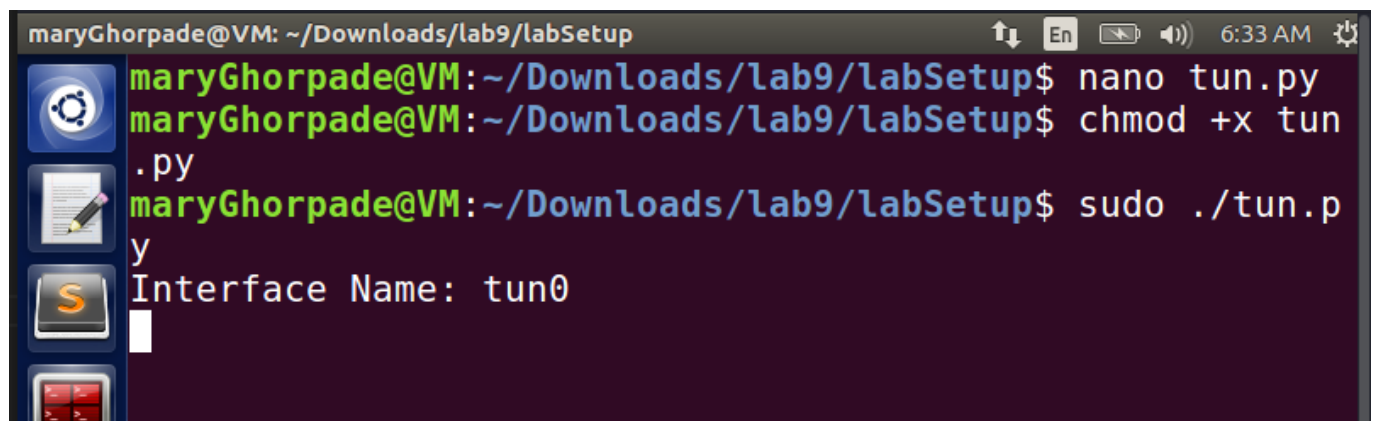


VPN Tunneling Lab

Objectives

- Virtual Private Network
- The TUN/TAP virtual interface
- IP tunneling
- Routing

Creating and configuring Tun interface

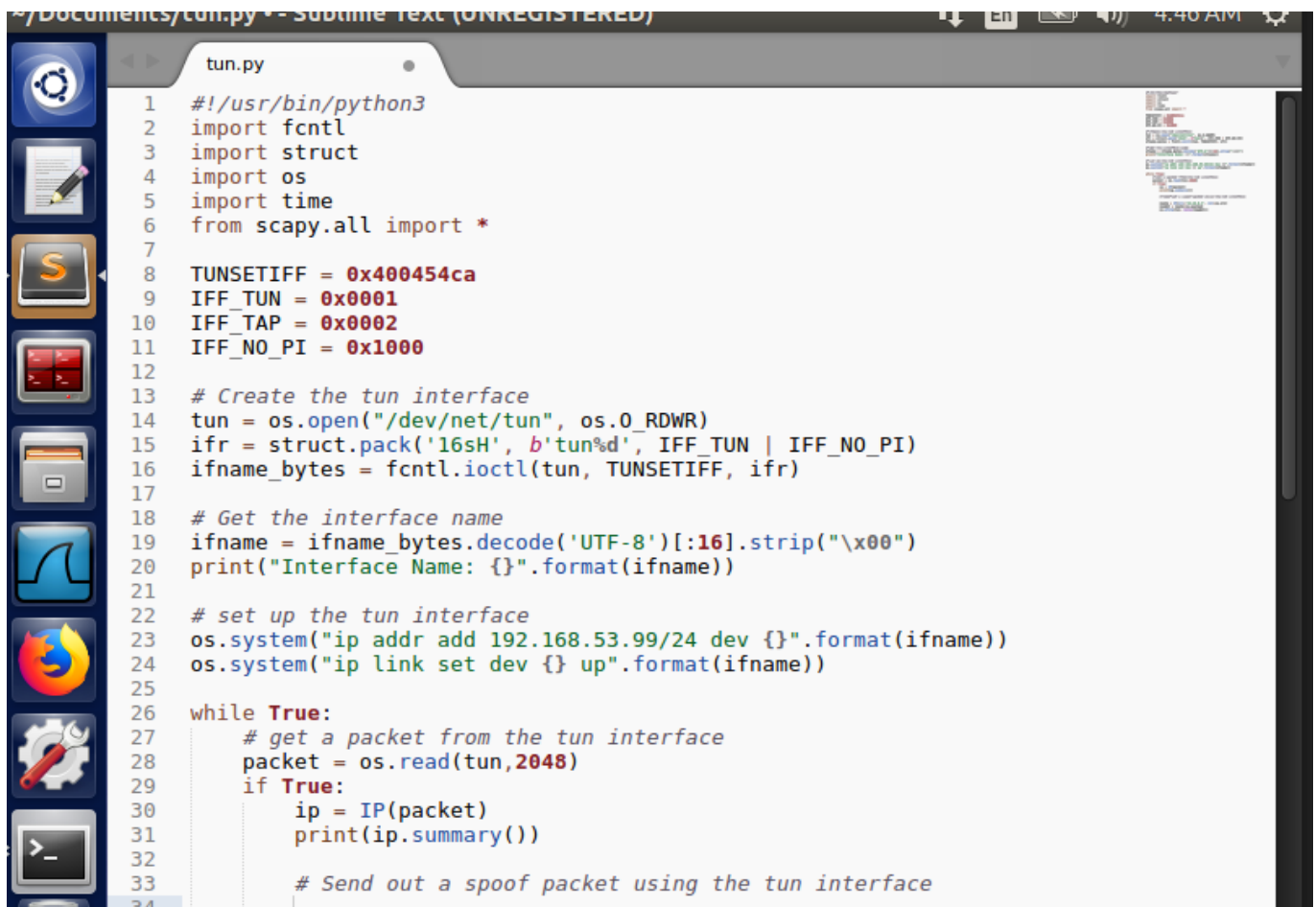
A terminal window titled 'maryGhorpade@VM: ~/Downloads/lab9/labSetup' with a status bar showing '6:33 AM'. The terminal shows the following commands and output:

```
maryGhorpade@VM:~/Downloads/lab9/labSetup$ nano tun.py
maryGhorpade@VM:~/Downloads/lab9/labSetup$ chmod +x tun.py
maryGhorpade@VM:~/Downloads/lab9/labSetup$ sudo ./tun.py
Interface Name: tun0
```

