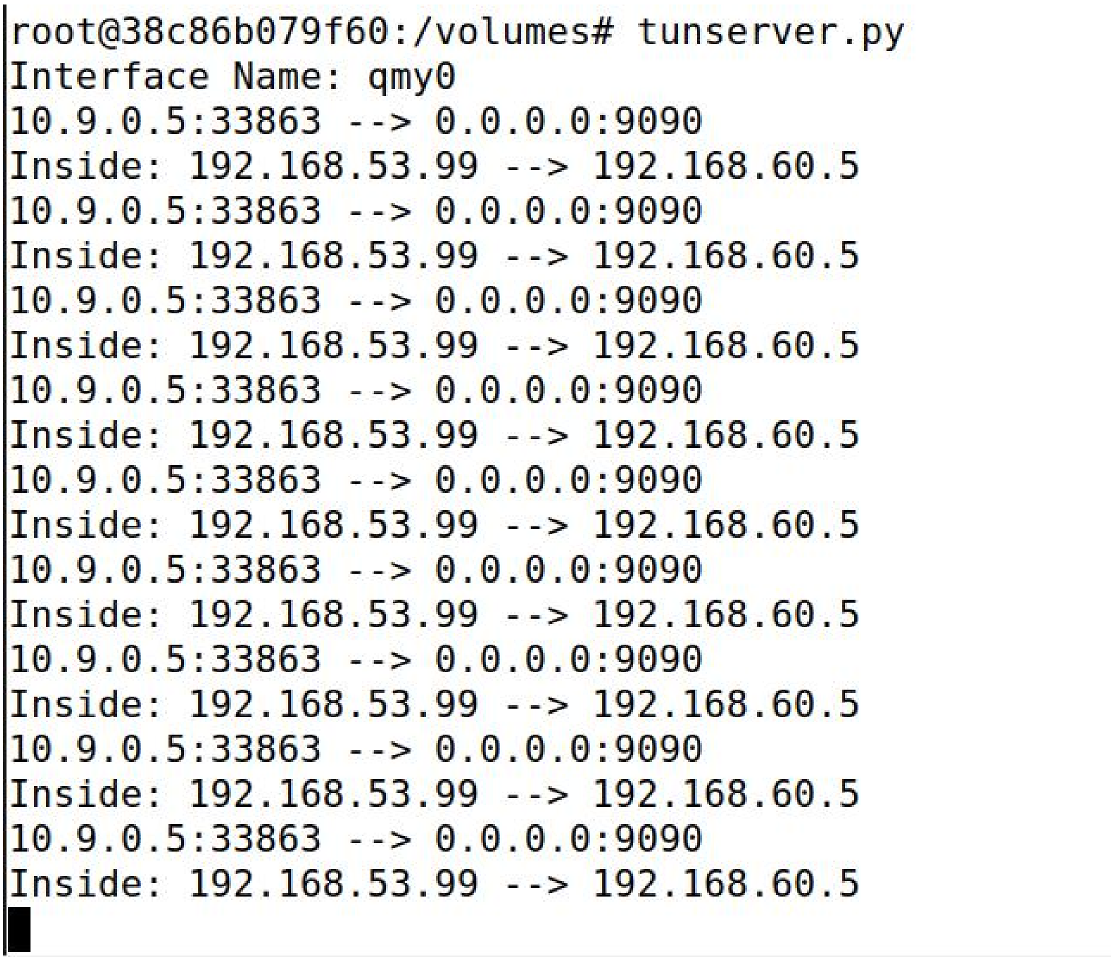
tunserver.py 如下：

```
#!/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'qmy%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))


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

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

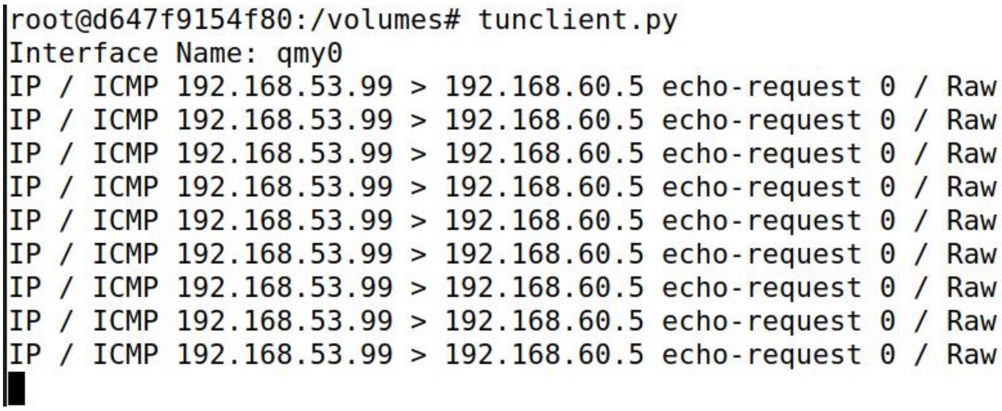
在 10.9.0.11 和 10.9.0.5 分别运行服务器端和客户端的程序。新开一个窗口，10.9.0.5 ping 192.168.60.5

在服务器端，可见管道外部是 10.9.0.5-->0.0.0.0，管道内部是 192.168.53.99-- >192.168.60.5 。



