VPN Lab

Config:

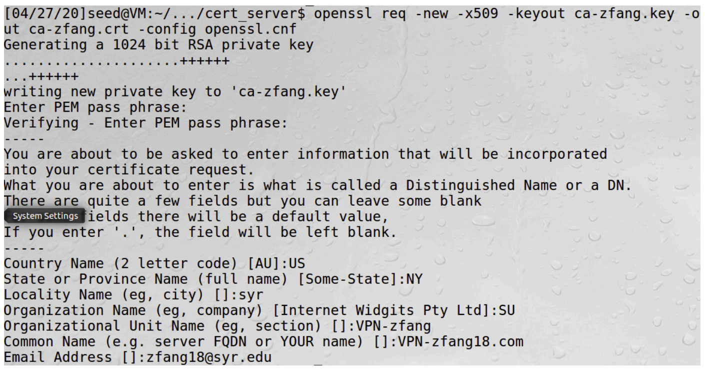
VPN Client: 10.0.2.7

Gateway two interface: 10.0.2.8, 192.168.60.1

Host V: 192.168.60.101

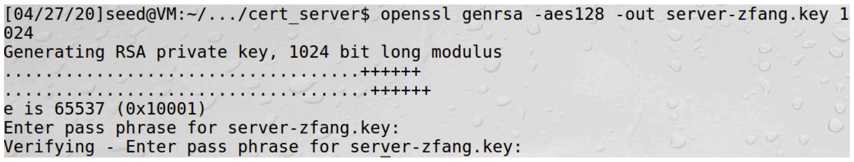
Task3:

On the client side, first we want create the self-signed root CA and write the private key and certificate:



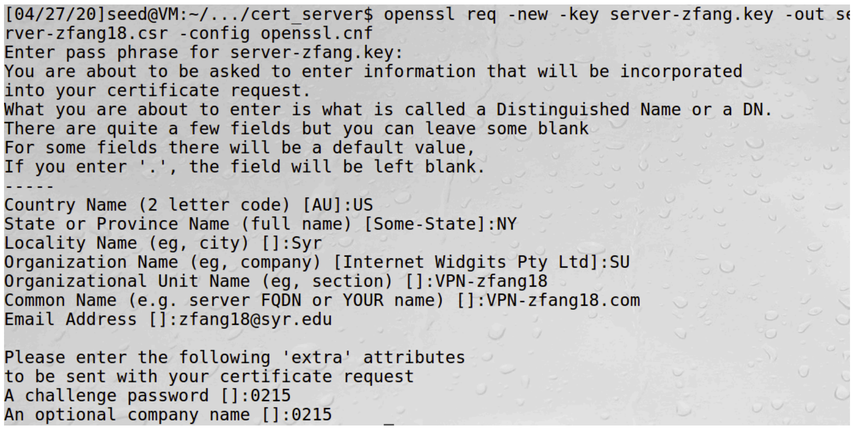
```
[04/27/20]seed@VM:~/.../cert_server$ openssl req -new -x509 -keyout ca-zfang.key -o
ut ca-zfang.crt -config openssl.cnf
Generating a 1024 bit RSA private key
....................++++++
...++++++
writing new private key to 'ca-zfang.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
          ields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NY
Locality Name (eg, city) []:syr
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SU
Organizational Unit Name (eg, section) []:VPN-zfang
Common Name (e.g. server FQDN or YOUR name) []:VPN-zfang18.com
Email Address []:zfang18@syr.edu
```

