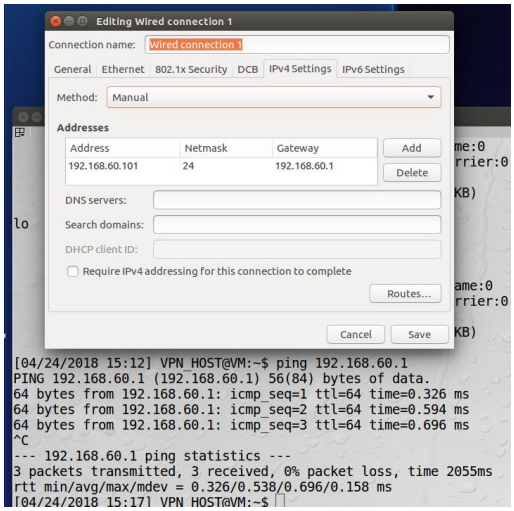
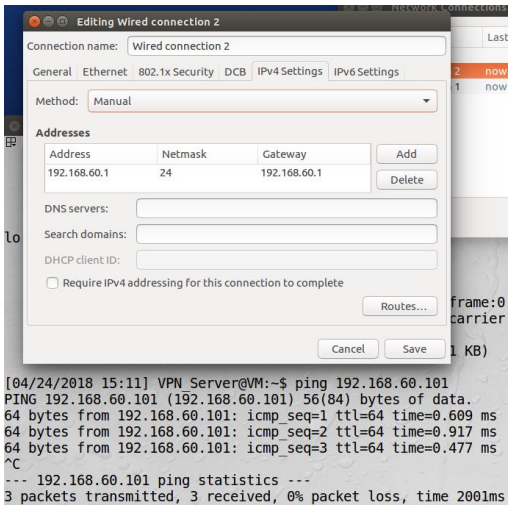
VPN

Task 1:



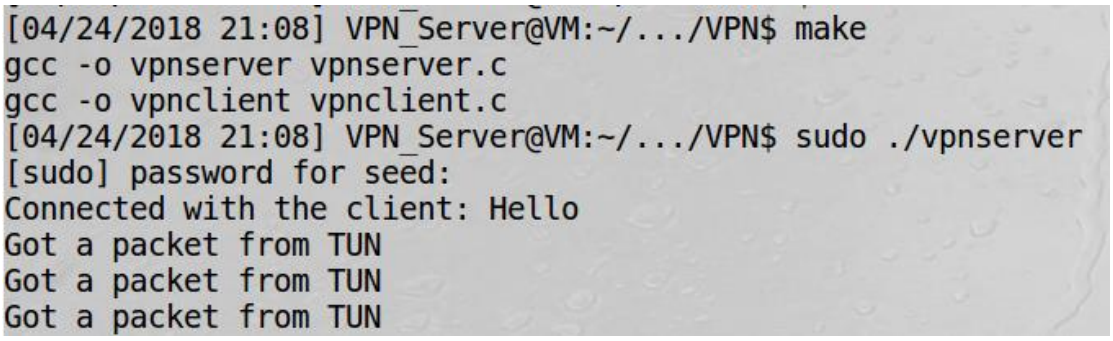
Edit the configuration on machine VPN_HOST



Edit internal configuration on machine VPN_Server. These two machines can conmmunicate each other.

Task 2:

Step 1:



```
[04/24/2018 21:09] VPN_Server@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[sudo] password for seed:
[04/24/2018 21:09] VPN_Server@VM:~$ sudo sysctl net.ipv4_ip_forward=1
sysctl: cannot stat /proc/sys/net/ipv4_ip_forward: No such file or directory
[04/24/2018 21:10] VPN_Server@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00-00
          inet addr:192.168.53.1  P-t-P:192.168.53.1  Mask:255.255.255.0
          inet6 addr: fe80::dd46:98b7:b75f:220a/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:7892 (7.8 KB)  TX bytes:5582 (5.5 KB)
```

Compile and run the server vpn program on server machine. Also assign a new ip to the tun interface and enable the forwarding function on server side.

Step2:

```
[04/24/2018 21:07] Client1@VM:~/.../VPN$ sudo ./vpnclient
[sudo] password for seed:
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:192.168.53.5  P-t-P:192.168.53.5  Mask:255.255.255.0
          inet6 addr: fe80::d36a:94f:5d9c:d469/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:184 errors:0 dropped:0 overruns:0 frame:0
          TX packets:290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:11174 (11.1 KB)  TX bytes:15737 (15.7 KB)
```

Run the client program and assign new ip to the tun interface.

