VPN FINAL PROJECT

Task 1 : VM setup

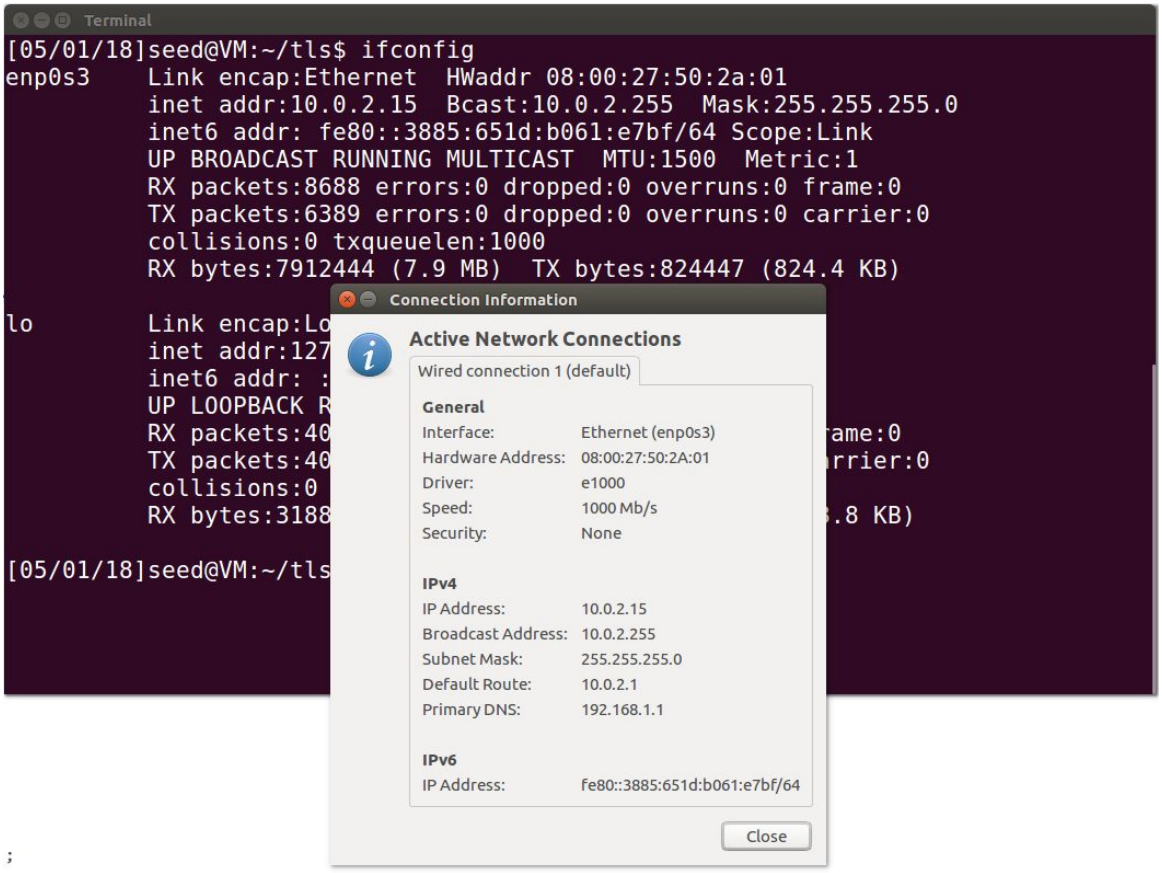

Fig 1.1 Client ip address and interface info

```
127.0.0.1       localhost
10.0.2.4        vpnlabserver.com
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
```

