Taoufik Laaroussi

CNT 4403 / Spring 2021

Prof.: Shahram Jahani

4/10/2021

**Secure Chat Room**

The goal of this assignment is to create a secure chat room by implementing a secure connection using SSL. I modified the provided code to implement a simple chat room with user interface from the following link <https://cs.lmu.edu/~ray/notes/javanetexamples>. Then I changed the mode of connection between the client and server using Java SSLSocket instead of Java socket. The process to use SSLsocket is slightly different from using simple socket, because first we need to generate a private key, public key, trust store and key store. Then implement a handshake process to verify the keys between client and server. Finally, after the handshake process is done, the server and client will be able to communicate securely.

**Implementation steps:**

**First step: key generation**

1 - First Generate the server Certificate and public/private key and store it in keystore file

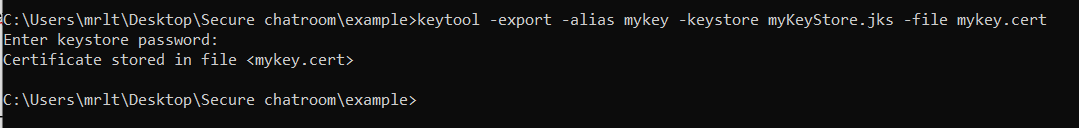
keytool -genkey -keyalg RSA -keysize 2048 -validity 360 -alias mykey -keystore myKeyStore.jks

Text

Description automatically generated

2 - Export the certficate and the public key that should be send to the client.

keytool -export -alias mykey -keystore myKeyStore.jks -file mykey.cert



3 - Add the key at the client side to a TrustedStore to trust the server

keytool -import -file mykey.cert -alias mykey -keystore myTrustStore.jts

Text

Description automatically generated

**Second step: Server and client implementation**

After keys creation I had to modify the provided code of a simple chat room from the link provided above. I modified a code on how to implement an SSLSocket connection from the following link: <https://docs.oracle.com/javase/10/security/sample-code-illustrating-secure-socket-connection-client-and-server.htm#JSSEC-GUID-B9103D0C-3E6A-4301-B558-461E4CB23DC9>.

The following screenshot shows the server side SSLSocket implementation:

**Handshake process**

Text

Description automatically generated

**Multiconnection handling process**

Text

Description automatically generated

The following screenshot shows the client side SSLSocket implementation:

**Handshake process with the server**

Graphical user interface, text, application

Description automatically generated

The loop in the code above is used to handle user input and send it to the server.

**How to compile and use the secure chat room**

1. All keys and files must be in the same folder.
2. Using the command line compile SSLClient.java and SSLServer.java using the following commands:

* javac SSLServer.java
* Javac SSLClient.java

A screenshot of a computer

Description automatically generated with low confidence

1. Then execute in order, the SSLServer only one time to keep it running, then in a different command line pages execute SSLClient as many clients as you want using the following commands.

* java SSLServer

Text

Description automatically generated with medium confidence

* java SSLClient (Note: this command is used to run a client each time) a user interface will shows up each time you execute this command.

Graphical user interface

Description automatically generated

1. Enter the username and then start performing some text exchange between clients.

Graphical user interface

Description automatically generated

1. To quit the chat tape ‘exit’ in the chat input text.Graphical user interface, text

   Description automatically generated

**For output, there is a small demo video showing how the program work. This video will be in the zip file.**

Finally, implementing a secure chat room was really a fun assignment, where I learned how to implement client server connection and secure this connection using SSL. During the whole semester we were learning about security and how it works in theory. In this assignment I got the opportunity to discover how security is implemented in real life.