Name:- MOHAMMED TOUSIF
Roll No:- CB.EN.P2CYS22008

IMPLEMENTING VIRTUAL PRIVATE NETWORK

Date :- 16-01-2023

Aim :- To implement virtual private network using virtual machines.

Tools Required :- Docker Files , Seed VM v20.

Procedure :-

Task 1: Network Setup

```
[01/16/23]seed@VM:~/.../Labsetup$ dcbuild
VPN_Client uses an image, skipping
Host1 uses an image, skipping
Host2 uses an image, skipping
Router uses an image, skipping
[01/16/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Pulling VPN_Client (handsonsecurity/seed-ubuntu:large)...
large: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Pulling fs layer
14428a6d4bcd: Pulling fs layer
14428a6d4bcd: Downloading [==================================
da7391352a9b: Downloading [>
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
b5e99359ad22: Pull complete
3d2251ac1552: Pull complete
1059cf087055: Pull complete
b2afee800091: Pull complete
c2ff2446bab7: Pull complete
4c584b5784bd: Pull complete
```

Host-U

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh client-10.9.0.5
root@be17a5159f9d:/# echo $PS1
\u@\h:\w\$
root@be17a5159f9d:/# export PS1="U-10.9.0.5:\w\n\$>"
U-10.9.0.5:/
$>
```

VPN Server

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh server-router
root@53539db6613d:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
N 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
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc no
queue state UP group default
    link/ether 02:42:0a:09:00:0b brd ff:ff:ff:ff:ff:ff link-netnsi
d 0
    inet 10.9.0.11/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
14: eth1@if15: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc no
queue state UP group default
    link/ether 02:42:c0:a8:3c:0b brd ff:ff:ff:ff:ff:ff link-netnsi
d 0
    inet 192.168.60.11/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
root@53539db6613d:/# export PS1="router-10.9.0.11-192.168.60.11:\w
\n\$>"
router-10.9.0.11-192.168.60.11:/
$>
```

Host-V

```
[01/16/23]seed@VM:~/.../Labsetup$ docksh host-192.168.60.5
root@98f9ea036905:/# export PS1="V-192.168.60.5:\w\n\$>"
V-192.168.60.5:/
$>
```

Host U can communicate with VPN server

```
U-10.9.0.5:/
$>ping 10.9.0.11 -c 4
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=0.350 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.548 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.354 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.297 ms

--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3075ms
rtt min/avg/max/mdev = 0.297/0.387/0.548/0.095 ms
U-10.9.0.5:/
$>
```

VPN server can communicate with Host V

```
router-10.9.0.11-192.168.60.11:/
$>ping 192.168.60.5 -c 4
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.125 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.774 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.105 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.406 ms

--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3026ms
rtt min/avg/max/mdev = 0.105/0.352/0.774/0.270 ms
router-10.9.0.11-192.168.60.11:/
$>
```

Host U should not be able to communicate with Host V

```
$>ping 192.168.60.5 -c 3
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2049ms

U-10.9.0.5:/
```

Run tcpdump on router and sniff the traffic.

```
$>tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decod
e
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
12:30:55.276731 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 1,
 length 64
12:30:55.276884 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 1, l
ength 64
12:30:56.279126 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 14, seq 2,
 length 64
12:30:56.279233 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 14, seq 2, l
ength 64
12:31:00.528040 ARP, Request who-has 10.9.0.5 tell 10.9.0.11, length 28
12:31:00.528577 ARP, Request who-has 10.9.0.11 tell 10.9.0.5, length 28
12:31:00.528712 ARP, Reply 10.9.0.11 is-at 02:42:0a:09:00:0b, length 28
12:31:00.528735 ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05, length 28
```

Task 2: Create and configure TUN interface

Task 2.a : Name of the interface

Running the tun.py code to create the TUN interface.

```python
#!/usr/bin/env python3

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

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

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'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:
   time.sleep(10)
```

Running the tun.py

```
tun.py
U-10.9.0.5:/volumes
$>chmod +x tun.py
U-10.9.0.5:/volumes
$>./tun.py
Interface Name: tun0
^Z
[1]+  Stopped                 ./tun.py
U-10.9.0.5:/volumes
$>
```

Interface is created

```
$>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
2: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN gro
up default qlen 500
    link/none
```

Changing code to name it as our own interface

```
14 # Create the tun interface
15 tun = os.open("/dev/net/tun", os.O_RDWR)
16 ifr = struct.pack('16sH', b'tousif%d', IFF_TUN | IFF_NO_PI)
17 ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
18
```

Running the code again we will get our interface

```
3: tousif0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN
group default qlen 500
    link/none
```

Task 2.b: Setup the TUN interface

It can be done through adding this two lines in tun.py code.

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

```
4: tousif1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_c
odel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global tousif1
       valid_lft forever preferred_lft forever
```

Task 2.c: Read from the TUN interface

Modifying while loop code.

```
while True:

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

First , On Host U , we ping 192.168.53.1 network as a part of 192.168.53.0/24 network. As the interface is not yet configured to get a reply back , we can see the packet loss gets place.

```
./tun.py &
[2] 61
[1]   Exit 127               ./tunpy
U-10.9.0.5:/volumes
$>Interface Name: tousif0

U-10.9.0.5:/volumes
$>ping 192.168.53.1 -c 3
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw

--- 192.168.53.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2054ms

U-10.9.0.5:/volumes
```

Now pinging 192.168.60.0/24 network , still no packets are received as client tries sending out packets to network outside of the host network which is unreachable.

```
$>./tun.py &
[1] 79
U-10.9.0.5:/volumes
$>Interface Name: tousif0

U-10.9.0.5:/volumes
$>ping 192.168.60.5 -c 3
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.

--- 192.168.60.5 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2036ms

U-10.9.0.5:/volumes
```

Task 2.d: Write to the TUN interface

Adding below lines of code in tun.py file.

```
# Send out a spoof packet using the tun interface
newip = IP(src='192.168.53.3', dst=ip.src)
newpkt = newip/ip.payload
os.write(tun, bytes(newpkt))
```

```
$>ping 192.168.53.99 -c 3
PING 192.168.53.99 (192.168.53.99) 56(84) bytes of data.
64 bytes from 192.168.53.99: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 192.168.53.99: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 192.168.53.99: icmp_seq=3 ttl=64 time=0.158 ms

--- 192.168.53.99 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.028/0.088/0.158/0.053 ms
```

```
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
```

Task 3: Send the IP Packet to VPN server through a tunnel

tun-server.py

```
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))
```

client.py

```
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'Upadh%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

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

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

SERVER_PORT = 9090
SERVER_IP = "10.9.0.11"

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:
                sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```