

Observational Report LAB 6: VPN LAB

ACS 545| Cryptography and Network Security

February 28, 2024

Ashutosh Mishra

Professor: ZesHeng Chen

* Observation Criteria: -

Note: -

1. Used Ubuntu Seed (v20.04 Focal) on Google Cloud Platform
2. Used Real VNC Viewer to perform the actions on GUI
3. All actions are being performed under username - **seed**

* Environment Setup: -

A folder named Crypto was created and the Labsetup file was downloaded from the Seed Documentation site. Post unzipping the folder we find a docker-compose.yml file which contains the configurations for three machines that are connected to the same LAN. The four machines are: -

1. Server-router
2. host-192.168.60.5
3. host-192.168.60.6
4. client-10.9.0.5

To create the four containers using the .yml file, firstly one can use either **docker-compose build** or **dcbuild** (alias).

A screenshot of a computer

Description automatically generated

Now we will use either the **docker-compose up** or **dcup** (alias) to start the containers.

A screenshot of a computer

Description automatically generated

Now, as per the above screenshot, we can see that the containers are up. However, just to cross check we can use either **docker ps –format “{{.ID}} {{.Names}}”** or **dockps** (alias) to show the status of the containers.

A screenshot of a computer

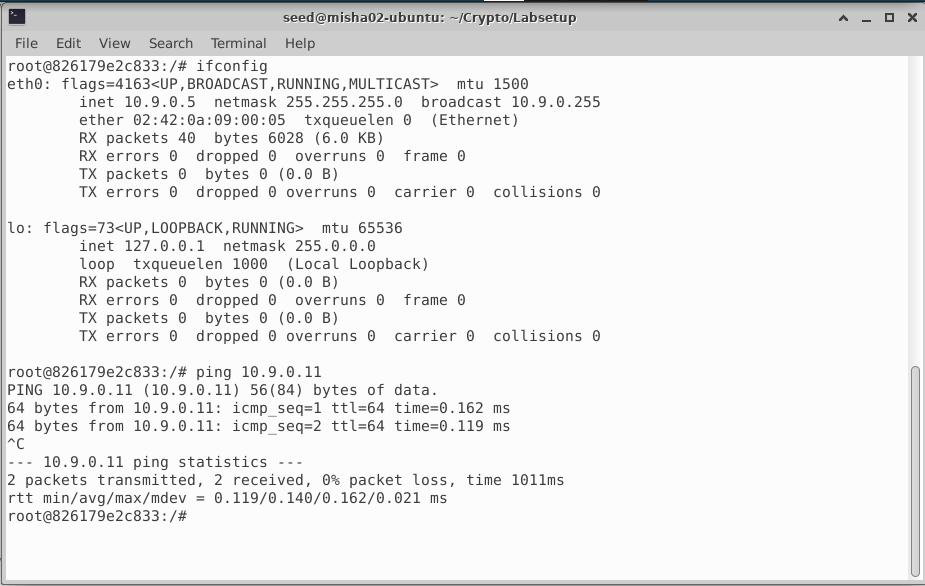
Description automatically generated

* Task 1 (Network Setup): -

The setup steps are shown above. Now, we will be performing various tests as mentioned in the task.

1. Host U can communicate with VPN Server.

**Ping to VPN Server from Host U**

****

**Ping to Host U from VPN Server**



1. VPN Server can communicate with Host V.

**Ping to VPN Server from Host V**

**A screenshot of a computer

Description automatically generated**

**Ping to Host V from VPN Server**

**A screenshot of a computer

Description automatically generated**

1. Host U should not be able to communicate with Host V.

A screenshot of a computer

Description automatically generated

1. Run tcpdump on the router and sniff the traffic on each of the network.

**Capture between V and Router**

**A screenshot of a computer

Description automatically generated**

**Capture between U and Router**

**A screenshot of a computer

Description automatically generated**

* Task 2 Create and Configure TUN Interface: -
  + Task 2A (Name of Interface): -

For the first task, I have referenced the code from seedLabs site. The requirement is to refer to the same steps mentioned in the handout. Firstly, the code file is as below: -

A screenshot of a computer

Description automatically generated

The very first step for this lab is to run the code and observe the changes in ip address on client U.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now, trying to change the interface name from tun0 to MISHRA. Code changes are as below.

