

Setup

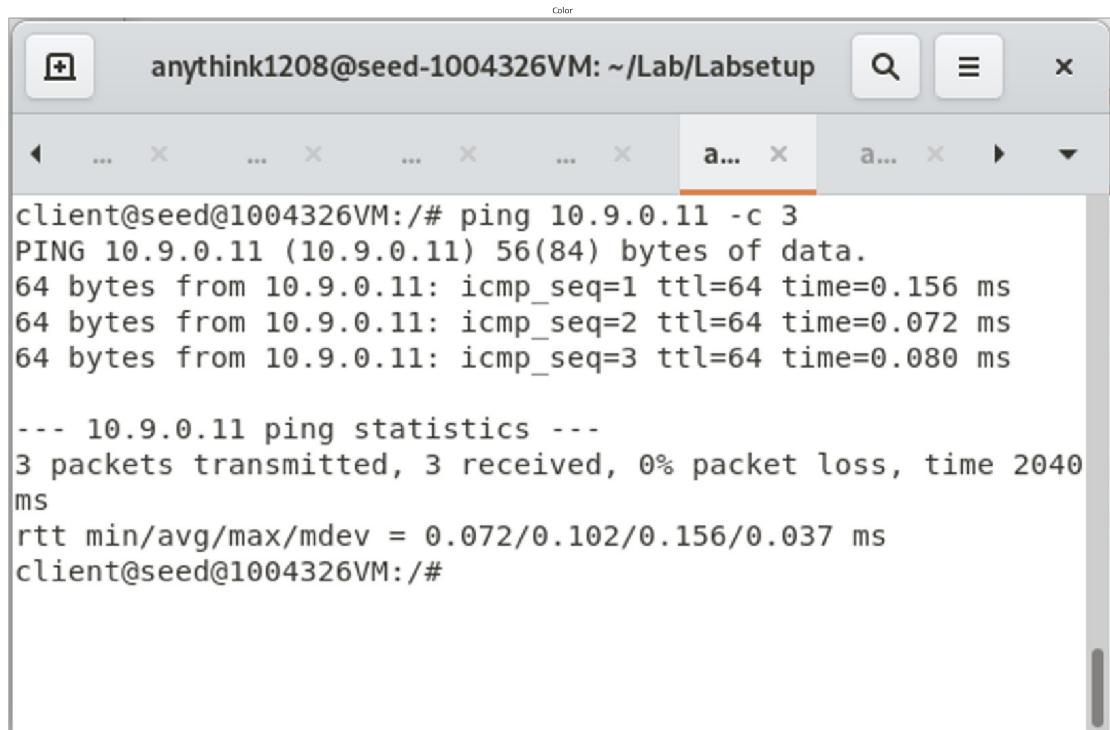
```
import cv2
from matplotlib import pyplot as plt

# This is a bit of magic to make matplotlib figures appear inline in
# the notebook
# rather than in a new window.
%matplotlib inline
plt.rcParams['figure.figsize'] = (100.0, 80.0) # set default size of
plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

def show_img(img):
    img = cv2.imread(img,-1)
    plt.subplot(131),plt.imshow(img),
    plt.title('Color'),plt.xticks([]), plt.yticks([])
    plt.show()
```

U (10.9.0.5) can communicate with VPN Server (10.9.0.11)

```
show_img('setup/client_server_comm.png')
```



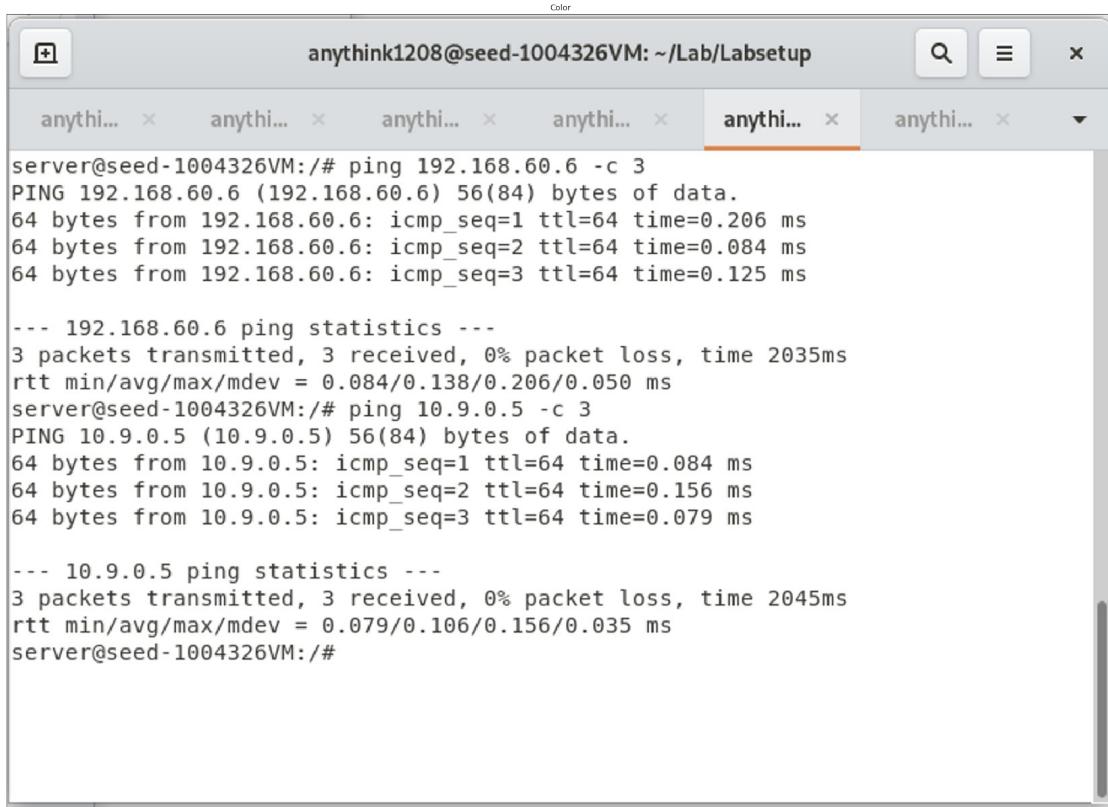
The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window displays the output of a ping command from the client machine to the VPN server. The output is as follows:

```
client@seed@1004326VM:/# ping 10.9.0.11 -c 3
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=0.156 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.072 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.080 ms

--- 10.9.0.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040 ms
rtt min/avg/max/mdev = 0.072/0.102/0.156/0.037 ms
client@seed@1004326VM:/#
```

VPN Server (192.168.60.11) is able to communicate with V (192.168.60.6)

```
show_img('setup/server_host_comm.png')
```



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window contains several tabs, with the fourth tab from the left active and highlighted in orange. The terminal output displays two ping sessions. The first session is from the server at 192.168.60.6 to the client at 10.9.0.5, showing three successful packets with round-trip times around 0.084 ms. The second session is from the client at 10.9.0.5 to the server at 192.168.60.6, also showing three successful packets with round-trip times around 0.125 ms.

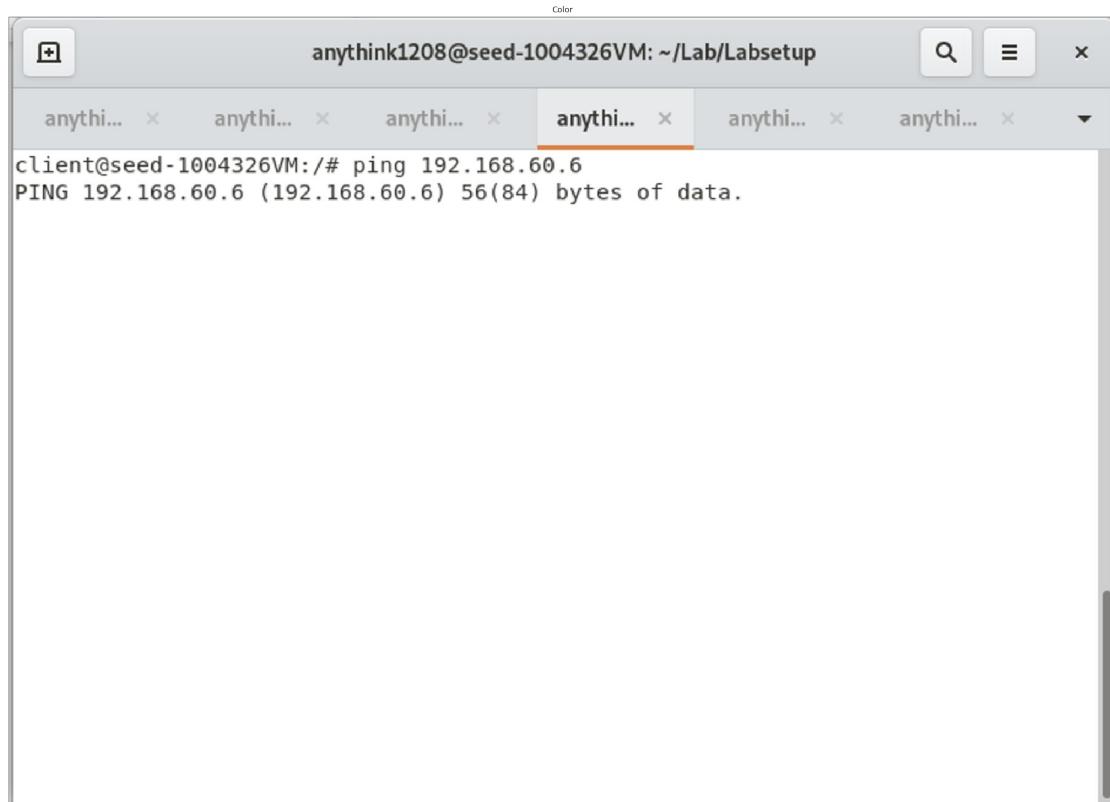
```
server@seed-1004326VM:/# ping 192.168.60.6 -c 3
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
64 bytes from 192.168.60.6: icmp_seq=1 ttl=64 time=0.206 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 192.168.60.6: icmp_seq=3 ttl=64 time=0.125 ms

--- 192.168.60.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.084/0.138/0.206/0.050 ms
server@seed-1004326VM:/# ping 10.9.0.5 -c 3
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=0.084 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.156 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.079 ms

--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.079/0.106/0.156/0.035 ms
server@seed-1004326VM:/#
```

U (10.9.0.5) is unable to ping V (192.168.60.6)

```
show_img('setup/client_host_comm.png')
```



A screenshot of a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window has multiple tabs, with the fourth tab from the left active and highlighted with an orange bar. The active tab displays the command "ping 192.168.60.6" and its output: "PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data."

