INTERNET PROTOCOL LAB ASSIGNMNET -12

Name: Siriparapu Sparshika

Roll No: CYS22006

Task 1: VM Setup We need two machines, one inside the firewall, and the other outside the firewall. The objective is to help the machine inside the firewall to reach out to the external sites blocked by the firewall. We use two virtual machines, VM1 and VM2, for these two machines. VM1 and VM2 are supposed to be two machines connected via the Internet through routers. This setup may require more than two VMs. For the sake of simplicity, we use a LAN to emulate the Internet connection. Basically, we simply connect VM1 and VM2 to a LAN using the NAT Network adapter. Figure 1 depicts the lab setup.

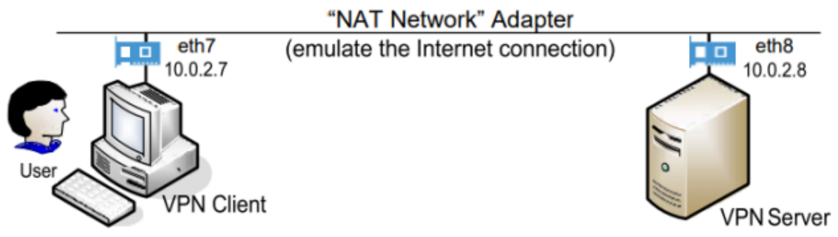
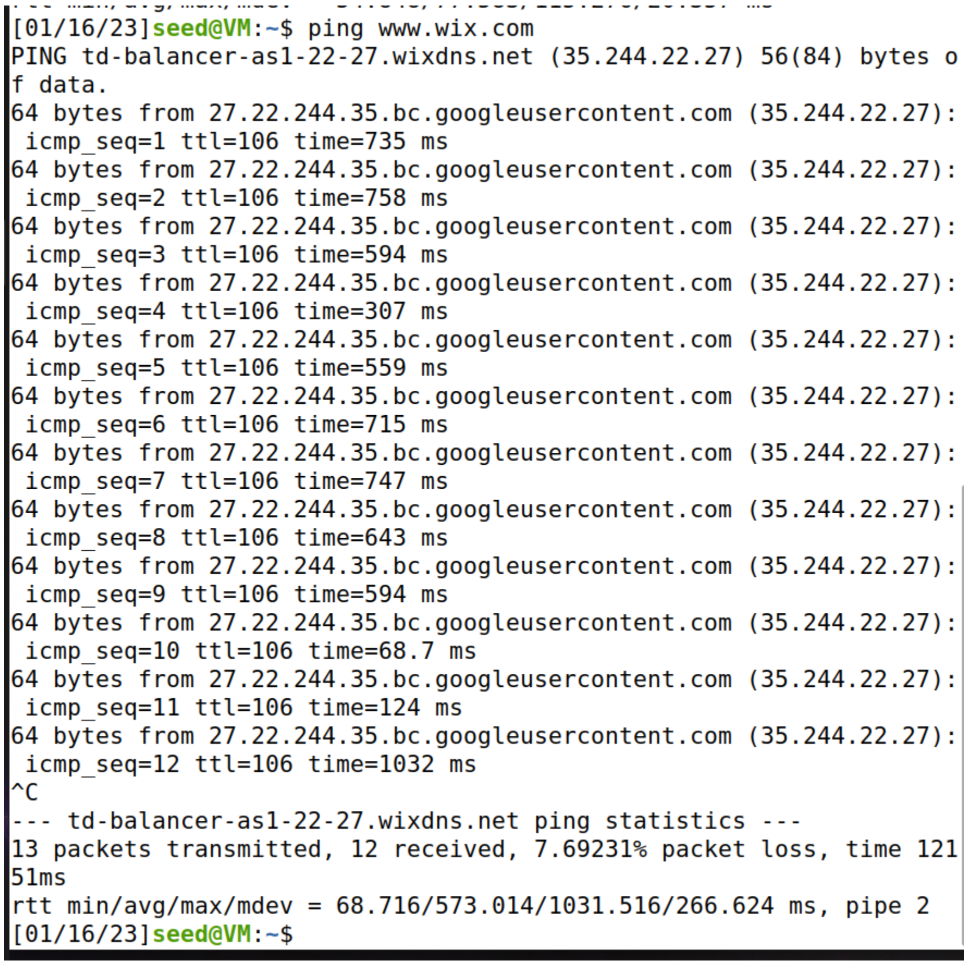