A screenshot of a computer

Description automatically generated

Now executing the code and seeing the change

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* + Task 2B (Setup the TUN Interface): -

For part B task, I have reused the same code provided in the Lab Setup file and altered it slightly as mentioned below.

A screenshot of a computer

Description automatically generated

Now, running the script and observing changes in output.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We can observe that the port MISHRA0 is now up and ip is assigned to it.

* + Task 2C (Read from TUN Interface): -

For part C task, I have reused the same C code provided in the Lab Setup file and altered it slightly as mentioned below.

A screenshot of a computer

Description automatically generated

Executing the code and observing following: -

A screenshot of a computer

Description automatically generated

1. **On Host U, ping a host in the 192.168.53.0/24 network. What are printed out by the tun.py program? What has happened? Why?A screenshot of a computer

   Description automatically generated**

**A screenshot of a computer

Description automatically generated**

The main observation here is that tun could receive the data packet but couldn’t return it.

1. **On Host U, ping a host in the internal network 192.168.60.0/24, Does tun.py print out anything? Why?**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

The main observation is that ping is successful, however, tun is not receiving any.

* + Task 2D (Write to TUN Interface): -

For part D task, I have reused the same code provided in the Lab Setup file and altered it slightly as mentioned below.

A screenshot of a computer

Description automatically generated

Executing the code

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The ip is written.

Even for arbitrary message, the message doesn’t pop up.

* Task3 (Send IP Packets To VPN Server Through a Tunnel): -

We first create a tun\_server.py and tun\_client.py as mentioned below:-

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Now, we will run tun\_client.py in U and run tun\_server.py in Server.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Let’s ping 192.168.53.0/24 and see output.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now, we will ping an ip from 192.168.60.0/24

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Task 4(Setup VPN Server): -

In this task the tun\_server.py file is slightly modified, and the other one is as is. Please find a screenshot of both the codes below.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Now we run the client script on U, perform tcpdump on V and run the server script on server.

On client

A screenshot of a computer

Description automatically generated

On Server

A screenshot of a computer

Description automatically generated

Ping request on client

A screenshot of a computer

Description automatically generated

Response on Server

A screenshot of a computer

Description automatically generated

* Task 5(Handling Traffic in Both Directions): -

For this task the code files are as below.

Client

A screenshot of a computer program

Description automatically generated

Server

A screenshot of a computer program

Description automatically generated

Let’s run the client script on U. and the server script on server.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now, we will ping one of host V’s and check if it is pingable.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

It is pingable. Below is the wireshark screenshot.

A screenshot of a computer

Description automatically generated

Now, let’s try telnet.

A screenshot of a computer

Description automatically generated

Telnet was successful. Please find attached screenshot of wireshark for telnet.

A screenshot of a computer

Description automatically generated

**Host U sends a ping request to VPN server for ping, then VPN server sends ping request to Host V. Host V replies to the request to VPN Server and then VPN server passes the request to Host U.**

* Task 6(Tunnel Breaking Experiment): -

Let’s telnet to host V from host U.

A screenshot of a computer

Description automatically generated

Telnet is successful.

Let’s stop the tun\_client.py script and observe the changes.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The shell becomes unresponsive. Let’s try running back the script.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Initially, things were as it was, then suddenly as soon as we type, all the missed commands are sent. The TCP connection was broken, and input was stored in a buffer. Post first key enter, the buffer was released, and connection was restored as soon as the script was retriggered.

* Task 7(Routing Experiment on Host V): -

Let’s view the IP Route.

A screenshot of a computer

Description automatically generated

As we can see that the route is via the Server. Let’s delete the default route and add a new route.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The ping is successful.

A screenshot of a computer

Description automatically generated

* Conclusion: -

The overall experience was exciting. I will be going deeper into the subject and try to discover new ways to handle how we had practiced in lab. Having a base knowledge of how VPN works, I will like to do a deep dive in this subject to find out more about how to protect the target machines from attacks that occur in a simple manner. One thing to think of is can we use the concepts of this to create free VPN services to connect to other parts of the world. Also, I will like to go deep into how decentralized VPN works.