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
client@seed-1004326VM:/# ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
```

Run tcpdump on the router, and sniff the packet on each network. Capture the packet

Testing eth0

```
show_img('setup/eth0_test_ping.png')
show_img('setup/tcpdump_eth0.png')
```

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x

client@seed-1004326VM:/# ping 10.9.0.11 -c 2
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=0.223 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.107 ms

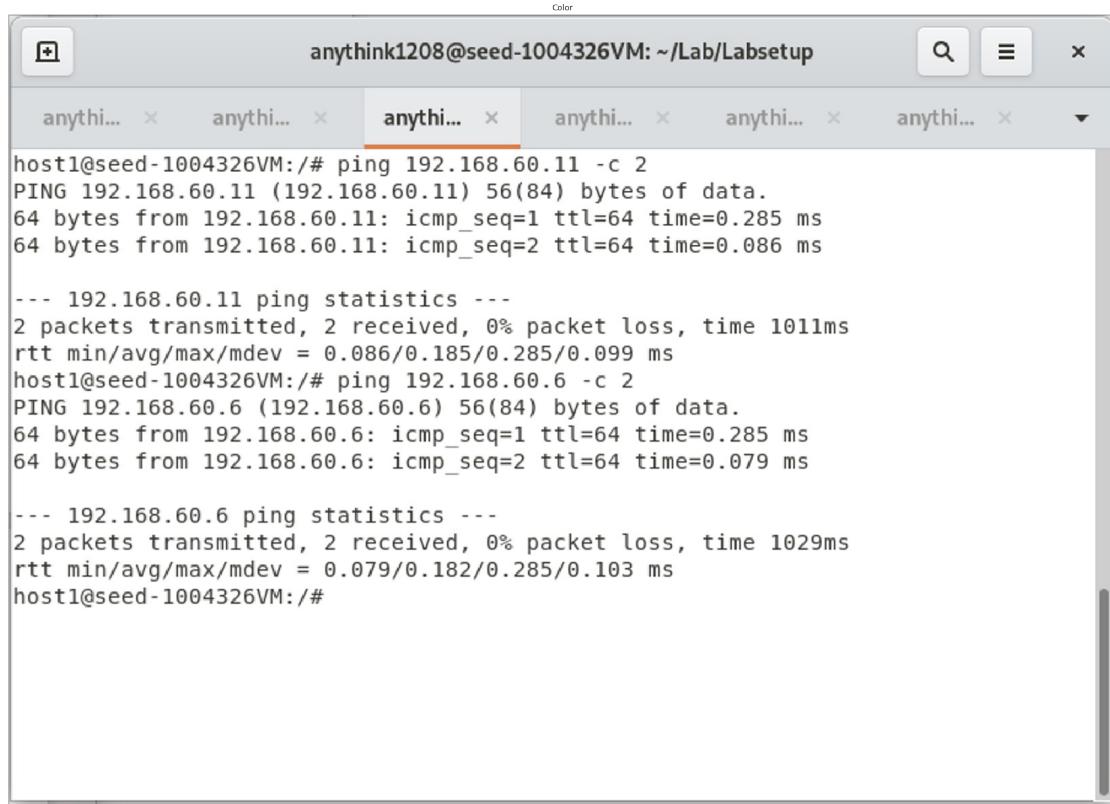
--- 10.9.0.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.107/0.165/0.223/0.058 ms
client@seed-1004326VM:/#
```

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x

server@seed-1004326VM:/# 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
02:00:35.126665 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 9, seq 1, length 64
02:00:35.126707 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 9, seq 1, length 64
02:00:36.146154 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 9, seq 2, length 64
02:00:36.146196 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 9, seq 2, length 64
02:00:40.242097 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
02:00:40.242252 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
02:00:40.242273 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
02:00:40.242276 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

Testing eth1

```
show_img('setup/eth1_test_ping.png')
show_img('setup/tcpdump_eth1.png')
```



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window contains several tabs, with the one labeled "anythi..." currently active. The terminal output displays two ping sessions. The first session, at the top, shows ping results from host1 to 192.168.60.11. The second session, below it, shows ping results from host1 to 192.168.60.6. Both sessions show 100% packet loss and low round-trip times.

```
host1@seed-1004326VM:~# ping 192.168.60.11 -c 2
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data.
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.285 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.086 ms

--- 192.168.60.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.086/0.185/0.285/0.099 ms
host1@seed-1004326VM:~# ping 192.168.60.6 -c 2
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
64 bytes from 192.168.60.6: icmp_seq=1 ttl=64 time=0.285 ms
64 bytes from 192.168.60.6: icmp_seq=2 ttl=64 time=0.079 ms

--- 192.168.60.6 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1029ms
rtt min/avg/max/mdev = 0.079/0.182/0.285/0.103 ms
host1@seed-1004326VM:~#
```

```
anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x
server@seed-1004326VM:/# 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
02:04:43.391312 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
02:04:43.391345 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
02:04:43.391422 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 11, seq 1
, length 64
02:04:43.391441 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 11, seq 1,
length 64
02:04:44.402170 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 11, seq 2
, length 64
02:04:44.402194 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 11, seq 2,
length 64
02:04:48.562076 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
02:04:48.562195 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
02:06:24.877152 ARP, Request who-has 192.168.60.6 tell 192.168.60.5, length 28
```

Task 2.a: Name of the Interface

```
show_img('Task_2a/check_ip_addr.png')
```

```

client@seed-1004326VM:/volumes# ./tun.py &
[1] 53
client@seed-1004326VM:/volumes# Interface Name: tun0

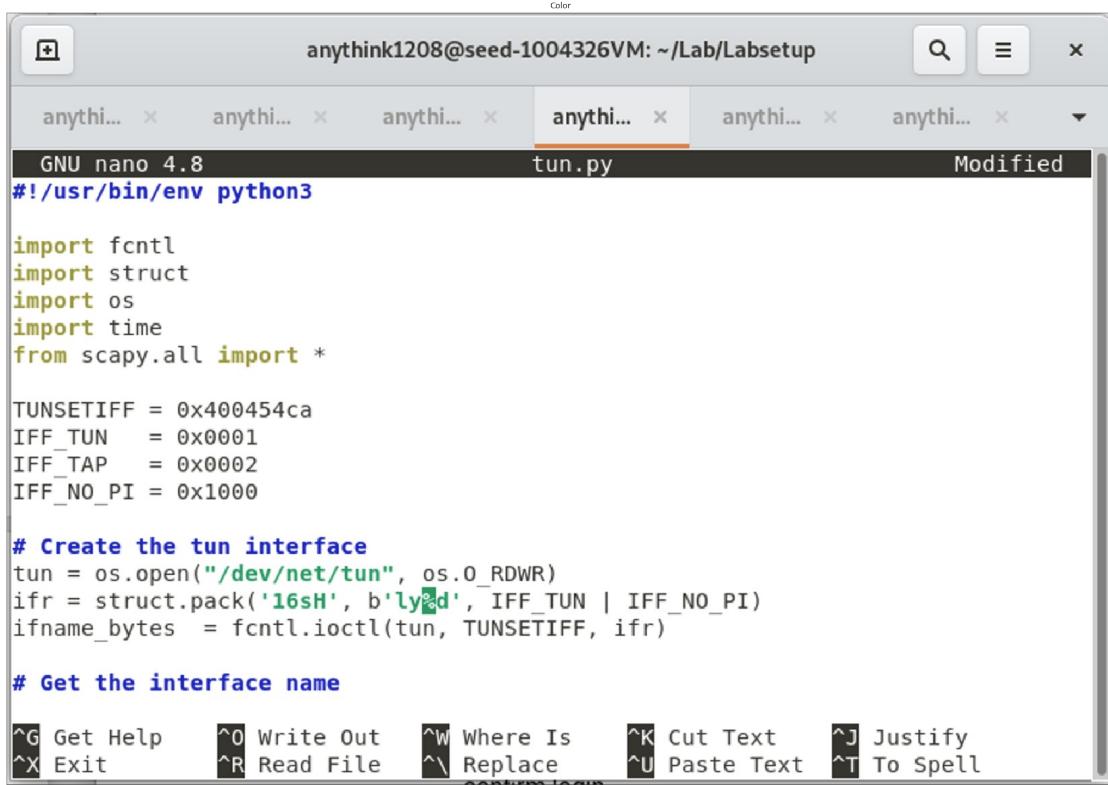
client@seed-1004326VM:/volumes# jobs
[1]+  Running                  ./tun.py &
client@seed-1004326VM:/volumes# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.9.0.5 netmask 255.255.255.0 broadcast 10.9.0.255
        ether 02:42:0a:09:00:05 txqueuelen 0 (Ethernet)
          RX packets 149 bytes 16726 (16.7 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 52 bytes 4480 (4.4 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
          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

client@seed-1004326VM:/volumes# ip addr
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: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
8: eth0@if9: <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
client@seed-1004326VM:/volumes# █

```

`show_img('Task 2a/rename_tun.png')`



```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
GNU nano 4.8
#!/usr/bin/env python3

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

TUNSETIFF = 0x4000454ca
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'lyd', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

Task 2.b: Set up the TUN Interface

Instead of running `sudo ip addr add 192.168.53.99/24 dev tun0 && sudo ip link set dev tun0 up`, I added it in the `tun.py` script, configuration is automatically performed by the program

```
show_img('Task 2b/set_ly0_tun.png')
```

The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The current tab is "tun.py". The code in the editor is as follows:

```

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'ly0', 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))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

```

The terminal window also displays a menu bar with "Color" and a status bar at the bottom showing keyboard shortcuts for various functions like Get Help, Write Out, Where Is, Cut Text, Justify, Exit, Read File, Replace, Paste Text, and To Spell.

`show_img('Task 2b/new_ip_addr_out.png')`

The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The current tab is "anythink...". The terminal output is as follows:

```

client@seed-1004326VM:/volumes# ./tun.py &
[1] 89
client@seed-1004326VM:/volumes# Interface Name: ly0

