VPN Lab: The Container Version

57118211 谢瑞

Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface

编写 tun.py 文件，保存至 volumes 目录。配置 TUN 接口的 IP（192.168.53.99/24）

```
#!/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))

while True:
    packet = os.read(tun, 2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())
    time.sleep(10)
```

在 10.9.0.5 上运行 tun.py：

```
root@887e57ef8bed:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.53.0/24 dev tun0 proto kernel scope link src 192.168.53.99
```

当运行 tun.py 后，通过“ip route”命令可看到成功注册了 tun0 网口，可以通过该网口连接到 192.168.53.0/24 网段。

Task 2.b: Set up the TUN Interface

给 tun.py 加上两行：

```
1.  os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
2.  os.system("ip link set dev {} up".format(ifname))
```

再次运行后可以看到接口有具体网段了：

```
tun0: 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
(UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        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
```

Task 2.c: Read from the TUN Interface

```
root@887e57ef8bed:/volumes# tun.py
Interface Name: tun0
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.2 echo-request 0 / Raw
```

当主机用户 U 使用“ping 192.168.53.2”命令时，可以得到 tun.py 相应的输出。因为 tun0 位于当前子网，所以 ping 命令相当于经过了 tun0 网口。

```
root@887e57ef8bed:/# 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=0.085 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.045 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.041 ms
```

当主机用户 U 使用“ping 192.168.60.1”命令时， tun.py 没有相应的输出。这是因为 ping 命令找不到子网的网口就会从默认网口发出，而不是我们指定的 tun0 网口。

Task 2.d: Write to the TUN Interface

修改程序发送回复包：

```python
while True:
    packet = os.read(tun, 2048)
    if packet:
        ip = IP(packet)
        print(ip.summary())
        newip = IP(src=ip.dst, dst=ip.src)
        newpkt = newip/ip.payload
        os.write(tun, bytes(newpkt))
```

可以看到回复包：

```
root@887e57ef8bed:/volumes# tun.py
Interface Name: tun0
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-reply 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-reply 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-reply 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-reply 0 / Raw
```

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

在 Host U 上编写 tun_client.py 文件，保存至 volumes 目录：

```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
```

```
SERVER_IP = "10.9.0.11"
SERVER_PORT = 9090

# 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")
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 tun0 via 192.168.53.99".format(ifname))
print("Interface Name: {}".format(ifname))

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
  # Get a packet from the tun interface
  packet = os.read(tun, 2048)
  if packet:
  # Send the packet via the tunnel
    sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

在 Router 编写 udp_server.py，接收 UDP 报文：

```
#!/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
IP_A = "0.0.0.0"
PORT = 9090

# Create the tun interface
```

```
17.
18.tun = os.open("/dev/net/tun", os.O_RDWR)
19.ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
20.ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
21.
22.# Get the interface name
23.ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
24.os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))
25.os.system("ip link set dev {} up".format(ifname))
26.
27.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
28.sock.bind((IP_A, PORT))
29.
30.while True:
31.  data, (ip, port) = sock.recvfrom(2048)
32.  print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
33.  pkt = IP(data)
34.  print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
```

在 tun_client.py 中添加以下代码后，ping 192.168.60.2，udp_server.py 有输出：

```
1. os.system("ip route add 192.168.60.0/24 dev tun0 via
   192.168.53.99".format(ifname))
```

```
root@af7e8332c479:/volumes# udp_server.py
10.9.0.5:57143 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.2
10.9.0.5:57143 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.2
10.9.0.5:57143 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.2
```

主机 U 对 192.168.53.1 进行 ping 的时候会使用 tun 网口，tun_client 把 tun 网口收到的报文封装成为了 UDP 报文，因此 VPN 服务器可以直接收到。但是第一次 ping 192.168.60.2 时，没有经过 tun0 网口而是经过本地默认网口 eth0，此时 VPN 不会收到报文。当将 192.168.60.0/24 网段配置到 tun0 网口上后再 ping，VPN 就会收到报文了。

Task 4: Set Up the VPN Server

在 Router 上编写 tun_server.py 并执行：

```
1. #!/usr/bin/env python3
```

```
2.
3. import fcntl
4. import struct
5. import os
6. import time
7. from scapy.all import *
8.
9. TUNSETIFF = 0x400454ca
10.IFF_TUN = 0x0001
11.IFF_TAP = 0x0002
12.IFF_NO_PI = 0x1000
13.IP_A = "0.0.0.0"
14.PORT = 9090
15.
16.# Create the tun interface
17.tun = os.open("/dev/net/tun", os.O_RDWR)
18.ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
19.ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
20.
21.# Get the interface name
22.ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
23.os.system("ip addr add 192.168.53.1/24 dev {}".format(ifname))
24.os.system("ip link set dev {} up".format(ifname))
25.
26.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
27.sock.bind((IP_A, PORT))
28.
29.while True:
30.  data, (ip, port) = sock.recvfrom(2048)
31.  print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
32.  pkt = IP(data)
33.  print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
34.  print("Sending raw:{}".format(data))
35.  os.write(tun,data)
```

在 Host U 执行 tun_client.py，并发送 ping 数据包：

