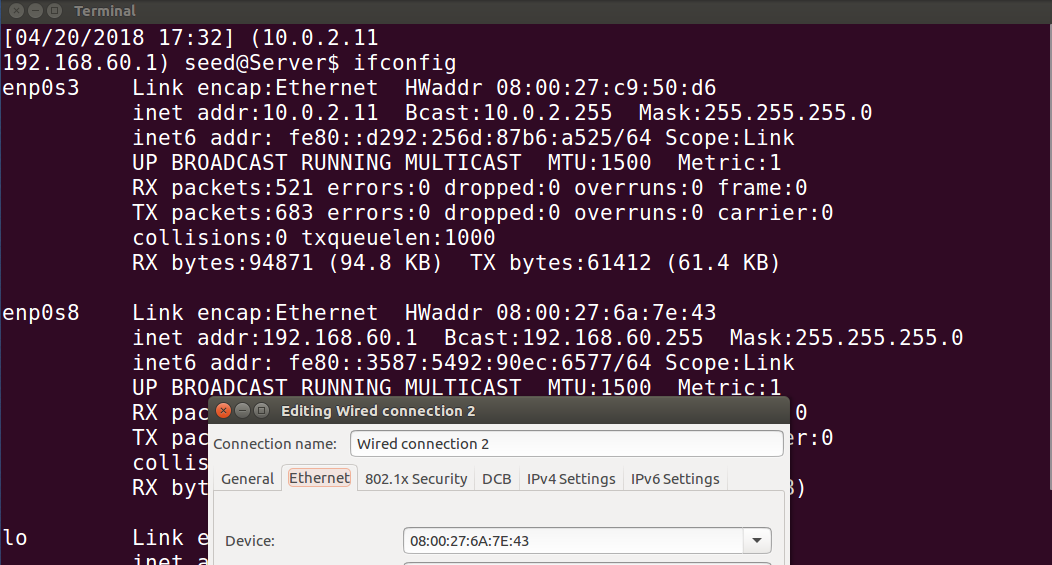
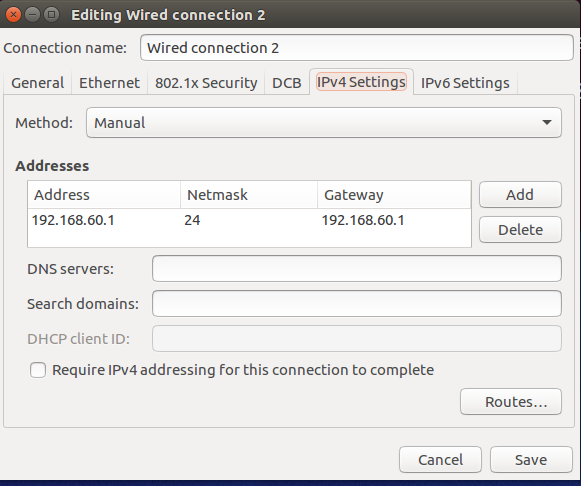
**Virtual Private Network (VPN) Lab**

**Karan Amrutesh**

**Task 1: VM Setup:**

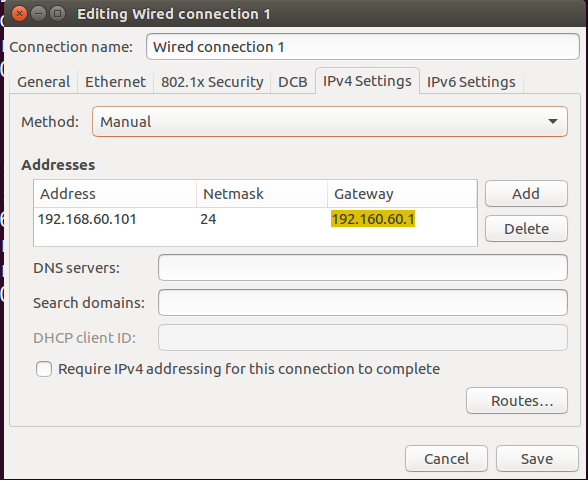
* VPN Server: We manually set up the IP address for the "Internal Network" adaptor on VPN Server:





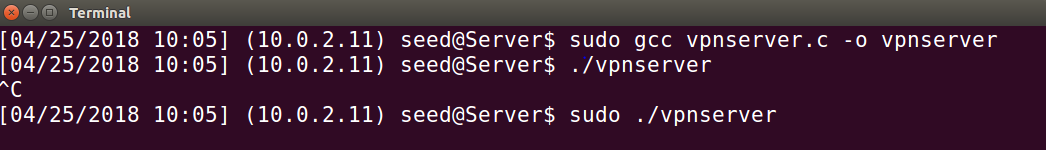
* Host V: We set up the Host V by giving the gateway as the server’s IP, both of which are connected through internal network by device “enp0s3



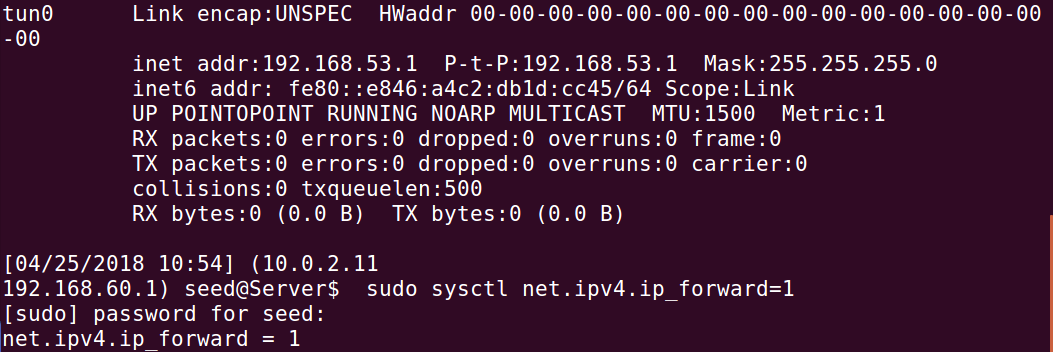


**Task 2: Creating a VPN Tunnel using TUN/TAP**

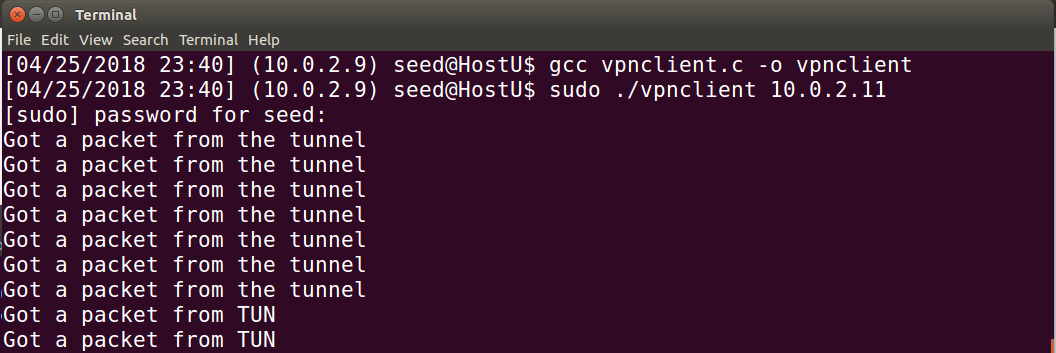
* Step 1 Run VPN Server:



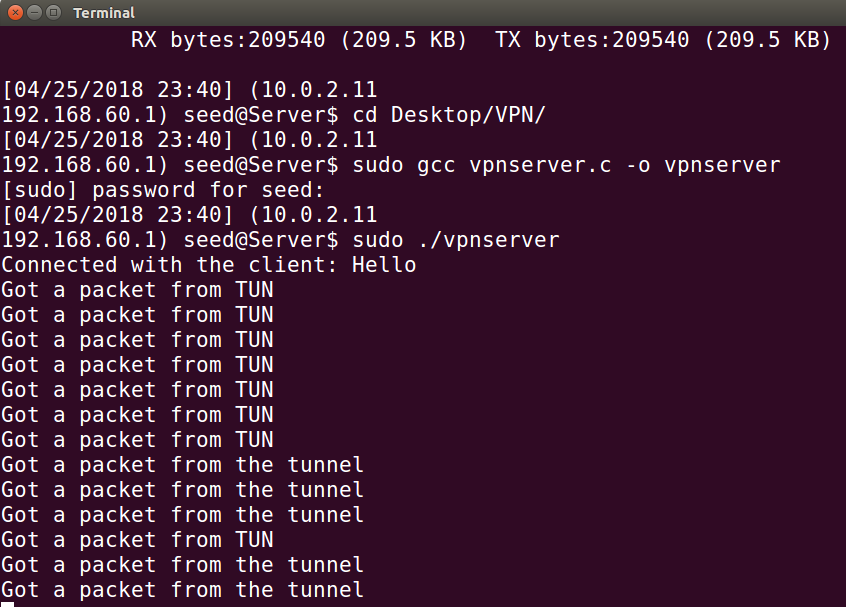
* Assigning IP address 192.168.53.1 to the tun0 device and bringing it up.
* We also enable the IP forwarding on the VPN server so that it behaves like a gateway:



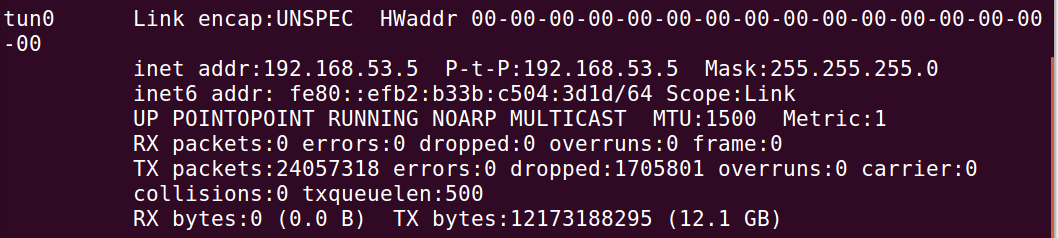
* Step 2: Running the VPN Client:



* The status of the server when the client connects to the tunnel:

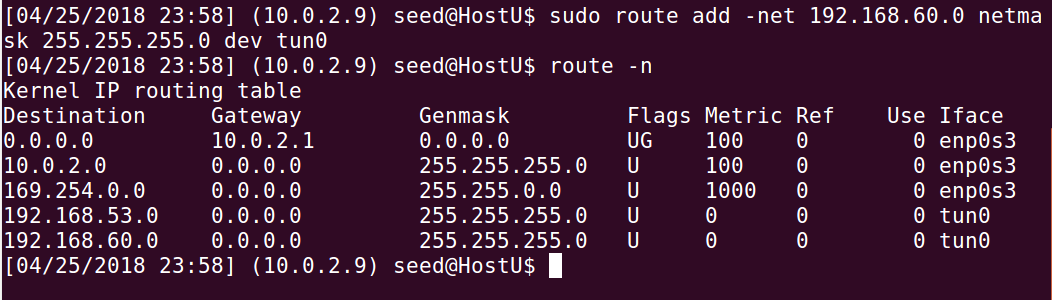


* Assigning IP address 192.168.53.5 to the tun0 device and bringing it up.

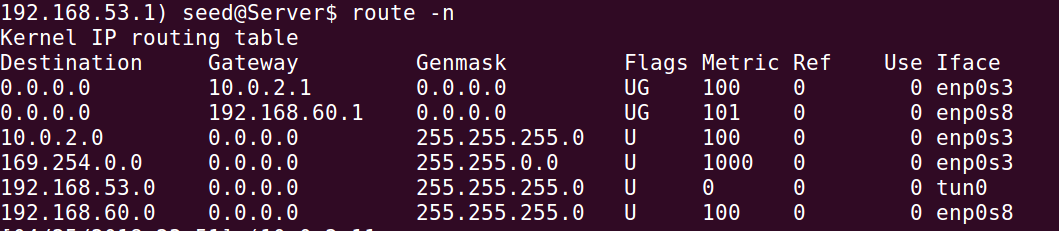


Step 3: Adding Routing entries on the Client and Server VMs:

* Add 192.168.60.0/24 on the client so that it forwards the packet directed to the host V on the network, 192.168.60.0 towards the tun interface.

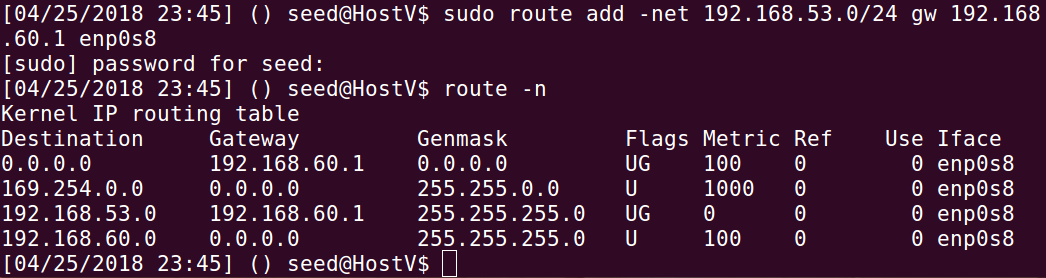


* On the server: We do not add any routing entry explicitly to the server. The entry for the traffic directed to the tun0 interface will be automatically added when the IP address is assigned to tun interface.



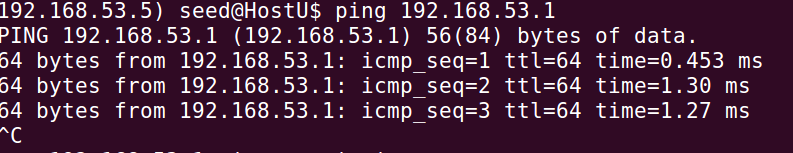
Step 4: Set Up Routing on Host V

* We add the following routing entry in the Host V so that the machine knows that the traffic directed to the network 192.168.53.0/24 goes through the VPN server acting as the gateway.

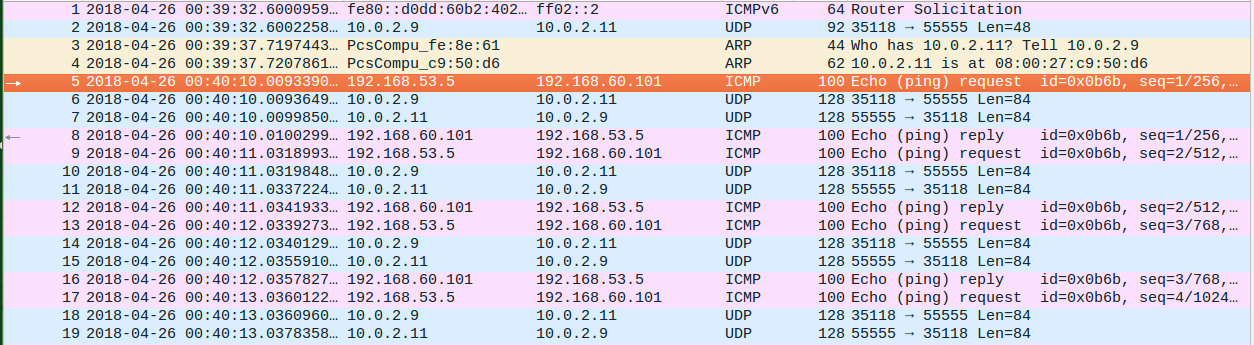


Step 5: Test the VPN Tunnel:

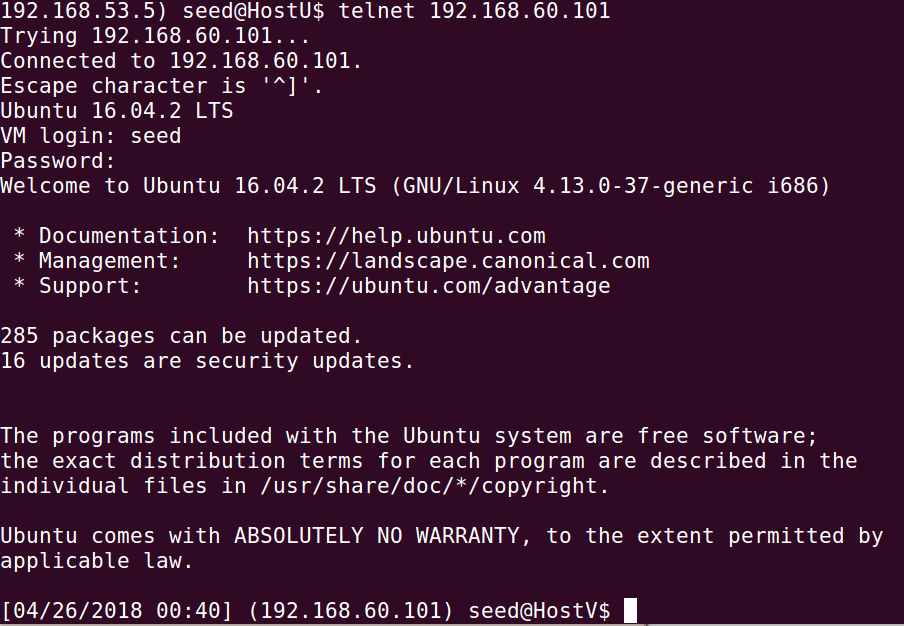
* We ping from the VPN Client (i.e Host U) to Host V:



* From the wireshark we can observe that the ICMP packets are exchanged between the tun0 interface of the Host U and the Host V:



* The ICMP packet is enclosed in the UDP packet and sent out from 10.0.2.9 to the server, 10.0.2.11.
* The server on receiving, takes the data from the UDP packet and finds out that the packet is destined to Host V. So it forwards the packet to Host V.
* Host V receives the ICMP packet having the source IP, 192.168.53.5 and sends out an ICMP reply through the gateway which is the server.
* **Telnet:** We telnet to Host V, from the Host U:

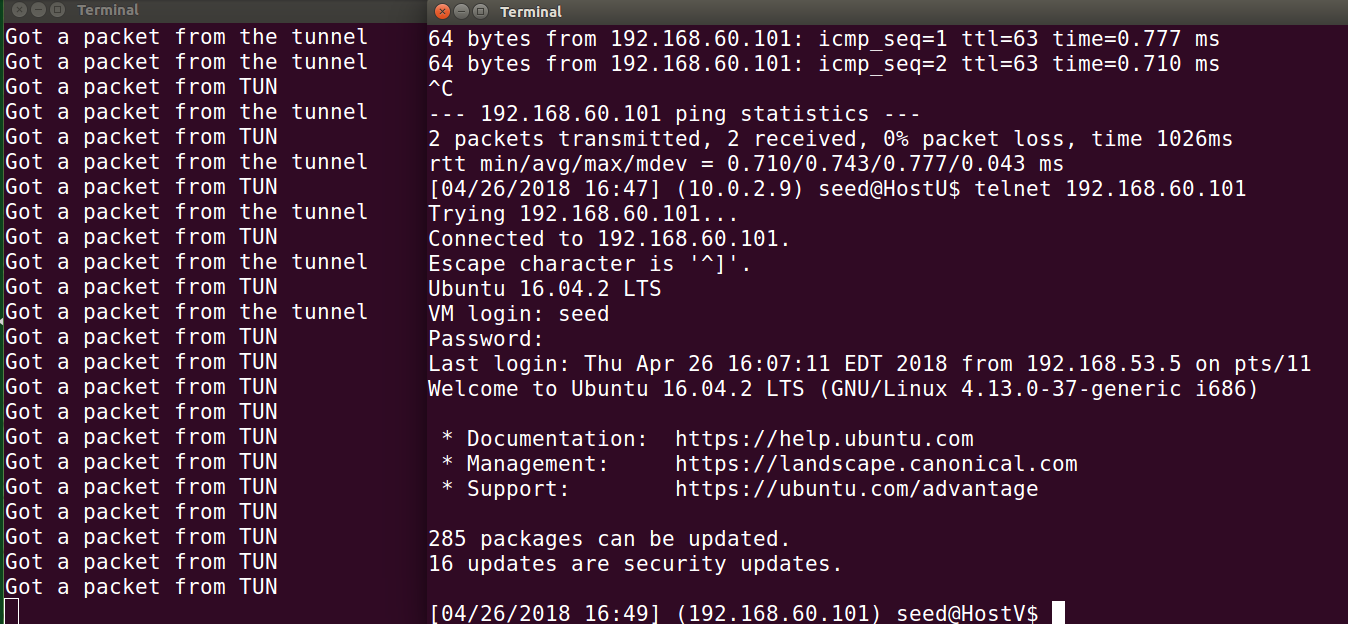


* From the wireshark capture, we can see the telnet connection being established between Host U and Host V through the server, similar to the ping operation.

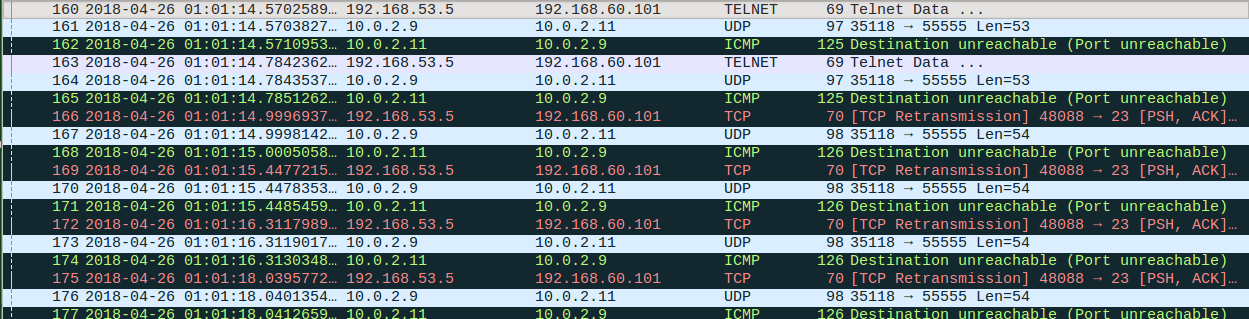


**Tunnel Breaking Test:**

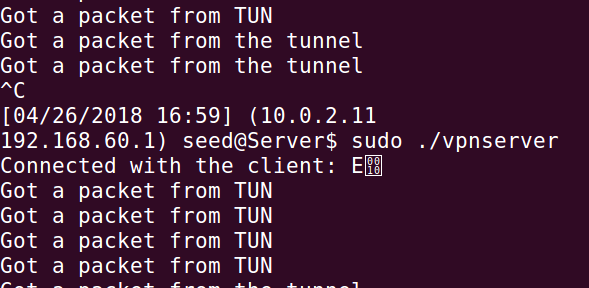
* We break the tunnel on the server.
* Because of this, we can see that the telnet connection on the Host U, has frozen – anything we typt on the terminal, it keeps sending to the tun interface (Got a packet from TUN) but from there, it is not able to complete the transaction.



* As a result of this, in the wireshark capture, we see the error message “Destination unreachable”.



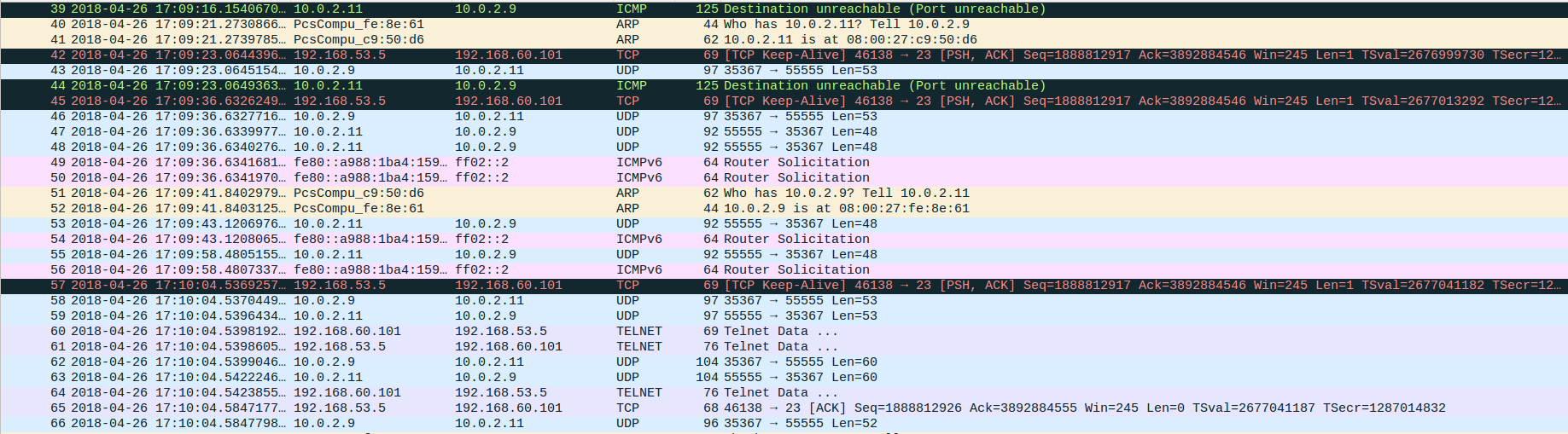
* Then we re-establish the tunnel on the server side:



* When the tunnel is re-established, the characters that we typed will eventually reach the telnet server due to TCP transmission and these characters will show up on the client side.