Step 3:

```
[04/24/2018 22:01] Client1@VM:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
192.168.53.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
```

Set up routing on client.

```
[04/24/2018 22:02] VPN_Server@VM:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Ifac
e
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 enp0
s3
0.0.0.0         192.168.60.1    0.0.0.0         UG        0 0          0 enp0
s8
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0
s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0
s3
192.168.53.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0
s8
[04/24/2018 22:07] VPN_Server@VM:~$
```

Set up routing on server.

Step 4:

```
[04/24/2018 21:00] VPN_HOST@VM:~$ sudo route add -net 192.168.53.0/24 gw 192.168
.60.1
[sudo] password for seed:
[04/24/2018 21:15] VPN_HOST@VM:~$
```

```
[04/24/2018 21:15] VPN_HOST@VM:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.60.1    0.0.0.0         UG        0 0          0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
192.168.53.0    192.168.60.1    255.255.255.0   UG        0 0          0 enp0s3
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
[04/24/2018 22:12] VPN_HOST@VM:~$
```

The ip should be the tun0 ip on client. Since this is the source ip of packets received by the host.

Step 5:

1	2018-04-24 21:16:48.8064704…	192.168.53.5	192.168.60.101	ICMP	100 Echo (ping) request id=0x0f00, seq=1/256, ttl=64 (reply i…
2	2018-04-24 21:16:48.8151687…	10.0.2.17	10.0.2.18	UDP	128 60003 → 55555 Len=84
3	2018-04-24 21:16:48.8161345…	10.0.2.18	10.0.2.17	UDP	128 55555 → 60003 Len=84
4	2018-04-24 21:16:48.8162013…	192.168.60.101	192.168.53.5	ICMP	100 Echo (ping) reply id=0x0f00, seq=1/256, ttl=63 (request…

1	2018-04-24 21:25:36.1275615…	192.168.53.5	192.168.60.101	TCP	76 45744 → 23 [SYN] Seq=3106409096 Win=29200
2	2018-04-24 21:25:36.1275971…	10.0.2.17	10.0.2.18	UDP	104 60003 → 55555 Len=60
3	2018-04-24 21:25:36.1284372…	10.0.2.18	10.0.2.17	UDP	104 55555 → 60003 Len=60
4	2018-04-24 21:25:36.1284943…	192.168.60.101	192.168.53.5	TCP	76 23 → 45744 [SYN, ACK] Seq=4057949308 Ack=

Ping and telnet host V from client. The first screenshot is ping result: 1,4 are packets monitored in the tun interface. The second screenshot is telnet result: 1 and 4 are packets monitored in the tun interface.

Step 6:

```
Got a packet from the tunnel
^C
```

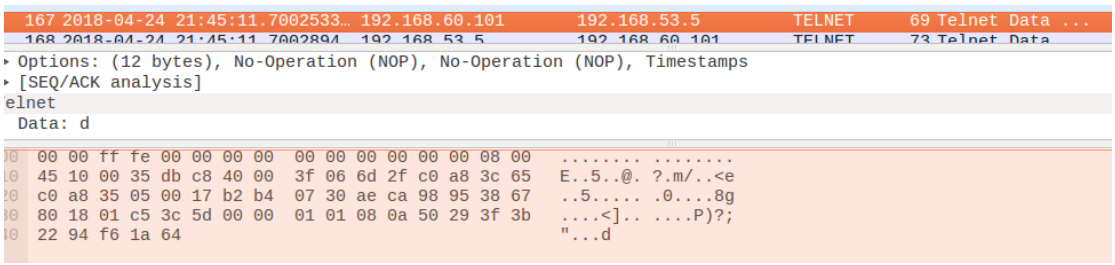
After we break the vpn server connection on the server side.

123	2018-04-24 21:44:44.2182618…	10.0.2.18	10.0.2.17	ICMP	125 Destination unreachable (Port unrea.
124	2018-04-24 21:44:44.4267177…	192.168.53.5	192.168.60.101	TCP	69 [TCP Keep-Alive] 45748 → 23 [PSH, A.
125	2018-04-24 21:44:44.4269042…	10.0.2.17	10.0.2.18	UDP	97 60003 → 55555 Len=53
126	2018-04-24 21:44:44.4283257…	10.0.2.18	10.0.2.17	ICMP	125 Destination unreachable (Port unrea.