Write public key and private key for the server



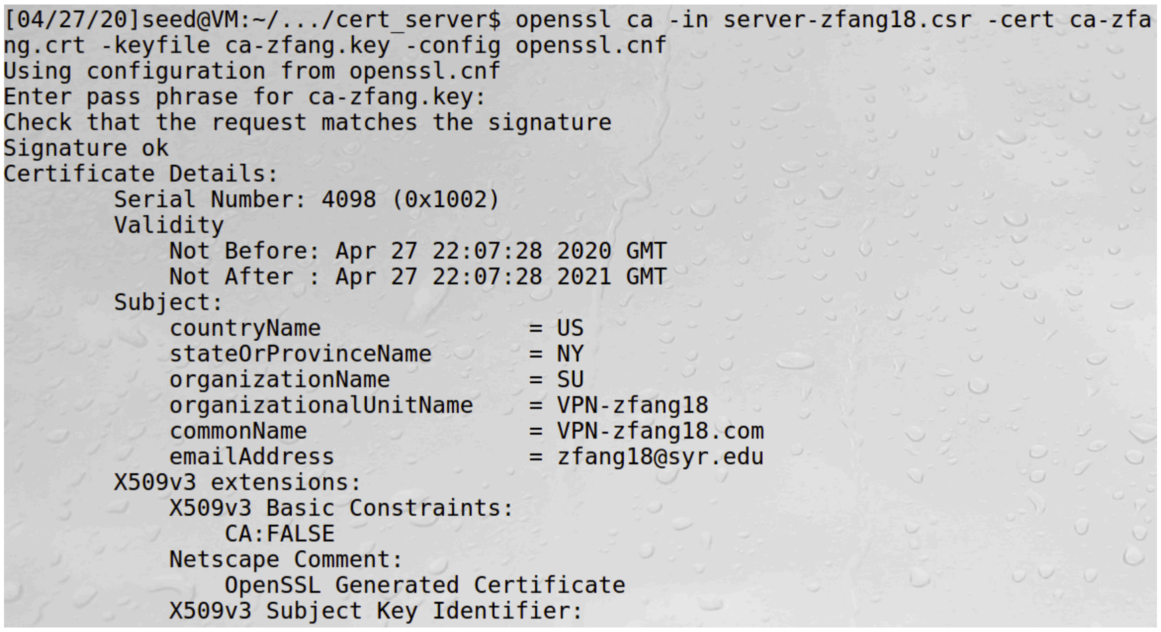
```
[04/27/20]seed@VM:~/.../cert_server$ openssl genrsa -aes128 -out server-zfang.key 1
024
Generating RSA private key, 1024 bit long modulus
..................................++++++
...................................++++++
e is 65537 (0x10001)
Enter pass phrase for server-zfang.key:
Verifying - Enter pass phrase for server-zfang.key:
```

Create CSR for website and we set the Common name VPN-zfang18.com

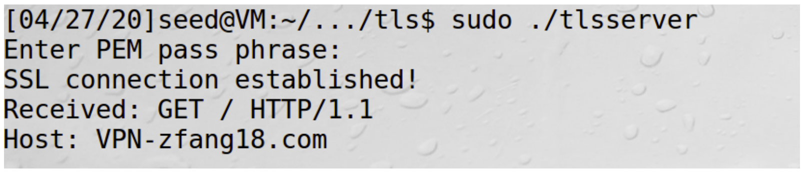


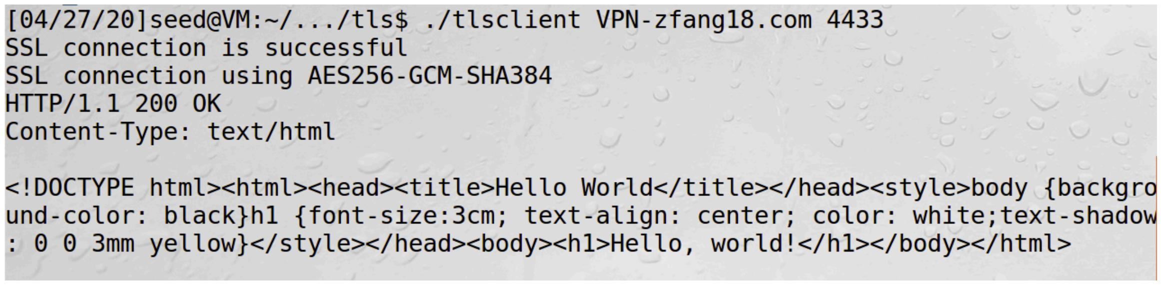
```
[04/27/20]seed@VM:~/.../cert_server$ openssl req -new -key server-zfang.key -out se
rver-zfang18.csr -config openssl.cnf
Enter pass phrase for server-zfang.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NY
Locality Name (eg, city) []:Syr
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SU
Organizational Unit Name (eg, section) []:VPN-zfang18
Common Name (e.g. server FQDN or YOUR name) []:VPN-zfang18.com
Email Address []:zfang18@syr.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:0215
An optional company name []:0215
```