Fig 1.2 : Client Hosts File

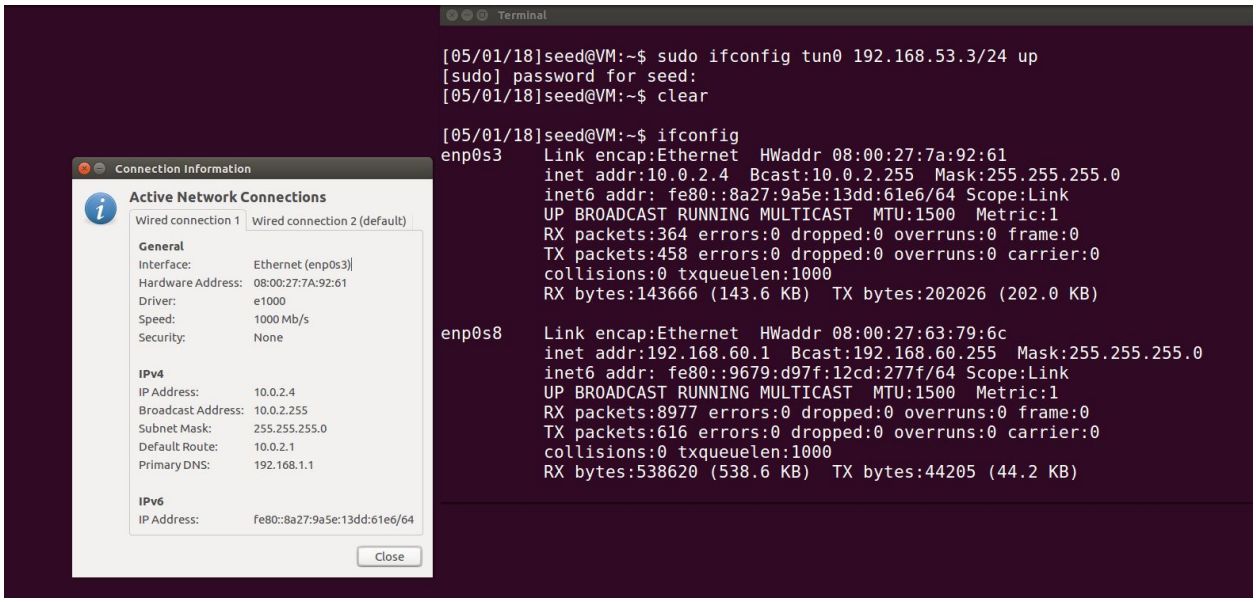

Fig 1.3 : Vpn Server Address and interface info

The Server is connected through a NAT network(enps03) to the client and through an internal host network(enps08) to an internal host.