We observed that packets send from the client side is not reachable.

```
[04/24/2018 21:44] VPN_Server@VM:~/.../VPN$ sudo ./vpnserver
[sudo] password for seed:
Connected with the client: E
```

```
[04/24/2018 21:43] Client1@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:
Last login: Tue Apr 24 21:25:57 EDT 2018 from 192.168.53.5 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-38-generic i686)

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

282 packages can be updated.
3 updates are security updates.

[04/24/2018 21:44] VPN_HOST@VM:~$ dfddfsddsfdsfds
```



When we reconnect the vpn on the server side. From the wirshark we notice that the unreachable data is sent out through the tunnel, the connection is resume.

Explanation:

Even the server is disconnected with client, the telnet program is still working. Since the connection is broken, the packets send from client is not reachable. TCP keeps resending the packets. That why we can see multiple error messages from the wirshark. But the data we typed in the telnet does not lost, they are buffered. If we now reconnect the server side, all data typed into telnet will eventually reach the telnet server due to TCP transmission and all the characters will show up in the terminal.

Task 3

Client code:

```
/*----------------TLS initialization ----------------*/
SSL *ssl   = setupTLSClient(hostname);

/*----------------Create a TCP connection ---------------*/
int sockfd = setupTCPClient(hostname, port);

/*----------------TLS handshake ----------------------*/

SSL_set_fd(ssl, sockfd);
int err = SSL_connect(ssl); CHK_SSL(err);
printf("SSL connection is successful\n");
printf ("SSL connection using %s\n", SSL_get_cipher(ssl));
```

First setup a ssl and tcp socket and combine them together.

```
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, ssl);
  if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, ssl);
}
```

In the while loop monitor the tun file descriptor and tcp file descriptor.

Sever code

```
ssl = SSL_new (ctx);

struct sockaddr_in sa_client;
size_t client_len;
int listen_sock = setupTCPServer();
 tunfd  = createTunDevice();
  sock = accept(listen_sock, (struct sockaddr*)&sa_client, &client_len);
SSL_set_fd (ssl, sock);
  err = SSL_accept (ssl);
  CHK_SSL(err);
  printf ("SSL connection established!\n");
```

The server also creates tcp and ssl, combine them together.

```
[04/29/2018 20:24] Server@VM:~/.../vpn$ sudo ./vpnserver
Enter PEM pass phrase:
SSL connection established!
```

Run the server code on machine server.

```
[04/29/2018 20:24] Client1@VM:~/.../vpn$ sudo ./vpnclient du.com 4433
SSL connection is successful
SSL connection using AES256-GCM-SHA384
```

Run the client on client machine.

```
[04/29/2018 20:52] Server@VM:~/.../vpn$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Ifac
e
0.0.0.0         192.168.60.1    0.0.0.0         UG        0 0          0 enp0
s8
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 enp0
s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0
s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0
s8
192.168.53.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.53.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0
s8
```

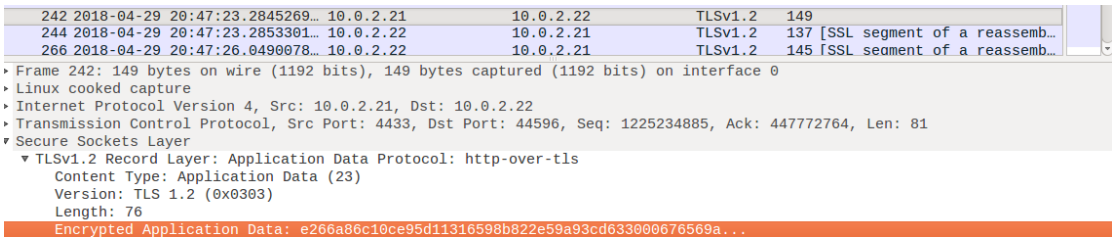
Edit the server routing table on server machine.

```
[04/29/2018 20:47] Client1@VM:~/.../vpn$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
192.168.53.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
192.168.60.0    0.0.0.0         255.255.255.0   U         0 0          0 tun0
```

