Firewall Evasion Lab: Bypassing Firewalls using VPN

Objective:

Organizations, Internet Service Providers (ISPs), and countries often block their internal users from accessing certain external sites. This is called egress filtering. For example, to prevent work-time distraction, many companies set up their egress firewalls to block social network sites, so their employee cannot access those sites from inside their network. These firewalls can be easily bypassed, and services or products that help users bypass firewalls are widely available on the Internet. The most commonly used technology to bypass egress firewalls is Virtual Private Network (VPN). The goal of this lab is to see how VPN works in action and how VPN can help bypass egress firewalls. This lab covers the following topics:

• Firewall
• VPN

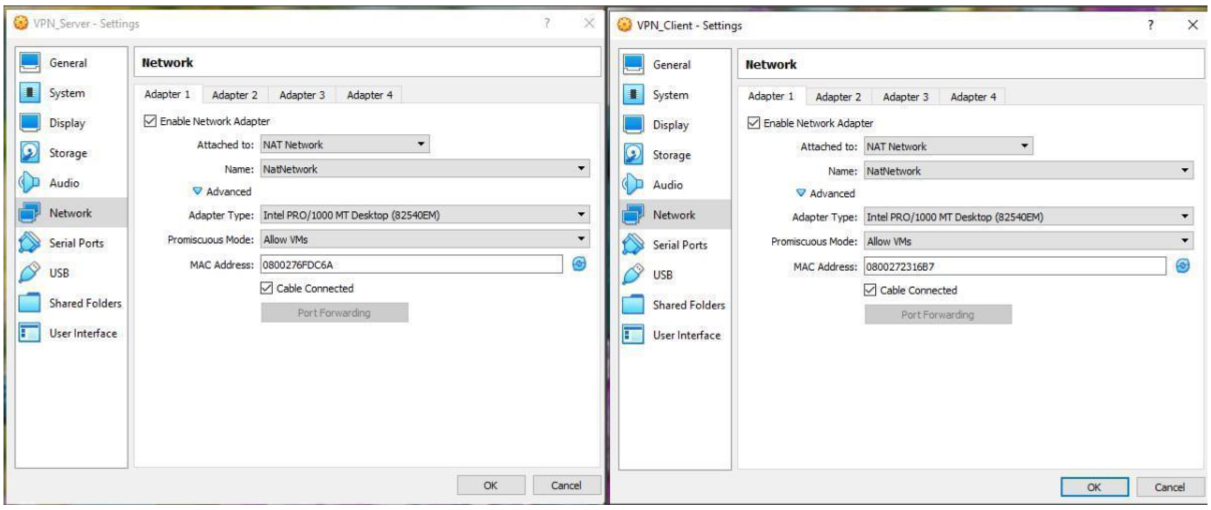
Lab Tasks

Task 1: VM Setup

Objective:

Two machines are needed where one machine should be inside the firewall, and the other machine should be outside the firewall. So that the machine inside the firewall will be able to reach out to the external sites blocked by the firewall.

- We have used two VM's named VPN_Server and VPN_Client connected to LAN using the NAT Network adapter.



VPN_Server - Settings

Network — Adapter 1

☑ Enable Network Adapter
Attached to: NAT Network
Name: NatNetwork
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow VMs
MAC Address: 0800276FDC6A
☑ Cable Connected

VPN_Client - Settings

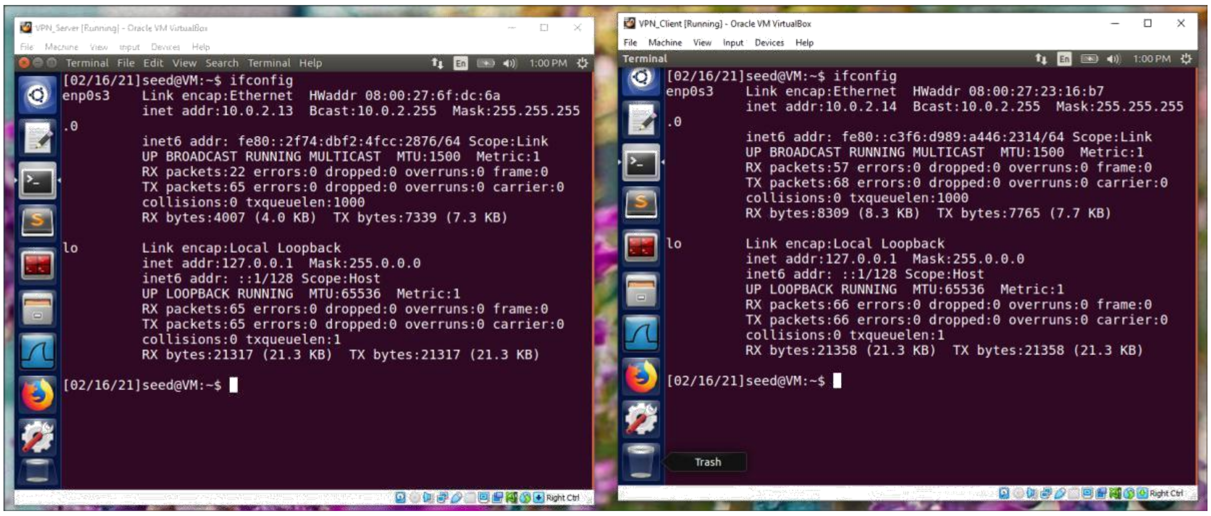
Network — Adapter 1

☑ Enable Network Adapter
Attached to: NAT Network
Name: NatNetwork
Advanced
Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode: Allow VMs
MAC Address: 0800272316B7
☑ Cable Connected

Observation:

The NAT network is enabled on both VM's

- Ran "*ifconfig*" command on both VM's to get the IP address of both.


VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
1:00 PM

```
[02/16/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:6f:dc:6a
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255
.0
          inet6 addr: fe80::2f74:dbf2:4fcc:2876/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4007 (4.0 KB)  TX bytes:7339 (7.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:65 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21317 (21.3 KB)  TX bytes:21317 (21.3 KB)

[02/16/21]seed@VM:~$
```

VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
1:00 PM
Trash

```
[02/16/21]seed@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:23:16:b7
          inet addr:10.0.2.14  Bcast:10.0.2.255  Mask:255.255.255
.0
          inet6 addr: fe80::c3f6:d989:a446:2314/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8309 (8.3 KB)  TX bytes:7765 (7.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:66 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21358 (21.3 KB)  TX bytes:21358 (21.3 KB)

[02/16/21]seed@VM:~$
```

Observation:
The IP addresses of both machines are:

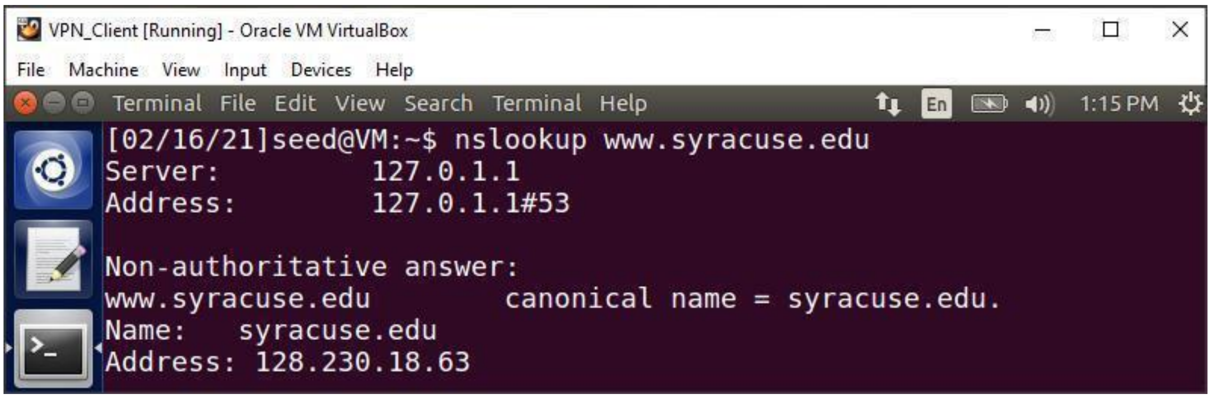
- VPN_Client: 10.0.2.14
- VPN_Server: 10.0.2.13

Task 2: Set up Firewall

Objective:

To set up a firewall on VM1 to block the access of a target website. We need to make sure that the IP address of the target web site is either fixed or in a fixed range; otherwise, we may have trouble completely blocking this website.