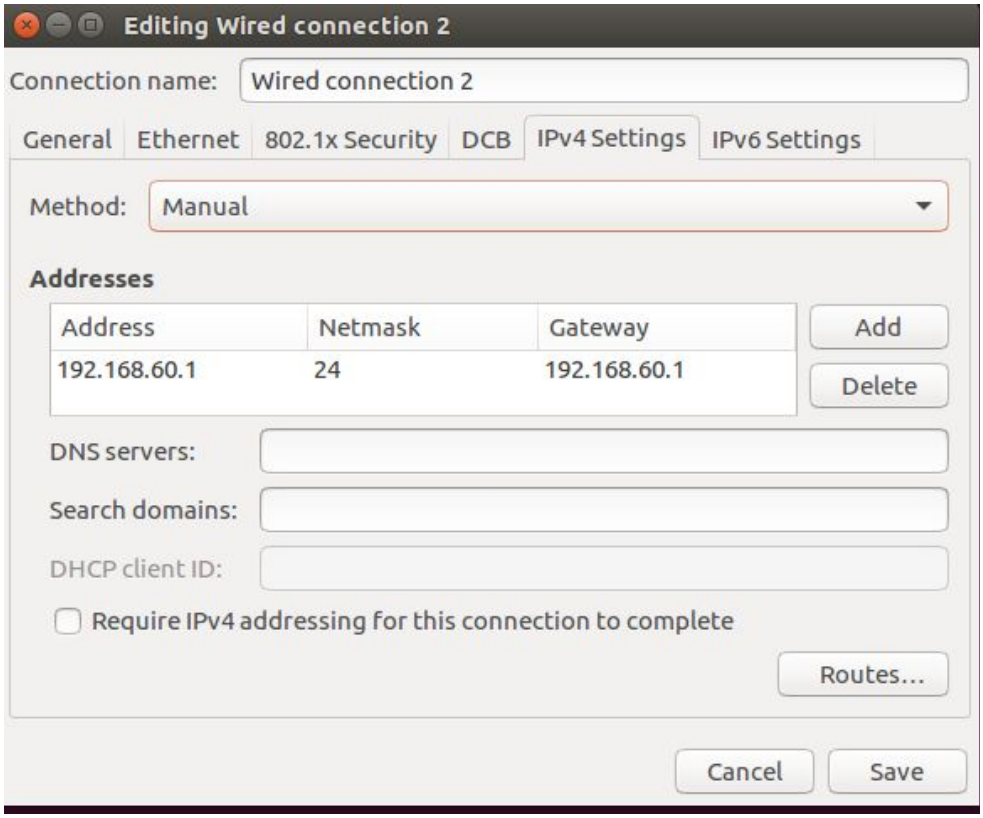

Fig 1.4 : Internal Host Connection from VPN

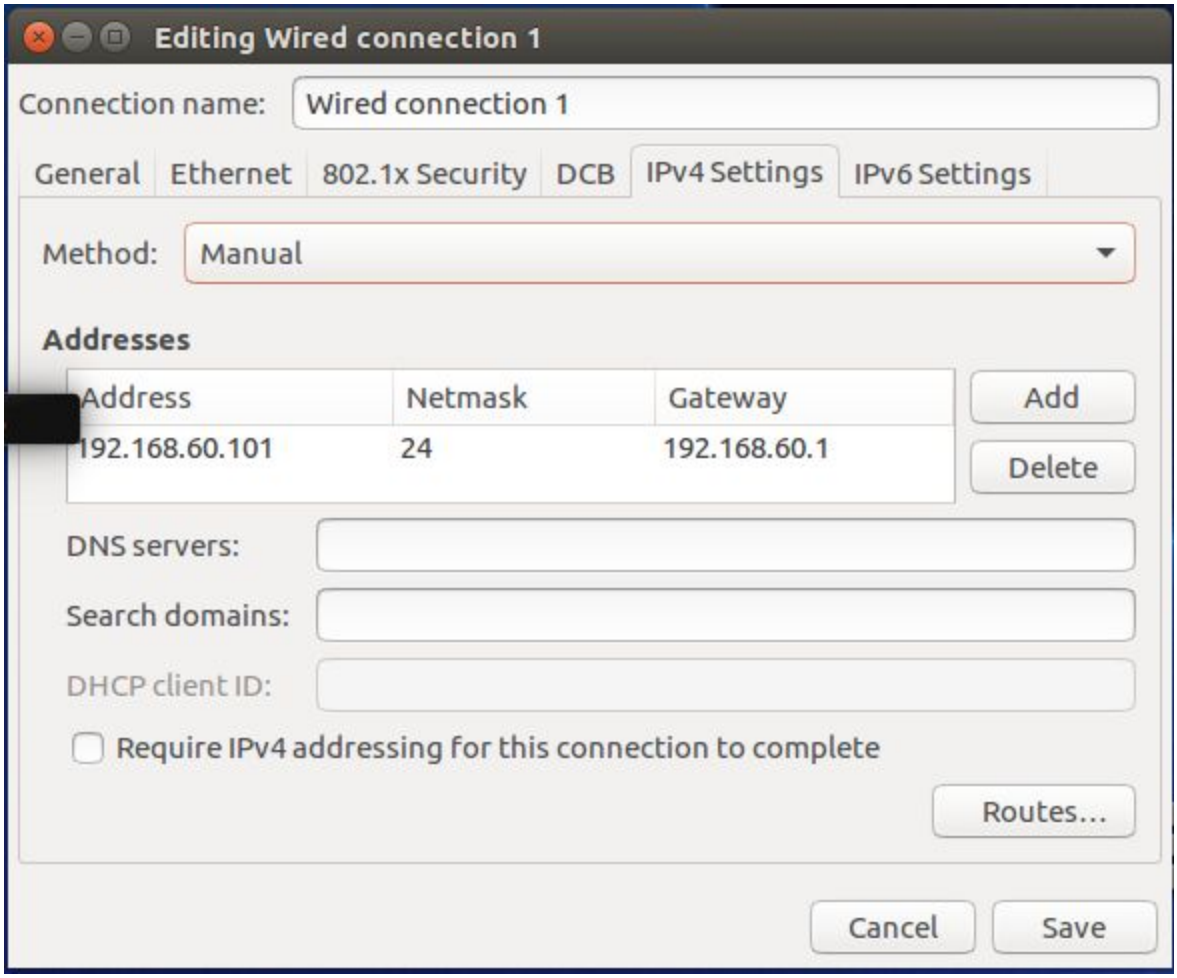

Fig 1.4 : Internal Host setup

Task 2 : Creating a VPN tunnel Using TUN/TAP

Step 1 : Run VPN server

```
[05/01/18]seed@VM:~/tls$ gcc tlsserver.c -o ./tlsserver -lssl -lcrypto -lcrypt
[05/01/18]seed@VM:~/tls$ sudo ./tlsserver
```

Fig 2.1 : Started VPN Tunnel

```
[05/01/18]seed@VM:~$ sudo ifconfig tun0 192.168.53.3/24 up
[sudo] password for seed:
[05/01/18]seed@VM:~$
```

Fig 2.2 : Config tun0 on server

Step 2 : Start client and config

```
[05/01/18]seed@VM:~/tls$ sudo ./tlsclient vpnlabserver.com 4433
[sudo] password for seed:
TLSCLIENT SETUP DONE!--
TcpCLIENT SETUP DONE!--
SSL connection is successful
SSL connection using AES256-GCM-SHA384
Enter UID:seed
ENter Pwd:dees
auth:i:seed
,p:dees
rec ip: 192.168.53.2ip
IP : 192.168.53.2
Execute `ifconfig tun0 192.168.53.2/24 up`
Execute `route add -net 192.168.60.0/24 tun0`
got packet from TUN0: 48
got packet from TUN0: 48
```

Fig 2.3 : Start client and config

The server establishes connection , asks for a user id and password , if the authentication is succesful , it sends back an ip address and executes client routing.

Step 5 : Testing the Tunnel

```
[05/01/18]seed@VM:~$ ping 192.168.60.101
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=2.61 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=2.20 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=2.39 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=2.08 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=63 time=2.15 ms
64 bytes from 192.168.60.101: icmp_seq=6 ttl=63 time=0.784 ms
64 bytes from 192.168.60.101: icmp_seq=7 ttl=63 time=2.03 ms
```

Fig 2.4 Try ping to Internal Host

```
[05/01/18]seed@VM:~$ telnet 192.168.60.101
Trying 192.168.60.101...
Connected to 192.168.60.101.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

509 packages can be updated.
281 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[05/01/18]seed@VM:~$ ls
bin            Desktop    Downloads          Music     Public  Templates
Customization  Documents  examples.desktop  Pictures  source  Videos
[05/01/18]seed@VM:~$
```

