VPN Tunneling Lab

1. Task 1: Network Setup

虚拟机 A(seed)
虚拟机 B（ubuntu）
内部网络 10.0.0.1/24，网关 10.0.0.1
虚拟机 C（security onion）
10.0.0.2/24，网关为 10.0.0.1
虚拟机 A 和 B 桥接至宿主机的网卡 ip 为自动分配，不过都在 192.168.1.0/24 内。

B ping C

```
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.382 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.284 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.284/0.325/0.382/0.044 ms
```

A ping C

```
[09/21/20]seed@VM:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6142ms
```

2. Task 2: Create and Configure TUN Interface

```
#!/usr/bin/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'nie%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)
```

```
[09/21/20]seed@VM:~/Lab/lab7$ sudo ./tun.py
Interface Name: nie0
```

```
5: nie0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc
ult qlen 500
    link/none
```

```
5: nie0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qd
te UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.88/24 scope global nie0
       valid_lft forever preferred_lft forever
    inet6 fe80::32f:73d7:c461:105c/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

ping192.168.53.87：

```
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 84
  id        = 22787
  flags     = DF
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xf5a5
  src       = 192.168.53.88
  dst       = 192.168.53.87
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0x96eb
     id        = 0x1424
     seq       = 0x5
###[ Raw ]###
        load      = '\x06li \xf0\x1c\x02\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x
```

```
#!/usr/bin/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'nie%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")
os.system("ip addr add 192.168.53.88/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
                ip = IP(packet)
                ip.show()
                newip = IP(src='1.2.3.4', dst='192.168.53.88')
                newpkt = newip/ip.payload
                os.write(tun, bytes(newpkt))
```

修改程序：

```
#!/usr/bin/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'nie%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")
os.system("ip addr add 192.168.53.88/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
                ip = IP(packet)
                ip.show()
                newpkt = "eeeeeeeeeeee".encode('utf-8')
                os.write(tun, bytes(newpkt))
```

运行程序并 ping192.168.53.87

3	2020-09-23 03:45:31.835111963	192.168.53.88	192.168.53.87	ICMP	84 Echo (ping) reque
4	2020-09-23 03:45:31.836029942	N/A	N/A	IPv6	12 Invalid IPv6 head
5	2020-09-23 03:45:32.847756750	192.168.53.88	192.168.53.87	ICMP	84 Echo (ping) reque
6	2020-09-23 03:45:32.848624354	N/A	N/A	IPv6	12 Invalid IPv6 head
7	2020-09-23 03:45:33.902507895	192.168.53.88	192.168.53.87	ICMP	84 Echo (ping) reque
8	2020-09-23 03:45:33.903259452	N/A	N/A	IPv6	12 Invalid IPv6 head
9	2020-09-23 03:45:34.927340858	192.168.53.88	192.168.53.87	ICMP	84 Echo (ping) reque
10	2020-09-23 03:45:34.928015520	N/A	N/A	IPv6	12 Invalid IPv6 head
11	2020-09-23 03:45:35.951189229	192.168.53.88	192.168.53.87	ICMP	84 Echo (ping) reque
12	2020-09-23 03:45:35.951853436	N/A	N/A	IPv6	12 Invalid IPv6 head

可以收到包，还是 12 个 e，没有添加任何头。

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

Client 程序

```
#!/usr/bin/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'nie%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")
os.system("ip addr add 192.168.53.88/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
print("Interface Name: {}".format(ifname))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
                sock.sendto(packet, ('192.168.1.102', 9090))
```

两边执行程序后，在 client 端虚拟机上 ping192.168.53.87，server 端如下：

```
192.168.1.104:38577 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 192.168.53.87
192.168.1.104:38577 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 192.168.53.87
192.168.1.104:38577 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 192.168.53.87
```

在 client 中添加路由表项：

```
sudo ip route add 10.0.0.0/24  via 192.168.53.88
```

再 ping10.0.0.2

server 端如下：

```
192.168.1.104:44816 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 10.0.0.2
192.168.1.104:44816 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 10.0.0.2
192.168.1.104:44816 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 10.0.0.2
192.168.1.104:44816 --> 0.0.0.0:9090
 Inside: 192.168.53.88 --> 10.0.0.2
```

4. Task 4: Set Up the VPN Server

tun_server.py 如下：

```
#!/usr/bin/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'nie%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")
os.system("ip addr add 192.168.53.87/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
IP_A = "10.0.2.4"
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))
        os.write(tun, data)
```

tun.py（seed2 上的客户端程序）如下：

```python
#!/usr/bin/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'nie%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")
os.system("ip addr add 192.168.53.88/24 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))
print("Interface Name: {}".format(ifname))
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if True:
                print(1)
                sock.sendto(packet, ('10.0.2.4', 9090))
```

```
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
10.0.2.7:59655 --> 10.0.2.4:9090
 Inside: 192.168.53.88 --> 192.168.60.2
```