Then generate certificate

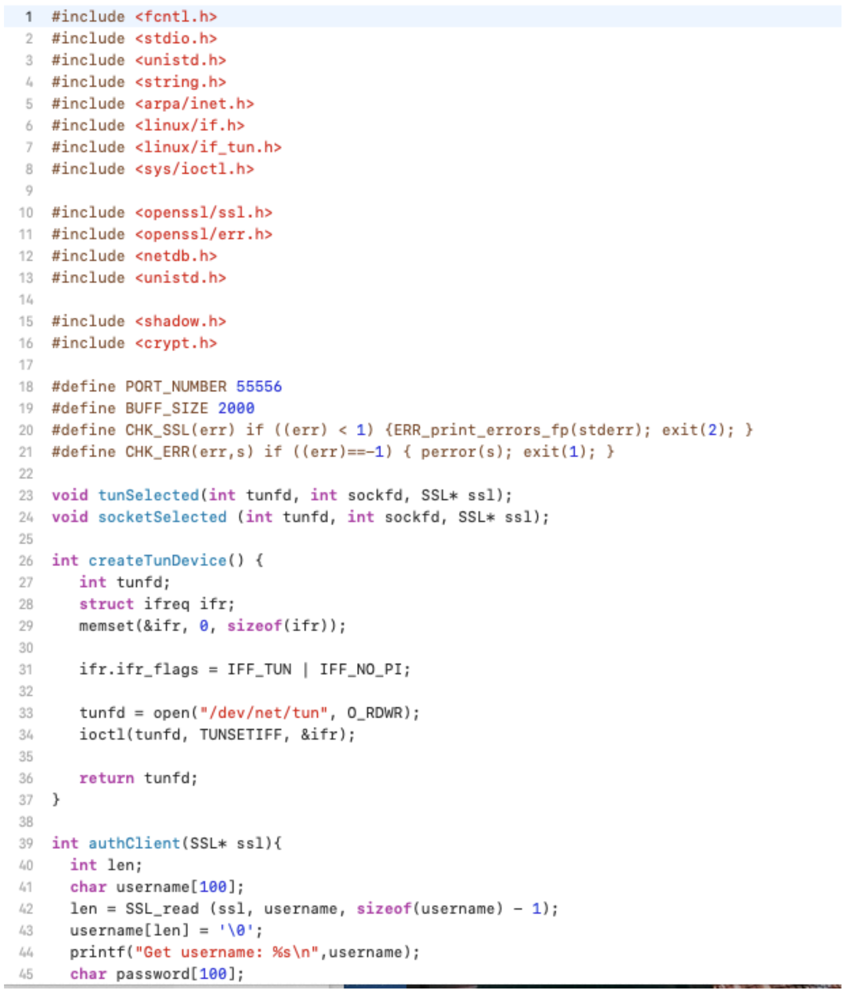

```
[04/27/20]seed@VM:~/.../cert_server$ openssl ca -in server-zfang18.csr -cert ca-zfa
ng.crt -keyfile ca-zfang.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca-zfang.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4098 (0x1002)
        Validity
            Not Before: Apr 27 22:07:28 2020 GMT
            Not After : Apr 27 22:07:28 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = NY
            organizationName          = SU
            organizationalUnitName    = VPN-zfang18
            commonName                = VPN-zfang18.com
            emailAddress              = zfang18@syr.edu
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
```

Then after we set the code pointing to our key and certificate file, we run the server on Gateway machine first, after inputting the passwd then run tls client:

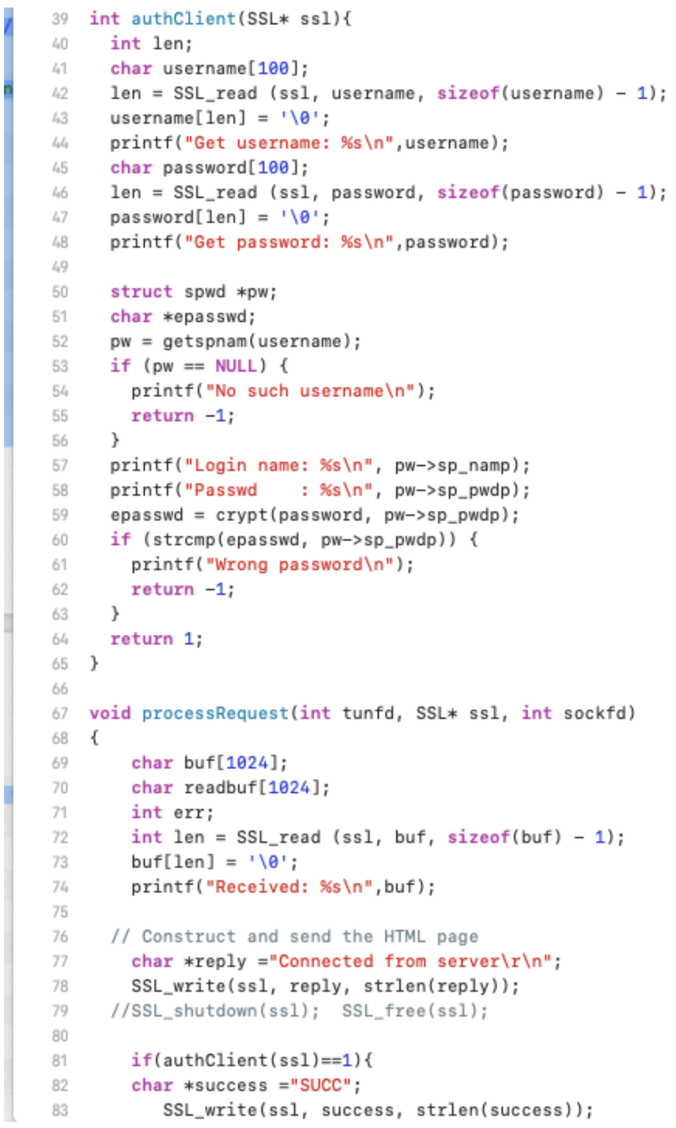

```
[04/27/20]seed@VM:~/.../tls$ sudo ./tlsserver
Enter PEM pass phrase:
SSL connection established!
Received: GET / HTTP/1.1
Host: VPN-zfang18.com
```


```
[04/27/20]seed@VM:~/.../tls$ ./tlsclient VPN-zfang18.com 4433
SSL connection is successful
SSL connection using AES256-GCM-SHA384
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html><html><head><title>Hello World</title></head><style>body {backgro
und-color: black}h1 {font-size:3cm; text-align: center; color: white;text-shadow
: 0 0 3mm yellow}</style></head><body><h1>Hello, world!</h1></body></html>
```

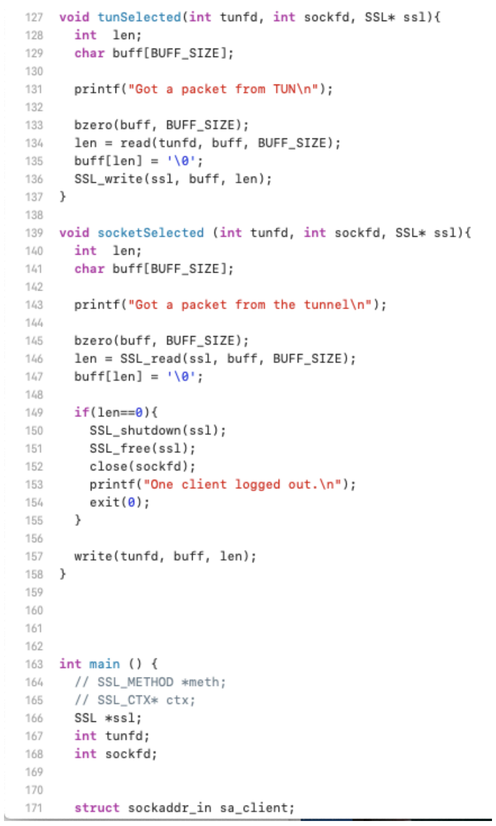
The following is VPN server source code:



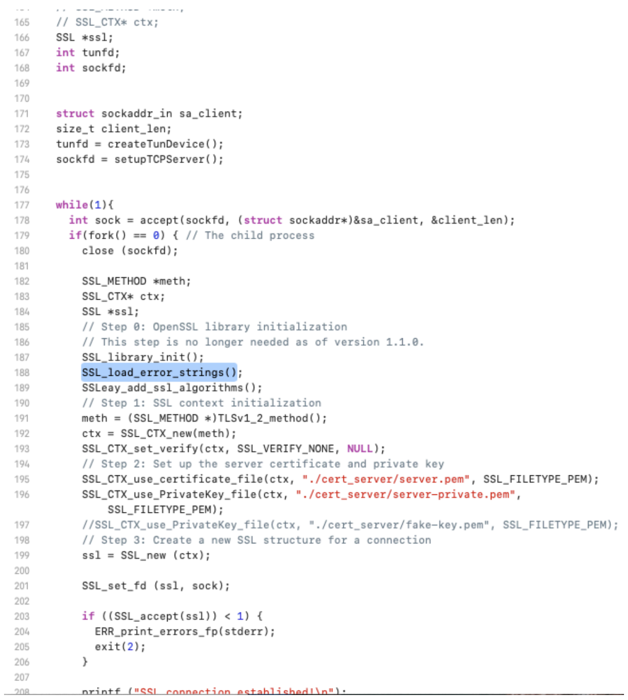
```c
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>

#include <openssl/ssl.h>
#include <openssl/err.h>
#include <netdb.h>
#include <unistd.h>

#include <shadow.h>
#include <crypt.h>

#define PORT_NUMBER 55556
#define BUFF_SIZE 2000
#define CHK_SSL(err) if ((err) < 1) {ERR_print_errors_fp(stderr); exit(2); }
#define CHK_ERR(err,s) if ((err)==-1) { perror(s); exit(1); }

void tunSelected(int tunfd, int sockfd, SSL* ssl);
void socketSelected (int tunfd, int sockfd, SSL* ssl);

int createTunDevice() {
   int tunfd;
   struct ifreq ifr;
   memset(&ifr, 0, sizeof(ifr));

   ifr.ifr_flags = IFF_TUN | IFF_NO_PI;

   tunfd = open("/dev/net/tun", O_RDWR);
   ioctl(tunfd, TUNSETIFF, &ifr);

   return tunfd;
}

int authClient(SSL* ssl){
  int len;
  char username[100];
  len = SSL_read (ssl, username, sizeof(username) - 1);
  username[len] = '\0';
  printf("Get username: %s\n",username);
  char password[100];
```



```c
int authClient(SSL* ssl){
  int len;
  char username[100];
  len = SSL_read (ssl, username, sizeof(username) - 1);
  username[len] = '\0';
  printf("Get username: %s\n",username);
  char password[100];
  len = SSL_read (ssl, password, sizeof(password) - 1);
  password[len] = '\0';
  printf("Get password: %s\n",password);

  struct spwd *pw;
  char *epasswd;
  pw = getspnam(username);
  if (pw == NULL) {
    printf("No such username\n");
    return -1;
  }
  printf("Login name: %s\n", pw->sp_namp);
  printf("Passwd    : %s\n", pw->sp_pwdp);
  epasswd = crypt(password, pw->sp_pwdp);
  if (strcmp(epasswd, pw->sp_pwdp)) {
    printf("Wrong password\n");
    return -1;
  }
  return 1;
}

void processRequest(int tunfd, SSL* ssl, int sockfd)
{
    char buf[1024];
    char readbuf[1024];
    int err;
    int len = SSL_read (ssl, buf, sizeof(buf) - 1);
    buf[len] = '\0';
    printf("Received: %s\n",buf);

  // Construct and send the HTML page
    char *reply ="Connected from server\r\n";
    SSL_write(ssl, reply, strlen(reply));
  //SSL_shutdown(ssl);  SSL_free(ssl);

    if(authClient(ssl)==1){
    char *success ="SUCC";
       SSL_write(ssl, success, strlen(success));
  else{
    char *fail = "FAIL";
    SSL_write(ssl, fail, strlen(fail));
    // here might need a better function
    return;
  }

  int i=0;
  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, sockfd, ssl);
    if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd, ssl);
    }
    printf("finished\n");
}

int setupTCPServer()
{
  struct sockaddr_in sa_server;
  int listen_sock;
    //We may have to use AF_INET here, dont know why
  listen_sock= socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  CHK_ERR(listen_sock, "socket");