Figure 1: Lab Environment Setup

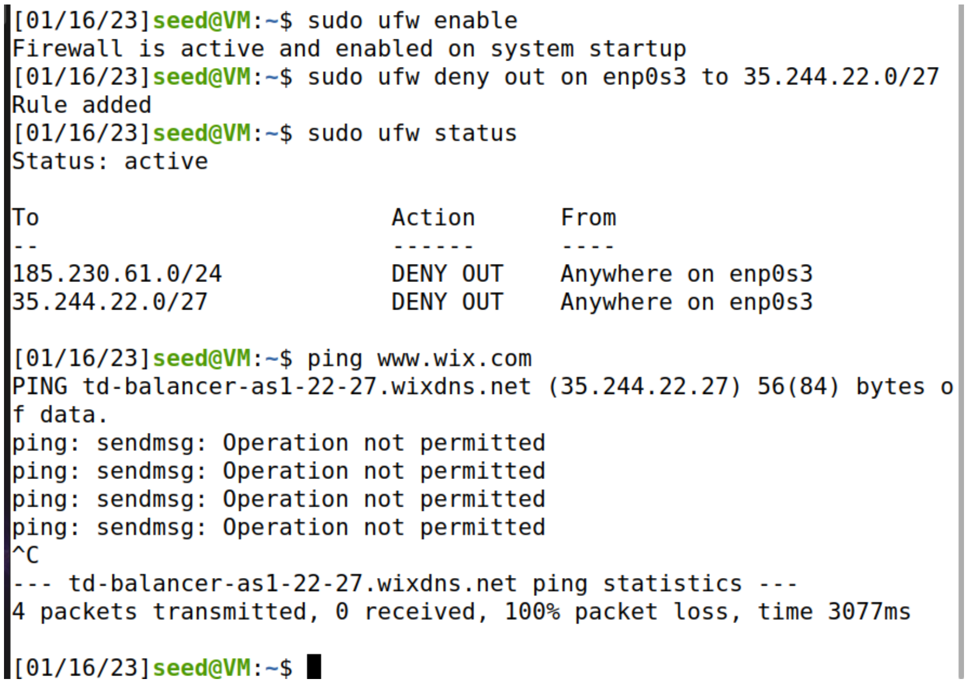
Task 2: Set up Firewall

In this task, you will set up a firewall on VM1 to block the access of a target website. You need to make sure that the IP address of the target web site is either fixed or in a fixed range; otherwise, you may have trouble completely blocking this website. Please refer to the Firewall Lab for details about how to blocking websites.

In the real world, the firewall should run on a separate machine, not on VM1. To minimize the number of VMs used in this lab, we put the firewall on VM1. Setting up the firewall on VM1 requires the superuser privilege, and so does the setup of the VPN tunnel. One may immediately say that if we already have the superuser privilege, why cannot we just simply disable the firewall on VM1. This is a good argument, but keep in mind, we put the firewall on VM1 simply because we do not want to create another VM in the lab environment. Therefore, although you have the superuser privilege on VM1, you are not allowed to use the privilege to reconfigure the firewall. You have to use VPN to bypass it. Compared to putting the firewall on an external machine, putting the firewall on VM1 does have a small issue that we need to deal with. When we set up the firewall to block packets, we need to make sure not to block the packets from getting to the virtual interface used by the VPN, or even our VPN will not be able to get the packets. Therefore, we cannot set the firewall rule before the routing, nor can we set the firewall rule on the virtual interface. We just need to set the rule on VM1's real network interface, so it will not affect the packets that go to the virtual interface



```
[01/16/23]seed@VM:~$ ping www.wix.com
PING td-balancer-as1-22-27.wixdns.net (35.244.22.27) 56(84) bytes o
f data.
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=1 ttl=106 time=735 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=2 ttl=106 time=758 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=3 ttl=106 time=594 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=4 ttl=106 time=307 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=5 ttl=106 time=559 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=6 ttl=106 time=715 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=7 ttl=106 time=747 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=8 ttl=106 time=643 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=9 ttl=106 time=594 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=10 ttl=106 time=68.7 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=11 ttl=106 time=124 ms
64 bytes from 27.22.244.35.bc.googleusercontent.com (35.244.22.27):
 icmp_seq=12 ttl=106 time=1032 ms
^C
--- td-balancer-as1-22-27.wixdns.net ping statistics ---
13 packets transmitted, 12 received, 7.69231% packet loss, time 121
51ms
rtt min/avg/max/mdev = 68.716/573.014/1031.516/266.624 ms, pipe 2
[01/16/23]seed@VM:~$
```



```
[01/16/23]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[01/16/23]seed@VM:~$ sudo ufw deny out on enp0s3 to 35.244.22.0/27
Rule added
[01/16/23]seed@VM:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
185.230.61.0/24            DENY OUT    Anywhere on enp0s3
35.244.22.0/27             DENY OUT    Anywhere on enp0s3

[01/16/23]seed@VM:~$ ping www.wix.com
PING td-balancer-as1-22-27.wixdns.net (35.244.22.27) 56(84) bytes o
f data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- td-balancer-as1-22-27.wixdns.net ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3077ms

[01/16/23]seed@VM:~$
```

Task 3: Bypassing Firewall using VPN

The idea of using VPN to bypass firewall is depicted in Figure 2. 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.

