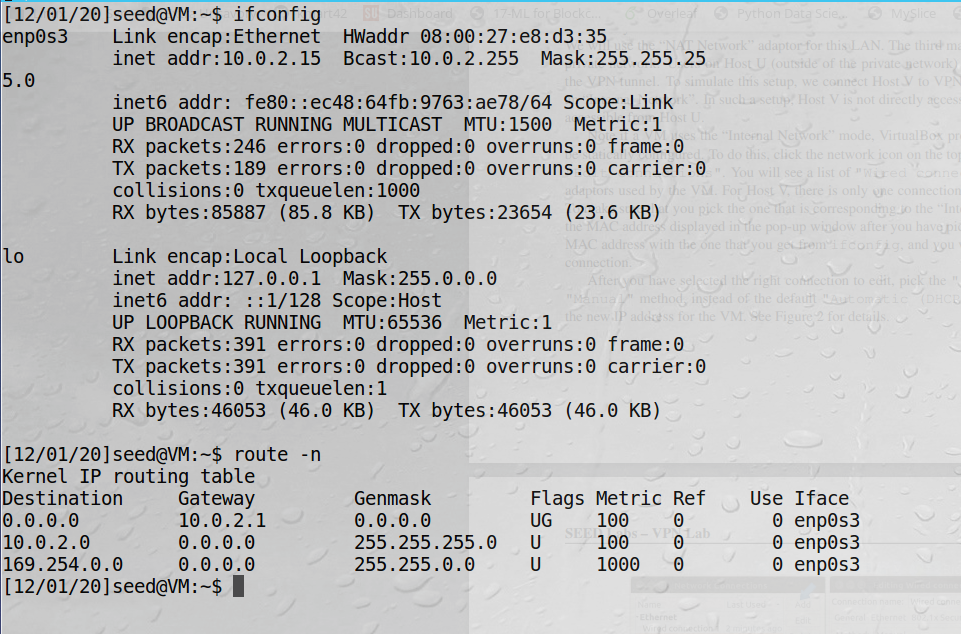
Lab 7 – Kevin Martin

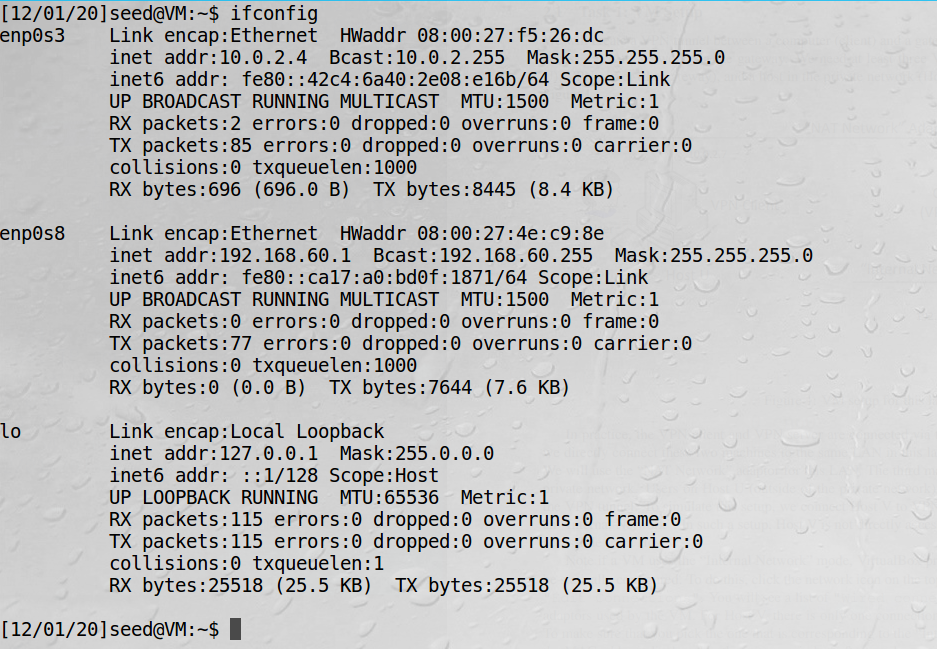
**Task 1 – VM Setup**

To set up the VPN, first we have to configure the three machines. Host U will be VM A, the VPN Server/Gateway will be VM C, and the destination/Host V will be VM C.