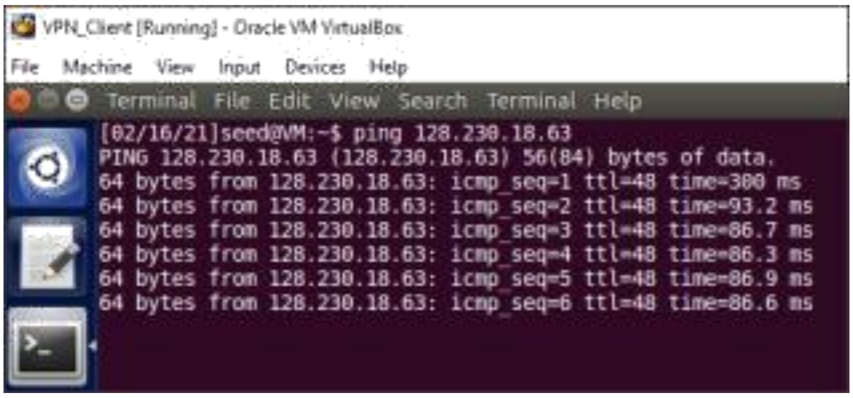
- A website which we would like to block by setting up the firewall is www.syracuse.edu
- We have used "*nslookup*" command to look up the IP address of a www.syracuse.edu on a network.
  nslookup www.syracuse.edu


VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
1:15 PM

```
[02/16/21]seed@VM:~$ nslookup www.syracuse.edu
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
www.syracuse.edu        canonical name = syracuse.edu.
Name:   syracuse.edu
Address: 128.230.18.63
```

Observation:
The IP address of www.syracuse.edu is **128.230.18.63** and this website has fixed IP address.

- Also validated the connectivity of www.syracuse.edu from VPN_Client machine (10.0.2.14) by using ping command.
  ping 128.230.18.63


VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help

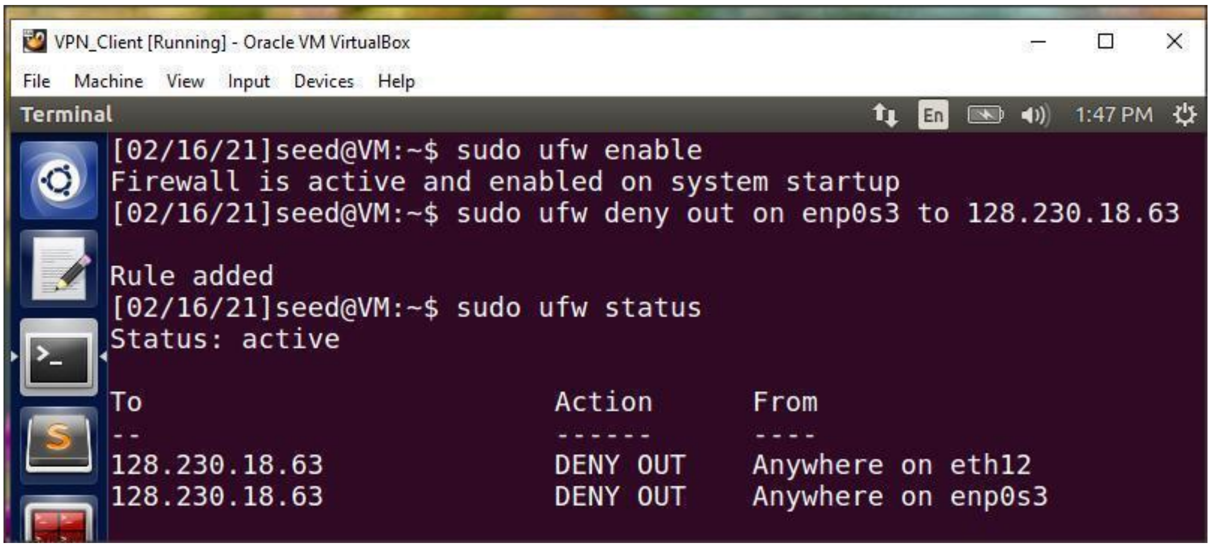
```
[02/16/21]seed@VM:~$ ping 128.230.18.63
PING 128.230.18.63 (128.230.18.63) 56(84) bytes of data.
64 bytes from 128.230.18.63: icmp_seq=1 ttl=48 time=300 ms
64 bytes from 128.230.18.63: icmp_seq=2 ttl=48 time=93.2 ms
64 bytes from 128.230.18.63: icmp_seq=3 ttl=48 time=86.7 ms
64 bytes from 128.230.18.63: icmp_seq=4 ttl=48 time=86.3 ms
64 bytes from 128.230.18.63: icmp_seq=5 ttl=48 time=86.9 ms
64 bytes from 128.230.18.63: icmp_seq=6 ttl=48 time=86.6 ms
```

Observation:

We are successfully able to connect to 128.230.18.63 (www.syracuse.edu) from VPN_Client (10.0.2.14) VM before applying firewall.

- Enabled firewall on VPN_Client (10.0.2.14) VM using given command
  sudo ufw enable
- Applied firewall rules on VPN_Client (10.0.2.14) VM having interface as enp0s3 to block

  128.230.18.63 (www.syracuse.edu) website. We executed given command
  to do so: *sudo ufw deny out on enp0s3 to 128.230.18.63*
- Validated the status of firewall rule applied by executing
  given command: *sudo ufw status*


VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
1:47 PM

```
[02/16/21]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[02/16/21]seed@VM:~$ sudo ufw deny out on enp0s3 to 128.230.18.63

Rule added
[02/16/21]seed@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
128.230.18.63              DENY OUT    Anywhere on eth12
128.230.18.63              DENY OUT    Anywhere on enp0s3
```

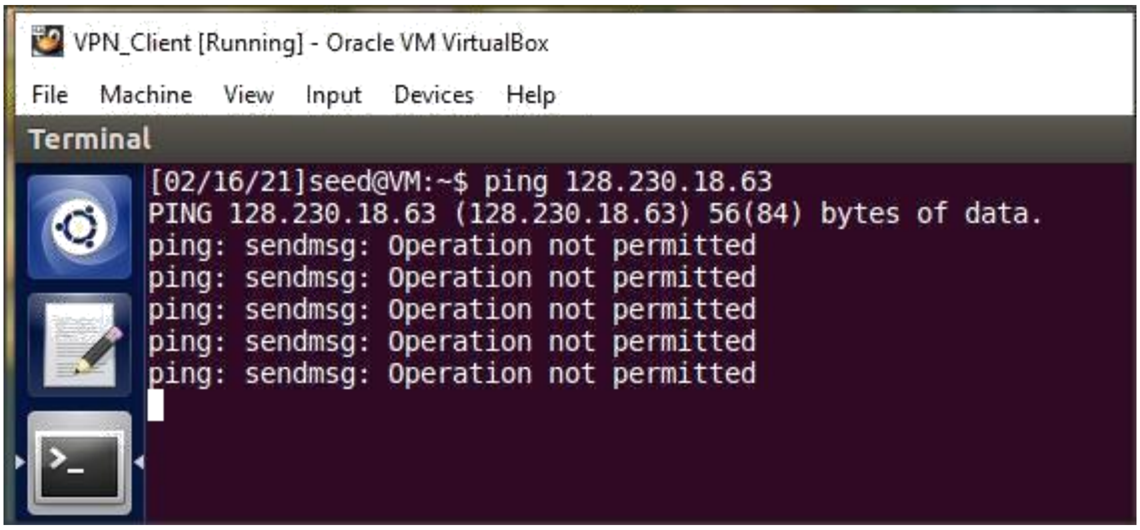
Observation:

The firewall was successfully enabled on the system showing that "*Firewall is active and enabled on system startup*"

After adding rule, it shows- "*Rule added*" which means that the firewall rule which we implemented has been added.

