Virtual Private Network Lab

Table of Contents

Overview

A Virtual Private Network (VPN) is used for creating a private scope of computer communications or providing a secure extension of a private network into an insecure network such as the Internet. VPN is a widely used security technology. VPN can be built upon IPSec or TLS/SSL (Transport Layer Security/Secure Socket Layer). These are two fundamentally different approaches for building VPNs. In this lab, we focus on the TLS/SSL-based VPNs. This type of VPNs is often referred to as TLS/SSL VPNs.

The learning objective of this lab is for students to master the network and security technologies underlying VPNs. To achieve this goal, students will be asked to implement a simple TLS/SSL VPN. Although this VPN is simple, it does include all the essential elements of a VPN. The design and implementation of TLS/SSL VPNs exemplify a number of security principles, including the following:

- Virtual Private Network
- TUN/TAP, and IP tunnelling
- Routing
- Public-key cryptography, PKI, and X.509 certificate
- TLS/SSL programming
- Authentication

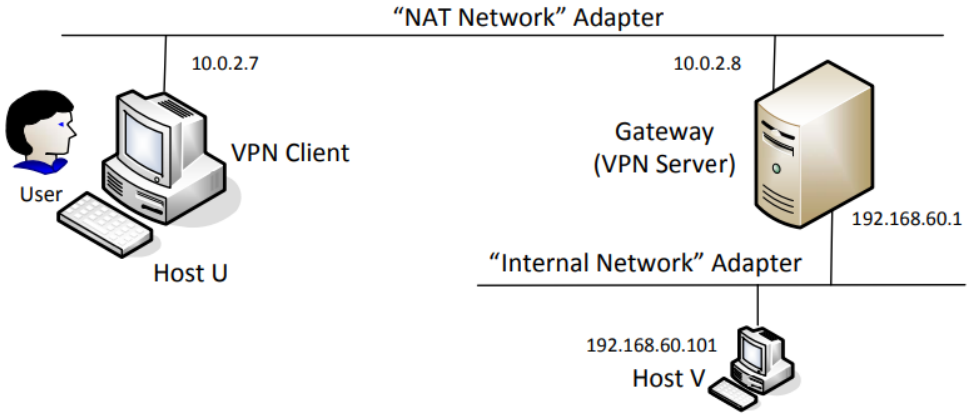
Note: We will be using openssl package in this lab. This lab has been tested on the prebuilt Ubuntu 16.04 VM.

Tasks

We will implement a simple VPN for Linux, called *miniVPN.*

Task 1: VM Setup

We will create a VPN tunnel between a computer (client) and a gateway, allowing the computer to securely access a private network via the gateway. We need at least three VMs: VPN client (also serving as Host U), VPN server (the gateway), and a host in the private network (Host V). The network setup is depicted in Figure below.



In practice, the VPN client and VPN server are connected via the Internet. For the sake of simplicity, we directly connect these two machines to the same LAN in this lab, i.e., this LAN simulates the Internet. We will use the "NAT Network" adaptor for this LAN. The third machine, Host V, is a computer inside the private network. Users on Host U (outside of the private network) want to communicate with Host V via the VPN tunnel. To simulate this setup, we connect Host V to VPN Server (also serving as a gateway) via an "Internal Network". In such a setup, Host V is not directly accessible from the Internet; nor is it directly accessible from Host U.

Before starting the VMs, in settings of the VM in Virtual Box, under the Network Tab, make sure the following are enabled:

VPN client – Adapter 1 – NAT Network

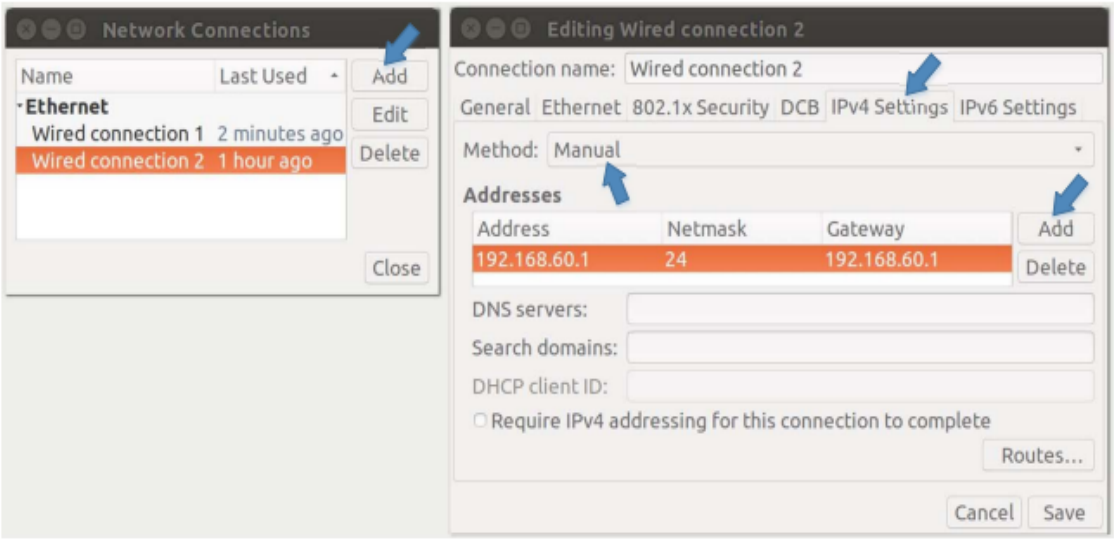
VPN Server – Adapter 1 – NAT network, Adapter 2 – Internal Network

HOST V – Adapter 1 – Internal Network

Note if a VM uses the "Internal Network" mode, VirtualBox provides no DHCP to it, so the VM must be statically configured. To do this, click the network icon on the top-right corner of the desktop, and select "Edit Connections". You will see a list of "Wired connections", one for each of the network adaptors used by the VM. For Host V, there is only one connection, but for VPN Server, we will see two. To make sure that you pick the one that is corresponding to the "Internal Network" adapter, you can check the MAC address displayed in the pop-up window after you have picked a connection to edit. Compare this MAC address with the one that you get from ifconfig, and you will know whether you picked the right connection.