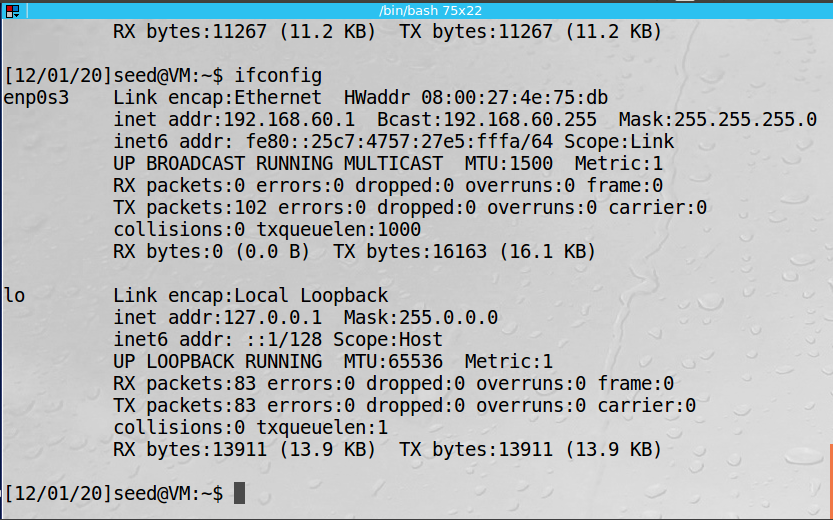
Host U (VM A), the NAT Network and IP 10.0.2.15:



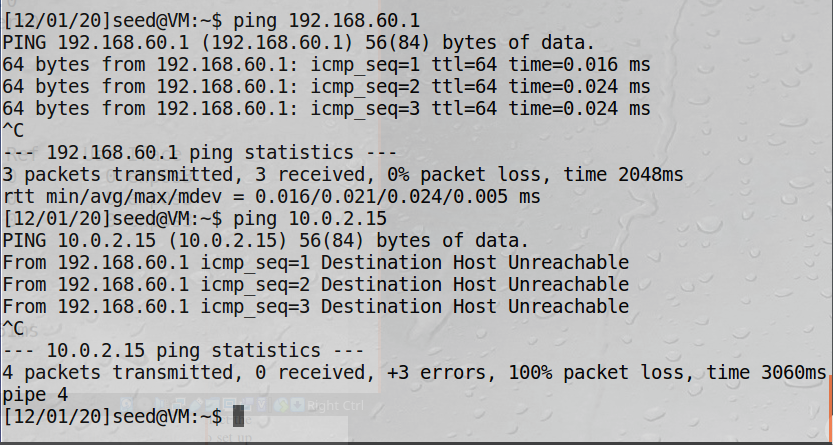
VPN Server (VM B) needs both the shared ethernet network as well as the statically configured connection. This is done through the ip4 settings of the VM. However, the **internal network** had to be configured outside of the VM first, then it was recognized in the booted VM.Once correctly added, the output is can be seen in the **ifconfig** screenshot:



Note the VM’s IP address as 10.0.2.4, as it has always been, and the new connection at 192.168.60.1.  
  
Finally, Host V (VM C) is configured to ONLY be on the internal network. Again, this was first done outside the VM, and is now recognized inside the booted instance via **ifconfig**:



From Host V (VM C), we can only ping the VPN server (VM B) but not Host U (VM A). Before modifying the connections, Host V would have been reachable:

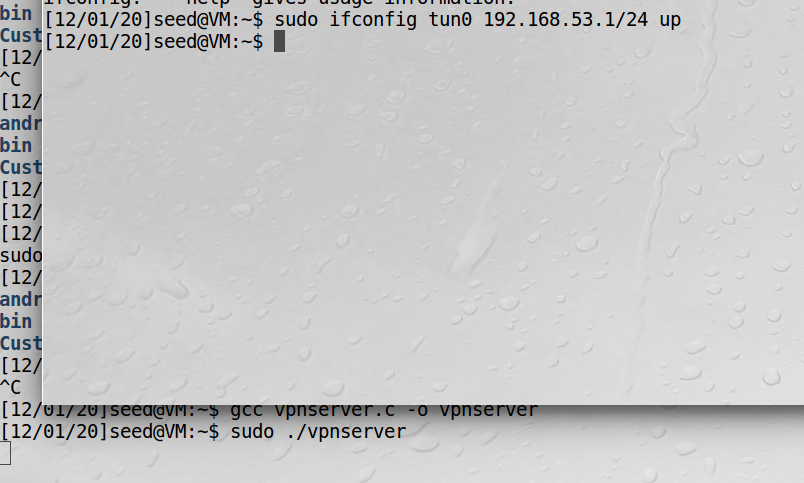


**Observation:** We set up a secondary “private” connection between the VPN Server and Host V. This network is not accessible for Host U. The VPN Server has connections to both the normal NAT Network as well as this newly established private network. Host V and Host U are, at the moment, unable to communicate with each other.

**Explanation:** The new private network allows connection between VPN Server and Host V to simulate a private network for something like a company’s internal network. Host U is simulating the user and will try to connect to V using the VPN tunnel.

**Task 2 – Creating a VPN Tunnel using TUN/TAP**

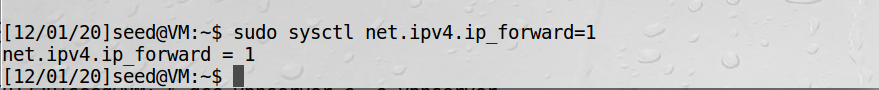
To create a VPN tunnel, we will use the **vpnserver** program provided. First, we will add the IP address in the VPN Server (VM B) and compile the program **vpn\_server.c**, and the **vpnserver** program can be ran. Finally it is confirmed using **ifconfig**:



We can also see the tunnel (tun0) from a standard **ifconfig**:

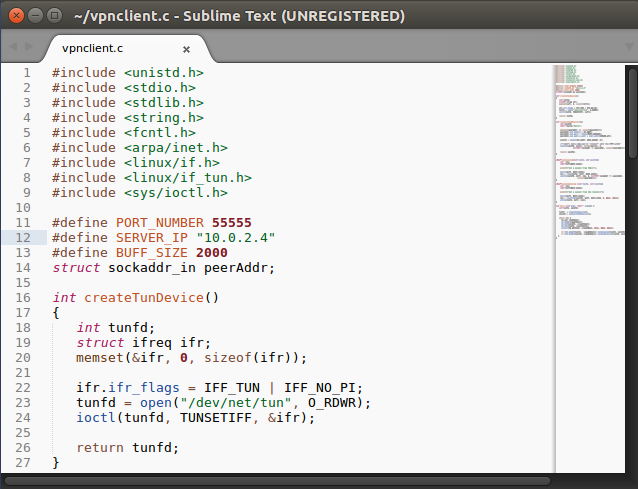


To complete the setup of the VPN server, we must able forwarding so the server will act as a gateway. As it is currently configured, it is only a host. The following **sysctl** command will fix:

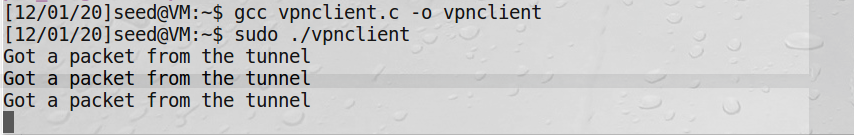


**Step 2: Run the VPN Client**

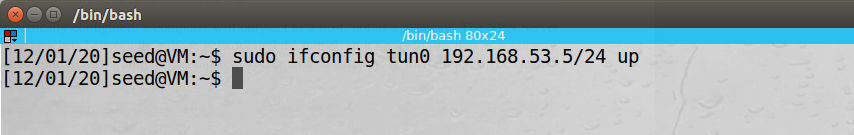
In order to run the VPN client, we will use the provided **vpnclient.c** program on Host U (VM A). We will set the server IP to the VPN Server (VM B):



