CS2003 Net2 Practical 160003493 Tutor: Miguel Nacenta 14 November 2017

Overview

The requirement of this practical was to design and implement a secure voting system to be used in a class election. Students should be able to vote by connecting to a server, entering authentication information, and placing their vote. At the end of the ballot, the results of the election should be returned to all students. The voting should be anonymous and secure.

Design and Implementation

I implemented the voting system using a multi-threaded SSL server, which can securely connect to multiple clients. I did this by creating classes for a SSL server, SSL client, and a server thread.

I tried a number of methods of implementing the SSL connection, and eventually found out how to create a self-signed certificate and key pair for the server key store (in electionKeyStore). I then extracted the public key and linked certificate, added the public key, and added the certificate to the trusted certificates of an empty client key store (in clientKeyStore). Using both these key stores allowed me to create a secure SSL connection, in which the client doesn't have access to the server's private key.

I use the inbuilt Java type KeyStore, along with factories for key stores, trust stores, and SSL server and client sockets, to make the SSL connection. I also create an instance of the JavaServerThread class each time a client connects to the server, which allows me to deal with multiple client requests at once.

The server is run from the command line, within the src file with the command:
java JavaSSLServer ../registeredStudents.csv ../candidates.txt
The client is also run from the command line within the src file, using the command:
java JavaSSLClient

a) To ensure that only registered students can vote, I read in a .csv file containing the registered students, hash each line using the Java MessageDigest type, and then store the digests in an ArrayList which is passed to each server thread. When the client enters their authentication information, this is constructed back into .csv format, hashed, and then the digest is compared with those in the ArrayList.

b) To ensure each student has only one vote, after the vote is stored, that student's authentication hash is removed from the ArrayList.

c) Hashing the authentication information of the students ensures that the server doesn't store the plain text of the client's authentication information, and therefore can't tell which student votes for which candidate.

d) Voting was secured using the SSL connection, preventing message contents from being read using WireShark or similar programs.

e) Client authentication ensures that the only people who can place votes are students registered on the course. The file of registered students is only required by the server program, preventing distribution of personal information to clients.

f) The authentication and voting systems are simple text based protocols, with clear steps. Errors in authentication or voting are dealt with cleanly, by simply allowing the client to re-enter information.

g) The clients maintain a connection to the server until the ballot is closed. When this happens, the final results of the election are returned to each client.

h) A separate thread keeps track of the time of the ballot. The thread will close the server when an hour has elapsed, and also checks each second if all registered students have voted, in which case the ballot closes early.

Testing

To check that my client key store doesn't have access to the server's private key, I ran the following code in my createSSLClient method (since removed).

```
Certificate certificate = ks.getCertificate( alias: "cs2003client");
System.out.println(certificate);

Key privateKey = ks.getKey( alias: "cs2003client", password);
System.out.println(privateKey);
```

This prints out the public key, certificate and private key contained in the key store, and returns the following (public key and certificate abbreviated):

```
pc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$ java JavaSSLClient
[
[
  Version: V3
  Subject: CN=Daniel Key, OU=CS, O=St Andrews, L=St Andrews, ST=Fife, C=UK
  Signature Algorithm: SHA256withDSA, OID = 2.16.840.1.101.3.4.3.2

  Key:  Sun DSA Public Key
    Parameters:DSA
        p:     8f7935d9 b9aae9bf abed887a cf4951b6 f32ec59e 3baf3718 e8eac496 1f3
efd36
    06e74351 a9c41833 39b809e7 c2ae1c53 9ba7475b 85d011ad b8b47987 75498469
    5cac0e8f 14b33608 28a22ffa 27110a3d 62a99345 3409a0fe 696c4658 f84bdd20
```