A tun interface has been created after running the program

```
maryGhorpade@VM: /home/seed
disc pfifo_fast state UP group default qlen 1000
  link/ether 08:00:27:e0:5e:f8 brd ff:ff:ff:ff:ff:ff
  inet 192.168.214.101/24 brd 192.168.214.255 scope g
lobal dynamic enp0s3
  valid_lft 3421sec preferred_lft 3421sec
  inet6 2c0f:fe38:2100:a78e:18b4:9b79:e6c1:ef91/64 sc
ope global temporary dynamic
  valid_lft 7023sec preferred_lft 7023sec
  inet6 2c0f:fe38:2100:a78e:1390:fc9d:9ab:93fa/64 sco
pe global mngtmpaddr noprefixroute dynamic
  valid_lft 7023sec preferred_lft 7023sec
  inet6 fe80::6eff:ee59:4864:5768/64 scope link
  valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 150
0 qdisc noqueue state DOWN group default
  link/ether 02:42:72:27:dd:c4 brd ff:ff:ff:ff:ff:ff
  inet 172.17.0.1/16 scope global docker0
    valid_lft forever preferred_lft forever
8: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc n
oop state DOWN group default qlen 500
  link/none
maryGhorpade@VM: /home/seed$
```

client_tun.py



```

1  #!/usr/bin/python3
2  import fcntl
3  import struct
4  import os
5  import time
6  from scapy.all import *
7
8  TUNSETIFF = 0x400454ca
9  IFF_TUN = 0x0001
10 IFF_TAP = 0x0002
11 IFF_NO_PI = 0x1000
12
13 # Create the tun interface
14 tun = os.open("/dev/net/tun", os.O_RDWR)
15 ifr = struct.pack('16sH', b'tun%d', IFF_TUN | IFF_NO_PI)
16 ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
17
18 # Get the interface name
19 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
20 print("Interface Name: {}".format(ifname))
21
22 # set up the tun interface
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
26 while True:
27     # get a packet from the tun interface
28     packet = os.read(tun, 2048)
29     if True:
30         ip = IP(packet)
31         print(ip.summary())
32
33     # Send out a spoof packet using the tun interface
34