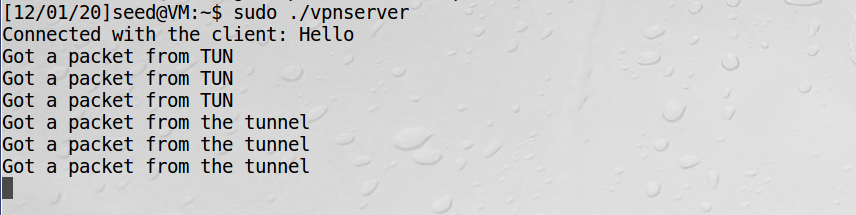
The **vpnclient** running:

****

Next we configure the tun0 interface to the previously configured IP address at the VPN Server:

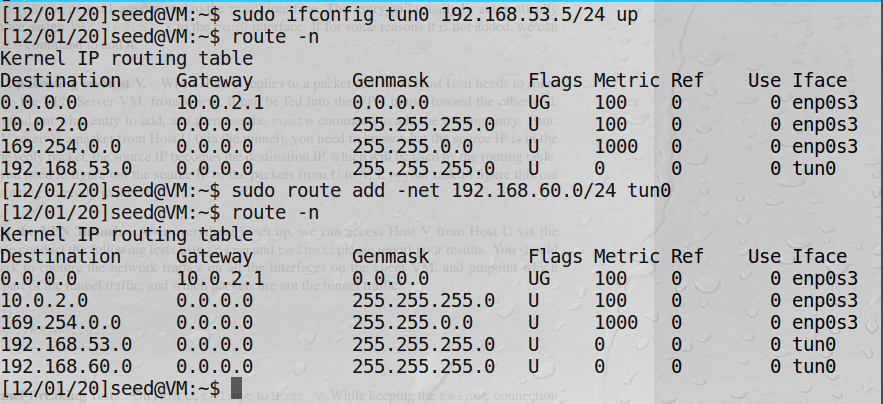
****

Back on the VPN Server, we can see a successful connection:

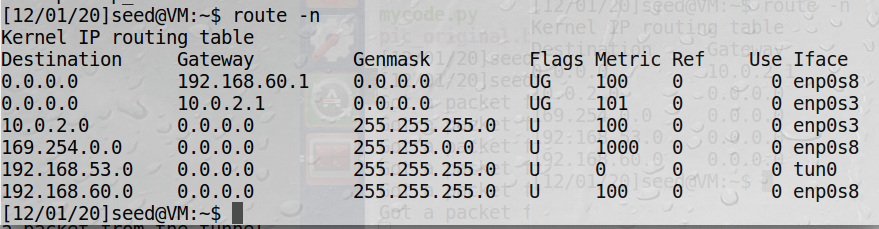


**Step 3: Routing on the Client and Server VMs**

Now that the connection has been established, we must direct traffic to the tunnel or else it is useless. On Host U, we use the **route** command to establish this protocol. Note the before and after output of running **route -n**, the later showing the last entry of the newly established tunnel:

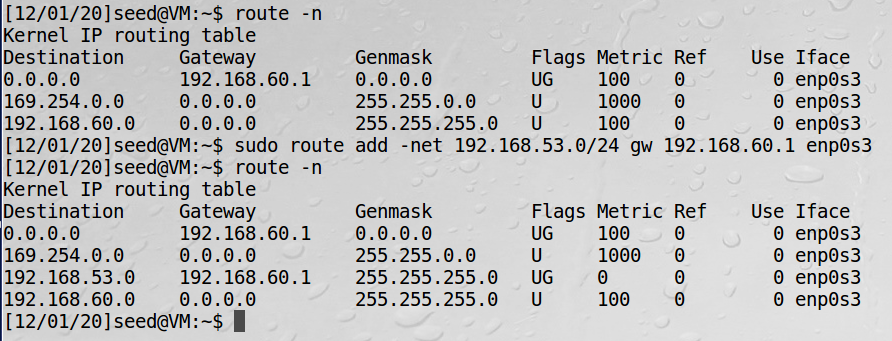
****

Back on the VPN Server, we can see that this route is recognized:

****

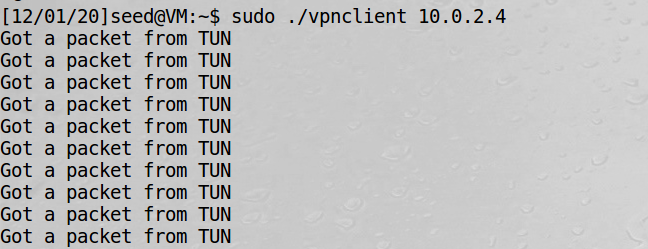
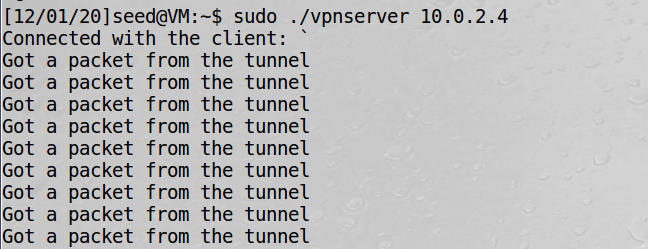
**Step 4: Set up Routing Host V**

To allow for packets to be sent back to Host U from Host V, we will need to establish this route via the **route** command on VM C, specifiying our new network:

****

**Step 5: Test the VPN**

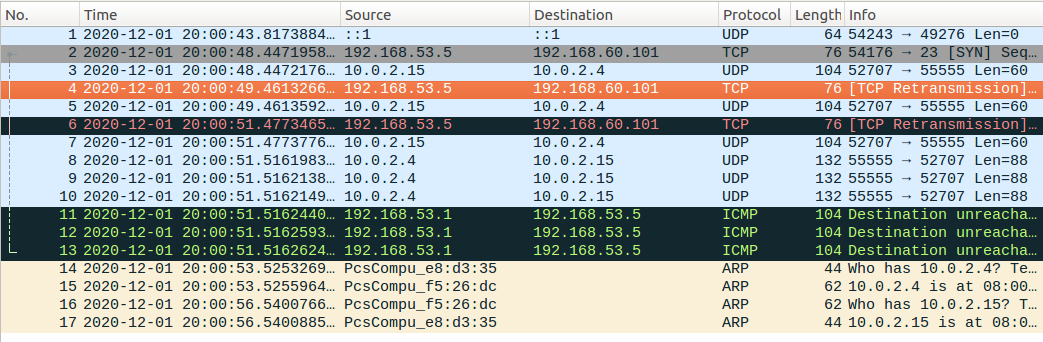
With everything in place and the routes established, we can now reach Host V from Host U by first using the **ping** command. Unfortunately I was not able to successfully transfer packets, however I did see the message “Got a packet from the tunnel” on both Host U and the VPN Server:

****

So the tunnel is working, but the fully established route does not seem to be. Telnet was also not able to connect.

**Step 6: Tunnel-Breaking Test**

Unfortunately I was not able to establish a proper Telnet connection. However, using Wireshark, we can see the same effect as if there had been a connection which was then broken. The output from Host V:



The source and destination appear to be correct, and we can see that output is similar to the expected broken Telnet output.

**Observation:** In order to properly connect to Host U and make use of the private network established between VPN Server and Host U, we must properly configure the entire route. That includes specifying the types of the connections and the IP address (actual) of the VPN Server, as well as the newly defined tunnel IP Addresses.

**Explanation:** Unfortunatley the final path did not work, but we were able to see the conneection attempted. All three VM’s appeared to be set up correctly and working as intended. The expectation was that a ping from VM A would reach VM C through VM B, but that did not quite happen.