  memset (&sa_server, '\0', sizeof(sa_server));
  sa_server.sin_family      = AF_INET;
  sa_server.sin_addr.s_addr = htonl(INADDR_ANY);
  sa_server.sin_port        = htons (PORT_NUMBER);

  int err = bind(listen_sock, (struct sockaddr*)&sa_server, sizeof(sa_server));
  CHK_ERR(err, "bind");

  err = listen(listen_sock, 5);
  CHK_ERR(err, "listen");
  return listen_sock;
}

void tunSelected(int tunfd, int sockfd, SSL* ssl){
  int  len;
  char buff[BUFF_SIZE];
```



```c
void tunSelected(int tunfd, int sockfd, SSL* ssl){
  int  len;
  char buff[BUFF_SIZE];

  printf("Got a packet from TUN\n");

  bzero(buff, BUFF_SIZE);
  len = read(tunfd, buff, BUFF_SIZE);
  buff[len] = '\0';
  SSL_write(ssl, buff, len);
}

void socketSelected (int tunfd, int sockfd, SSL* ssl){
  int  len;
  char buff[BUFF_SIZE];

  printf("Got a packet from the tunnel\n");

  bzero(buff, BUFF_SIZE);
  len = SSL_read(ssl, buff, BUFF_SIZE);
  buff[len] = '\0';

  if(len==0){
    SSL_shutdown(ssl);
    SSL_free(ssl);
    close(sockfd);
    printf("One client logged out.\n");
    exit(0);
  }

  write(tunfd, buff, len);
}




int main () {
  // SSL_METHOD *meth;
  // SSL_CTX* ctx;
  SSL *ssl;
  int tunfd;
  int sockfd;


  struct sockaddr_in sa_client;
```



```c
  // SSL_CTX* ctx;
  SSL *ssl;
  int tunfd;
  int sockfd;


  struct sockaddr_in sa_client;
  size_t client_len;
  tunfd = createTunDevice();
  sockfd = setupTCPServer();


  while(1){
    int sock = accept(sockfd, (struct sockaddr*)&sa_client, &client_len);
    if(fork() == 0) { // The child process
      close (sockfd);

      SSL_METHOD *meth;
      SSL_CTX* ctx;
      SSL *ssl;
      // Step 0: OpenSSL library initialization
      // This step is no longer needed as of version 1.1.0.
      SSL_library_init();
      SSL_load_error_strings();
      SSLeay_add_ssl_algorithms();
      // Step 1: SSL context initialization
      meth = (SSL_METHOD *)TLSv1_2_method();
      ctx = SSL_CTX_new(meth);
      SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);
      // Step 2: Set up the server certificate and private key
      SSL_CTX_use_certificate_file(ctx, "./cert_server/server.pem", SSL_FILETYPE_PEM);
      SSL_CTX_use_PrivateKey_file(ctx, "./cert_server/server-private.pem",
          SSL_FILETYPE_PEM);
      //SSL_CTX_use_PrivateKey_file(ctx, "./cert_server/fake-key.pem", SSL_FILETYPE_PEM);
      // Step 3: Create a new SSL structure for a connection
      ssl = SSL_new (ctx);

      SSL_set_fd (ssl, sock);

      if ((SSL_accept(ssl)) < 1) {
        ERR_print_errors_fp(stderr);
        exit(2);
      }

      printf ("SSL connection established!\n");
```