Validation of firewall status shows that the firewall status is "*active*" and also shows the firewall rule added to block *128.230.18.63* from the *enp0s3* along with the "*DENY OUT*" action.

- After setting up the firewall rules, validated the connectivity of website (www.syracuse.edu) from VPN_Client (10.0.2.14) VM by executing given command:
  Ping 128.230.18.63


VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal

```
[02/16/21]seed@VM:~$ ping 128.230.18.63
PING 128.230.18.63 (128.230.18.63) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

Observation:
The VPN_Client (10.0.2.14) VM is not able to connect to the website 128.230.18.63 (www.syracuse.edu) after setting up the firewall rules. Getting error "**ping: sendmsg: Operation not permitted**"
Hence the firewall is working, and the target IP address (128.230.18.63) is no longer reachable from VPN_Client VM.

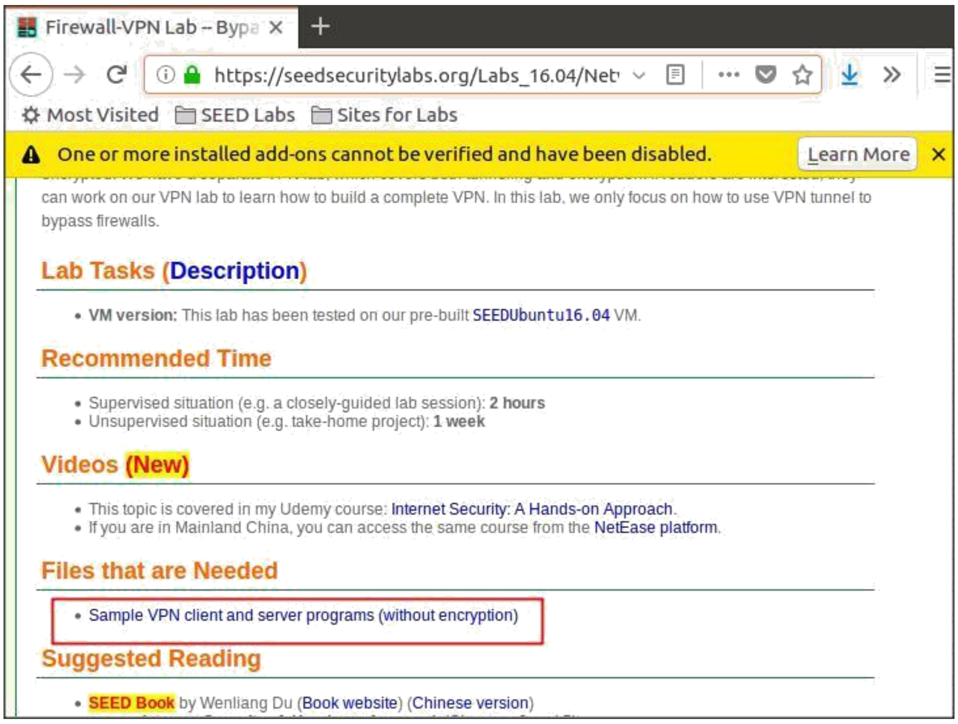
Task 3: Bypassing Firewall using VPN

Objective:

We establish a VPN tunnel between VM1 (VPN Client VM) and VM2 (VPN Server VM). When a user on VM1 tries to access a blocked site, the traffic will not directly go through its network adapter, because it will be blocked. Instead, the packets to the blocked site from VM1 will be routed to the VPN tunnel and arrive at VM2. Once they arrive there, VM2 will route them to the final destination. When the reply packets come back, it will come back to VM2, which will then redirect the packets to the VPN tunnel, and eventually get the packet back to VM1. That is how the VPN helps VM1 to bypass firewalls.

Step 1: Run VPN Server
To run the VPN server we will perform given steps:

- We have downloaded sample VPN programs (both vpnclient and vpnserver) from seed lab's website https://seedsecuritylabs.org/Labs_16.04/Networking/Firewall_VPN/ on both VPN_Client and VPN_Server VMs


Firewall-VPN Lab – Bypa

https://seedsecuritylabs.org/Labs_16.04/Net

Most Visited · SEED Labs · Sites for Labs

One or more installed add-ons cannot be verified and have been disabled. Learn More

can work on our VPN lab to learn how to build a complete VPN. In this lab, we only focus on how to use VPN tunnel to bypass firewalls.

Lab Tasks (Description)

- VM version: This lab has been tested on our pre-built SEEDUbuntu16.04 VM.

Recommended Time

- Supervised situation (e.g. a closely-guided lab session): 2 hours
- Unsupervised situation (e.g. take-home project): 1 week

Videos (New)

- This topic is covered in my Udemy course: Internet Security: A Hands-on Approach.
- If you are in Mainland China, you can access the same course from the NetEase platform.

Files that are Needed

- Sample VPN client and server programs (without encryption)

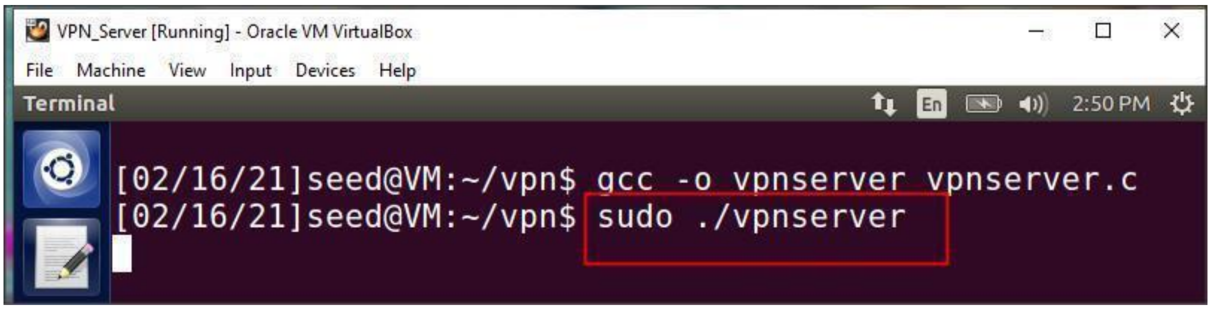
Suggested Reading

- SEED Book by Wenliang Du (Book website) (Chinese version)

Observation:

The extracted folder named “vpn” has been downloaded from the seed’s lab website. Then unextracted this folder and found both vpnserver and vpnclient programs in vpn folder.

- Now compile vpnserver.c program using given command: *gcc -o vpnserver vpnserver.c*
- Executed the compiled program using root privileges as shown in given snapshot: *sudo ./vpnserver*


VPN_Server [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

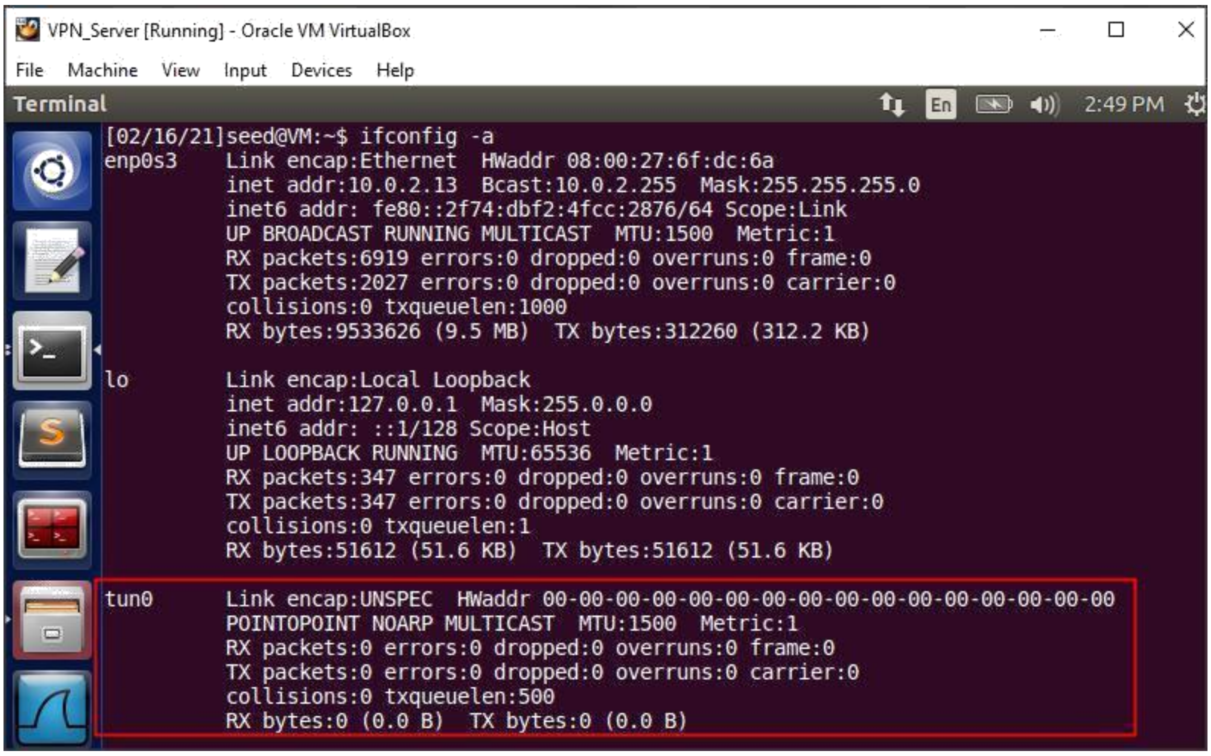
Terminal 2:50 PM