```
root@38c86b079f60:/volumes# tunserver.py
Interface Name: qmy0
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:33863 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```

客户端:



```
root@d647f9154f80:/volumes# tunclient.py
Interface Name: qmy0
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
█
```

Task 4: Set Up the VPN Server

打开路由器上的路由转发

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

使用 task3 中的代码,程序步骤与 task3 步骤一样。
10.9.0.5ping192.168.60.5
新开一个 server 窗口， server 的 eth1 端口收到返回。

```
root@38c86b079f60:/volumes# 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
02:49:47.060074 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 8, length 64
02:49:47.060175 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 8, length 64
02:49:48.083423 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 9, length 64
02:49:48.083526 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 9, length 64
02:49:49.106920 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 10, length 64
02:49:49.107024 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 10, length 64
02:49:50.198643 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 11, length 64
02:49:50.198742 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 11, length 64
02:49:51.219581 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 12, length 64
02:49:51.219682 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 12, length 64
02:49:52.243782 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 13, length 64
02:49:52.243983 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 13, length 64
02:49:53.267629 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 163, seq 14, length 64
02:49:53.267656 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 163, seq 14, length 64
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
```

Task 5: Handling Traffific in Both Directions

代码修改如下：

Open　tunclient.py　~/Desktop/Labs_20.04/Network Security/VPN Tunneling Lab/Labsetup/volumes

tunclient.py　tunserver.py

```python
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'qmy%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)
SERVER_IP = "10.9.0.11"
SERVER_PORT = 9090


while True:
        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))
                        oswrite(tun,bytes(pkt))
                if fd is tun:
                        packet=os.read(tun,2048)
                        print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
                        sock.sendto(packet,('10.9.0.11',9090))
```

Open tunserver.py ~/Desktop/Labs_20.04/Network Security/VPN Tunneling Lab/Labsetup/volumes

tunclient.py tunserver.py

```python
# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'qmy%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))


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

while True:
        while True:
        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))
                oswrite(tun,bytes(pkt))
        if fd is tun:
                packet=os.read(tun,2048)
                print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
                sock.sendto(packet,('10.9.0.5',9090))
```

ping 192.168.60.5 可以 ping 通，并且能看到返回报文。

telnet 192.168.60.5 ，连接成功，并且能看到返回报文。

Task 6: Tunnel-Breaking Experiment

一旦 client 或 server 程序中断，这时候敲击键盘没有任何反应，所有的敲击结果都在缓冲区不停地重发；当程序恢复运行， VPN 又建立起来，敲击结果就会显示在终端。