```
  1 2020-09-25 23:03:56.060402316    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
  2 2020-09-25 23:03:56.060471302    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
  3 2020-09-25 23:03:57.084413256    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
  4 2020-09-25 23:03:57.084432374    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
  5 2020-09-25 23:03:58.109730914    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
  6 2020-09-25 23:03:58.109755354    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
  7 2020-09-25 23:03:59.133509046    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
  8 2020-09-25 23:03:59.133534056    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
  9 2020-09-25 23:04:00.156899624    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
 10 2020-09-25 23:04:00.156954650    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
 13 2020-09-25 23:04:01.183123091    192.168.53.88          192.168.60.2           ICMP         98 Echo (ping) request
 14 2020-09-25 23:04:01.183147164    192.168.60.2           192.168.53.88          ICMP         98 Echo (ping) reply
```

5. Task 5: Handling Traffic in Both Directions

tun_client.py

```
#!/usr/bin/python3
import fcntl
import struct
import os
import time
import select
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'nie%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")
os.system("ip addr add 192.168.53.88/24 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))
print("Interface Name: {}".format(ifname))
IP_A = "10.0.2.7"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
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))
                        os.write(tun,data)
                if fd is tun:
                        packet = os.read(tun, 2048)
                        pkt = IP(packet)
                        print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
                        sock.sendto(packet, ("10.0.2.4", 9090))
```

tun_server.py

```python
#!/usr/bin/python3
import fcntl
import struct
import os
import time
from select import *
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'nie%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")
os.system("ip addr add 192.168.53.87/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
IP_A = "10.0.2.4"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
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))
                        os.write(tun,data)
                if fd is tun:
                        packet = os.read(tun, 2048)
                        pkt = IP(packet)
                        print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
                        sock.sendto(packet, ("10.0.2.7", 9090))
```

从 U ping V：

```
PING 192.168.60.2 (192.168.60.2) 56(84) bytes of data.
64 bytes from 192.168.60.2: icmp_seq=1 ttl=63 time=3.51 ms
64 bytes from 192.168.60.2: icmp_seq=2 ttl=63 time=3.09 ms
64 bytes from 192.168.60.2: icmp_seq=3 ttl=63 time=2.47 ms
64 bytes from 192.168.60.2: icmp_seq=4 ttl=63 time=2.60 ms
64 bytes from 192.168.60.2: icmp_seq=5 ttl=63 time=2.59 ms
^C
```

```
From tun ==>: 192.168.53.88 --> 192.168.60.2
From socket <==: 192.168.60.2 --> 192.168.53.88
From tun ==>: 192.168.53.88 --> 192.168.60.2
From socket <==: 192.168.60.2 --> 192.168.53.88
From tun ==>: 192.168.53.88 --> 192.168.60.2
From socket <==: 192.168.60.2 --> 192.168.53.88
From tun ==>: 192.168.53.88 --> 192.168.60.2
From socket <==: 192.168.60.2 --> 192.168.53.88
```

```
From socket <==: 192.168.53.88 --> 192.168.60.2
From tun ==>: 192.168.60.2 --> 192.168.53.88
From socket <==: 0.0.0.0 --> 77.235.183.63
From socket <==: 192.168.53.88 --> 192.168.60.2
From tun ==>: 192.168.60.2 --> 192.168.53.88
From socket <==: 192.168.53.88 --> 192.168.60.2
From tun ==>: 192.168.60.2 --> 192.168.53.88
From socket <==: 192.168.53.88 --> 192.168.60.2
From tun ==>: 192.168.60.2 --> 192.168.53.88
From socket <==: 192.168.53.88 --> 192.168.60.2
From tun ==>: 192.168.60.2 --> 192.168.53.88
```

```
[09/26/20]seed@VM:~$ telnet 192.168.60.2
Trying 192.168.60.2...
Connected to 192.168.60.2.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sat Sep 26 04:47:59 EDT 2020 from 192.168.53.88 on pts/2
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

6. Task 6: Tunnel-Breaking Experiment

如果停止 tun_client.py，通过已连接的 telnet 输入一串 “zzzzzz” ，什么都不会发生，然后再启动 tun_client.py，这一串 z 会同时跳出来。
telnet 所在进程的 socket 一直在利用 tcp 特性向 tun 重发，未超时前 client 程序启动，那么这些重发就被 U 的 tun 卡接收到，然后继续进行。

7. Task 7: Routing Experiment on Host V

V 的路由表修改

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
192.168.53.0    192.168.60.1    255.255.255.0   UG    0      0        0 enp0s3
192.168.60.0    *               255.255.255.0   U     100    0        0 enp0s3
```

U 仍然可以 ping 通 V：

```
[09/26/20]seed@VM:~$ ping 192.168.60.2
PING 192.168.60.2 (192.168.60.2) 56(84) bytes of data.
64 bytes from 192.168.60.2: icmp_seq=1 ttl=63 time=2.81 ms
64 bytes from 192.168.60.2: icmp_seq=2 ttl=63 time=2.44 ms
64 bytes from 192.168.60.2: icmp_seq=3 ttl=63 time=2.28 ms
```

8. Task 8: Experiment with the TUN IP Address

9. Task 9: Experiment with the TAP Interface

ping 192.168.53.80

可以看到接收到的包是 arp 包

```
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 6a:4d:19:6b:26:8d
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 6a:4d:19:6b:26:8d
     psrc      = 192.168.53.88
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.53.80
```

程序中改为 IP(packet)，解码出来的包也都是错的，ipversion 都是 15

说明 tun 卡接收到包时以太帧头部已经没有了。