```
[02/16/21]seed@VM:~/vpn$ gcc -o vpnserver vpnserver.c
[02/16/21]seed@VM:~/vpn$ sudo ./vpnserver
```

Observation:

This program will establish a VPN tunnel from server connects to the hosting system via a TUN interface. It does not encrypt the tunnel traffic.

- Since, the execution of vpnserver program block and wait for connections, so we have used another window to run other commands.
- Validating a virtual TUN network interface in the system using “*ifconfig -a*” commandas shown in snapshot:

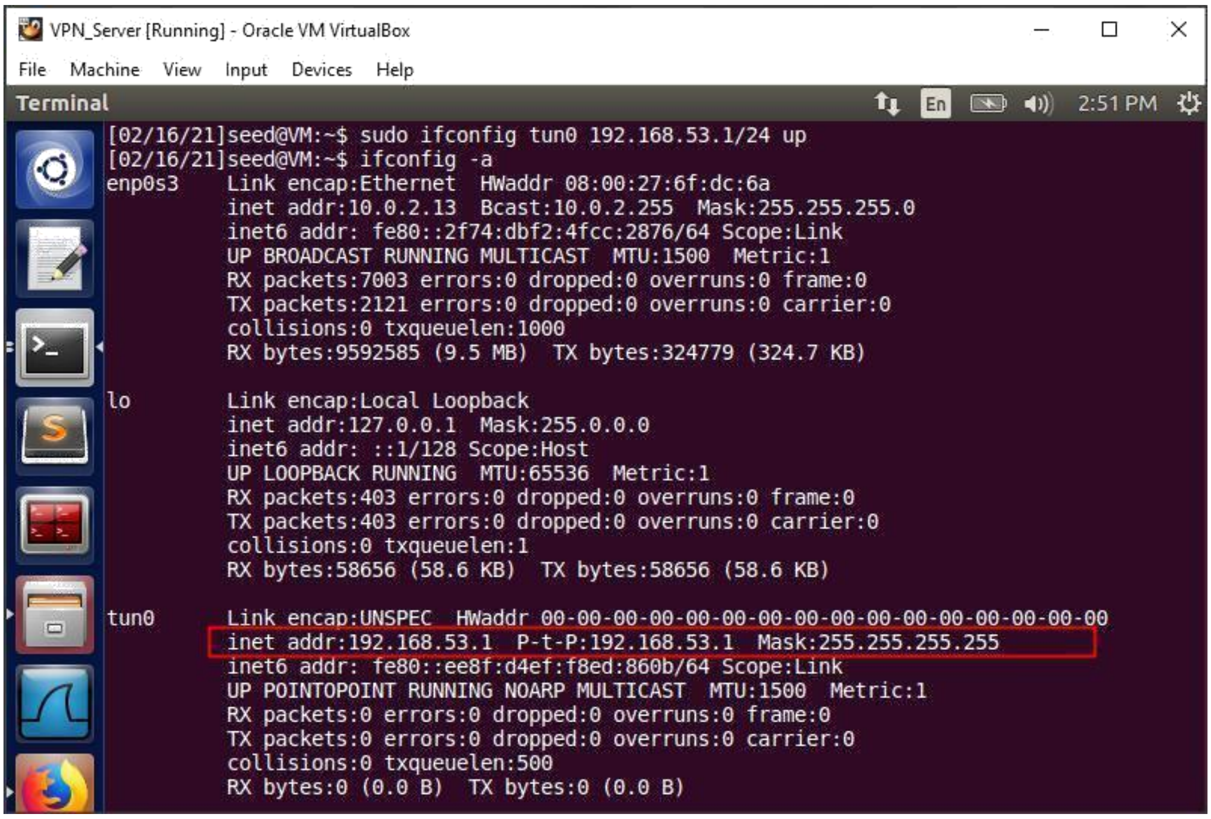

VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En
2:49 PM

```
[02/16/21]seed@VM:~$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:6f:dc:6a
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2f74:dbf2:4fcc:2876/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6919 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2027 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9533626 (9.5 MB)  TX bytes:312260 (312.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:347 errors:0 dropped:0 overruns:0 frame:0
          TX packets:347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:51612 (51.6 KB)  TX bytes:51612 (51.6 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          POINTOPOINT NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Observation:

A virtual TUN network interface has appeared on the VPN_Server VM (10.0.2.13) where the name of interface is **tun0**.

- Now configuring tun0 interface by assigning an IP address “192.168.53.1” and then activating it by using given command:
  sudo ifconfig tun0 192.168.53.1/24 up
- Again, validating the configuration of tun0 interface by executing “*ifconfig -a*” command on VPN_Server VM.