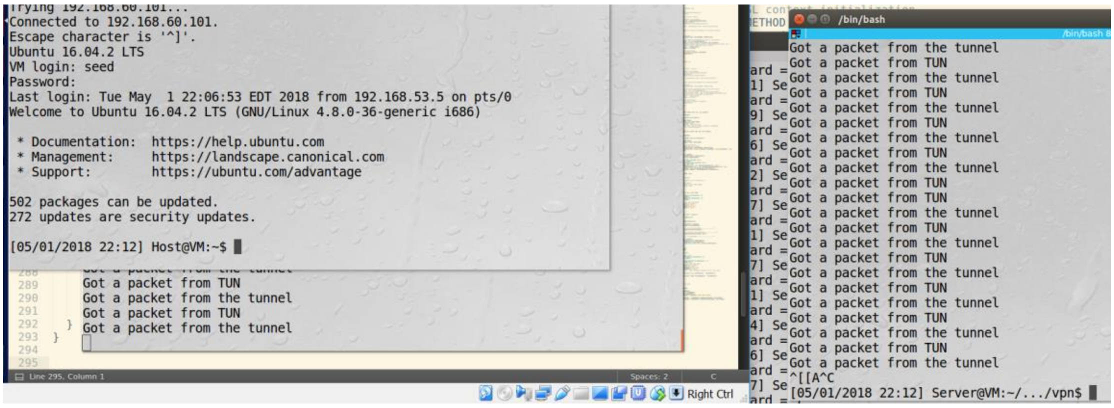
Edit the client routing table on client machine.

```
[04/29/2018 20:47] Client1@VM:~/.../vpn$ 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=1.17 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=3.61 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=3.29 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=1.72 ms
64 bytes from 192.168.60.101: icmp_seq=5 ttl=63 time=1.58 ms
^C
--- 192.168.60.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.179/2.279/3.610/0.981 ms
```

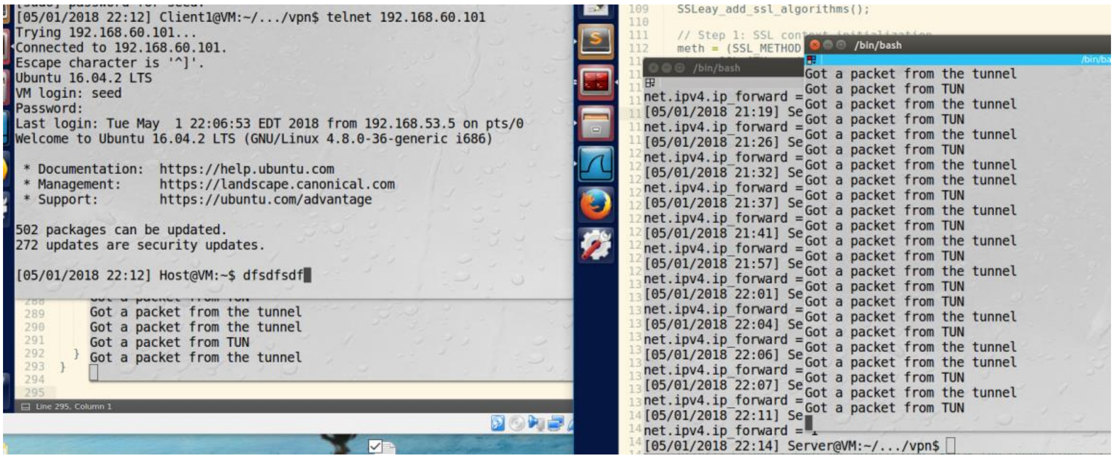
Ping host V on user machine.



Form the wirshark, we can see the data all encrypted.

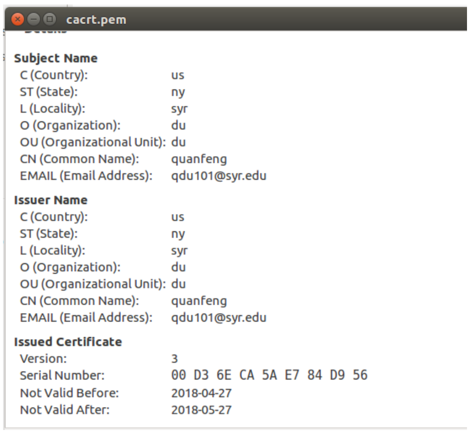


I telnet 192.168.60.101, then I disconnect from the server side. The telnet program cannot show what I typed in the terminal.



However, when I connect the server again, I can see what I typed in the terminal just showed out.

Task 4:



This is the root CA generated followed PKI lab instruction.