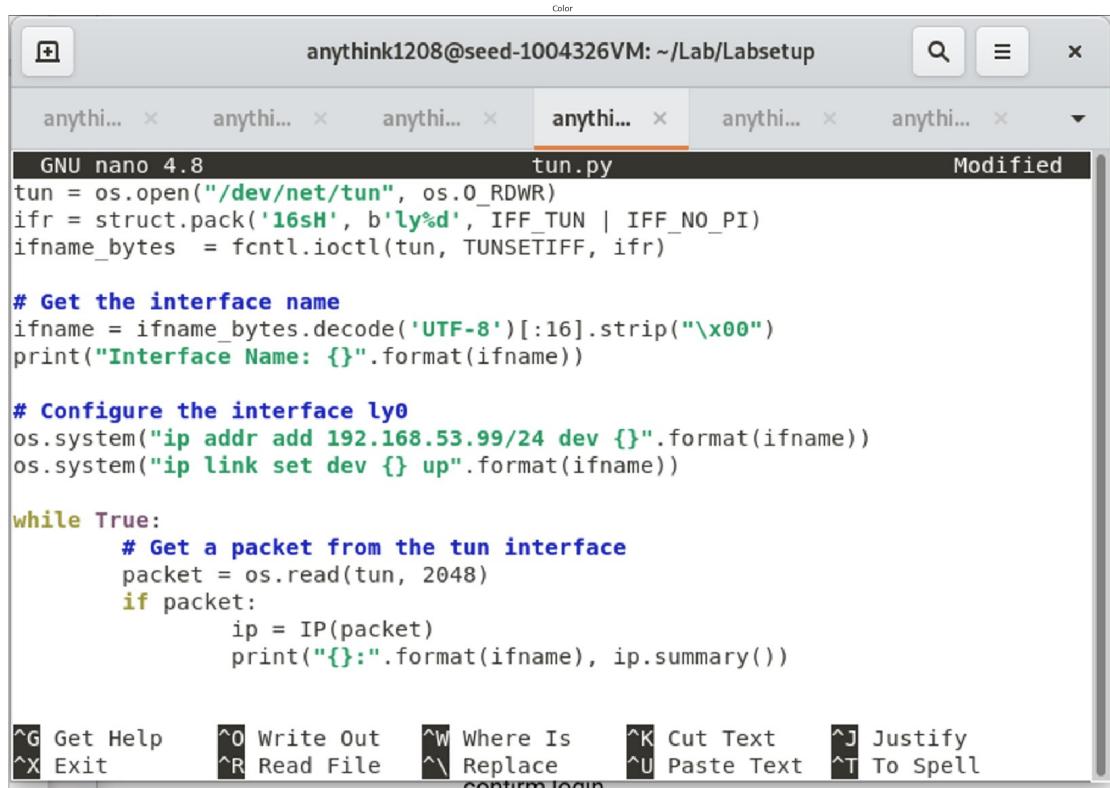
client@seed-1004326VM:/volumes# jobs
[1]+  Running                  ./tun.py &
client@seed-1004326VM:/volumes# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
8: eth0@if9: <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
10: ly0: <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 ly0
        valid_lft forever preferred_lft forever
client@seed-1004326VM:/volumes#

```

At line 10, ly0 has a newly added IP address 192.168.53.99 and an UNKNOWN state as compared to line 4 tun0 that has no IP address and a DOWN state

Task 2.c: Read from the TUN Interface

```
show_img('Task 2c/print_ip_scapy.png')
```



```
anythi... x anythi... x anythi... x tun.py anythi... x anythi... x anythi... x Modified
GNU nano 4.8
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'ly%d', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

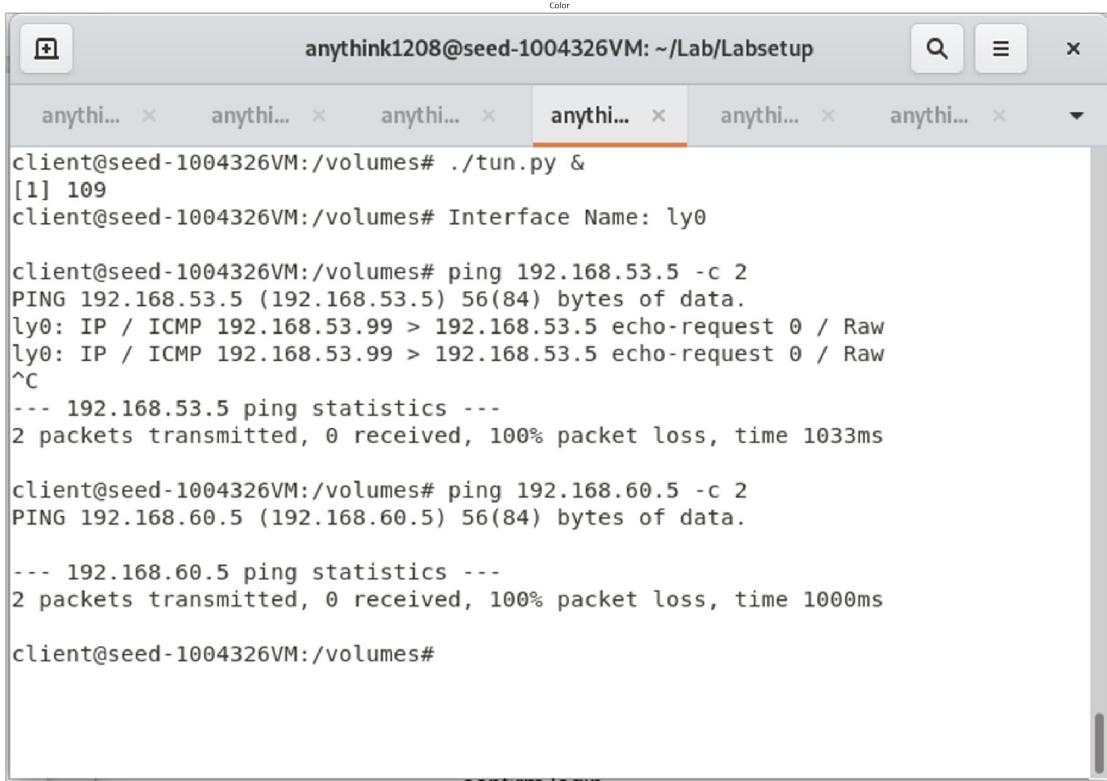
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

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

^G Get Help      ^O Write Out   ^W Where Is     ^K Cut Text    ^J Justify
^X Exit         ^R Read File   ^\ Replace      ^U Paste Text  ^T To Spell
                                          confirm login
```

```
show_img('Task 2c/ping_net.png')
```



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window contains several tabs, with the fourth tab from the left active and highlighted in orange. The terminal output is as follows:

```
anythi... x anythi... x anythi... x anythi... x anythi... x anythi... x
client@seed-1004326VM:/volumes# ./tun.py &
[1] 109
client@seed-1004326VM:/volumes# Interface Name: ly0

client@seed-1004326VM:/volumes# ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
ly0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
ly0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
^C
--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1033ms

client@seed-1004326VM:/volumes# ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms

client@seed-1004326VM:/volumes#
```

When pinging 192.168.53.5, the packet is directed to the ly0 interface (192.168.53.99) which then captures the packet for inspection. However, 192.168.53.5 does not exist, so there is 100% packet loss.

When pinging 192.168.60.5, there is no interface located in the 192.168.60.0/24 network, so the packet is not captured for inspection. 192.168.60.5 does not exist as well, so a 100% packet loss is expected.

Task 2.d: Write to the TUN Interface

```
show_img('Task_2d/spoof_icmp.png')
```

```

Activities Terminal Mar 28 08:11
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
GNU nano 4.8 tun.py
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', IFF_TUN | IFF_NO_PI)
ifname_bytes = Tctl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface l0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print("{}:{}".format(ifname, pkt.summary()))

    # Send out a spoof packet using the tun interface
    if ICMP in pkt and pkt[ICMP].type != 8:
        return
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
    icmp = ICMP(type=8, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
    data = pkt[Raw].load
    newpkt = ip/icmp/data

    print("Sending spoofed pkt from: " + pkt[IP].src + " back to " + pkt[IP].dst)
    os.write(tun, bytes(newpkt))

Get Help Write Out Where Is Cut Text Justify Cur Pos Undo Mark Text To Bracket
Exit Read File Replace Paste Text To Spell Go To Line Redo Copy Text Where Was

```

Check if ICMP packet type is 8 (Echo request), and construct an echo reply and write it to the TUN interface. The code was taken from the sniff_and_spoof.py code from Week 1 Lab.

The new packet that was created has its source and destination swapped with a random data being added

`show_img('Task_2d/spoofed_icmp_out.png')`

```

anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythink1208@seed-1004326VM: /volumes# ./tun.py &
[2] 186
[1]  Terminated                  ./tun.py
client@seed-1004326VM: /volumes# Interface Name: ly0

client@seed-1004326VM: /volumes# tcpdump -i ly0 -n 2> /dev/null &
[3] 194
client@seed-1004326VM: /volumes# ping 192.168.53.5 -c 1
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
08:46:11.279453 IP 192.168.53.99 > 192.168.53.5: ICMP echo request, id 20, seq 1, length 64
ly0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Sending spoofed pkt from: 192.168.53.99 back to 192.168.53.5
08:46:11.282010 IP truncated-ip - 29435 bytes missing! 114.105.116.105 > 110.103.32.97: ip-proto-102

--- 192.168.53.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

client@seed-1004326VM: /volumes# 

```

Part 2: write some arbitrary data to the interface

`show_img('Task 2d/data_changed.png')`

```

Activities Terminal Mar 28 08:51
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythink1208@seed-1004326VM: /volumes# ./tun.py
tun.py

GNU nano 4.8
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'lynd', 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))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        pkt = IP(packet)
        print("{}:{}".format(ifname), pkt.summary())