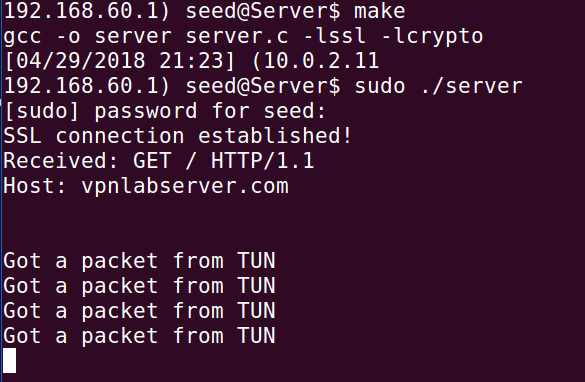
* Wireshark capture showing the telnet communication after re-establishing the tunnel:



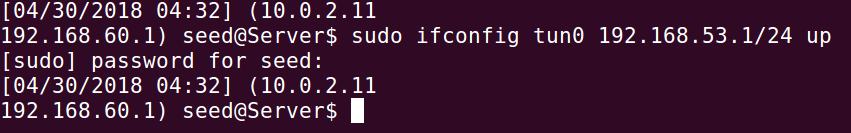
* The characters that we typed into the telnet after we break the tunnel are not lost. They are buffered, waiting to be sent to the telnet server. So when the tunnel is got back again, it will reach the telnet server and appear on the client.

**Task 3: Encrypting the Tunnel**

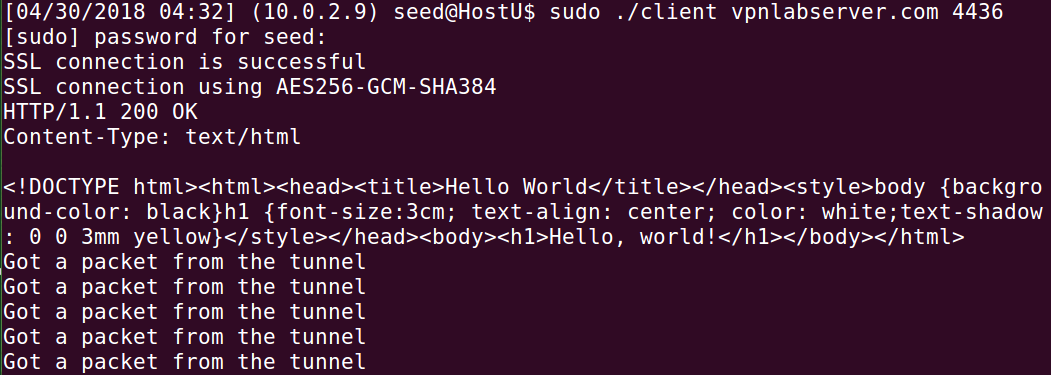
* Running the tlsserver program:



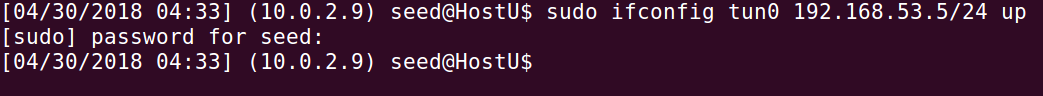
* Assigning IP address 192.168.53.1 to the tun0 interface of the server and bringing it up:



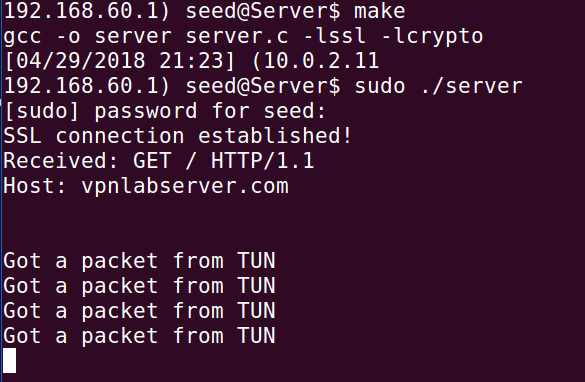
* We run the client giving the domain and the port number. SSL connection is established between the server and the client. It sends an HTTP Get request to the server and receives a reply:



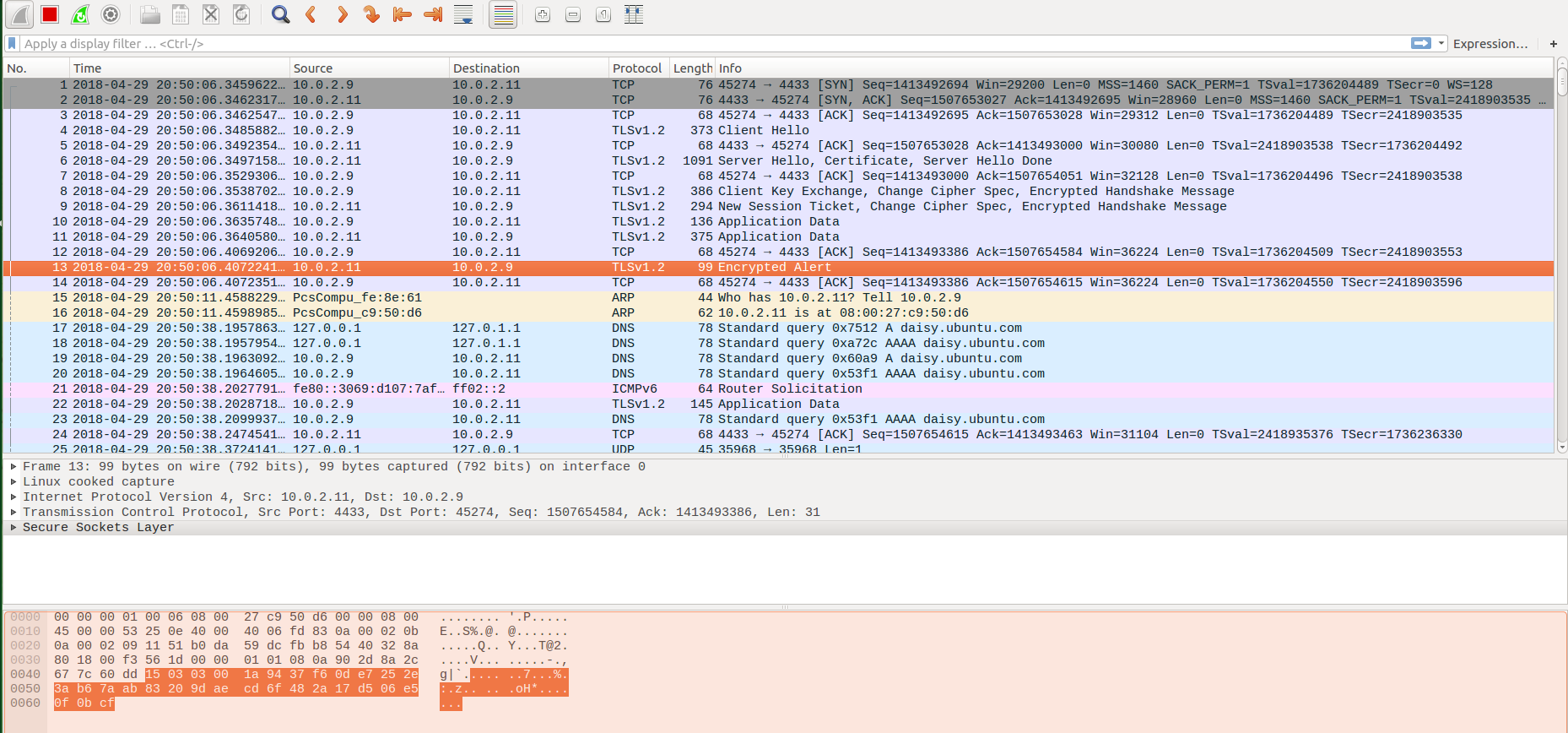
* Assigning IP address 192.168.53.5 to the tun0 device of the client and bringing it up:



* The server side after running the client program.
* We can see that SSL connection is established and the request got from client:



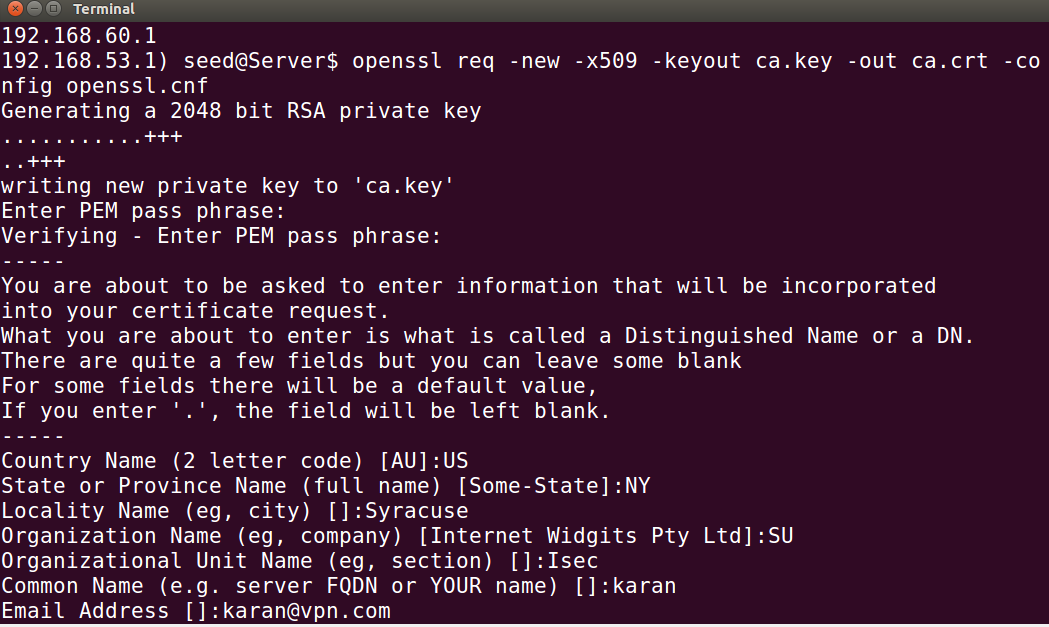
* From the Wireshark capture, we can see that the communication between the client and server happens through the TLS layer and also that the data is encrypted:



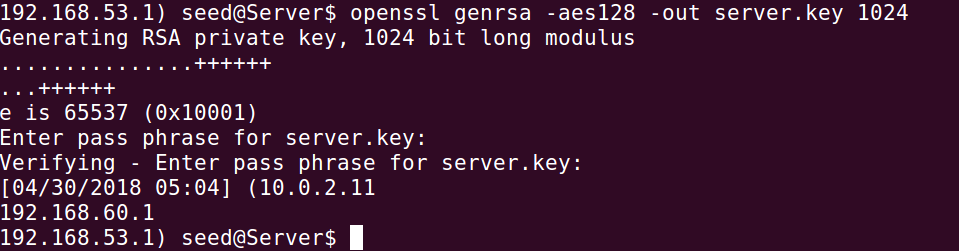
**TASK 4: Authenticating Server and Client:**

**Creating a server certificate for amrutesh.com:**

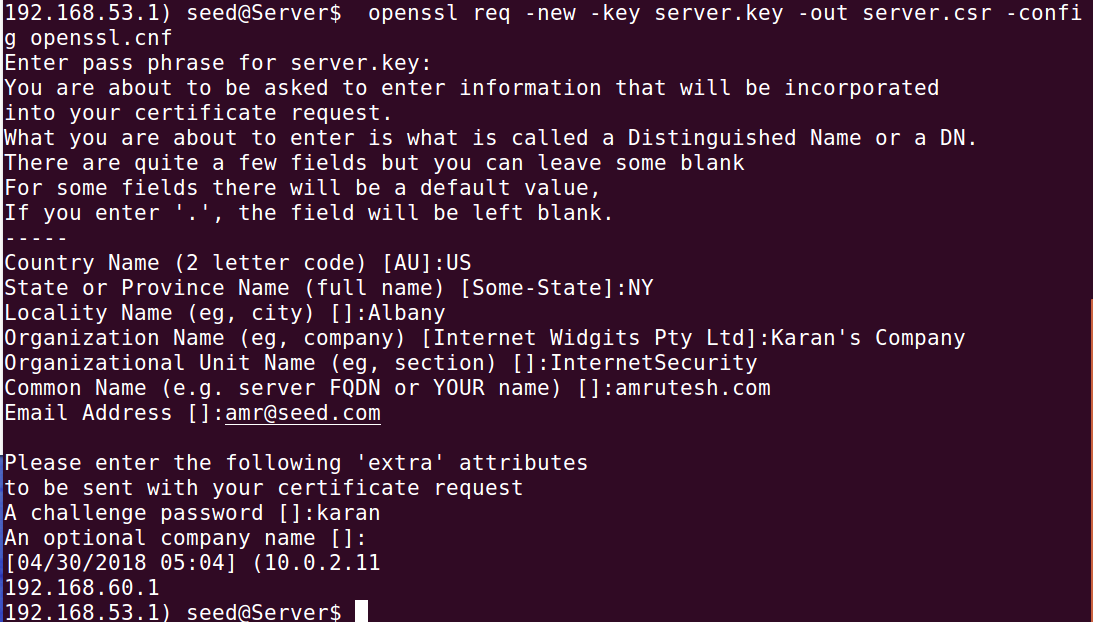
* Step 1: Becoming CA
* We generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. The output of the command are: CA’s private key and the CA’s public-key certificate



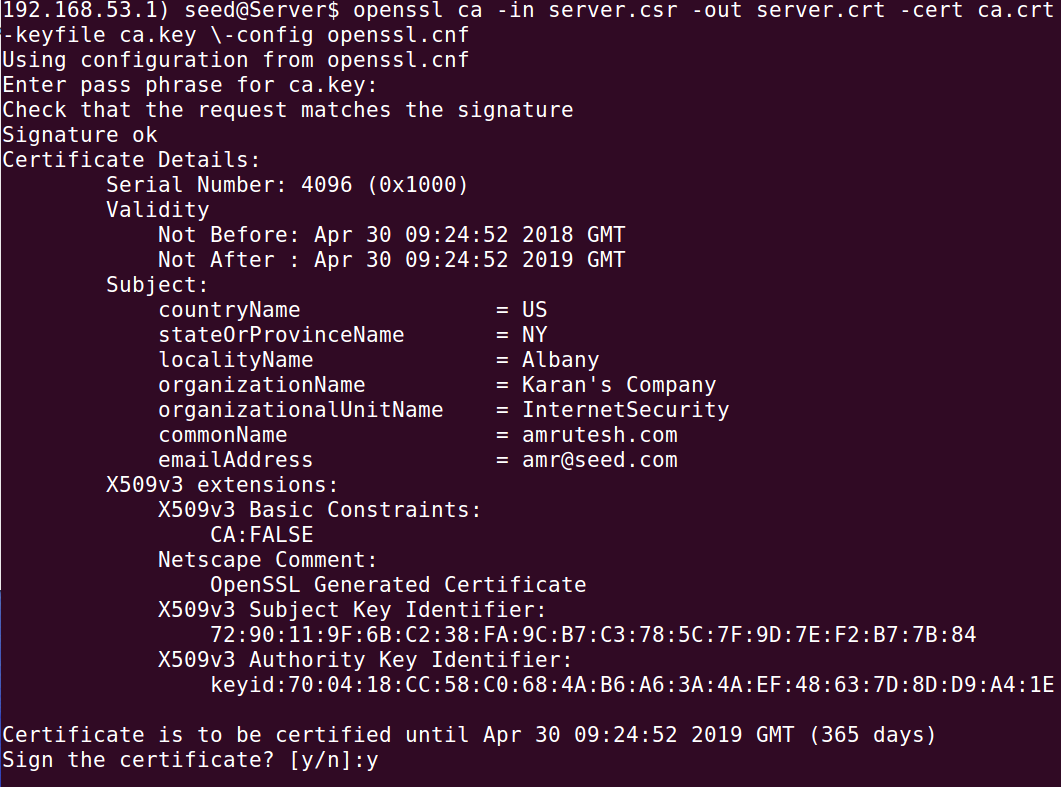
* Step 2: Creating a Certificate for *amrutesh.com*
* Generating public/private key pair: We can run the following command to generate an RSA key pair (both private and public keys). We also provide a password to encrypt the private key.



