Chat Portal Application With TLS

Note

I have used the following assumption throughout this readme file: **Alice wants to chat with Bob.**

So, **Alice** is the **communication initiator** and **Bob** is the **comunication responder**.

Directory Structure

Follwing is the directory structure of my system:

1. CA_Cert: The directory contains X.509 certificate (**cacert.pem**) and private key (**cakey.pem**) of the CA.

2. Client_Cert: The directory contains X.509 certificates and private keys of the users. The directory has two sub directories:

(**i) cert –** The directory contains the X.509 certificates of the users.
(ii) key – The directory contains private keys of the users.

3. header: The directory contains all the header files that are used by my system.

5. helper: The directory contains all the helper code that is used by my server and client code, such as, initializing the chat server, initializaing TLS connection, etc.

System Overview?

The system contains a CA (**CA.c**), a server (**server.c**) and a client (**client.c**).

* The CA issues and signs client certificates.

* Alice and Bob logs-in to the server and then the Alice communicates to the chat server to forward a request to Bob which starts listening process, waiting for connection on a certain port number.

* Alice thereafter establishes a TLS connection to the Bob's listening port. Bob (communication responder) responds with a regular **mutual TLS handshake** which involves sending the Bob's certificate to Alice (connection initiator) and Alice's certificate to Bob. Both parties (Alice and Bob) validate each other certificates. Upon successful authentication a TLS connection is established which the two parties (Alice and Bob) can now use to communicate securely.

Commands To Use The System?

Once Alice and Bob are logged in to the chat server, they can use the following commands:

1. chat : Alice sends chat request to Bob. Using **chat** command. Bob starts a listening process on a certain port number.
3. connect : Once Bob starts a listening process and acknowledges Alice about it, Alice will connect to Bob using **connect** command to establish a TLS connection.
3. msg : Once the TLS connection is established, Alice and Bob can send chat messages to each other using **msg** command.
4. logout : To logout of the chat server.

Assumptions made by me

Following are the assumptions that i made while making the code :

1. Username and password will be maximum 40 characters long.
2. Username and password will be continuous, i.e, no space is present in username and password.
3. Usernames are case sensitive.
4. Every message sent by a user will be maximum 920 characters long.
5. Every username is unique.

Corner cases handled by me

1. Maximum 20 users can login at a time on the chat portal.

2. If a user already has a valid certificate and he/she issues CSR to CA, then CA will not issue a new certificate. I am saving the X.509 certificate of the user with the username of the user. So, if a user - already having a certificate - issues a CSR (Certificate Signing Request) to the CA, then i am checking the validity period of the current X.509 certificate of the user. If the current certificate is invalid, then a new certificate is issued, else an an appropraite error message is displayed to the user.

3. A user (say Alice) can only issue chat request (to chat with another user, say Bob) if and only if both Alice and Bob have a X.509 certificate.

4. Two users A and B can only chat if both of them are logged in. If user A is not logged in and user B wants to chat with user A, then the server will send appropriate error message to the user B.

4. A user (say Alice) cannot connect to another user (say Bob) if Alice has not earlier sent a chat request to Bob or Bob has not yet accepted Alice’s chat request.