    # Send out a spoof packet using the tun interface
    if ICMP in pkt and pkt[ICMP].type != 8:
        break
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
    icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
    data = pkt[Raw].load
    newpkt = ip/icmp/data

    print("Sending spoofed pkt from: " + pkt[IP].src + " back to " + pkt[IP].dst)
    data = b"Instead of writing an IP packet to the interface, write some arbitrary data to the interface, and report your observation"
    os.write(tun, data)

    print("Raw data: " + data)

```

`show_img('Task 2d/dump_out.png')`

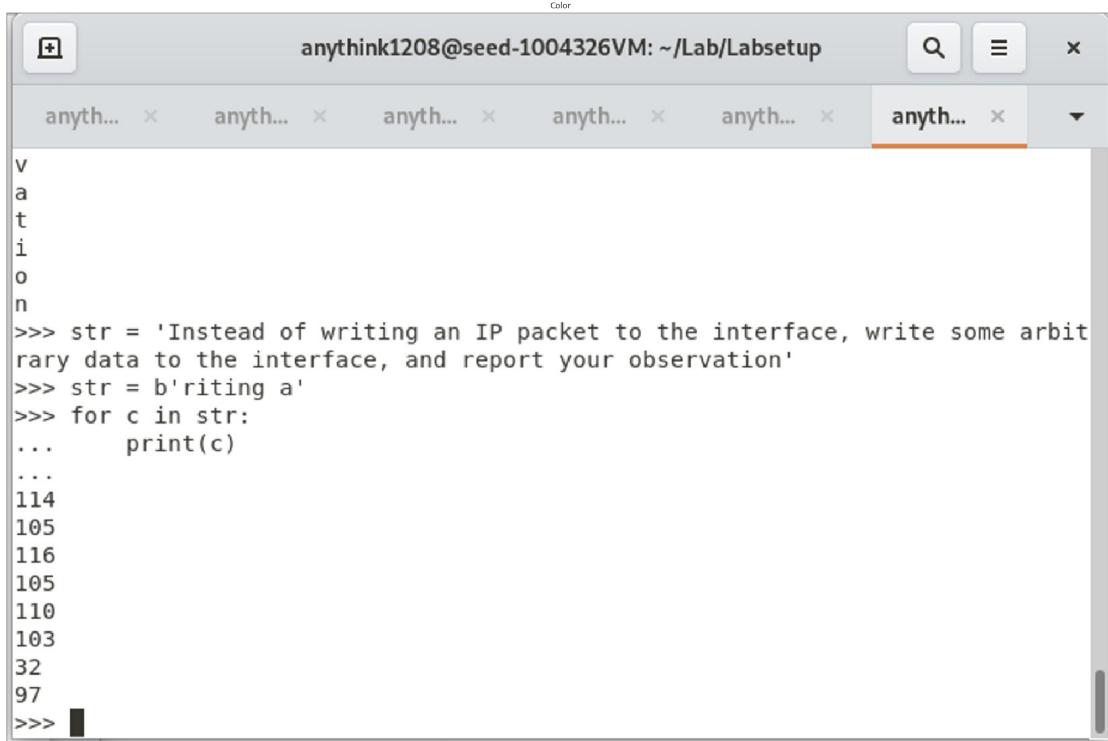
```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anyth... x anyth... x anyth... x anyth... x anyth... x anyth... x anyth... x
client@seed-1004326VM:/volumes# ./tun.py &
[2] 186
[1]  Terminated                  ./tun.py
client@seed-1004326VM:/volumes# Interface Name: ly0

client@seed-1004326VM:/volumes# tcpdump -i ly0 -n 2> /dev/null &
[3] 194
client@seed-1004326VM:/volumes# ping 192.168.53.5 -c 1
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
08:46:11.279453 IP 192.168.53.99 > 192.168.53.5: ICMP echo request, id 20, seq 1, length 64
ly0: IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
Sending spoofed pkt from: 192.168.53.99 back to 192.168.53.5
08:46:11.282010 IP truncated-ip - 29435 bytes missing! 114.105.116.105 > 110.103.32.97: ip-proto-102
--- 192.168.53.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

client@seed-1004326VM:/volumes#
```

After changing the data to a binary data, with random text, some random IP address was provided. The IP address provided is actually the binary values of 'riting a', which is seen below:

```
show_img('Task 2d/bin_val.png')
```



```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anyth... x anyth... x anyth... x anyth... x anyth... x anyth... x
v
a
t
i
o
n
>>> str = 'Instead of writing an IP packet to the interface, write some arbitrary data to the interface, and report your observation'
>>> str = b'riting a'
>>> for c in str:
...     print(c)
...
114
105
116
105
110
103
32
97
>>> 
```

The source data starts from the 13th byte, inclusive of spacing. The 4 bytes (13-16) refers to the source IP and after the next corresponding 4 bytes (17-20) refers to the destination IP

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

```
show_img('Task 3/tun_server.png')
```

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anyt... x anyt... x anyth... x anyt... x anyth... x anyth... x anyth... x
GNU nano 4.8 tun server.py Modified
#!/usr/bin/env python3
from scapy.all import *
IP_A = "0.0.0.0"
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))

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^Y Replace ^U Paste Text ^T To Spell
```

```
show_img('Task 3/tun_client_1.png')
show_img('Task 3/tun_client_2.png')
```

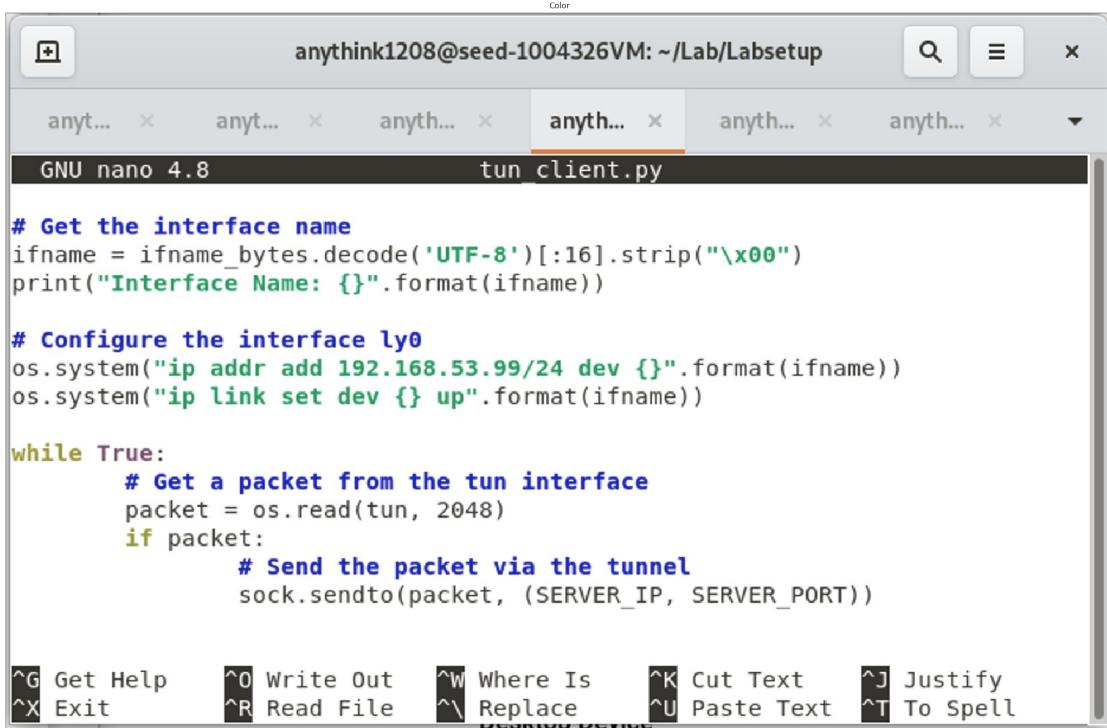
```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
anyt... x anyt... x anyth... x anyt... x anyth... x anyth... x anyth... x
GNU nano 4.8 tun client.py Modified
#!/usr/bin/env python3

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

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP, SERVER_PORT = "10.9.0.11", 9090

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^Y Replace ^U Paste Text ^T To Spell
```



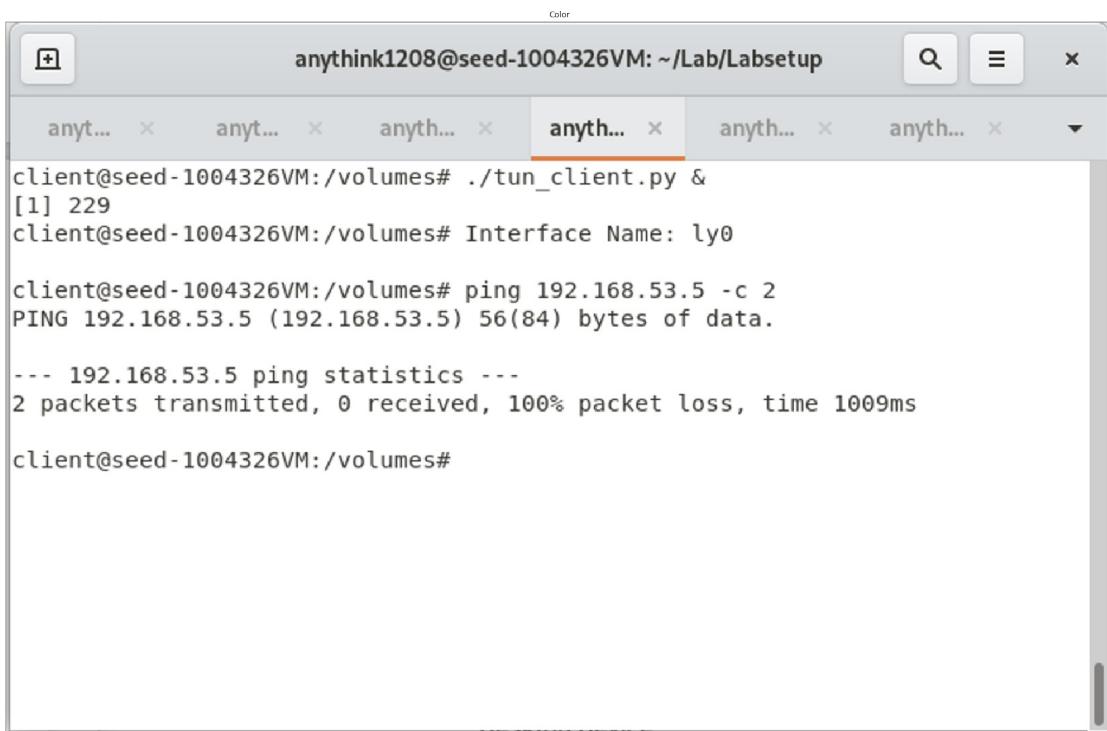
```
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

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))

^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File     ^Y Replace       ^U Paste Text    ^T To Spell
```

```
show_img('Task 3/ping_net.png')
```



```
client@seed-1004326VM:/volumes# ./tun_client.py &
[1] 229
client@seed-1004326VM:/volumes# Interface Name: ly0

client@seed-1004326VM:/volumes# ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.

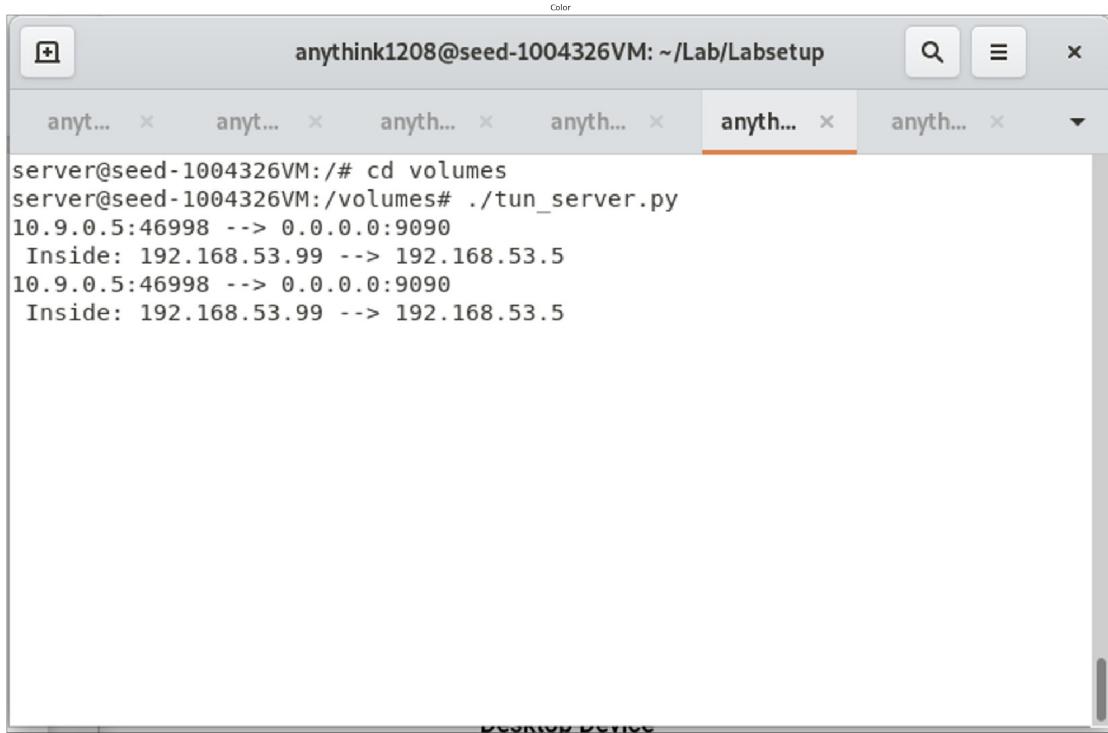
--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1009ms

client@seed-1004326VM:/volumes#
```

