Computer Networks and Applications, Fall 2023

Programming Assignment Part 3

B11705009 An-Che Liang

Compile

I use the `g++` compiler to complie the c++ source code into executable program. The command is as follow:

Server

```
g++ -o client-secure client-secure.cpp -lssl -lcrypto
```

Client

```
g++ -o server-secure server-secure.cpp -lssl -lcrypto
```

Execution

The `server-secure` program is a text-based terminal appilcation. To run the program, type `./server-secure <port_number>` in the containing folder's terminal. Then a text-based interface will appear in the terminal if the provided port number is valid.

Once the server is successfully attachad to the socket with provided port number, it will spawn a new thread to listen to incoming clients. The program statically allocate memory for 64 concurrent threads, one thread will start when accepted a new connection and terminate when the client disconnects.

Also, the server program provides authentication capability. One client can only perform the `List` command once logged in, and will prevent ant other client to login to active accounts.

The `client-secure` program is a text-based terminal appilcation. To run the program, type `./client-secure` in the containing folder's terminal. Then a text-based interface will appear in the terminal.

First, the client program will ask where to connect to the server application, include the IP address and port numeber. The client program will exit if it fails to connect to the socket.

Then, you will be given 5 commands to choose from:

1. Register
2. Login
3. Check server status (equivalent to List command on the server)
4. Terminate connection (equivalent to Exit command on the server)
5. Initiate payment

Enter the desired command accordingly, the client will ask additional information to perform some commands, such as it will ask for payee's name and payment amount when you initiate a new payment.

Encryption

To enable encryption for all the socket communication, I use the `openssl` library to enforce RSA encryption on the content of the messages.

First, I create a self-signing CA using the CLI interface of `openssl` with the following script:

```
openssl req -new -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem -days 3650
```

Then with our own CA, we can sign the `serverkey.pem` and `serverkey.pem` files. Then in the server application and client application, we can use `SSL_set_fd()` to direct the stream of socket into ssl connection and `SSL_accept()` to initiate automatic handshake across two SSL sockets. Then we can simply use `SSL_write()` and `SSL_read()` to tranceive information securely.

Environment

This program is written and tested on the Ubuntu 22.04 linux distribution with x86-64 architecture CPU.

Reference

1. GeekForGeeks link
2. Baeldung link
3. OpenSSL Documentation link