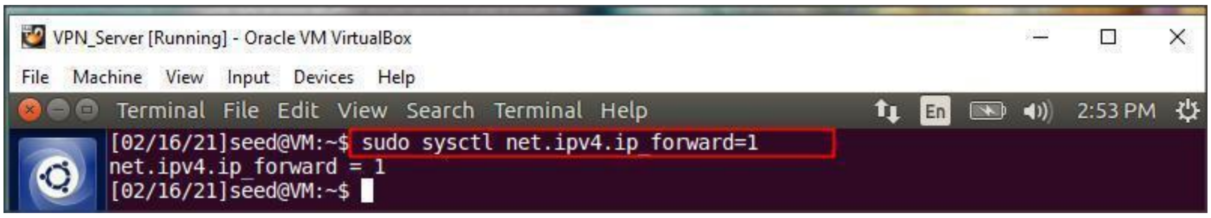

VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
2:51 PM

```
[02/16/21]seed@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[02/16/21]seed@VM:~$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:6f:dc:6a
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2f74:dbf2:4fcc:2876/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7003 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2121 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9592585 (9.5 MB)  TX bytes:324779 (324.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:403 errors:0 dropped:0 overruns:0 frame:0
          TX packets:403 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:58656 (58.6 KB)  TX bytes:58656 (58.6 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:192.168.53.1  P-t-P:192.168.53.1  Mask:255.255.255.255
          inet6 addr: fe80::ee8f:d4ef:f8ed:860b/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

<u>Observation</u>:

The IP address (192.168.53.1) assigned to tun0 interface has been successfully configured as shown in above snapshot.

- For now, a VPN_Server VM will only act as a host not as a gateway unless specifically configured. The VPN Server needs to forward packets to other destinations, so it needs to function as a gateway. Therefore, we have enabled the IP forwarding for a VPN_Server VM to behave like a gateway. Ran given command to enable IP forwarding: *sudo sysctl net.ipv4.ip_forward=1*


VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
2:53 PM

```
[02/16/21]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[02/16/21]seed@VM:~$ 
```

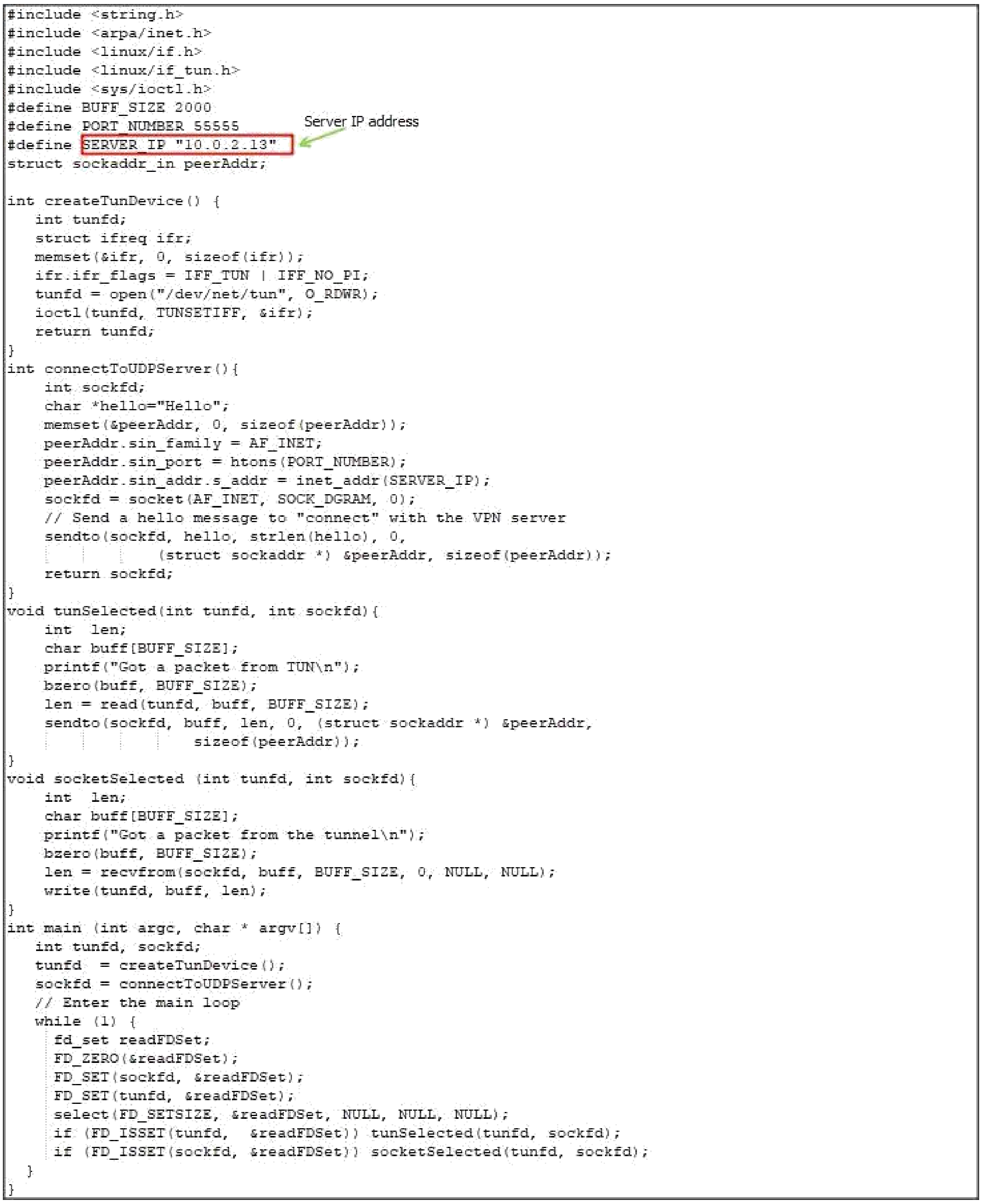
<u>Observation</u>:

The IP forwarding on VPN_Server VM (10.0.2.13) where it should behave as a gateway has been enabled successfully.

Step 2: Run VPN Client

To run the VPN Client, we will perform given steps:

- We will perform the same process for downloading the vpnclient program from seed labs website (as did in Step 1).
- Before executing the program, we have updated the SERVER_IP address from “127.0.0.1” to “10.0.2.13” in the program because our VPN_SERVER VM IP address is 10.0.2.13. Refer following screenshot:



```
#include <string.h>
#include <arpa/inet.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>
#define BUFF_SIZE 2000
#define PORT_NUMBER 55555
#define SERVER_IP "10.0.2.13"
struct sockaddr_in peerAddr;