There is no reply to the ping echo request, but the packet is routed to the tun interface ly0 (192.168.53.99), so the tun server is able to capture the packet. The UDP packet is sent

from the client to the server at port 9090. Inside the UDP packet, it shows that the packet is received from ly0 IP address to a host that is non-existent (192.168.53.5)

```
show_img('Task 3/tun_server_out.png')
```



A terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window has a tab bar with several tabs, one of which is highlighted in orange and labeled "anyth...". The main pane displays the following log output:

```
server@seed-1004326VM:/# cd volumes
server@seed-1004326VM:/volumes# ./tun_server.py
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
```

After an IP route is added to 192.168.60.0/24 network, the virtual address is able to connect to hosts in the network, so the ping to 192.168.60.5 is routed to the ly0 interface.

```
show_img('Task 3/ip_route.png')
show_img('Task 3/tun_server_out_2.png')
```

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
```

```
client@seed-1004326VM:/volumes# ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.

--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1009ms

client@seed-1004326VM:/volumes# ip route add 192.168.60.0/24 dev ly0
client@seed-1004326VM:/volumes# 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 ly0 proto kernel scope link src 192.168.53.99
192.168.60.0/24 dev ly0 scope link
client@seed-1004326VM:/volumes# ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1015ms

client@seed-1004326VM:/volumes#
```

```
anythink1208@seed-1004326VM: ~/Lab/Labsetup
```

```
any... x any... x any... x any... x any... x any... x
```

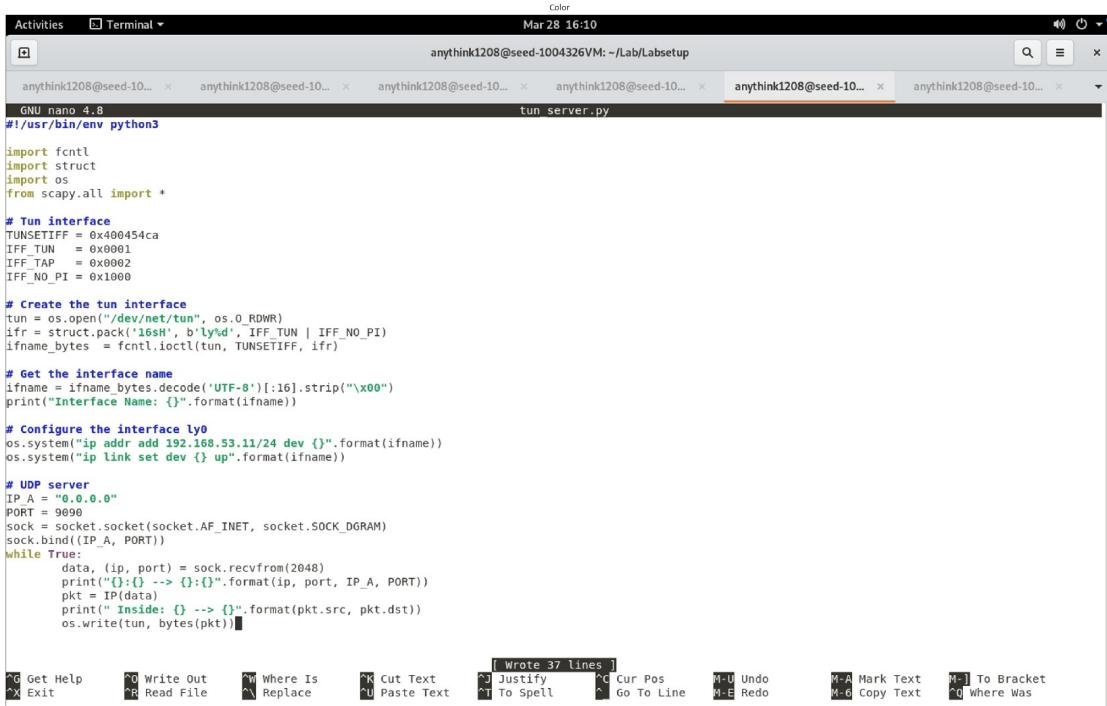
```
server@seed-1004326VM:# cd volumes
server@seed-1004326VM:/volumes# ./tun_server.py
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.53.5
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:46998 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```

Task 4: Set Up the VPN Server

- Create a TUN interface and configure it
- Get the data from the socket interface; treat the received data as an IP packet

- Write the packet to the TUN interface

```
show_img('Task 4/config_tun_server.png')
```



```
anythink1208@seed-1004326VM:~/Lab/Labsetup
anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
GNU nano 4.8 tun server.py
#!/usr/bin/env python3

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

# Tun interface
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('16s', b'lyd', IFF_TUN | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tun, TUNSETIFF, ifr)

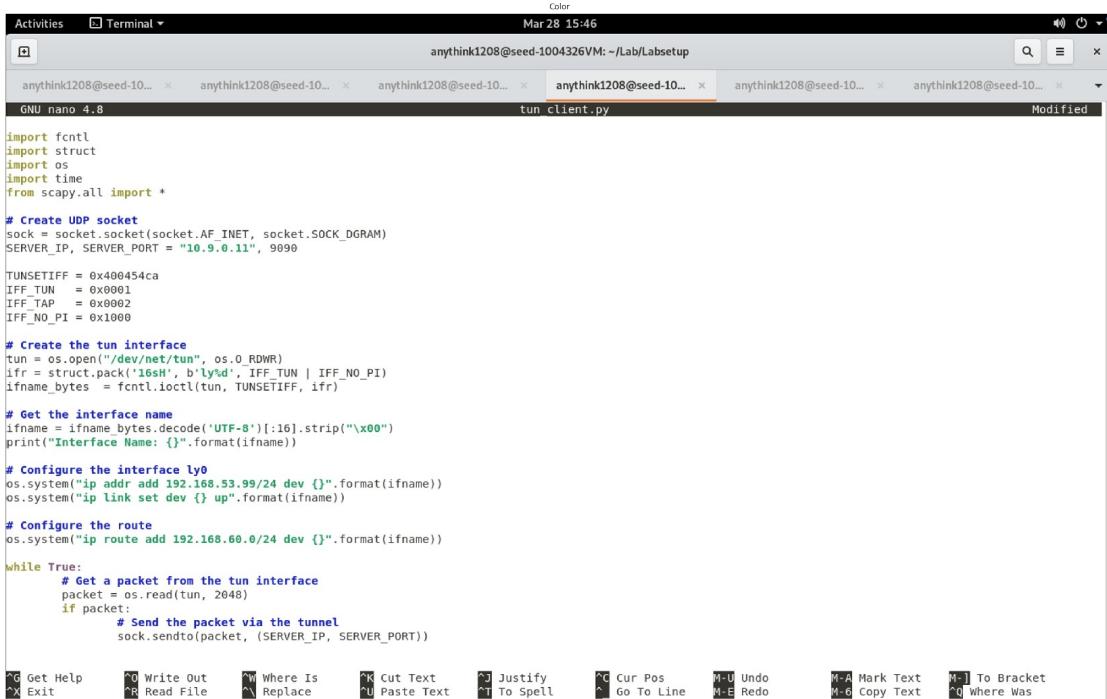
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# UDP server
IP_A = "0.0.0.0"
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, bytes(pkt))
```