We turned off the verification first on SSL_CTX_set_verify. Then we also set up the server certificate and private key file that we have generated before in the cert_server folder.

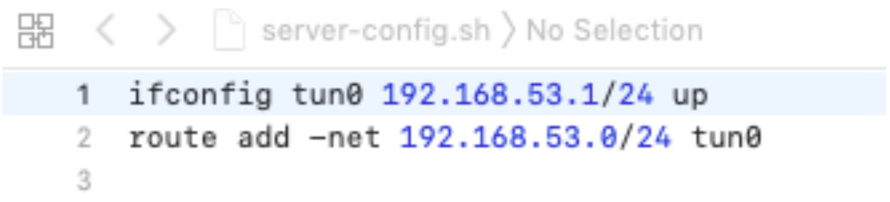
For VPN client we have the following code:

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <netdb.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <termios.h>

#define BUFF_SIZE 2000
#define CHK_SSL(err) if ((err) < 1) { ERR_print_errors_fp(stderr); exit(2); }
#define CA_DIR "ca_client"
 struct sockaddr_in peerAddr;
 struct addrinfo hints, *result;
 int PORT_NUMBER = 55556;
 const char *hostname;

 int createTunDevice() {
 int tunfd;
 struct ifreq ifr;
 memset(&ifr, 0, sizeof(ifr));

 ifr.ifr_flags = IFF_TUN | IFF_NO_PI;
 tunfd = open("/dev/net/tun", O_RDWR);
 ioctl(tunfd, TUNSETIFF, &ifr);

 return tunfd;
 }

 int verify_callback(int preverify_ok, X509_STORE_CTX *x509_ctx)
 {
 char buf[300];

 X509* cert = X509_STORE_CTX_get_current_cert(x509_ctx);
```

```
  X509_NAME_oneline(X509_get_subject_name(cert), buf, 300);
  printf("subject= %s\n", buf);

  if (preverify_ok == 1) {
  printf("Verification passed.\n");
  } else {
  int err = X509_STORE_CTX_get_error(x509_ctx);
  printf("Verification failed: %s.\n",X509_verify_cert_error_string(err));
  }
  }
SSL* setupTLSClient(const char* hostname, SSL_CTX* ctx)
  {

// Step 0: OpenSSL library initialization
// This step is no longer needed as of version 1.1.0.
SSL_library_init();
SSL_load_error_strings();

  SSLeay_add_ssl_algorithms();
  SSL_METHOD *meth;
  SSL* ssl;

  meth = (SSL_METHOD *)TLSv1_2_method();
  ctx = SSL_CTX_new(meth);

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

  X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);
X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);

  return ssl;
}

int connectToTCPServer(const char *hostname){

    //get host IP address
```

```
        hints.ai_family = AF_INET;
        int error = getaddrinfo(hostname, NULL, &hints, &result);
        if (error) {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(error));
        exit(1); }
        struct sockaddr_in* ip = (struct sockaddr_in *) result->ai_addr;
        int sockfd= socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
          // Create a TCP socket

        memset(&peerAddr, 0, sizeof(peerAddr));
        peerAddr.sin_family = AF_INET;
        peerAddr.sin_port = htons(PORT_NUMBER);
        peerAddr.sin_addr.s_addr = inet_addr((char *)inet_ntoa(ip->sin_addr));

        connect(sockfd, (struct sockaddr*) &peerAddr, sizeof(peerAddr));
        return sockfd;
}