int createTunDevice() {
   int tunfd;
   struct ifreq ifr;
   memset(&ifr, 0, sizeof(ifr));
   ifr.ifr_flags = IFF_TUN | IFF_NO_PI;
   tunfd = open("/dev/net/tun", O_RDWR);
   ioctl(tunfd, TUNSETIFF, &ifr);
   return tunfd;
}
int connectToUDPServer(){
    int sockfd;
    char *hello="Hello";
    memset(&peerAddr, 0, sizeof(peerAddr));
    peerAddr.sin_family = AF_INET;
    peerAddr.sin_port = htons(PORT_NUMBER);
    peerAddr.sin_addr.s_addr = inet_addr(SERVER_IP);
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    // Send a hello message to "connect" with the VPN server
    sendto(sockfd, hello, strlen(hello), 0,
                (struct sockaddr *) &peerAddr, sizeof(peerAddr));
    return sockfd;
}
void tunSelected(int tunfd, int sockfd){
    int  len;
    char buff[BUFF_SIZE];
    printf("Got a packet from TUN\n");
    bzero(buff, BUFF_SIZE);
    len = read(tunfd, buff, BUFF_SIZE);
    sendto(sockfd, buff, len, 0, (struct sockaddr *) &peerAddr,
                    sizeof(peerAddr));
}
void socketSelected (int tunfd, int sockfd){
    int  len;
    char buff[BUFF_SIZE];
    printf("Got a packet from the tunnel\n");
    bzero(buff, BUFF_SIZE);
    len = recvfrom(sockfd, buff, BUFF_SIZE, 0, NULL, NULL);
    write(tunfd, buff, len);
}
int main (int argc, char * argv[]) {
   int tunfd, sockfd;
   tunfd  = createTunDevice();
   sockfd = connectToUDPServer();
   // Enter the main loop
   while (1) {
     fd_set readFDSet;
     FD_ZERO(&readFDSet);
     FD_SET(sockfd, &readFDSet);
     FD_SET(tunfd, &readFDSet);
     select(FD_SETSIZE, &readFDSet, NULL, NULL, NULL);
     if (FD_ISSET(tunfd,  &readFDSet)) tunSelected(tunfd, sockfd);
     if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd);
  }
}
```

Observation:

The buffer size defined in the program is 2000 and port number is 5555.
We have modified only SERVER_IP.

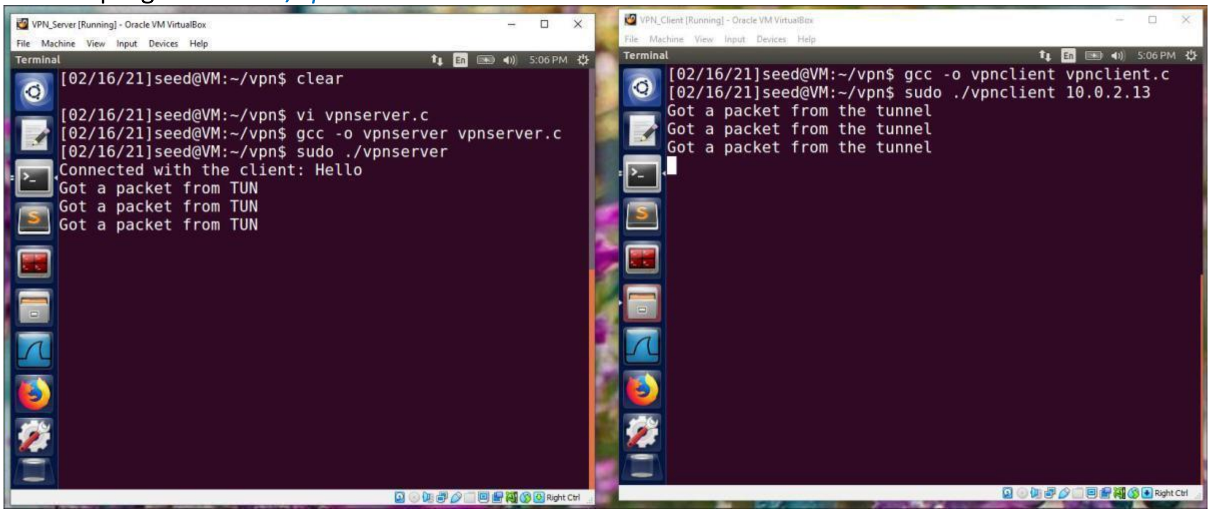
This program has five functions- createTunDevice, connectToUDPServer, tunSelected, socketSelected and main function.

- ➢ createTunDevice () – This function is creating tun interface with number 0 i.e tun0
- ➢ connectToUDPServer () – When the UDP packets are received from VPN Client VM, then the socket will send hello msg showing that it is connecting with VPN server.
- ➢ tunSelected () – This function will execute whenever VPN server gets packet from tun interface

➢
socketSelected () - This function will execute whenever VPN server recieves packet from the tunnel

➢
main () – The execution of program starts from this function.

- Now compiling the vpnclient.c program on VPN_Client (10.0.2.14) VM using given command: *gcc -o vpnclient vpnclient.c*
- After compiling the program, executed following command to run the vpnclient program: *sudo ./vpnclient 10.0.2.13*



```
[02/16/21]seed@VM:~/vpn$ clear

[02/16/21]seed@VM:~/vpn$ vi vpnserver.c
[02/16/21]seed@VM:~/vpn$ gcc -o vpnserver vpnserver.c
[02/16/21]seed@VM:~/vpn$ sudo ./vpnserver
Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

```
[02/16/21]seed@VM:~/vpn$ gcc -o vpnclient vpnclient.c
[02/16/21]seed@VM:~/vpn$ sudo ./vpnclient 10.0.2.13
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

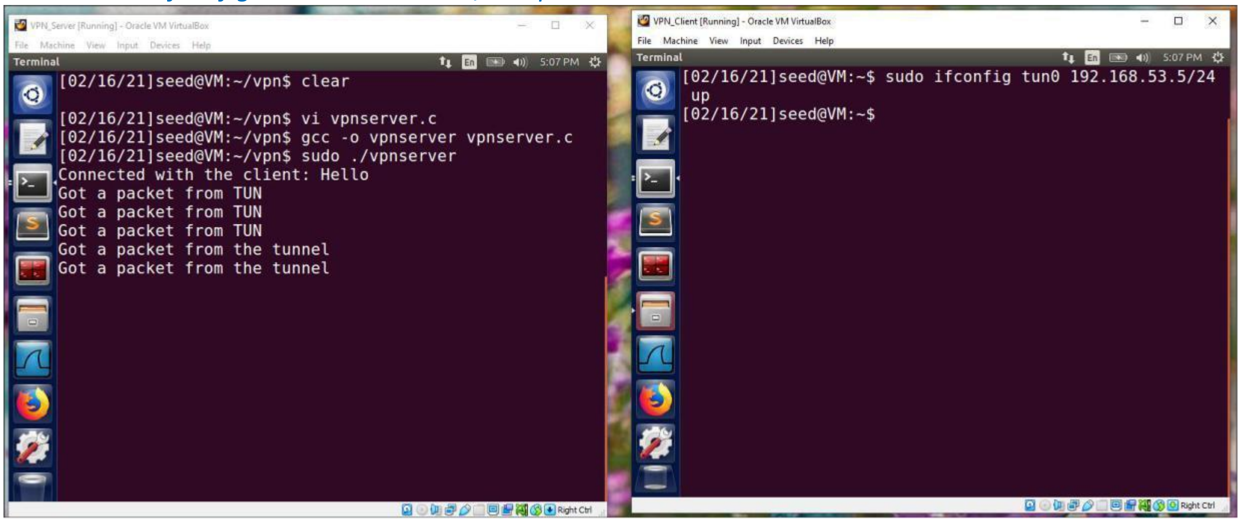
Observation:

On VPN_Client VM, this program will establish a VPN tunnel from client to server and connects to the hosting system via a TUN interface. It does not encrypt the tunnel traffic.

Also, observed that VPN server has send hello message to connect with the VPN server. (As shown in snapshot- "Connected with the client").

Once the client is connected to the VPN server, then VPN server see "Got a packet from TUN" message.
And parallel observed "Got a packet from the tunnel" on VPN Client VM.

- Since, the execution of vpnclient program block and wait for connections, so we have used another window to run other commands.
- We assign IP address 192.168.53.5 to the tun0 interface on VPN_Client VM (10.0.2.14) by executing given command:
  sudo ifconfig tun0 192.168.53.5/24 up



```
[02/16/21]seed@VM:~/vpn$ clear

[02/16/21]seed@VM:~/vpn$ vi vpnserver.c
[02/16/21]seed@VM:~/vpn$ gcc -o vpnserver vpnserver.c
[02/16/21]seed@VM:~/vpn$ sudo ./vpnserver
Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
```