Fig 2.4 : Telnet Connection Successful

3	2018-05-01 21:36:25.0610155…	192.168.53.2	192.168.60.1	ICMP	84	Echo (ping) request	id=0x3ada, seq=1/256, ttl=64 (reply in 4)
4	2018-05-01 21:36:25.0625179…	192.168.60.1	192.168.53.2	ICMP	1999	Echo (ping) reply	id=0x3ada, seq=1/256, ttl=64 (request in 3)
5	2018-05-01 21:36:26.0631094…	192.168.53.2	192.168.60.1	ICMP	84	Echo (ping) request	id=0x3ada, seq=2/512, ttl=64 (reply in 6)
6	2018-05-01 21:36:26.0646960…	192.168.60.1	192.168.53.2	ICMP	1999	Echo (ping) reply	id=0x3ada, seq=2/512, ttl=64 (request in 5)
7	2018-05-01 21:36:28.1096997…	fe80::3c:d378:34d0:…	ff02::2	ICMPv6	1999	Router Solicitation	
8	2018-05-01 21:36:29.7656600…	192.168.53.2	192.168.60.101	ICMP	84	Echo (ping) request	id=0x3ade, seq=1/256, ttl=64 (reply in 9)
9	2018-05-01 21:36:29.7682559…	192.168.60.101	192.168.53.2	ICMP	1999	Echo (ping) reply	id=0x3ade, seq=1/256, ttl=63 (request in 8)
0	2018-05-01 21:36:30.7672144…	192.168.53.2	192.168.60.101	ICMP	84	Echo (ping) request	id=0x3ade, seq=2/512, ttl=64 (reply in 11)
1	2018-05-01 21:36:30.7693930…	192.168.60.101	192.168.53.2	ICMP	1999	Echo (ping) reply	id=0x3ade, seq=2/512, ttl=63 (request in 10)

Fig 2.5 Wireshark capture of tun0

Step 6 : Tunnel Breaking Test

After breaking the tunnel , we try to type something but nothing registers

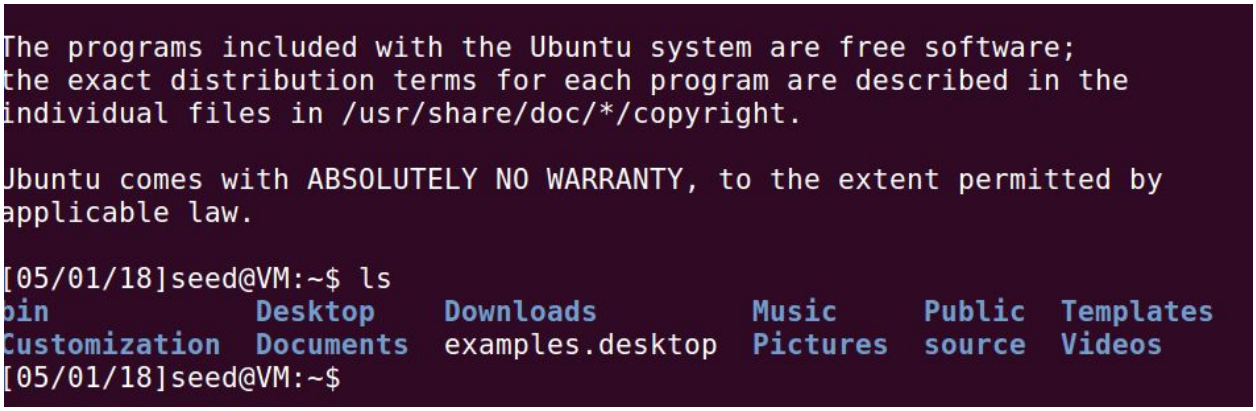

Fig 2.6 : Cannot type anything in telnet

Its not that we cannot type anything , its just not displayed because the connection has been lost .

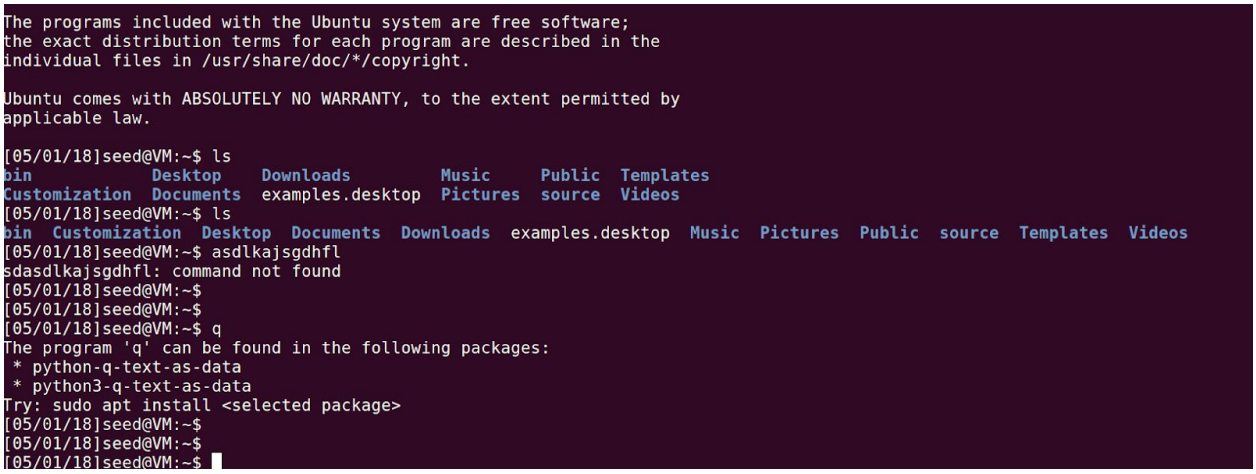

Fig 2.7 : Everything we typed will be executed

Once the connection is restored , it returns whatever command we typed during the time of the disconnection. All the commands are stored in a buffer , which is sent to the telnet once the connection is restored.