void tunSelected(int tunfd, int sockfd, SSL *ssl){
int len;
char buff[BUFF_SIZE];
printf("Got a packet from TUN\n");
bzero(buff, BUFF_SIZE);
len = read(tunfd, buff, BUFF_SIZE);
SSL_write (ssl, buff, sizeof(buff)-1);
}
void socketSelected (int tunfd, int sockfd, SSL *ssl){
int len;
char buff[BUFF_SIZE];
// printf("Got a packet from the tunnel\n");
bzero(buff, BUFF_SIZE);
int err = SSL_read (ssl, buff, sizeof(buff)-1);
buff[err] = '\0';
write(tunfd, buff, err);
}

int main (int argc, char * argv[]) {
```

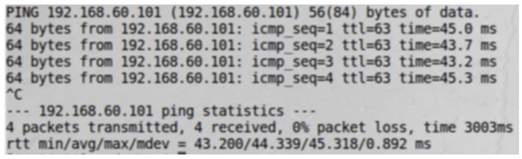
```
const char *hostname;
SSL_CTX *ctx;
int tunfd, sockfd;
printf("111");
if (argc > 1) hostname = argv[1];
else {
printf("Please enter a legal host name.\n");
return 0; }
if (argc > 2) PORT_NUMBER = atoi(argv[2]);
tunfd = createTunDevice();
//TLS initialization
SSL *ssl = setupTLSClient(hostname, ctx);


//create a TCP conncetion
  sockfd = connectToTCPServer(hostname);

  //TLS handshake
  char readbuf[2000];
  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));

  err = SSL_write (ssl, "Connect to Server!", strlen("Connect to Server!"
));
  err = SSL_read (ssl, readbuf, sizeof(readbuf)-1);
  readbuf[err] = '\0';
  printf("receive: %s\n", readbuf);

  int i=0;
  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, sockfd, ssl);
  if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd, ssl
);
 }
 printf("finished\n");
 }
```



```
ifconfig tun0 192.168.53.1/24 up
route add -net 192.168.53.0/24 tun0
```

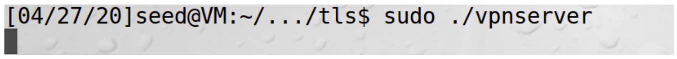
Run the script:

After up the tun0, add some entry in route table, then we can make a test. Try to ping HostV:

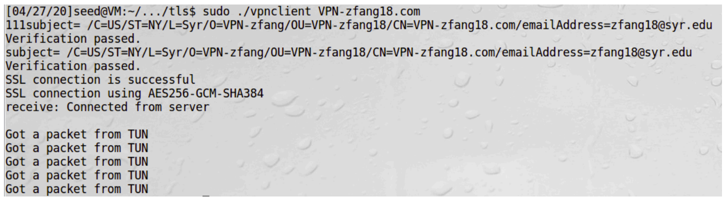


```
PING 192.168.60.101 (192.168.60.101) 56(84) bytes of data.
64 bytes from 192.168.60.101: icmp_seq=1 ttl=63 time=45.0 ms
64 bytes from 192.168.60.101: icmp_seq=2 ttl=63 time=43.7 ms
64 bytes from 192.168.60.101: icmp_seq=3 ttl=63 time=43.2 ms
64 bytes from 192.168.60.101: icmp_seq=4 ttl=63 time=45.3 ms
^C
--- 192.168.60.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 43.200/44.339/45.318/0.892 ms
```

Then We run server first then try run client:



```
[04/27/20]seed@VM:~/.../tls$ sudo ./vpnserver
```



```
[04/27/20]seed@VM:~/.../tls$ sudo ./vpnclient VPN-zfang18.com
111subject= /C=US/ST=NY/L=Syr/O=VPN-zfang/OU=VPN-zfang18/CN=VPN-zfang18.com/emailAddress=zfang18@syr.edu
Verification passed.
subject= /C=US/ST=NY/L=Syr/O=VPN-zfang/OU=VPN-zfang18/CN=VPN-zfang18.com/emailAddress=zfang18@syr.edu
Verification passed.
SSL connection is successful
SSL connection using AES256-GCM-SHA384
receive: Connected from server

Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```