Authentication

Goal

The purpose of this laboratory exercise is to provide hands on experience with the development of a user authentication mechanism.

Java Authentication

The traditional security mechanisms for the Java Virtual Machine (JVM) provide a means to enforce access controls based on where the code was downloaded from (*aka. the code base*) and who signed it. These access controls are needed because of the distributed nature of the Java platform where applications may consist of packages that are dynamically downloaded from different software providers and where an applet can be downloaded over a public network and then run locally.

However, the first versions of the Java 2 platform did not provide a way to enforce similar access controls based on the principal who runs the code. To provide this type of access control, the Java 2 security architecture requires the following:

- support for authentication of principals (determining who is actually running the code);
- extensions to the existing authorization components to enforce new access controls based on the authenticated principal.

The Authentication lab addresses the first of these problems by designing and implementing a simple authentication mechanism for a client/server application. Students who have not previously implemented an RMI application may consult the Java RMI Tutorial from Oracle or consult this short video on YouTube.

Lab Work

The first task is to write a simple client/server application using RMI. The example used in this lab is a mock-up of a simple authenticated print server, such as a print server installed in a small company. The print server must support the following operations:

> `print(String filename, String printer);` // prints file *filename* on the specified *printer*
> `queue(String printer);` // lists the print queue for a given printer on the user's display in lines of the form *<job number>* *<file name>*
> `topQueue(String printer, int job);` // moves *job* to the top of the queue
> `start();` // starts the print server
> `stop();` // stops the print server
> `restart();` // stops the print server, clears the print queue and starts the print server again
> `status(String printer);` // prints status of printer on the user's display
> `readConfig(String parameter);` // prints the value of the *parameter* on the user's display
> `setConfig(String parameter, String value);` // sets the *parameter* to *value*

These operations define the interface of the print server, but it is unnecessary to implement any printing capabilities for this lab, i.e. it is sufficient that the print server records the invocation of a particular operation in a logfile or prints it on the console. It must be possible to invoke all the print server operations defined in the interface from the client program.

This lab will design and implement a password based authentication mechanism for the print server, i.e. the print server must authenticate all requests from the client. For the purpose of this lab, it is not necessary to consider enrolment of users, i.e. authentication data structures can be populated by hand. The design and implementation of the print server must, however, consider the problems of password storage, password transport and password verification.

Password Storage

In relation to password storage, three possible solution must be considered: passwords stored in a "system" file, passwords stored in a "public" file, where cryptography is used to ensure confidentiality and integrity of the stored data; and passwords stored in a data base management system; these options are outlined below.

- **System File** Storing passwords in a system file relies on the operating system/file system protection mechanisms are used to ensure confidentiality and integrity of the stored data. This password file is normally not accessible to all users, so some mechanism (e.g. the SetUID mechanism in Unix) or system service is required to provide controlled access to the data stored in the file (similar to the DBMS storage described below.)
- **Public File** Storing passwords in a public file that can be read (but not necessarily written) by all users is the traditional way to store passwords on Unix systems. Confidentiality of the passwords is normally protected by cryptographic means, whereas integrity (e.g. binding users and passwords) may either be protected by the operating system/file system protection mechanism, i.e. normal users have no write access to the file (but how are passwords then updated?) or by cryptographic means.
- **DBMS** Storing passwords in a database (often unencrypted) relies on the security architecture, e.g. the access control mechanism, implemented in the DBMS to provide confidentiality and integrity of the stored passwords.

The analysis must briefly discuss how each of these three solutions may be implemented in the given context and compare and contrast the security offered by each of the proposed implementations (these considerations must be documented in the lab report outlined below). Other possible solutions may also be included in the analysis if they are considered relevant. Based on the discussions above, a solution should be selected and the reasons behind documented.

Password Transport

An analysis of the password transport problem must include a discussion of how to implement both individual request authentication and authenticated sessions, where the initial authentication is used to define an authenticated session, which implicitly authenticates messages exchanged. If authentication of invocation requires transfer of any additional parameters between client and server, these should simply be added to the interface defined above.

Password Verification

The password verification mechanism depends on the choice of password storage and must be explained in the report.

General Comments

For the purpose of this lab, it is acceptable to assume that secure communication between client and server is ensured by other means, e.g. TLS with server side certificates. It must, however, be explicitly stated if this assumption is made and the specific guarantees required by the channel (e.g. confidentiality, integrity and/or availability) must be specified. It is also possible to protect communications by cryptographic means, in this case the relevant techniques and technologies must be identified and discussed and it must be implemented correctly in the application. A complete and correct implementation of secure communication will count positively in the assessment of the report, *but only if everything else is well implemented and documented*.

Evaluation

This lab is a mandatory part of the course, which means that you have to hand in a small report, which will be evaluated and counts towards your final grade. The report should document your work, as defined above, and follow the normal structure of areport, and we recommend that you use the following structure:

1. **Introduction** (max 1 page)
   The introduction should provide a general introduction to the problem of authentication in client/server applications. It should define the scope of the answer, i.e. explicitly state what problems are considered, and outline the proposed solution. Finally, it should clearly state which of the identified goals are met by the developed software.
2. **Authentication** (max 3 pages)
   This section should provide a short introduction to the specific problem of password based authentication in client/server systems and analyse the problems relating to password storage (on the server), password transport (between client and server) and password verification.
3. **Design and Implementation** (max 3 pages including diagrams)
   A software design for the proposed solution must be presented and explained, i.e. why is this particular design chosen. The implementation of the designed authentication mechanism in the client server application must also be outlined in this section.
4. **Evaluation** (max 2 pages)
   This section should document that the requirements defined in Section 2 have been satisfied by the implementation. In particular, the evaluation should demonstrate that the user is always authenticated by the server before the service is invoked, e.g. the username and methodname may be written to a logfile every time a service is invoked.
   The evaluation should provide a simple summary of which of the requirements are satisfied and which are not.
5. **Conclusion** (max 1 page)
   The conclusions should summarize the problems addressed in the report and clearly identify which of the requirements are satisfied and which are not (a summary of Section 4). The conclusions may also include a brief outline of future work.

The full report should be limited to a maximum of 10 pages, excluding the source code. Source code for this lab should be included in a separate zip-archive.
The report and source code should be handed in electronicallyon Inside as indicated by the Authentication Lab assignment.

Useful Links

- Java RMI Tutorial

- Quick video introduction to RMI

Christian Damsgaard Jensen Christian.Jensen@imm.dtu.dk
Last modified 11 October 2021.