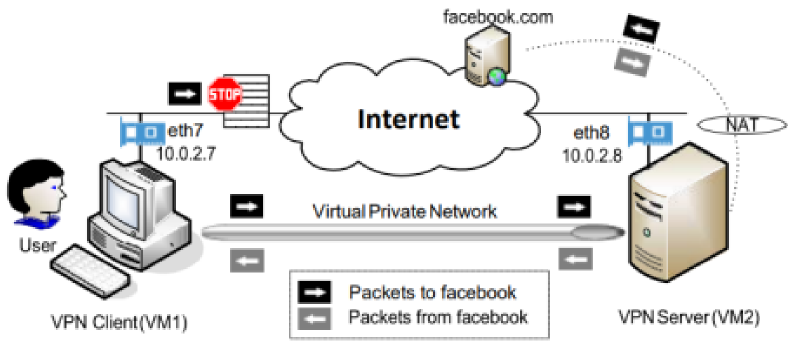


Figure 2: Bypassing firewall using VPN

We have created a sample VPN program, including a client program (vpnclient) and a server program (vpnserver), both of which can be downloaded from Microsoft Teams. This simple VPN program only establishes a VPN tunnel between the client and server; it does not encrypt the tunnel traffic. The program is explained in detail in Chapter 16 of the SEED book titled Computer Security: A Hands-on Approach; the chapter also explains how TUN/TAP works and how to use it to create VPN. The vpnclient and vpnserver programs are the two ends of a VPN tunnel. They communicate with each other using either TCP or UDP via the sockets depicted in Figure 3. In our sample code, we choose to use UDP for the sake of simplicity. The dotted line between the client and server depicts the path for the VPN tunnel. The VPN client and server programs connect to the hosting system via a TUN interface, through which they do two things: (1) get IP packets from the hosting system, so the packets can be sent through the tunnel, (2) get IP packets from the tunnel, and then forward it to the hosting system, which will forward the packet to its final destination. The following procedure describes how to create a VPN tunnel using the vpnclient and vpnserver programs.

We have created a sample VPN program, including a client program (vpnclient) and a server program (vpnserver), both of which can be downloaded from Microsoft Teams. This simple VPN program only establishes a VPN tunnel between the client and server; it does not encrypt the tunnel traffic. The program is explained in detail in Chapter 16 of the SEED book titled Computer Security: A Hands-on Approach; the chapter also explains how TUN/TAP works and how to use it to create VPN. The vpnclient and vpnserver programs are the two ends of a VPN tunnel. They communicate with each other using either TCP or UDP via the sockets depicted in Figure 3. In our sample code, we choose to use UDP for the sake of simplicity. The dotted line between the client and server depicts the path for the VPN tunnel. The VPN client and server programs connect to the hosting system via a TUN interface, through which

they do two things: (1) get IP packets from the hosting system, so the packets can be sent through the tunnel, (2) get IP packets from the tunnel, and then forward it to the hosting system, which will forward the packet to its final destination. The following procedure describes how to create a VPN tunnel using the vpnclient and vpnserver programs.

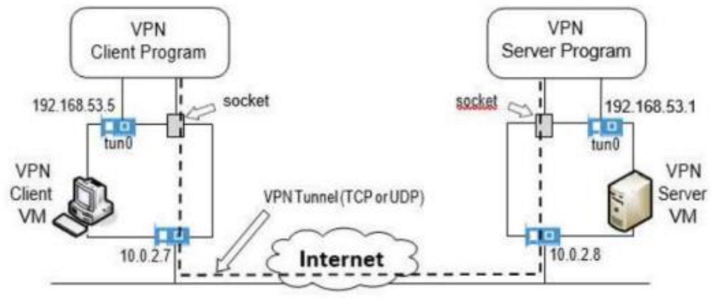
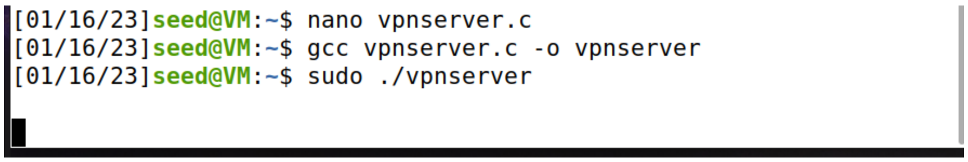


Figure 3: VPN client and server

Run the following commands. The first command will start the server program, and the second command assigns an IP address to the tun0 interface and then activates it. It should be noted that the first command will block and wait for connections, so we need to find another window run the second comman.



```
[01/16/23]seed@VM:~$ nano vpnserver.c
[01/16/23]seed@VM:~$ gcc vpnserver.c -o vpnserver
[01/16/23]seed@VM:~$ sudo ./vpnserver
```