* Generate a Certificate Signing Request (CSR)
* Now we have the key ﬁle, so we generate a Certiﬁcate Signing Request (CSR), which basically includes the company’s public key. The CSR will be sent to the CA, who will generate a certiﬁcate for the key.



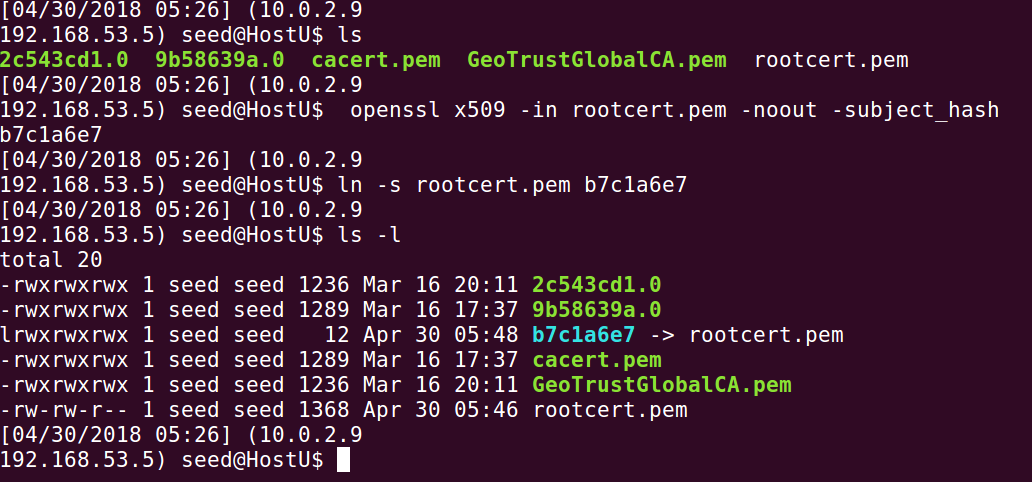
* Generating Certificates:
* The CSR file needs to have the CA’s signature to form a certificate. The following command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA’s ca.crt and ca.key:



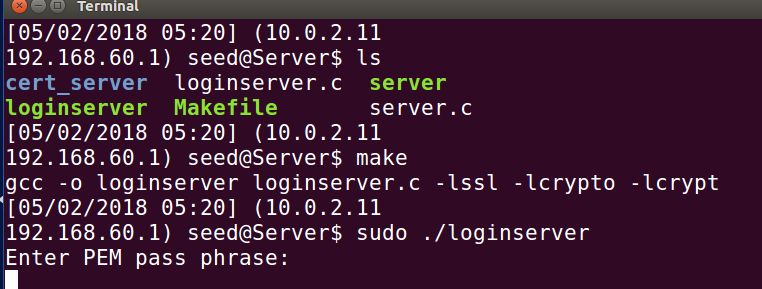
* We choose *amrutesh.com* as the name of our website and add it to /etc/hosts in the client.



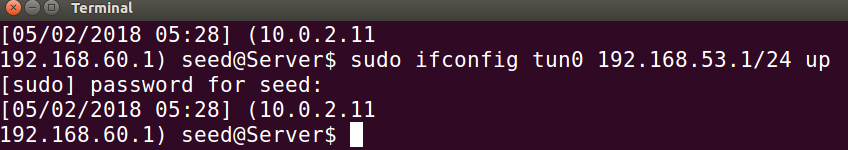
* We then copy the root CA’s certificate to the client and create a symbolic link to the file using its hash value:



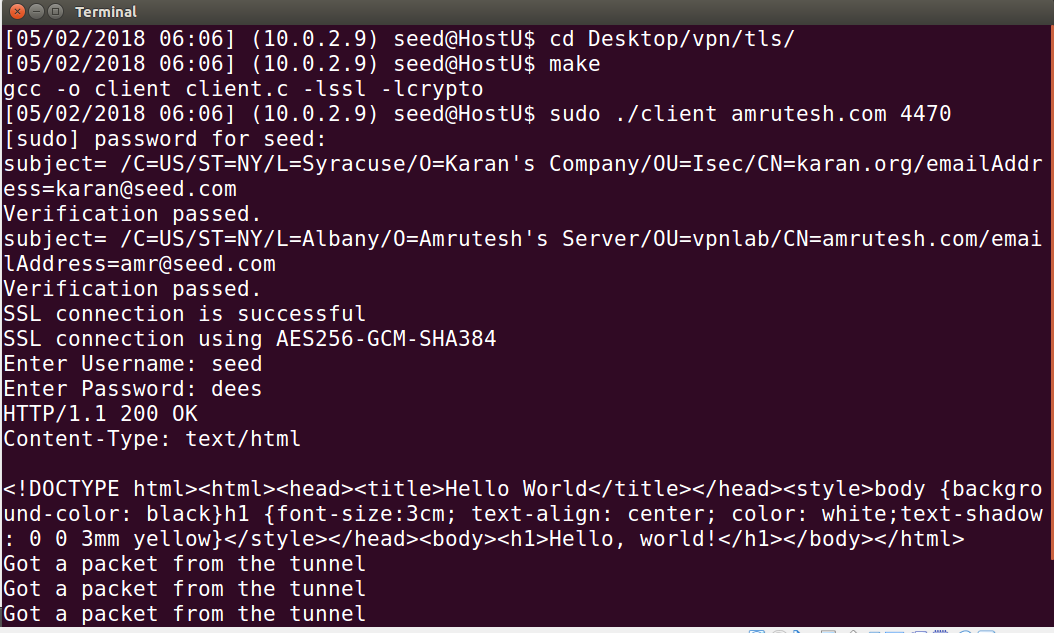
* We compile and run the server program:



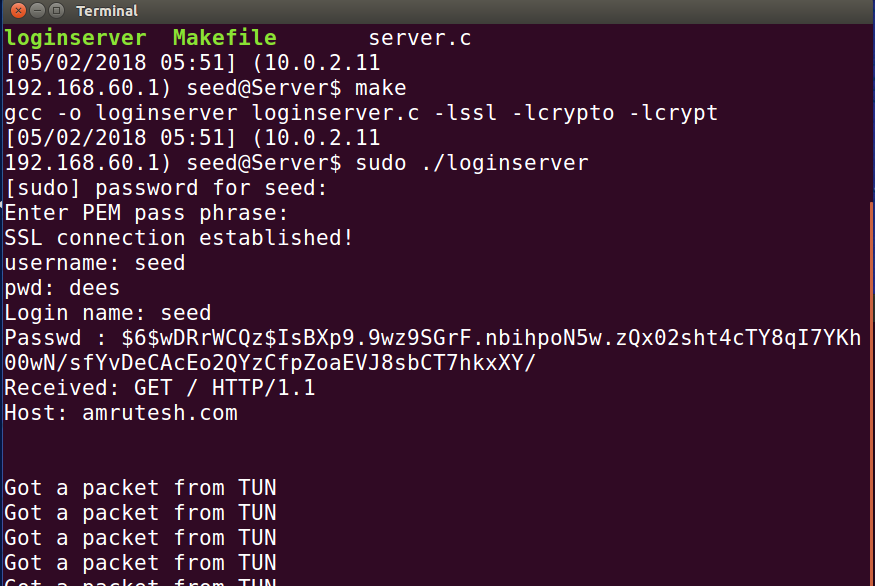
* Assigning IP address 192.168.53.1 to the tun0 device and bringing it up.