The terminal shows the script being run in a nano editor. The output indicates the creation of a TUN interface named 'lyd', assignment of IP 192.168.53.11/24, and binding to UDP port 9090. A loop is running to receive and forward data through the TUN interface.

```
show_img('Task 4/config_tun_client.png')
```



```
anythink1208@seed-1004326VM:~/Lab/Labsetup
anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
GNU nano 4.8 tun client.py Modified
import fcntl
import struct
import os
import time
from scapy.all import *

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
SERVER_IP, SERVER_PORT = "10.9.0.11", 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('16s', b'lyd', 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))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Configure the route
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

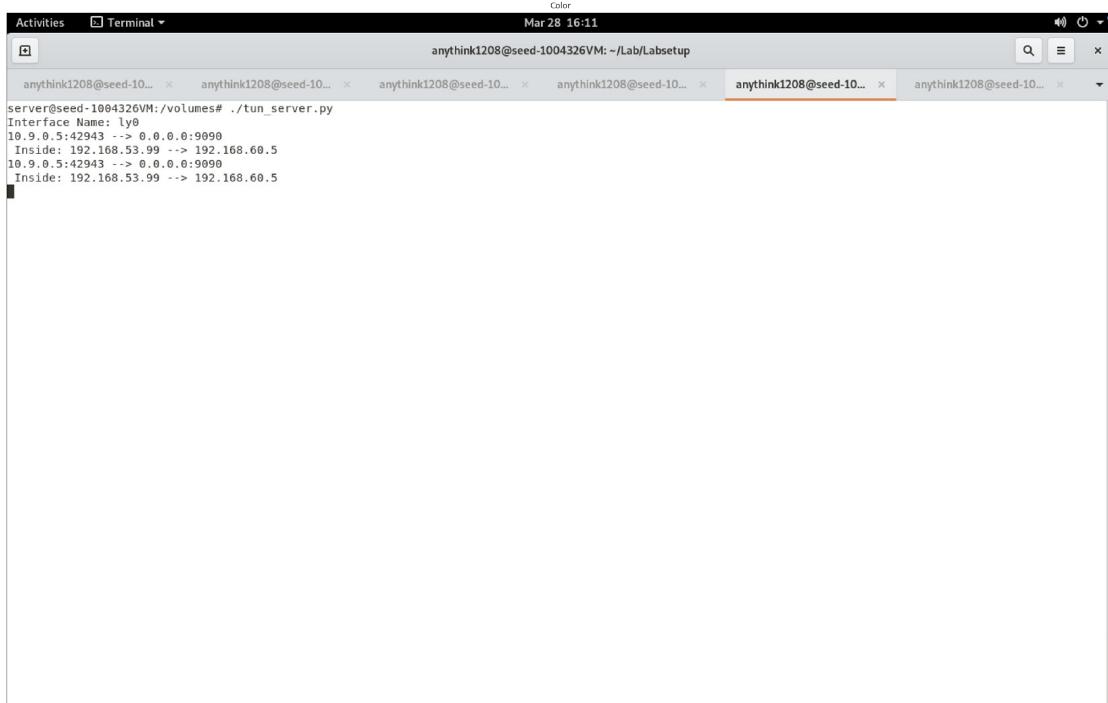
while True:
    # Get a packet from the tun interface
    packet = os.read(tun, 2048)
    if packet:
        # Send the packet via the tunnel
        sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

The terminal shows the script being run in a nano editor. The output indicates the creation of a TUN interface named 'lyd', assignment of IP 192.168.53.99/24, and sending UDP packets to a server at 10.9.0.11:9090 via the tunnel interface.

```
show_img('Task 4/tun_server_out.png')
```

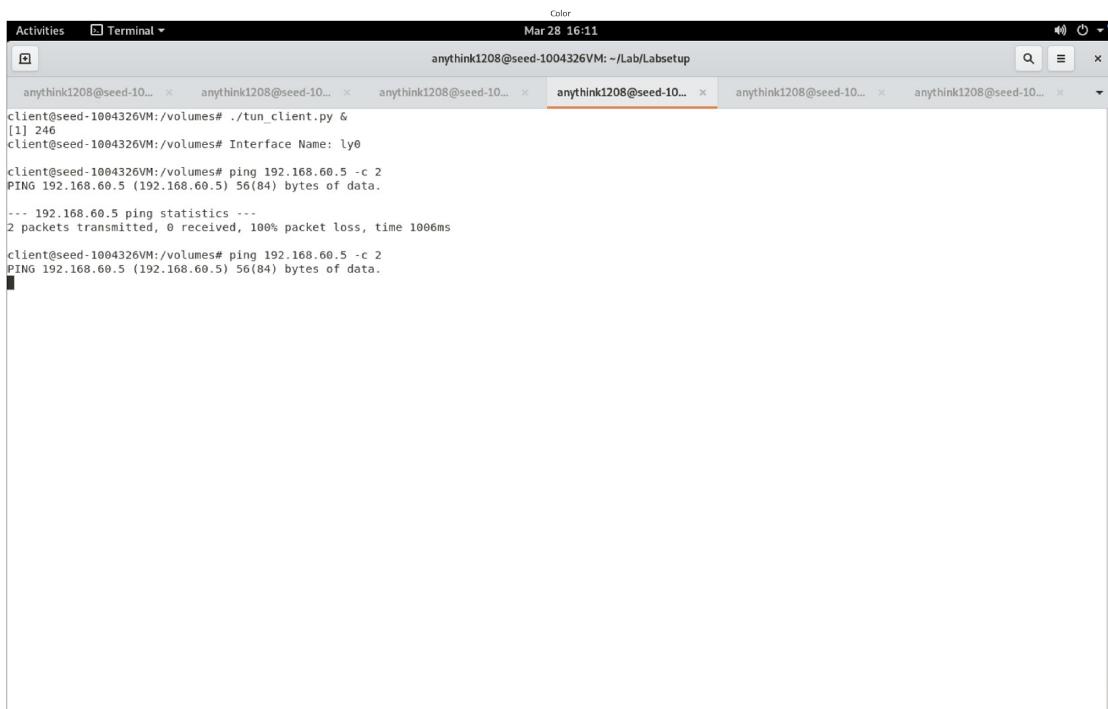
```
show_img('Task 4/tun_client_out.png')
```

```
show_img('Task 4/dump_out.png')
show_img('Task 4/wireshark_out.png')
```



A screenshot of a Linux terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The terminal shows the output of a command to start a TUN server. The output includes:

```
server@seed-1004326VM:/volumes# ./tun_server.py
Interface Name: ly0
10.9.0.5:42943 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
10.9.0.5:42943 --> 0.0.0.0:9090
Inside: 192.168.53.99 --> 192.168.60.5
```



A screenshot of a Linux terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The terminal shows the output of a command to start a TUN client. The output includes:

```
client@seed-1004326VM:/volumes# ./tun_client.py &
[1] 246
client@seed-1004326VM:/volumes$ Interface Name: ly0

client@seed-1004326VM:/volumes# ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1006ms

client@seed-1004326VM:/volumes# ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

```
Activities Terminal Mar 28 16:12 Color
anythink1208@seed-10... anythink1208@seed-10... anything1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
host1@seed-1004326VM:~/tcpdump -i eth0 -n 2>/dev/null
16:11:35.182618 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 26, seq 1, length 64
16:11:35.182676 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 26, seq 1, length 64
16:11:36.218962 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 26, seq 2, length 64
16:11:36.218989 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 26, seq 2, length 64
16:11:40.214072 ARP Request who-has 192.168.60.11 tell 192.168.60.5, length 28
16:11:40.214315 ARP Request who-has 192.168.60.11 tell 192.168.60.11, length 28
16:11:40.214353 ARP Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
16:11:40.214355 ARP Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```

Capturing from veth3b81ea0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|-------------------|-------------------|----------|--------|-----------------------------------|
| 1 | 0.000000000 | 10.9.0.5 | 10.9.0.11 | UDP | 126 | 40917 → 9090 Len=84 |
| 2 | 1.025843339 | 10.9.0.5 | 10.9.0.11 | UDP | 126 | 40917 → 9090 Len=84 |
| 3 | 5.249752813 | 02:42:0a:09:00:05 | 02:42:0a:09:00:0b | ARP | 42 | Who has 10.9.0.11? Tell 10.9.0.11 |
| 4 | 5.249947382 | 02:42:0a:09:00:0b | 02:42:0a:09:00:05 | ARP | 42 | 10.9.0.11 is at 02:42:0a:09:00:05 |

In the tcpdump and wireshark above, we can see that 2 ICMP echo request and replies are made, as expected from the 2 ping packets. The client has successfully sent a packet to the server and host V 192.168.60.5 and sent back.

Task 5: Handling Traffic in Both Directions

show_img('Task 5/config_tun_server.png')



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window contains a nano editor displaying a Python script named "tun_server.py". The script is used to handle traffic in both directions using a TUN interface. It includes code for creating the TUN interface, configuring it, and setting up a UDP server to receive and forward data between the TUN interface and a local port. The terminal window also shows several other tabs open, likely related to the lab setup.

```
GNU nano 4.8
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'lyd', 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))

# Configure the interface ly0
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# UDP server
IP_A = "0.0.0.0"
PORT = 9999
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:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            print("From UDP {}: {} -> {}:{}".format(ip, port, IP_A, PORT))
            pkt = IP(data)
            print("From socket <=:: {} -> {}".format(pkt.src, pkt.dst))
            sock.sendto(data, (ip, port))

        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==:: {} -> {}".format(pkt.src, pkt.dst))
            sock.sendto(packet, (ip, port))
```

show_img('Task 5/config_tun_client.png')

```

Activities Terminal Mar 29 07:11
anythink1208@seed-1004326VM: ~/Lab/Labsetup
GNU nano 4.8 tun_client.py

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'tynd', IFF_TUN | IFF_NO_PI)
ifname_bytes = Tctl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Configure the route
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

while True:
    # this will block until at least one interface is ready
    ready, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
            data, (ip, port) = sock.recvfrom(2048)
            print("From UDP {}: {} --> 10.9.0.5".format(ip, port))
            pkt = IP(data)
            print("From socket <==: {} --> {}".format(pkt.src, pkt.dst))
            os.write(tun, bytes(pkt))
        if fd is tun:
            packet = os.read(tun, 2048)
            pkt = IP(packet)
            print("From tun ==> {} --> {}".format(pkt.src, pkt.dst))
            # Get a packet from the tun interface
            sock.sendto(packet, (SERVER_IP, SERVER_PORT))

```

Get Help Write Out Where Is Cut Text Justify Cur Pos Undo Mark Text To Bracket
 Exit Read File Replace Paste Text To Spell Go To Line Redo Copy Text Where Was

When client or server receives data in the tun interface, it will forward the UDP socket, and when the data is received from the UDP socket, it sends to the tun interface.

```

show_img('Task 5/ping/ping_net.png')
show_img('Task 5/ping/tun_server_out.png')
show_img('Task 5/ping/dump_out.png')
show_img('Task 5/ping/wireshark_out.png')

```

```

anythink1208@seed-1004326VM: ~/Lab/Labsetup
anyt... anyt... anyt... anyt... anyt... anyt... anyt... anyt...