Task 3 : Encrypting the Tunnel

The tunnel is encrypted using TLS . The connection between the client and the server needs to be encrypted , especially when you will be sending your userID and password for the server to authenticate.

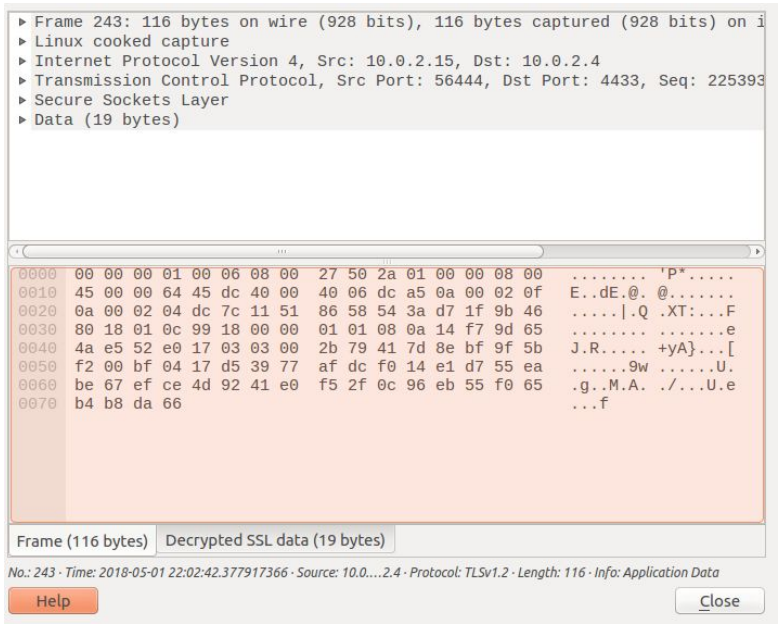

Fig 3.1 : Encrypted Id/password going to server

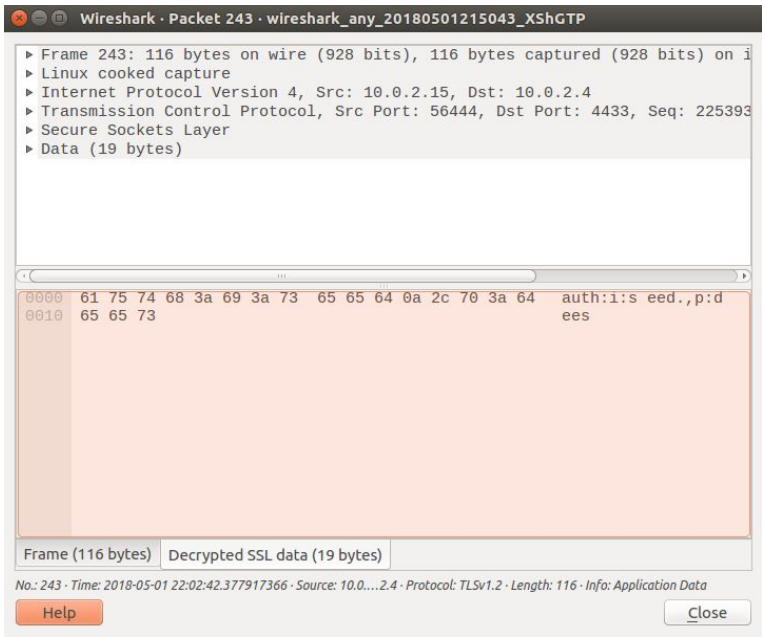

Fig 3.2 : Decrypted version of id/password

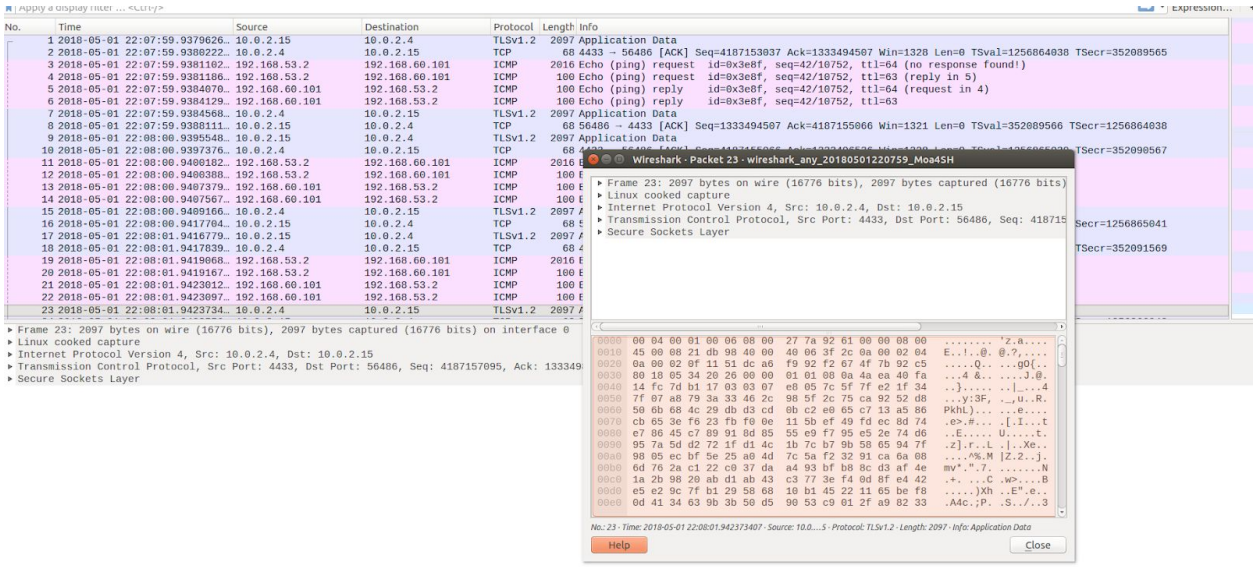

Fig 3.3 : Every packet going on the internet is encrypted

Task 4 : Authenticating the VPN server

We are using PKI to authenticate the server .