```
[02/16/21]seed@VM:~$ sudo ifconfig tun0 192.168.53.5/24
 up
[02/16/21]seed@VM:~$
```

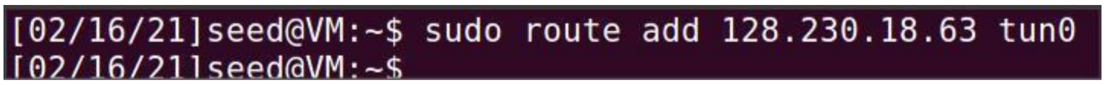
Observation:

On VPN_Server machine, we observed “Got a packet from the tunnel” message which means that tunnel has been established successfully between client and server VM.

Step 3: Set Up Routing on Client and Server VMs

Before we can use the tunnel, we need to set up routing paths on both client and server machines to direct the intended traffic through the tunnel.

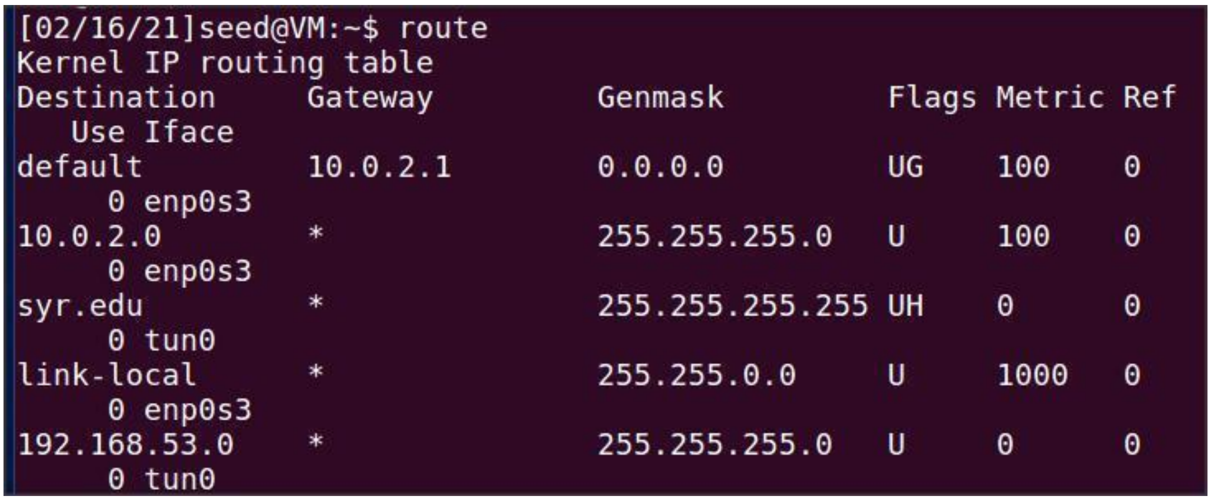
- We have used given command to add the routing on both server and client VM: *sudo route add 128.230.18.63 tun0*



```
[02/16/21]seed@VM:~$ sudo route add 128.230.18.63 tun0
[02/16/21]seed@VM:~$
```

Observation:
This command will add the entry in the kernel IP routing table with the interface as tun0.

- Now validating the route entry using given command: *route*



```
[02/16/21]seed@VM:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
   Use Iface
default         10.0.2.1        0.0.0.0         UG    100    0
     0 enp0s3
10.0.2.0        *               255.255.255.0   U     100    0
     0 enp0s3
syr.edu         *               255.255.255.255 UH    0      0
     0 tun0
link-local      *               255.255.0.0     U     1000   0
     0 enp0s3
192.168.53.0    *               255.255.255.0   U     0      0
     0 tun0
```

Observation:

This will provide the kernel IP routing table having details about the destination IP’s, Gateway, Genmask, Flags, Metric, Ref, User Interface.

Step 4: Set Up NAT on Server VM

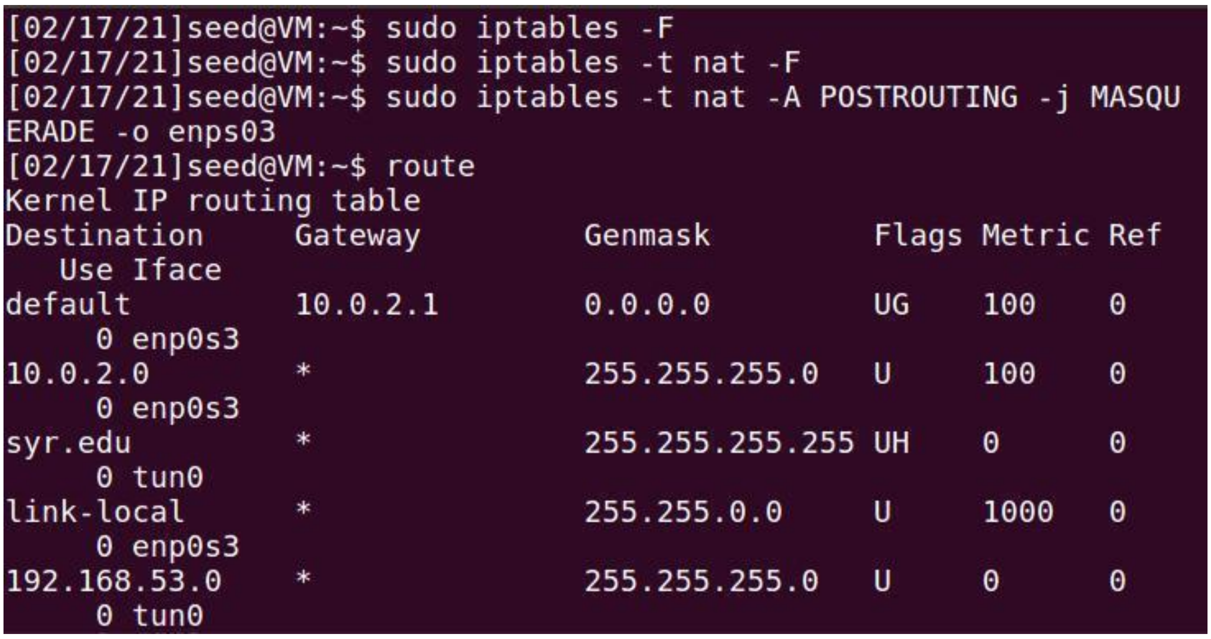
When the final destination sends packets back to users, the packet will be sent to the VPN Server first (think about why and write down your answer in the report). Answer:

The return packet will reach the VPN Server’s NAT adapter first (because the source IPs of all the outgoing packets from the Server VM are changed to the NAT’s external IP address (which is basically the host computer’s IP address in our setup). Usually, the NAT will replace the destination IP address with the IP address of the original packet (i.e. 192.168.53.5 in our case), and give it back to whoever owns the IP address.

To create another NAT right on the Server VM, so all packets coming out of the Server VM will have this VM’s IP address as their source IP. To reach the Internet, these packets will go through another

NAT, which is provided by VirtualBox, but since the source IP is the Server VM, this second NAT will have no problem relaying back the returned packets from the Internet to the Server VM.

- Executing given commands on server VM which will enable the NAT on the Server VM *sudo iptables -F*
  sudo iptables -t nat -F
  sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enps03

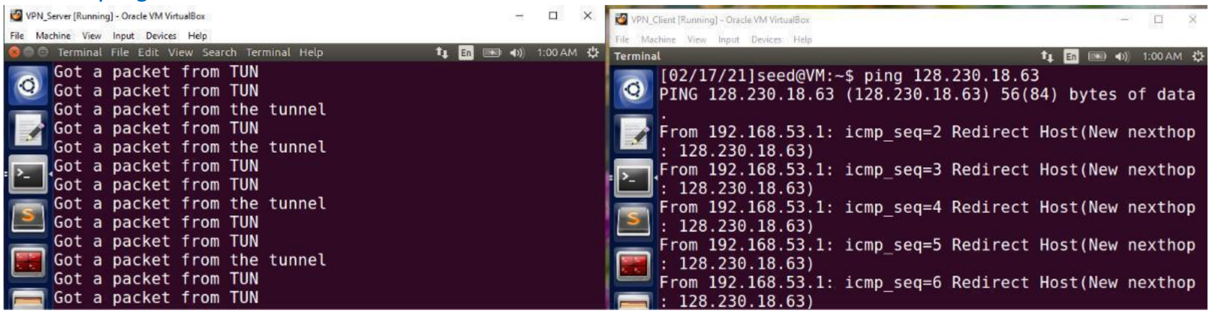


```
[02/17/21]seed@VM:~$ sudo iptables -F
[02/17/21]seed@VM:~$ sudo iptables -t nat -F
[02/17/21]seed@VM:~$ sudo iptables -t nat -A POSTROUTING -j MASQU
ERADE -o enps03
[02/17/21]seed@VM:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
   Use Iface
default         10.0.2.1        0.0.0.0         UG    100    0
     0 enp0s3
10.0.2.0        *               255.255.255.0   U     100    0
     0 enp0s3
syr.edu         *               255.255.255.255 UH    0      0
     0 tun0
link-local      *               255.255.0.0     U     1000   0
     0 enp0s3
192.168.53.0    *               255.255.255.0   U     0      0
     0 tun0
```

Observation:

We have successfullycleaned all the iptables rules and also added a rule on post-routing position to the NatNetwork adapter (tun0) connected to VPN server

- Validating using ping command from VPN client VM, whether we are able to bypass the firewall for the blocked website or not:
  ping 128.230.18.63



VPN_Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal File Edit View Search Terminal Help
1:00 AM