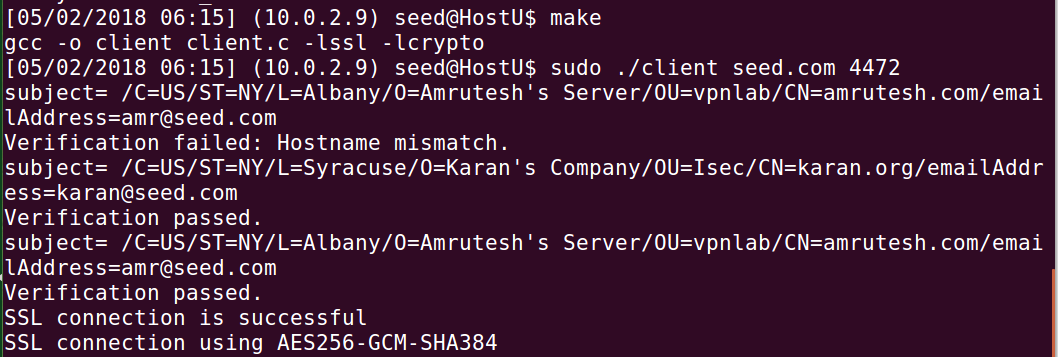
* We run the client program giving the correct domain and the port number which the server is using.
* Our server’s certificate and our CA’s certificates are verified.
* Also, the client is asked for authentication. Once he provides correct credentials, the requested data will be given. The username and password are checked in the server’s shadow file.



* Status in the server side once the SSL connection has been established.



* Hostname mismatch: If the given hostname does not match with the common name in the CA’s certificate, then the verification fails saying Hostname mismatch.

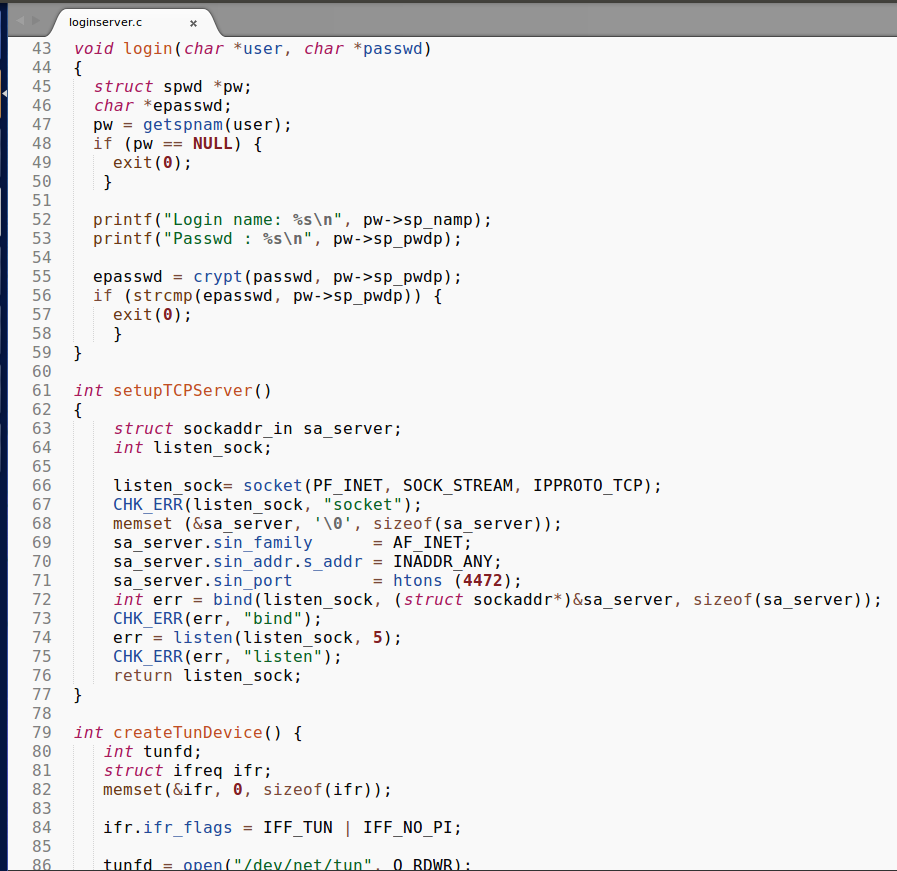


**Code:**

Server Code:



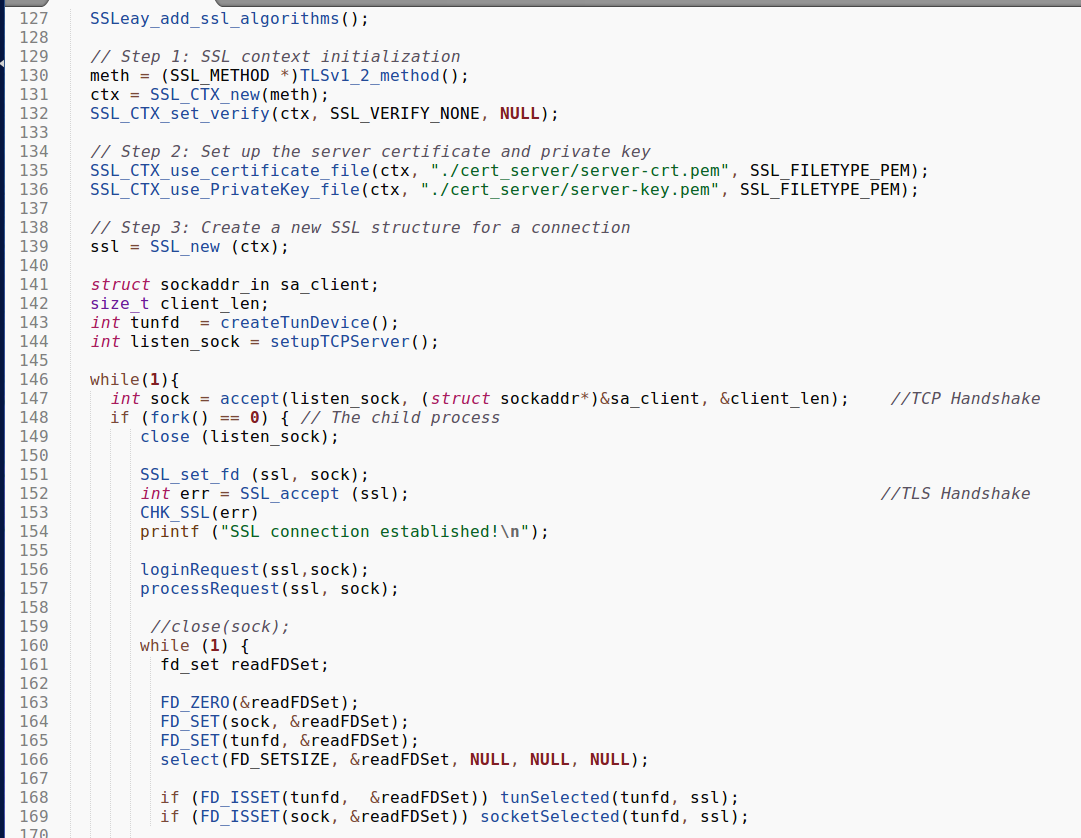
* The login function takes in the login credentials from the users
* The login function compares the given username and password with that contained in the shadow file.
* The bind call in the setupTCPServer binds the attribustes of the TCP protocol to the socket.