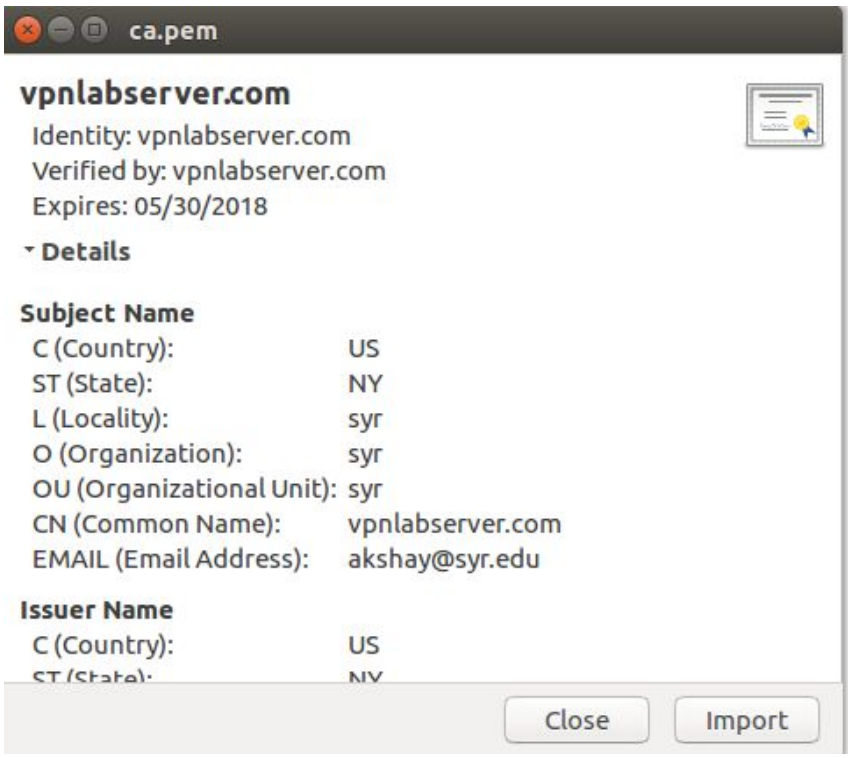

Fig 4.1 : Client CA certificate

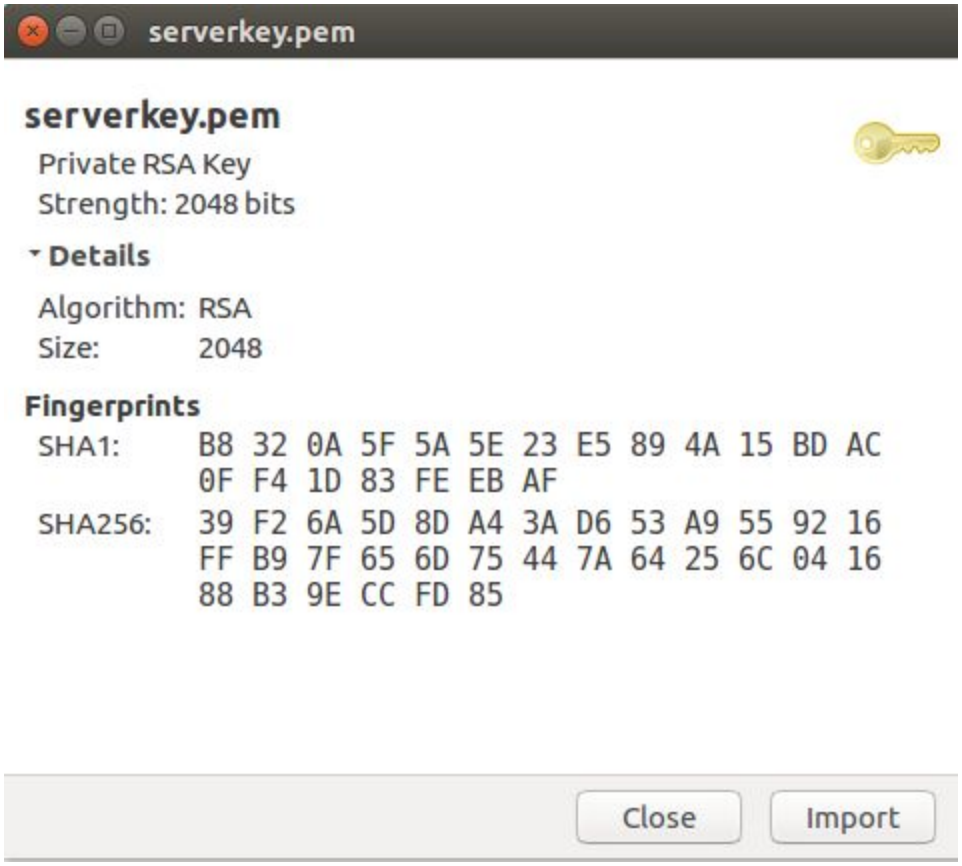

Fig 4.2 : Servers Key

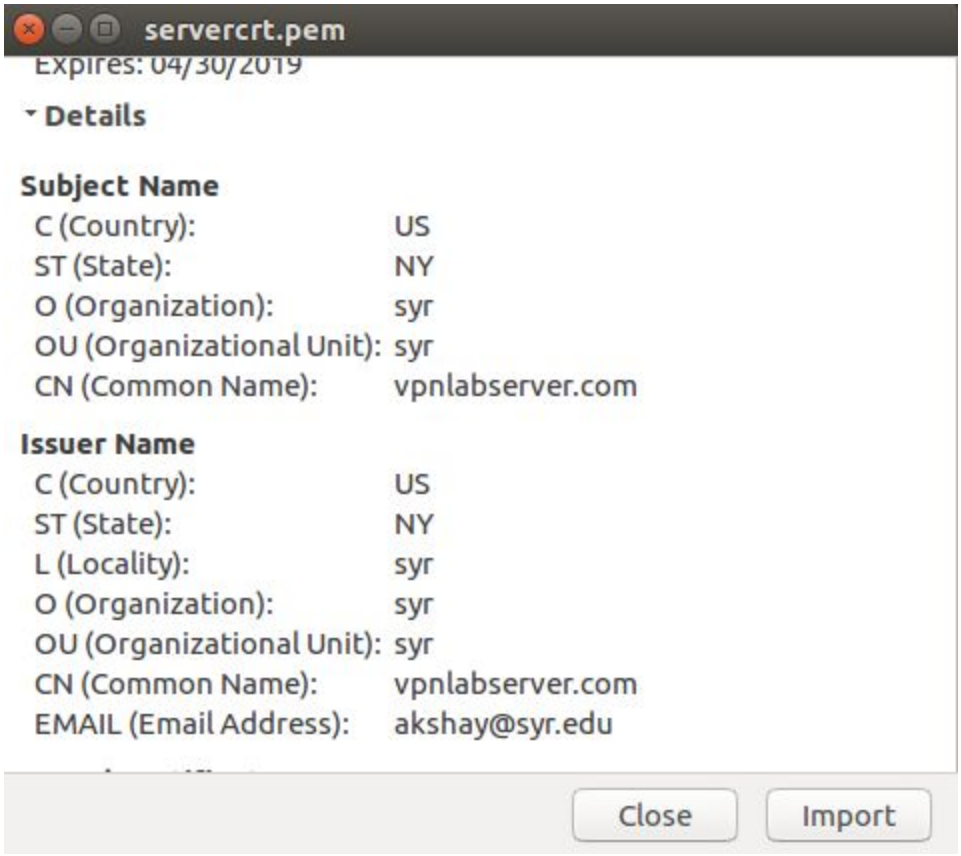

Fig 4.3 : Servers Certificate

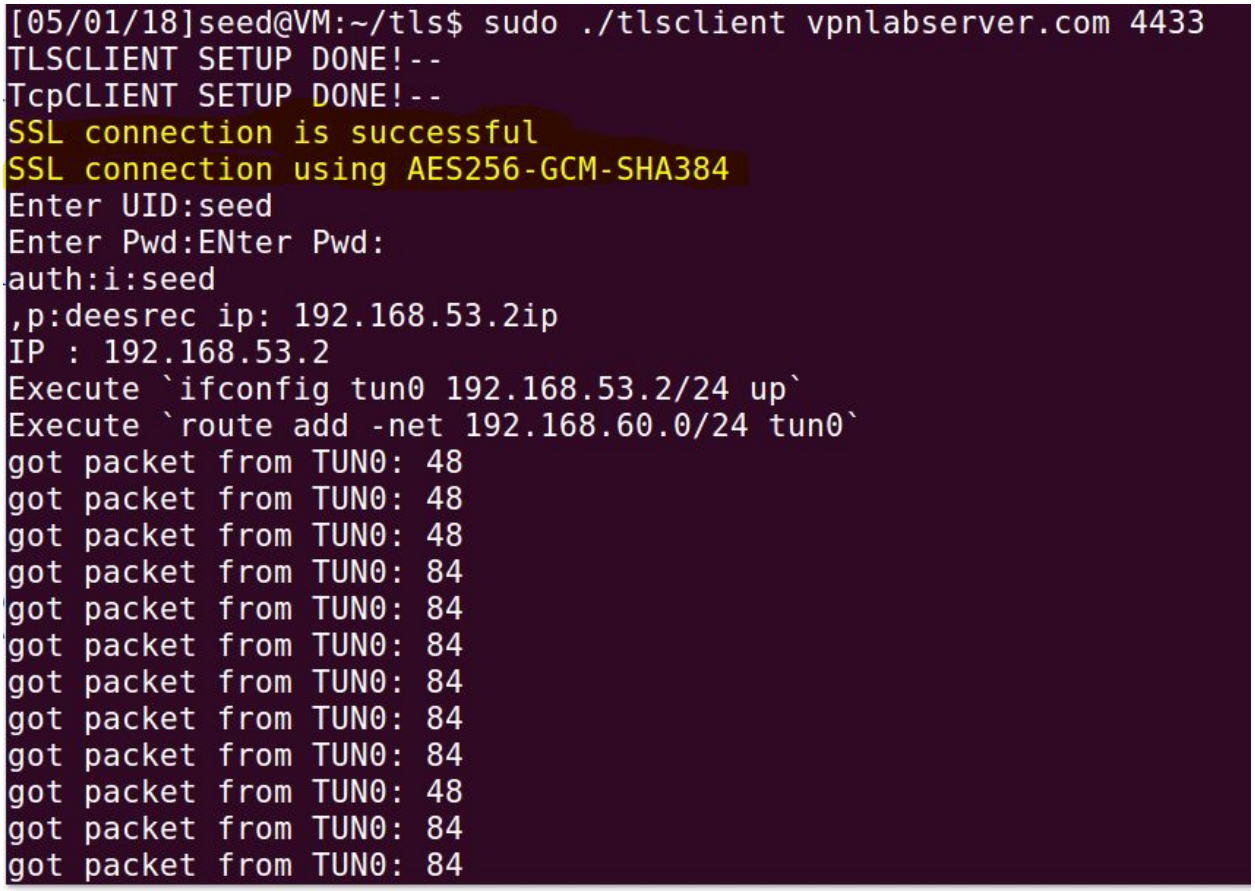

Fig 4.4 : Client successfully authenticates server

TASK 5 : Authenticating the VPN Client

We will be using the shadow file to authenticate the user on the server.

```
int login(char *user, char *passwd)
{
        struct spwd *pw;
        char *epasswd;
        pw = getspnam(user);
        if (pw == NULL) {
                return -1;
        }
        printf("Login name: %s\n", pw->sp_namp);
        printf("Passwd : %s\n", pw->sp_pwdp);
        epasswd = crypt(passwd, pw->sp_pwdp);
printf("recieved pwd : %s\n",epasswd);

if (strcmp(epasswd,pw->sp_pwdp)) {
//printf("match!\n");
        return -1;
        }
//printf("no match!\n");
        return 1;
}
```

Fig 5.1 : Function to check shadowfile.

The UID/password we get through the tunnel is fed into this function . The shadow file is encrypted and we cannot just compare the passwords.THe received password is encrypted using the crpyt() function and then compared.

```