[1] 322
client@seed-1004326VM:/volumes# Interface Name: ly0

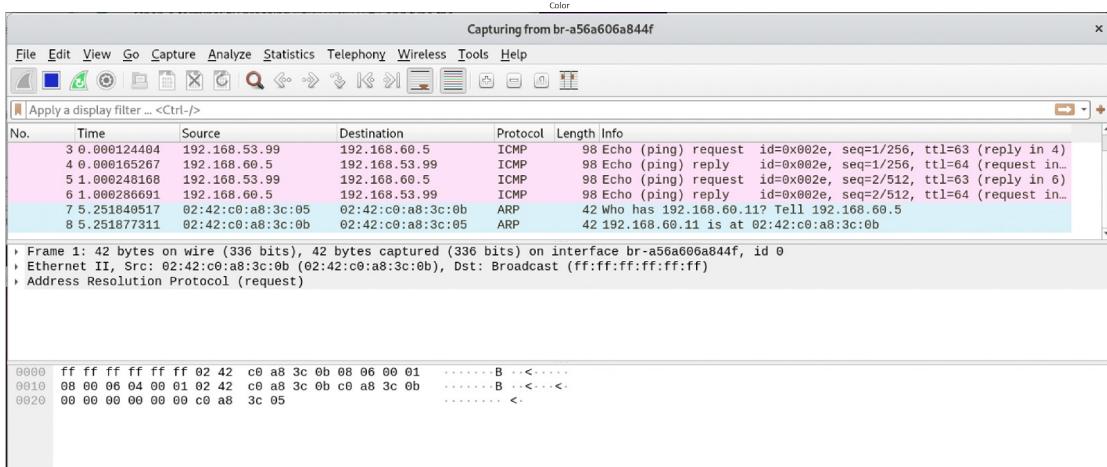
client@seed-1004326VM:/volumes# ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket <==: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=8.26 ms
From tun ==>: 192.168.53.99 --> 192.168.60.5
From UDP 10.9.0.11:9090 --> 10.9.0.5
From socket <==: 192.168.60.5 --> 192.168.53.99
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=2.19 ms

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.186/5.224/8.263/3.038 ms
client@seed-1004326VM:/volumes#

```

```
anyt... x anyt... x anyt... x anyt... x anyt... x
server@seed-1004326VM:/volumes# ./tun_server.py
Interface Name: ly0
From UDP 10.9.0.5:39015 --> 0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
From UDP 10.9.0.5:39015 --> 0.0.0.0:9090
From socket <==: 192.168.53.99 --> 192.168.60.5
From tun ==>: 192.168.60.5 --> 192.168.53.99
```

```
anyt... x anyt... x anyt... x anyt... x anyt... x anyt... x
07:01:20.306340 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
07:13:04.209232 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 32, seq 1, length 64
07:13:04.209448 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 32, seq 1, length 64
07:13:05.208228 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 32, seq 2, length 64
07:13:05.208252 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 32, seq 2, length 64
07:13:09.426102 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
07:13:09.426229 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
07:13:09.426264 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
07:13:09.426273 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
```



When ping is sent from 10.9.0.5 to the server at 10.9.0.11, port 9090, the server forwards it to 192.168.60.5 from address 192.168.53.99.

```
show_img('Task 5/telnet/telnet_net.png')
show_img('Task 5/telnet/tun_server_out.png')
show_img('Task 5/telnet/dump_out.png')
show_img('Task 5/telnet/wireshark_out.png')
```

```
Activities Terminal Mar 29 13:42
anythink1208@seed-1004326VM:~/Lab/Labsetup
anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...

client@seed-1004326VM:/volumes# ./tun_client.py &>/dev/null &
[1] 338
client@seed-1004326VM:/volumes# jobs
[1]+ Running ./tun_client.py &> /dev/null &
client@seed-1004326VM:/volumes# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
269fc02c1021 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.13.0-1019-gcp x86_64)

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

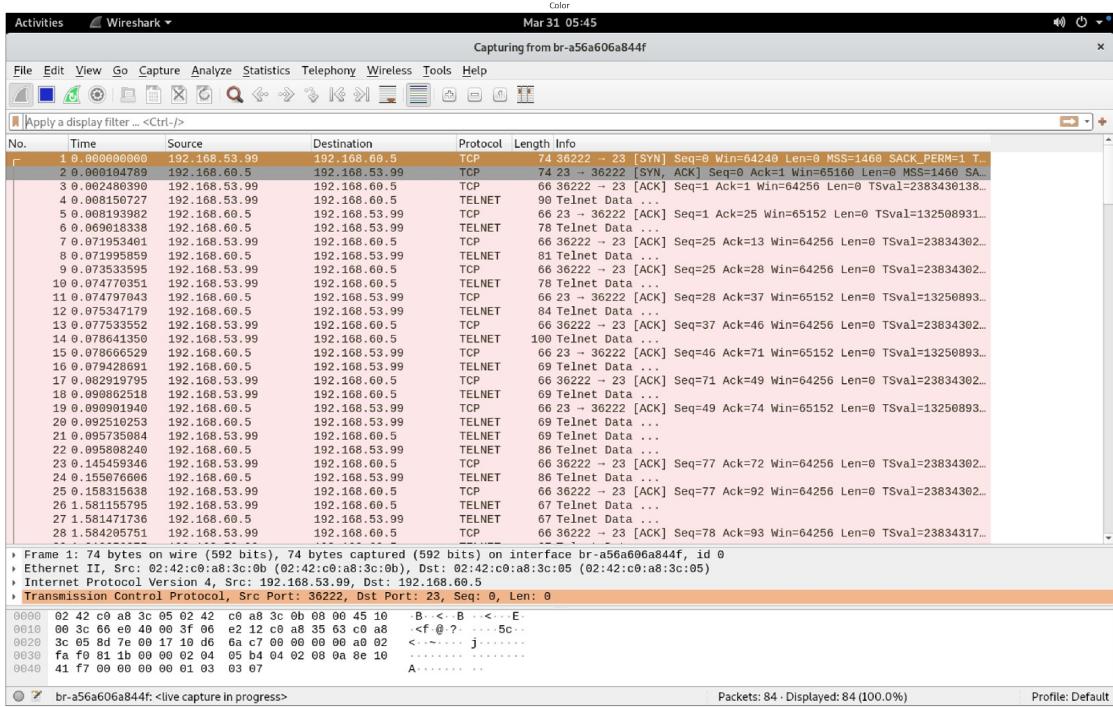
This system has been minimized by removing packages and content that are
not required on the system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

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

seed@269fc02c1021:~$ exit
logout
Connection closed by foreign host.
client@seed-1004326VM:/volumes#
```

When telnet is sent from 10.9.0.5 to the server at 10.9.0.11, port 36042, the server forwards it to 192.168.60.5 from address 192.168.53.99.

Task 6: Tunnel-Breaking Experiment

```
show_img('Task 6/break_server.png')
show_img('Task 6/telnet_hang.png')
```

The shell was not responsive when the server is stopped. However, when the server resumes, the shell reconnects and becomes responsive

```
show_img('Task 6/telnet_resume.png')
```

```

anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10... anythink1208@seed-10...
Connection closed by foreign host.
client@seed-1004326VM:/volumes# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
269fc02c1021 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.13.0-1019-gcp x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Mar 29 13:42:17 UTC 2022 from 192.168.53.99 on pts/2
seed@269fc02c1021:~$ exit
logout
Connection closed by foreign host.
client@seed-1004326VM:/volumes# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
269fc02c1021 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.13.0-1019-gcp x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Mar 29 13:43:13 UTC 2022 from 192.168.53.99 on pts/2
seed@269fc02c1021:~$ date
Tue Mar 29 14:03:41 UTC 2022
seed@269fc02c1021:~$ date

```

Task 7: Routing Experiment on Host V

show_img('setup/docker_2_setup.png')

```

anythink1208@seed-1004326VM:~/Lab/Labsetup$ sudo docker compose
-f docker-compose2.yml up
[+] Running 6/0
  # Container host-192.168.50.5 Created          0.0s
  # Container host-192.168.50.6 Created          0.0s
  # Container client-10.9.0.5      C...          0.0s
  # Container host-192.168.60.5 Created          0.0s
  # Container host-192.168.60.6 Created          0.0s
  # Container server-router     Cre...          0.0s
Attaching to client-10.9.0.5, host-192.168.50.5, host-192.168.50.6, ho
st-192.168.60.5, host-192.168.60.6, server-router

```

show_img('Task 7/add_route.png')
show_img('Task 7/ping_net.png')

```
Activities Terminal Mar 29 16:31 anythink1208@seed-1004326VM: ~/Lab/Labsetup
anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208...
GNU nano 4.8 tun_server.py Modified
#!/usr/bin/env python3

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

ip, port = '10.9.0.5', 12345

# Tun interface
TUNSETIFF = 0x400045ca
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'lynd', 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))

# Configure the interface
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Configure the route
os.system("ip route add 192.168.50.0/24 dev {}".format(ifname))

# UDP server
IP_A = "0.0.0.0"
PORT = 9090
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], [], [])
    if ready[0] == tun:
        # read from tun interface
        data = tun.read(1500)
        if len(data) > 0:
            # send data to IP_A:PORT
            sock.sendto(data, (IP_A, PORT))
    else:
        # read from UDP socket
        data, addr = sock.recvfrom(1500)
        # write data to tun interface
        tun.write(data)

# Get Help   F0 Write Out   F1 Where Is   F2 Cut Text   F3 Justify   F4 Cur Pos   F5 Undo   F6 Mark Text   F7 To Bracket
X Exit      F8 Read File   F9 Replace    F0 Paste Text  F1 To Spell  F2 Go To Line  F3 Redo    F4 Copy Text  F5 Where Was
```

```
Activities Terminal Color Mar 30 00:58
anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208... anythink1208...
anythink1208... x anythink1208... x

host3 192.168.50.5@seed-1004326VM: # ping 192.168.60.5 -c 2
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=62 time=4.98 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=62 time=3.04 ms
--- 192.168.60.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 3.040/4.008/4.977/0.968 ms
host3:192.168.50.5@seed-1004326VM: #
```

Task 8: Experiment with the TUN IP Address

On Host U, change its TUN interface's IP address to 192.168.30.99

```
show_img('Task 8/change_ip.png')
show_img('Task 8/ping_net.png')
show_img('Task 8/tun_client_out.png')
```

The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window title bar includes multiple tabs labeled "anythink1208...". The main area displays the content of a file named "tun client.py". The code is a Python script that uses scapy to create a TUN interface, bind it to port 9090, and then listen for incoming UDP connections. It also configures the interface and routes traffic. The terminal window has a standard Linux-style interface with a menu bar, a toolbar at the bottom, and a status bar at the bottom right.

```
GNU nano 4.8
#!/usr/bin/env python3

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

# Create UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# SERVER_IP, SERVER_PORT = "10.9.0.11", 9090
SERVER_IP, SERVER_PORT = "192.168.30.99", 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'tun', IFF_TAP | 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))

# Configure the interface ly0
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Configure the route
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

while True:
    # this will block until at least one interface is ready
    ready, _ = select.select([sock, tun], [], [])
    for fd in ready:
        if fd is sock:
```

The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window title bar includes multiple tabs labeled "anythink1208...". The main area displays the output of a ping command from host3 to host1. The output shows two packets transmitted, zero received, 100% packet loss, and a time of 1028ms. The terminal window has a standard Linux-style interface with a menu bar, a toolbar at the bottom, and a status bar at the bottom right.