```

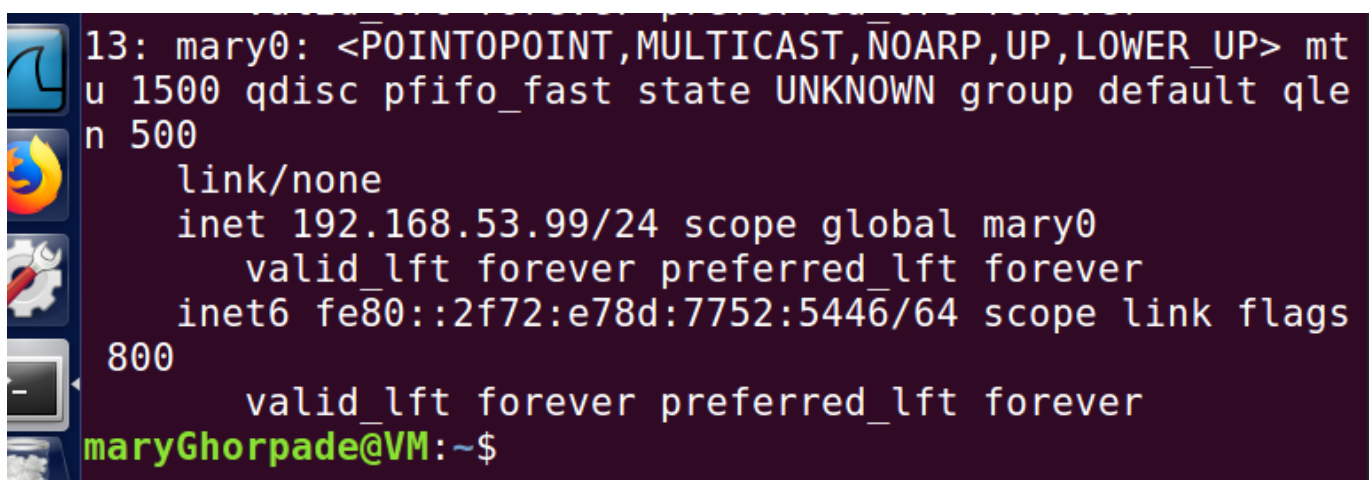
Setting up TUN interface

Assign it an IP address to it and bring the interface up

```
sudo ip addr add 192.168.53.99/24 dev tun0
```

```
sudo ip link set dev tun0 up
```

The interface is assigned an ip as seen below



```

13: mary0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mt
u 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global mary0
        valid_lft forever preferred_lft forever
    inet6 fe80::2f72:e78d:7752:5446/64 scope link flags
800
        valid_lft forever preferred_lft forever
maryGhorpade@VM:~$

```

Reading TUN interface

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

It indicates that the VPN Server received the ICMP packets from the tun interface (192.168.53.99) which are intended for the chosen host in 192.168.53.0/24 (192.168.53.20).

Write to the TUN Interface

```
# Send out a spoof packet using the tun interface
newip = IP(src='1.2.3.4', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

Set up VPN Server

```
import fcntl
import struct
import os
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN = 0x0001
IFF_TAP = 0x0002
IFF_NO_PI = 0x1000
TUN_IP = "192.168.53.98"

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lin%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)
```

IP forwarding can be enabled using the following command

```
sudo sysctl net.ipv4.ip_forward=1
```

It should be noted that although Host V will respond to the ICMP packets, the reply will not get back to Host U, because we have not set up everything yet. Therefore, for this task, it is sufficient to show the ICMP packets have arrived at Host V.

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
02:17:11.772881 IP 192.168.0.1.5353 > 224.0.0.251:5353: 0 [2q] PTR (QM)? _lpps_tcp.local. PTR (QM)? _lpp_tcp.local. (45)
02:17:11.773027 IP6 fe80::42:32ff:feae:f7a7:5353 > ff02::fb:5353: 0 [2q] PTR (QM)? _lpps_tcp.local. PTR (QM)? _lpp_tcp.local. (45)
02:17:11.773615 IP6 fe80::fcd1:4cff:feb2:2529:5353 > ff02::fb:5353: 0 [2q] PTR (QM)? _lpps_tcp.local. PTR (QM)? _lpp_tcp.local. (45)
```

```
9.0.!      listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
Inside:    03:46:21.707911 IP 192.168.53.99 > ced9612d0882: ICMP echo request, id 52, seq 196
9.0.!      , length 64
Inside:    03:46:21.708011 IP ced9612d0882 > 192.168.53.99: ICMP echo reply, id 52, seq 196,
9.0.5:36661 --> |length 64
Inside: 192.168.03:46:21.715608 IP ced9612d0882.35106 > 192.168.1.1.domain: 30836+ PTR? 99.53.168.
9.0.5:36661 --> |192.in-addr.arpa. (44)
Inside: 192.168.03:46:22.724393 IP 192.168.53.99 > ced9612d0882: ICMP echo request, id 52, seq 197
9.0.5:36661 --> |, length 64
Inside: 192.168.03:46:22.724428 IP ced9612d0882 > 192.168.53.99: ICMP echo reply, id 52, seq 197,
9.0.5:36661 --> |length 64
```