   char uid[20];
   char *pwd = malloc(20);
char *comb = malloc(strlen(uid)+strlen(pwd)+10);
        tunfd  = createTunDevice();
printf("Enter UID:");

   fgets(uid,20,stdin);
printf("ENter Pwd:");
pwd = getpass("Enter Pwd:");

strcpy(comb,"auth:i:");
strcat(comb,uid);
strcat(comb,",p:");
   strcat(comb,pwd);
//sprintf(sendBuf, "", hostname);
   memcpy(sendBuf,comb,strlen(comb));
        printf("%s",comb);
   SSL_write(ssl, sendBuf, strlen(sendBuf));
```

Fig 5.2 : Sending the authentication details from client

We send the uid and password to the server through the tunnel after the client(we) has authenticated the server and are sure that this is the correct server to access.

After receiving it at the server , we check if its an authentication packet (auth:i:uid,p:pwd format)
The uid and password are seperated and fed into the login function in fig 5.1.

```
Received: auth:i:seed
,p:dees
uid:seedpwd:deesLogin name: seed
Passwd : $6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN/sfYvDeCAc
Eo2QYzCfpZoaEVJ8sbCT7hkxXY/
recieved pwd : $6$wDRrWCQz$IsBXp9.9wz9SGrF.nbihpoN5w.zQx02sht4cTY8qI7YKh00wN/sfY
vDeCAcEo2QYzCfpZoaEVJ8sbCT7hkxXY/
login----1Login success ,send free ipaddress!
sending ip :192.168.53.2ip
```

If authentication is successfull , an ip address is sent from the lookup table.

The client recieves the ip and runs routing commands to setup and route tun0.

```
IP : 192.168.53.2
Execute `ifconfig tun0 192.168.53.2/24 up`
Execute `route add -net 192.168.60.0/24 tun0`
got packet from TUN0: 48
got packet from TUN0: 48
got packet from TUN0: 48
[05/01/18]seed@VM:~/tls$
```

Fig 5.5 : Executing routing commands and setup tun0

Task 6 : Multiple clients

```
//struct for assigning ip addresses
struct iplookup{
char *ip;
int avail;
SSL *ssl;
pid_t pid;
};
struct iplookup iplook[3];
```

Fig 6.1 : iplookup table for multiple clients.

```
char* getip()
{
int i=0;
for(i=0;i<4;i++)
{
        if(iplook[i].avail==1){
        iplook[i].avail=0;
        return iplook[i].ip;}
}
return "busy";
}
```

Fig 6.2 : getting a free ip address from table

Incomplete

I have issues with SSL maintaining sessions for each client . The server is able to authenticate multiple clients but cannot send receive and data because the SSL handler is going out of scope. If I try to get around this , I cannot get multiple clients to authenticate the server.