UNIVERSIDAD DEL BÍO-BÍO

INFORME TAREA 2

Seguridad Informática

NICOLÁS OYARCE ABURTO
19.090.216-1
5 DE DICIEMBRE DE 2017

Para la realización de esta tarea se utilizó como base el código ejemplo de sockets seguros en lenguaje Java, ya que esta contiene la base para poder realizar un intercambio de mensajes entre Cliente y Servidor, lo que facilitó el desarrollo de todo el trabajo.

Los sockets seguros (SSL) permiten la creación de un canal seguro para la transferencia de datos en cualquier protocolo de la capa de aplicación, esto se logra a través de la utilización de certificados criptográficos que son otorgados por entidades certificadoras.

Para que exista autenticación mutua se deben crear una keystore y truststore para el cliente y servidor, también el objeto tipo SSLServerSocket debe invocar el método setNeedClientAuth y pasarle el parámetro true.

En este caso se utilizó la clase SSLSocket junto a la clase SSLContext para realizar la conexión segura, esto implica que las keystores y truststores deben ser cargadas en el objeto SSLContext para funcione el programa.

El trabajo principal consistió en gestionar los almacenes, llaves y certificados que utilizarían nuestro servidor y cliente, para ello se utilizaron los siguientes comandos de la herramienta keytool.

- Primero se generó el almacén de llaves del servidor y del cliente:

```
keytool -genkeypair -alias llaveServidor -keystore
llavesServidor.jks
keytool -genkeypair -alias llaveCliente -keystore
llavesCliente.jks
```

- Luego se generó el certificado del servidor y del cliente:

```
keytool -export -alias llaveServidor -file
certificadoServidor.cer -keystore llavesServidor.jks
keytool -export -alias llaveCliente -file
certificadoCliente.cer -keystore llavesCliente.jks
```

- Finalmente se generaron los almacenes del cliente y del servidor (truststore) y adicionalmente se importó:

- Certificado del servidor en el truststore del cliente para indicar que el cliente confía en el servidor, por lo que permitirá una conexión con él.

```
keytool -import -trustcacerts -file certificadoServidor.cer -alias certUBB -keystore truststore_cliente.jks
```

- Certificado del cliente en el truststore del servidor para indicar que el servidor confía en el cliente, por lo que permitirá una conexión con él.

```
keytool -import -trustcacerts -file certificadoCliente.cer -alias certCliente1 -keystore truststore_servidor.jks
```

Por último, se agregó la porción de código que permite traspasar a mayúsculas el mensaje enviado por el cliente, el cual se le debe agregar la fecha de recibo en el servidor. Esto se logró utilizando los métodos que provee Java para obtener la hora y para concatenar Strings.

A continuación, el código utilizado:

```
String str = entrada.readLine();

System.out.println("El cliente envió: " + str);

str = str.toUpperCase();

str = str.concat(" - Recibido a las: " + obtenerHora());
```

Método para obtener hora:

```
Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy - HH:mm:ss");

return sdf.format(cal.getTime());
```