```
anythink1208@seed-1004326VM:~/Lab/Labsetup$ ping 192.168.60.5 -c 2
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1028ms

host3-192.168.50.5@seed-1004326VM:~#
```

```
anythink1208@seed-1004326VM:/volumes# ./tun_client.py
Interface Name: ly0
From tun ==>: 192.168.50.5 --> 192.168.60.5
From tun ==>: 192.168.50.5 --> 192.168.60.5
```

The packets are dropped at 192.168.30.0 as there is no corresponding interface and route at 192.168.30.0/24 network. Due to the lack of time, I used tcpdump to check the packet trace instead.

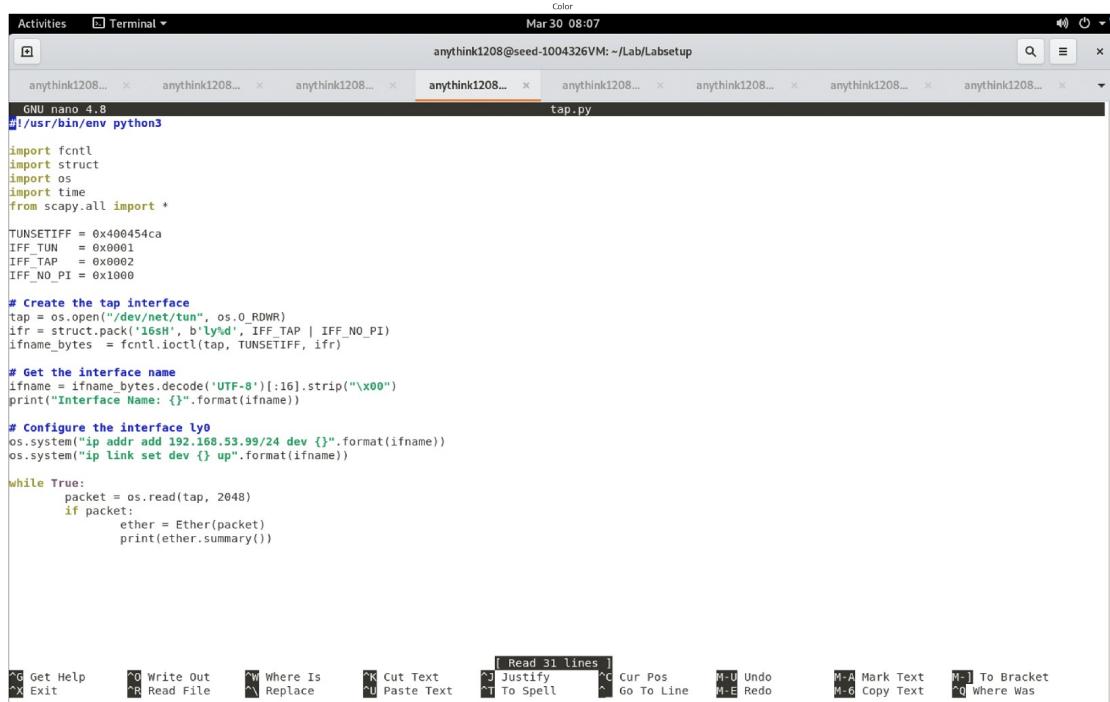
`show_img('Task 8/tun_server_out.png')`

```
anythink1208@seed-1004326VM:/volumes# ./tun_server.py
Interface Name: ly0
```

Since the tun_server is located 192.168.53.99/24, the server does not receive the packets

Task 9: Experiment with the TAP Interface

```
show_img('Task 9/create_tap.png')
show_img('Task 9/arping_without_spoof.png')
show_img('Task 9/arping_without_spoof_2.png')
```



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window has multiple tabs open, all labeled "anythink1208...". The current tab contains the following Python script:

```
GNU nano 4.8
#!/usr/bin/env python

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

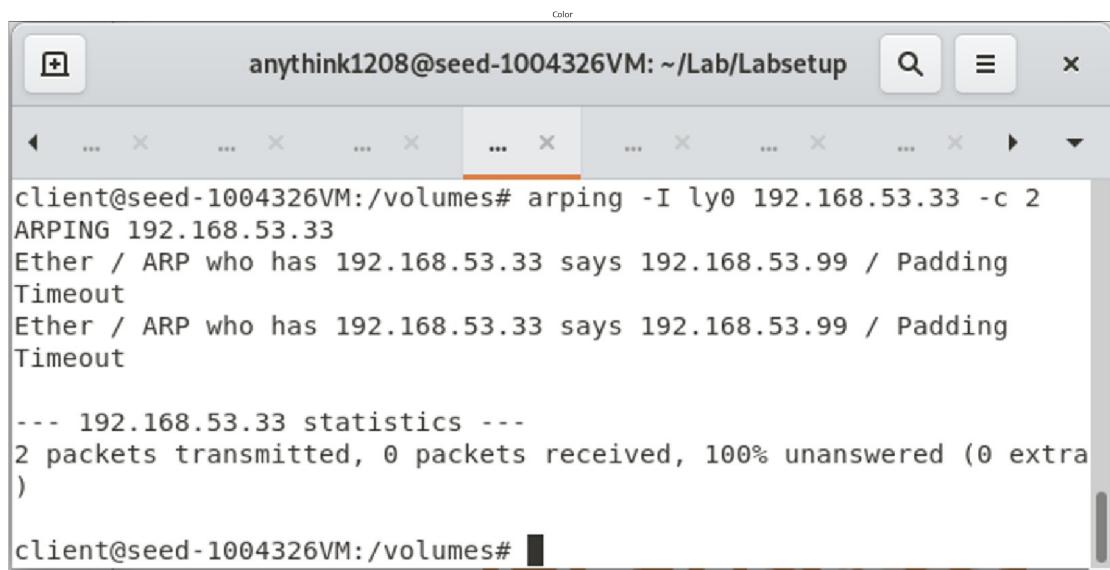
TUNSETEIFF = 0x4000454ca
IFF_TUN   = 0x00001
IFF_TAP   = 0x00002
IFF_NO_PI = 0x1000

# Create the tap interface
tap = os.open('/dev/net/tun', os.O_RDWR)
ifr = struct.pack('16s', b'ly0', IFF_TAP | IFF_NO_PI, ifr)
ifname_bytes = fcntl.ioctl(tap, TUNSETEIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
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(tap, 2048)
    if packet:
        ether = Ether(packet)
        print(ether.summary())
```



The screenshot shows a terminal window titled "anythink1208@seed-1004326VM: ~/Lab/Labsetup". The window displays the output of the "arping" command:

```
client@seed-1004326VM:/volumes# arping -I ly0 192.168.53.33 -c 2
ARPING 192.168.53.33
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
Timeout

--- 192.168.53.33 statistics ---
2 packets transmitted, 0 packets received, 100% unanswered (0 extra
)

client@seed-1004326VM:/volumes#
```

```

client@seed-1004326VM:/volumes# arping -I ly0 1.2.3.4 -c 2
ARPING 1.2.3.4
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
Timeout
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
Timeout

--- 1.2.3.4 statistics ---
2 packets transmitted, 0 packets received, 100% unanswered (0 extra
)

client@seed-1004326VM:/volumes#

```

When initiating a ping to 192.168.53.5, it starts off with sending an ARP packet to find MAC address of the host with the corresponding IP. Afterwards, it sends 2 ping requests to 192.168.53.5. Arping gives 100% unanswered as 192.168.53.5 or 1.2.3.4 is non-existent in the networks.

Spoofed ARP reply TAP

```

!cat tap.py

#!/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 tap interface
tap = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'ly%d', IFF_TAP | IFF_NO_PI)
ifname_bytes = fcntl.ioctl(tap, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))

# Configure the interface ly0
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(tap, 2048)
#     if packet:
#         ether = Ether(packet)
#         print(ether.summary())

while True:
    packet = os.read(tap, 2048)
    if packet:
        print("-----")
        ether = Ether(packet)
        print(ether.summary())
        # Send a spoofed ARP response
        FAKE_MAC = "aa:bb:cc:dd:ee:ff"
        if ARP in ether and ether[ARP].op == 1 :
            arp = ether[ARP]
            newether = Ether(dst=ether.src, src=FAKE_MAC)
            newarp = ARP(psrc=arp.pdst, hwsrc=FAKE_MAC,
                         pdst=arp.psrc, hwdst=ether.src, op=2)
            newpkt = newether/newarp
            print("***** Fake response:")
            print("{}".format(newpkt.summary()))
            os.write(tap, bytes(newpkt))

show_img('Task 9/ping_net.png')
show_img('Task 9/arping_with_spoof.png')

```

```

client@seed-1004326VM:/volumes# ./tap.py &
[1] 163
client@seed-1004326VM:/volumes# Interface Name: ly0

client@seed-1004326VM:/volumes# ping 192.168.53.5 -c 2
PING 192.168.53.5 (192.168.53.5) 56(84) bytes of data.
-----
Ether / ARP who has 192.168.53.5 says 192.168.53.99
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.5
-----
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
-----
Ether / IP / ICMP 192.168.53.99 > 192.168.53.5 echo-request 0 / Raw
--- 192.168.53.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1021ms

```

```
client@seed-1004326VM:/volumes# arping -I ly0 192.168.53.33 -c 2
ARPING 192.168.53.33

-----
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=0 time=10.650 usec

-----
Ether / ARP who has 192.168.53.33 says 192.168.53.99 / Padding
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 192.168.53.33
42 bytes from aa:bb:cc:dd:ee:ff (192.168.53.33): index=1 time=7.008 usec

--- 192.168.53.33 statistics ---
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.007/0.009/0.011/0.002 ms
client@seed-1004326VM:/volumes# arping -I ly0 1.2.3.4 -c 2
ARPING 1.2.3.4

-----
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 1.2.3.4
42 bytes from aa:bb:cc:dd:ee:ff (1.2.3.4): index=0 time=10.221 usec

-----
Ether / ARP who has 1.2.3.4 says 192.168.53.99 / Padding
***** Fake response: Ether / ARP is at aa:bb:cc:dd:ee:ff says 1.2.3.4
42 bytes from aa:bb:cc:dd:ee:ff (1.2.3.4): index=1 time=1.516 msec

--- 1.2.3.4 statistics ---
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.010/0.763/1.516/0.753 ms
```

With a successful ARP spoofing, a faked MAC address of aa:bb:cc:dd:ee:ff is received for an arping request of 192.168.53.33 or 1.2.3.4.