* ioctl command is used to create the tun device and return the file descriptor



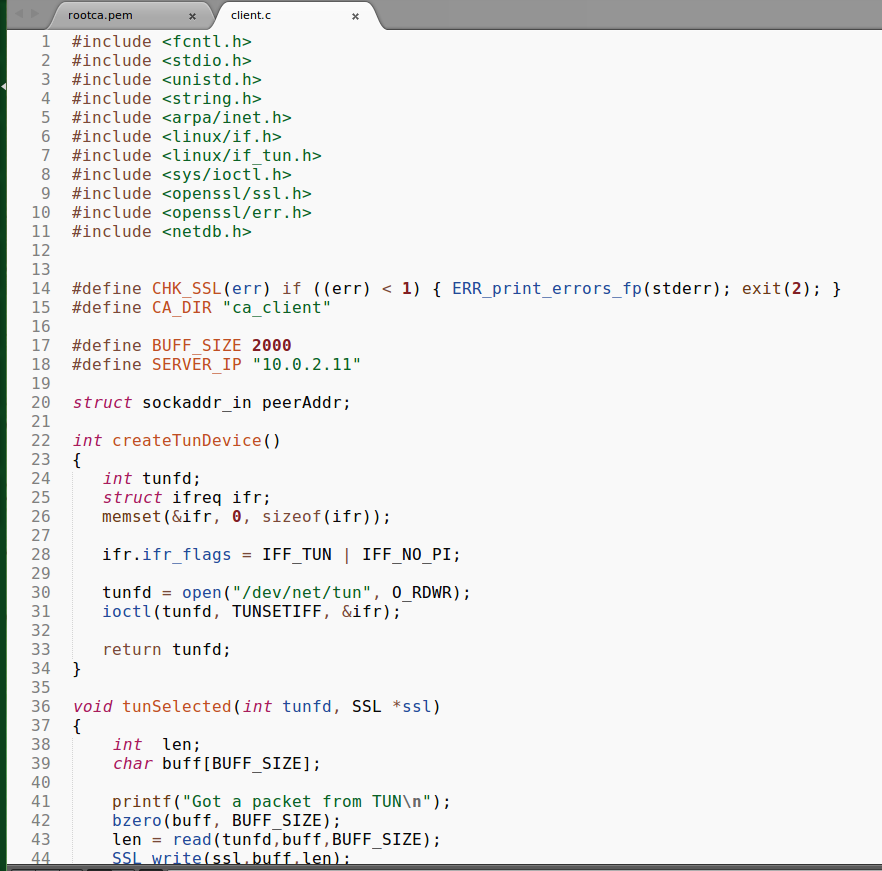
* The TCP handshake will be carried out first and then the socket will be sent to the SSL layer followed by the TLS handshake.

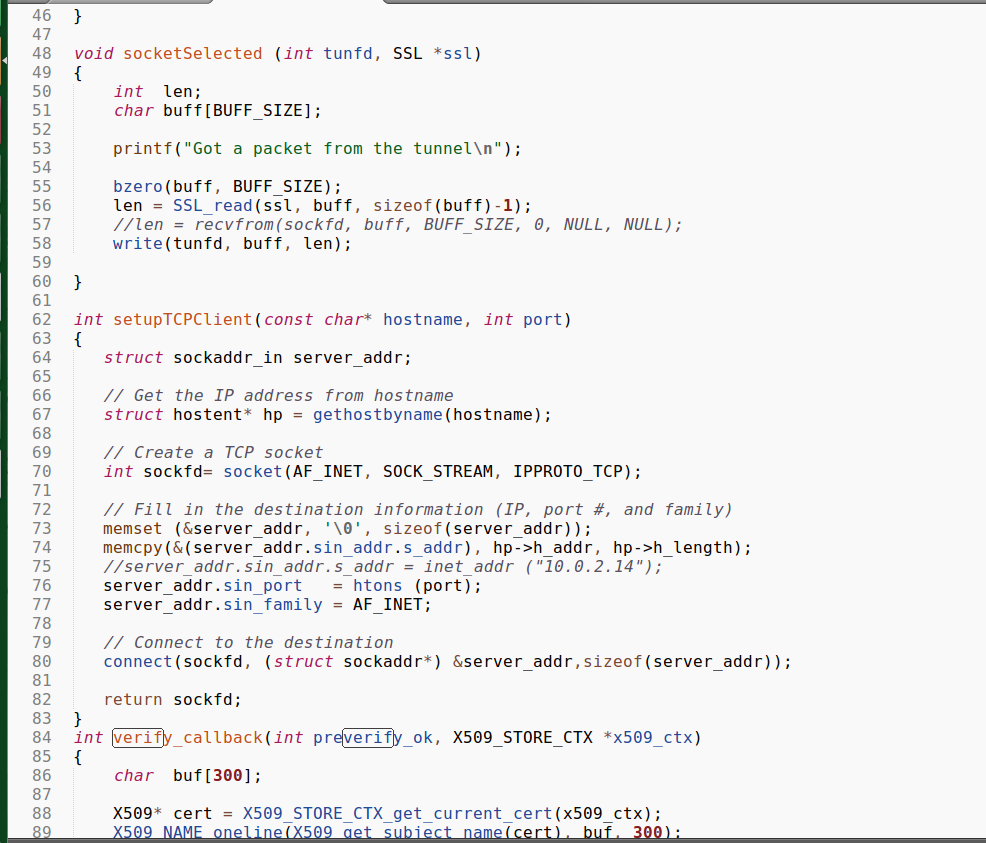


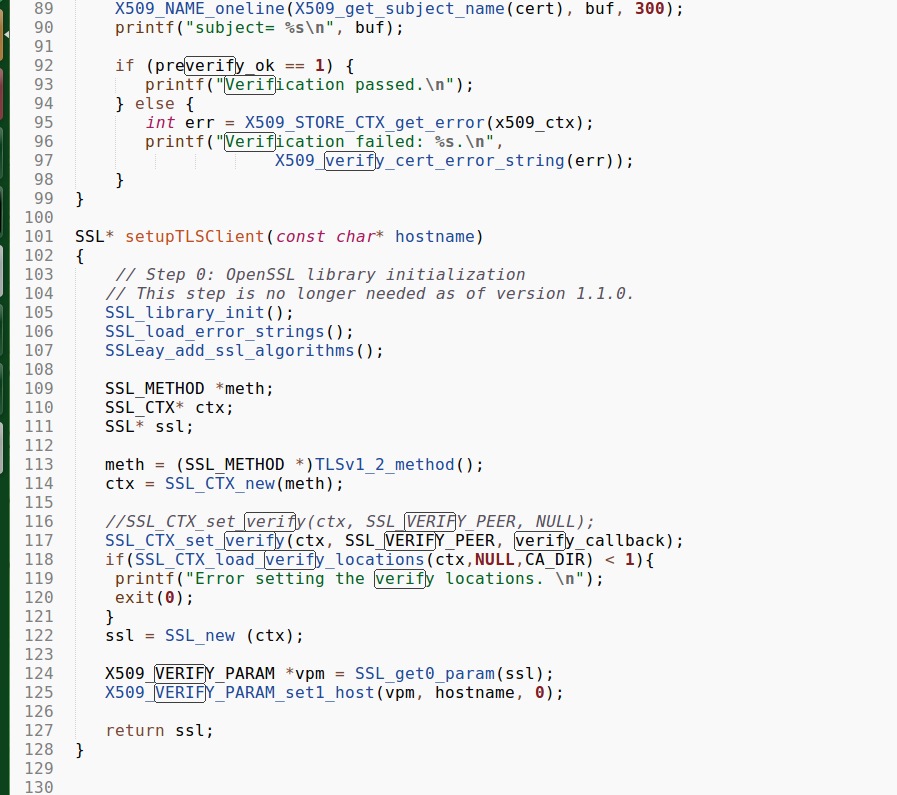
* The processRequest will process the SSL request from the client. Here we construct an HTML page and send it to the client.
* We use SSL\_WRITE() to send the message to the client.



Client Code:







1. **Verifying that the server certificate is valid: We do this by verifying the CA’s certificate stored in in a folder which is loaded by the line SSL\_CTX\_load\_verify\_locations(). (*Line No. 118*)**
2. **Verifying that the server is the owner of the certificate: This is done by verifying that the owner mentioned in the subject field of the server’s certificate is indeed own the public key of the certificate. Verifying with the CA’s certificate will tell us the owner of the public key. (*Line 118*)**
3. **Verifying that the server is the intended server: We do this by checking the hostname of the server. (*Line No. 125*)**