```
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
```

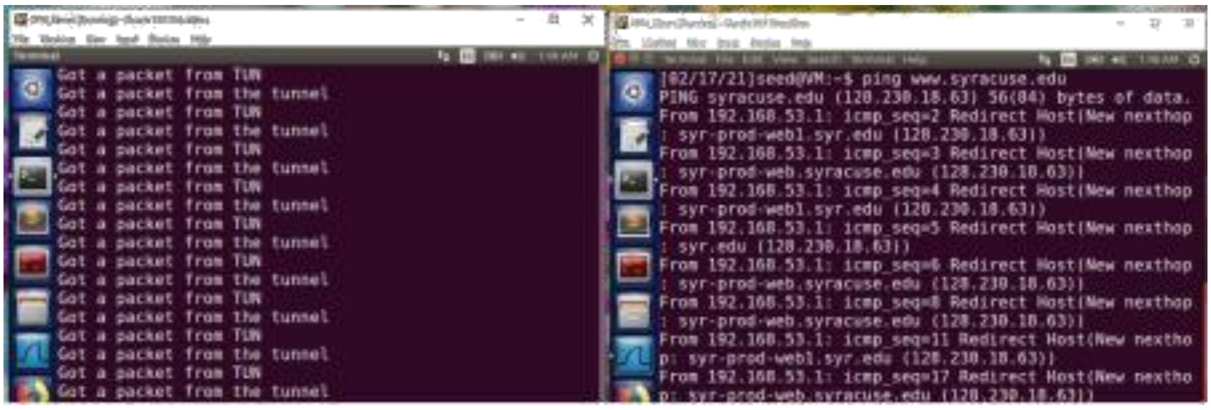
VPN_Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
1:00 AM

```
[02/17/21]seed@VM:~$ ping 128.230.18.63
PING 128.230.18.63 (128.230.18.63) 56(84) bytes of data
.
From 192.168.53.1: icmp_seq=2 Redirect Host(New nexthop
: 128.230.18.63)
From 192.168.53.1: icmp_seq=3 Redirect Host(New nexthop
: 128.230.18.63)
From 192.168.53.1: icmp_seq=4 Redirect Host(New nexthop
: 128.230.18.63)
From 192.168.53.1: icmp_seq=5 Redirect Host(New nexthop
: 128.230.18.63)
From 192.168.53.1: icmp_seq=6 Redirect Host(New nexthop
: 128.230.18.63)
```

Observation:

We are able to bypass the firewall and able to connect to the blocked website (128.230.18.63) as shown in above snapshot.

- Since, 128.230.18.63 doesn't run telnet server. So, we are unable to validate target website using telnet.
- But we have validated the website using *ping www.syracuse.edu* command and ith that also, we are able to connect the blocked website from VPN Client VM. Refer given snapshot:



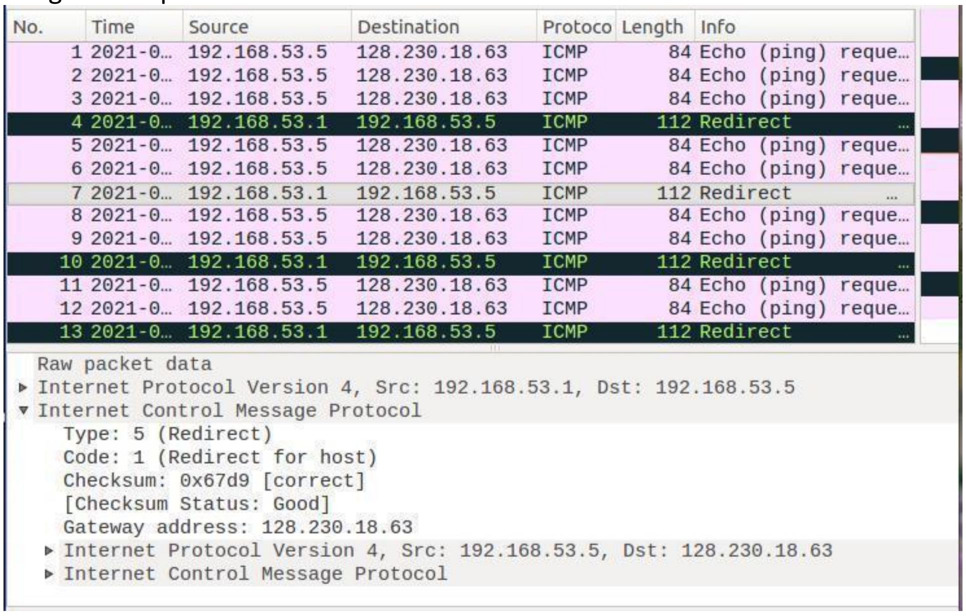
```
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
```

```
[02/17/21]seed@VM:~$ ping www.syracuse.edu
PING syracuse.edu (128.230.18.63) 56(84) bytes of data.
From 192.168.53.1: icmp_seq=2 Redirect Host(New nexthop
: syr-prod-web1.syr.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=3 Redirect Host(New nexthop
: syr-prod-web.syracuse.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=4 Redirect Host(New nexthop
: syr-prod-web1.syr.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=5 Redirect Host(New nexthop
: syr.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=6 Redirect Host(New nexthop
: syr-prod-web.syracuse.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=8 Redirect Host(New nexthop
: syr-prod-web.syracuse.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=11 Redirect Host(New nextho
p: syr-prod-web1.syr.edu (128.230.18.63))
From 192.168.53.1: icmp_seq=17 Redirect Host(New nextho
p: syr-prod-web.syracuse.edu (128.230.18.63))
```

Observation:

As the ping request from 192.168.53.1 goes to 128.230.18.63, then it shows "Got a packet from the tunnel" on the VPN server VM.

- Also, captured ping request packets in wireshark to validate the path of the packets. Refer given snapshot:



No.	Time	Source	Destination	Protoco	Length	Info
1	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
2	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
3	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
4	2021-0…	192.168.53.1	192.168.53.5	ICMP	112	Redirect …
5	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
6	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
7	2021-0…	192.168.53.1	192.168.53.5	ICMP	112	Redirect …
8	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
9	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
10	2021-0…	192.168.53.1	192.168.53.5	ICMP	112	Redirect …
11	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
12	2021-0…	192.168.53.5	128.230.18.63	ICMP	84	Echo (ping) reque…
13	2021-0…	192.168.53.1	192.168.53.5	ICMP	112	Redirect …

```
  Raw packet data
▸ Internet Protocol Version 4, Src: 192.168.53.1, Dst: 192.168.53.5
▾ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0x67d9 [correct]
    [Checksum Status: Good]
    Gateway address: 128.230.18.63
  ▸ Internet Protocol Version 4, Src: 192.168.53.5, Dst: 128.230.18.63
  ▸ Internet Control Message Protocol
```

Observation:

When we tried to access blocked website (128.230.18.63) from 192.168.53.5, the packet will be routed to the VPN tunnel and arrives at 192.168.53.1. Once it arrived, then 192.168.53.1 will route them to the final destination. When the reply packet comes back, it will come to 192.168.53.1 which will then redirect the packet to the VPN tunnel and eventually get the packet back to 192.168.53.5. That's how VPN helps 192.168.53.5 to bypass firewalls.

Conclusion:

The tunnelling technology is the most essential one to help bypass firewalls for protecting the content of the traffic that goes through the VPN tunnel. Hence, any blocked website (firewall rule active) can be accessible using VPN tunnel.

Note: The required code snippet has been attached on canvas.