```
root@af7e8332c479:/volumes# chmod a+x tun_server.py
root@af7e8332c479:/volumes# tun_server.py
10.9.0.5:38473 --> 0.0.0.0:9090
  Inside: 192.168.53.99 --> 192.168.60.5
Sending raw:b'E\x00\x00T\x8bP@\x00@\x01\xbc\x9f\xc0\xa85c\xc0\xa8<\x05\x08\x00\xfa)\x00b\x
00\x01\xdb\xd5\xeb`\x00\x00\x00\x00ti\x03\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\
x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
10.9.0.5:38473 --> 0.0.0.0:9090
```

在 Host V 上用 tcpdump 抓包：

```
root@8eebf64ba1a3:/# 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
05:47:21.846742 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 109, seq 14, length 64
05:47:21.846768 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 109, seq 14, length 64
05:47:22.871079 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 109, seq 15, length 64
05:47:22.871129 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 109, seq 15, length 64
05:47:23.894571 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 109, seq 16, length 64
05:47:23.894597 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 109, seq 16, length 64
```

Task 5: Handling Traffic in Both Directions

编写 tun_server_select.py 并在 Router 运行

```python
#!/usr/bin/env 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
IP_A = "0.0.0.0"
PORT = 9090

# 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")
os.system("ip addr add 192.168.53.1/24 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:
# this will block until at least one interface is ready
  ready, _, _ = select.select([sock, tun], [], [])

  for fd in ready:
```

```
35.
36.  if fd is sock:
37.    data, (ip, port) = sock.recvfrom(2048)
38.    pkt = IP(data)
39.    print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
40.    os.write(tun,data)
41.  if fd is tun:
42.    packet = os.read(tun, 2048)
43.    pkt = IP(packet)
44.    print("From tun ==>: {} -
   -> {}".format(pkt.src, pkt.dst))sock.sendto(packet,("10.9.0.5",po
   rt))
```

编写 tun_client_select.py 并在 Host U 运行：

```
1. #!/usr/bin/env python3
2.
3. import fcntl
4. import struct
5. import os
6. import time
7. import select
8. from scapy.all import *
9.
10.TUNSETIFF = 0x400454ca
11.IFF_TUN = 0x0001
12.IFF_TAP = 0x0002
13.IFF_NO_PI = 0x1000
14.SERVER_IP = "10.9.0.11"
15.SERVER_PORT = 9090
16.
17.# Create the tun interface
18.tun = os.open("/dev/net/tun", os.O_RDWR)
19.ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
20.ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
21.
22.# Get the interface name
23.ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
24.os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
25.os.system("ip link set dev {} up".format(ifname))
26.os.system("ip route add 192.168.60.0/24 dev tun0 via 92.168.53.99
   ".format(ifname))
27.print("Interface Name: {}".format(ifname))
28.
```

```
29.# Create UDP socket
30.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
31.
32.while True:
33.# this will block until at least one interface is ready
34.  ready, _, _ = select.select([sock, tun], [], [])
35.
36.for fd in ready:
37.  if fd is sock:
38.    data, (ip, port) = sock.recvfrom(2048)
39.    pkt = IP(data)
40.    print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
41.    os.write(tun,data)
42.  if fd is tun:
43.    packet = os.read(tun, 2048)
44.    pkt = IP(packet)
45.    print("From tun ==>: {} --> {}".format(pkt.src, pkt.dst))
46.    sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

ping 通 192.168.60.5 ：

```
root@a06069e1ab16:/# 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=2.29 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=1.64 ms
root@af7e8332c479:/volumes# chmod a+x tun_server_select.py
root@af7e8332c479:/volumes# tun_server_select.py
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

root@a06069e1ab16:/volumes# chmod a+x tun_client_select.py
root@a06069e1ab16:/volumes# 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
```

Task 6: Tunnel-Breaking Experiment

如上程序在 10.9.0.5 上 telnet 192.168.60.5 ：

```
root@a06069e1ab16:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
8eebf64ba1a3 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)
```

```
root@a06069e1ab16:/volumes# 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
```

```
root@af7e8332c479:/volumes# tun_server_select.py
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
```

一旦 client 或 server 程序断开， tunnel 重新建立， telnet 也会重新建立，此时敲击键盘不会有反应。