Use the root CA to sign for my VPN server, which is du.com

```
[04/29/2018 20:24] Client1@VM:~/.../vpn$ sudo ./vpnclient du.com 4433
SSL connection is successful
SSL connection using AES256-GCM-SHA384
```

Form task 3, when client visits du.com, the authority verification succeed and the ssl connection is successful.

```
    SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
    if(SSL_CTX_load_verify_locations(ctx,NULL, CA_DIR) < 1){
printf("Error setting the verify locations. \n");
exit(0);
    }
```

These two lines set the server verification and verifying is the server is the owner of the CA.

```
//enable host name check
X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);
X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);
```

These two lines code set the host name check.

```
SSL_CTX_use_certificate_file(ctx, "./server_cert/server-cert.pem", SSL_FILETYPE_PEM);
SSL_CTX_use_PrivateKey_file(ctx, "./server_cert/server-key.pem", SSL_FILETYPE_PEM);
```

In server, these two lines of code set up the certificates.

```
[04/29/2018 21:25] Client1@VM:~/.../vpn$ sudo ./vpnclient quanfeng.com 4433
3082016448:error:14090086:SSL routines:ssl3_get_server_certificate:certificate v
erify failed:s3 clnt.c:1264:
[04/29/2018 21:23] Server@VM:~/.../vpn$ sudo ./vpnserver
Enter PEM pass phrase:
3073644800:error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert certificat
e unknown:s3_pkt.c:1487:SSL alert number 46
```

When using quanfeng.com as the url, the connection cannot be established.

Even server provide the valid CA, this CA also can be verified by the root CA. The common name and the url typed in the client is not match. Therefore, the connection cannot be established.

Task 5:

```
char usr[60];
char *pwd;
char buf [2000];
char hpwd[200];
  //char hpwd[200];
size_t i;
 //usrAndpwd(usr, pwd);
printf("    User(most 60 characters): ");
fgets(usr, 60, stdin);
//printf();
pwd=getpass("Password(most 80 characters): ");
strncpy(hpwd, pwd, 15);
// exclude the '\n' from user input
if (usr[strlen(usr)-1] == '\n')
  usr[strlen(usr)-1] = '\0';
if (hpwd[strlen(hpwd)-1] == '\n')
  hpwd[strlen(hpwd)-1] = '\0';
//printf("Input     User: %s\nInput Password: %s\n", usr, pwd);
err = SSL_write (ssl, usr, strlen(usr));   CHK_SSL(err);
err = SSL_write (ssl, hpwd, strlen(hpwd));   CHK_SSL(err);
```

In the client, add client authentication code, send the user name and password to the server.

```
int isAuth = 0;
char usr[60]; char pwd[80];
  // get user name and password from client
err = SSL_read (ssl, usr, sizeof(usr) - 1);                    CHK_SSL(err);
usr[err] = '\0';
printf("Got username: %s\n", usr);
err = SSL_read (ssl, pwd, sizeof(pwd) - 1);                    CHK_SSL(err);
pwd[err] = '\0';
printf("Got password: %s\n", pwd);

isAuth = login(usr, pwd);
  //printf("\nisAuth: %d\n\n", isAuth);
if (isAuth == -1)
    return 0;
```

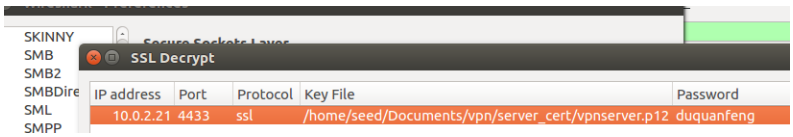
```
int login(char *user, char * pwd){
  struct spwd *pw;
  char *epasswd;
  if(user==null||pwd==null) return -1;

  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);
  if (strcmp(epasswd, pw->sp_pwdp)) {
    return -1;
  }
  return 1;
}
```

In server, read user and password, use shadow to verify the user.

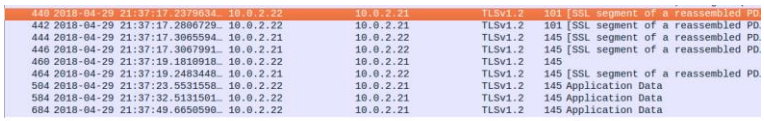
```
[05/01/2018 22:16] Client1@VM:~/.../vpn$ sudo
[sudo] password for seed:
SSL connection is successful
SSL connection using AES256-GCM-SHA384
    User(most 60 characters): seed
Password(most 80 characters):
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

```
[05/01/2018 22:15] Server@VM:~/.../vpn$ make
gcc -o vpnserver vpnserver.c -lssl -lcrypto -lcrypt
gcc -o vpnclient vpnclient.c -lssl -lcrypto -lcrypt
[05/01/2018 22:22] Server@VM:~/.../vpn$ sudo ./vpnserver
Enter PEM pass phrase:
SSL connection established!
Receive Client username: seed
Login name: seed
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

The server can verify the user.



Add the server key in the wirshark.



On the wirshark, the data can be decrypted. The user name and password all can be decrypted.