After you have selected the right connection to edit, pick the "ipv4 Settings" tab and select the "Manual" method, instead of the default "Automatic (DHCP)". Click the "Add" button to set up the new IP address for the VM.

Setup all the 3 VMs according to the diagram shown above and make the connections in each VM according to the image shown below as an example.



Task 2: Creating a VPN Tunnel using TUN/TAP

Sample VPN client program (vpnclient) and a server program (vpnserver), both of which can be downloaded from this lab's web site.

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.

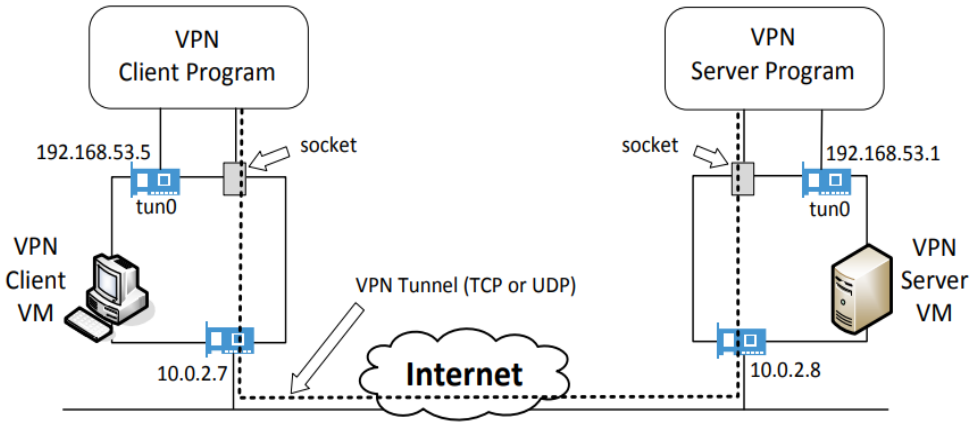


Figure 3: VPN client and server

Step 1: Run VPN server and set it's IP address of the interface – (Run on VPNServer VM)

Commands:

make

sudo ./vpnserver

Open another terminal

sudo ifconfig tun0 192.168.53.1/24 up

sudo sysctl net.ipv4.ip_forward=1

After performing this, run *ifconfig* command

Explain your observations and show screenshots of the above commands

Step 2: Run VPN Client and set IP address of the interface - (Run on VPNClient VM)

Make the following change to the functions in sample code of vpnclient.c

```
int main (int argc, char * argv[]) {
  int tunfd, sockfd;
  char* serverip;

  if(argc > 1) serverip = argv[1];

  tunfd  = createTunDevice();
  sockfd = connectToUDPServer(serverip);

  // 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);
  }
}
int connectToUDPServer(char* serverip){
   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(serverip);

   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;
}
```

Commands:

make

sudo ./vpnclient 10.0.2.8

The ip address is the IP address of your VPN server

Open another terminal and run:

sudo ifconfig tun0 192.168.53.5/24 up

Run ifconfig command

Explain your observations with screenshots of terminals in Client and Server VMs.

Step 3: Set up routing on Client and Server VMs

On VPN Client VM,

Run the following commands,

sudo route add -net 192.168.53.0/24 tun0

route -n

sudo route add -net 192.168.60.0/24 tun0

route -n

Show your observations and screenshots

On VPN server VM,

sudo route add -net 192.168.53.0/24 tun0

route -n

Show your observations and screenshots

Step 4: Set up routing on HOST V

On Host V,

sudo route add -net 10.0.2.0/24 enp0s3

route -n

Show your observations and screenshots

Step 5: Test the VPN tunnel (ping and telnet)

On the VPN Client VM,

Keep wireshark ready

ping 192.168.60.101

Show the screenshots and observations of the Natnetwork communication (UDP) which is the Tunnel traffic and show the ICMP communication

Next task is to perform telnet, keep wireshark ready

telnet 192.168.60.101

Show your wireshark screenshots. Show which packets are tunnel traffic and which aren't.

Run 'ls' command, create a folder on HOST V and then run 'ls' command on the telnet connection, you should notice that the new folder create is visible.

Step 6: Tunnel-Breaking Test

Break the VPN tunnel connection, by disconnecting the vpnserver program.

Continue to type 'ls' command on the telnet Window

Explain your observations, show wireshark screenshot.

Now, reconnect the VPN tunnel/ server and explain your observations

Show screenshots on the Server and Client side along with Wireshark captures.

Submission

You should submit a detailed lab report to describe your design and implementation. Ensure proper screenshots are provided. Explain each observation and provide justification as well

References

Computer & Internet Security (handsonsecurity.net)

SEED Project (seedsecuritylabs.org)