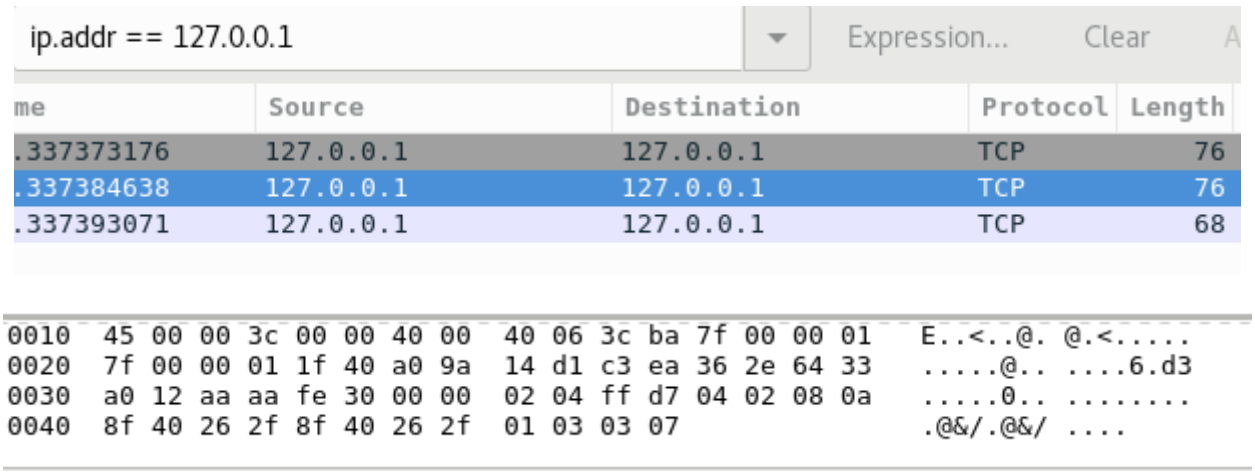
```
Certificate Extensions: 1
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 47 9A F9 61 1B BF 38 56   0B 16 7E 49 51 8C 29 87  G..a..8V...IQ.).
0010: FE 20 24 D2                                        . $.
]
]

]
  Algorithm: [SHA256withDSA]
  Signature:
0000: 30 3C 02 1C 25 44 BD 70   0E DB 8B 5D 02 C8 3F 33  0<..%D.p...]..?3
```

```
]
null
```

This shows that there is no reference to the server's private key in the client's key store.

I ran WireShark to attempt to decipher the messages being sent, and there didn't appear to be a filter you could use to decode them and understand the message contents.



Example client-side authentication and voting procedure:

```
^Cpc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$ java JavaSSLClient
Please enter your name
Bob
Please enter your matriculation number
16000004
Please enter your date of birth
05/08/1962

Authentication successful

Please cast your vote by typing the name of the candidate you wish to vote for.
The choices are as follows:
Julu
Ben
Jose

Ben

Your vote has been received and counted
The results of the election will be sent once the ballot closes

Final results:
Julu: 0 votes
Ben: 1 votes
Jose: 0 votes

pc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$
```

Example server-side authentication and voting procedure (showing two clients connecting):
A SocketException is thrown when the server closes down, which Java won't allow to be caught.

```
pc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$ java JavaSSLServer ../r
egisteredStudents.csv ../candidates.txt
SSL ServerSocket started
User is now connected to the server
User is now connected to the server
Ballot closed
java.net.SocketException: Socket closed
        at java.net.PlainSocketImpl.socketAccept(Native Method)
        at java.net.AbstractPlainSocketImpl.accept(AbstractPlainSocketImpl.java:
409)
        at java.net.ServerSocket.implAccept(ServerSocket.java:545)
        at sun.security.ssl.SSLServerSocketImpl.accept(SSLServerSocketImpl.java:
348)
        at JavaSSLServer.createSSLServer(JavaSSLServer.java:102)
        at JavaSSLServer.main(JavaSSLServer.java:32)
pc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$
```

Example results from all four students voting, and the server closing the ballot early:

```
Your vote has been received and counted
The results of the election will be sent once the ballot closes

Final results:
Julu: 1 votes
Ben: 1 votes
Jose: 2 votes

pc5-001-l:~/Documents/cs2003/Practicals/Week 9/src dgk2$
```

```
SSL ServerSocket started
User is now connected to the server
User is now connected to the server
User is now connected to the server
User is now connected to the server
Ballot closed
java.net.SocketException: Socket closed
```

Example of an unregistered student attempting to vote:

```
Please enter your name
Tim
Please enter your matriculation number
1
Please enter your date of birth
01/01/2000
Authentication failed, please try again
Please enter your name
```

Example of a student attempting to vote twice:

```
Please enter your name
Bob
Please enter your matriculation number
16000004
Please enter your date of birth
05/08/1962
Authentication failed, please try again
Please enter your name
```

Evaluation

This was a difficult practical to start, as generating keystores and creating an SSL connection in Java were not covered in lectures (a simple SSL server example was provided, but this didn't use keystores). I also had to self-teach multi-threaded servers, as this wasn't something covered in the module.

The actual implementation of the authentication and voting system wasn't too difficult, and planning out the basic protocols and methods required beforehand made the coding quicker and easier.

If I had more time, I would consider another way to store the registered student information in the program. Although the information is hashed, it is indexed in the same order as the plain text .csv file, meaning that a programmer could potentially infer which student was connected to the server if he had access to the .csv file.

Through writing this practical I learnt about creating self-signed certificates and key stores, implementing a multi-threaded server, using Java hashing functions, and creating methods which could work on multiple clients simultaneously.

Conclusion

This was a difficult practical, and I felt that more direction or instruction on parts of it, either in the specification or lectures, would have been very useful. Regardless, I managed to create a voting

system which met all of the specifications, and did so using SSL connections between a server and multiple clients.