Authentication and Access Control Lab - Fall 2024

based on Christian Damsgaard Jensen's lab

November 1, 2024

Goal

This laboratory is in two parts. The purpose of the first part is to provide hands on experience with the development of a user authentication mechanism. Java is the official language of instruction for computer science at DTU Compute, which is why it has been chosen for this assignment.

The purpose of the second part is twofold: to provide hands on experience with the specification and enforcement of authorization policies and to provide an experimental framework for discussing access control policies. Students are assumed to have completed the first part.

At the end of the assignment, groups are expected to hand-in at least the following:

- one report for both part of the assignment (see Evaluation)
- one zip file containing the source code for the first part
- one zip file containing the source code for the access control list implementation
- one zip file containing the source code for the role access control implementation

It is important to hand-in the zip file for the first part so that if the implementation breaks during the improvements from the second part, the first part can still be independently assessed.

Part I: Authentication

Java Authentication

The traditional security mechanisms for the Java Virtual Machine (JVM) provide a mean to enforce access controls based on where the code was downloaded from, i.e. the code base, and who signs it. These access controls are needed because of the distributed nature of the Java platform where application 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 print server to the user's display
- setConfig(String parameter, String value); // sets the parameter on the print server 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, password verification and session management.

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 and cryptography 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). In particular, the report must explain how the chosen solution prevents users from learning or changing the password of other users. 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.

Session management

Signing in for every action on the print server is cumbersome. Conversly, using a print server does not require to be logged in for more than a certain amount of time. Implement a simple session management mechanism so that requests to the print server are authenticated for a period of time and that users get disconnected after this period of time. The program must be able to store a user session and check whether the session is still valid when making a request. If the session is not valid anymore, the user should be logged out. This exercice abstracts from the problem of the secure storage of the session and storing it in a Java object is enough. However, the generation of the session and the session lifetime must be discussed 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. 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.

Part II: Access Control

Access Control Scenario

Consider the printserver scenario defined in the first part. Not everybody working in the company has the same rights to access the print server. Alice is managing the print server, so she has the rights to perform all operations. Bob is the janitor who doubles as service technician, he has the rights to start, stop and restart the print server as well as inspect and modify the service parameters, i.e., invoke the *status*, *readConfig* and *setConfig* operations. Cecilia is a power user, who is allowed to print files and manage the print queue, i.e., use *queue* and *topQueue* as well as restart the print server when everything seems to be stuck. Finally, David, Erica, Fred and George are ordinary users who are only allowed to print files and display the print queue.

Lab Work

The first task is to modify the prototype print server developed in the first part, so that it implements the necessary code to enforce the access control policy outlined above. This means that all registered users must be included in the password-file/-database defined in the first part. This first implementation should be based on an access control list for the print server, i.e. the print server is considered as a single object with the different methods as the possible operations. The access control list must be specified in an external policy

file, or another form of external media, that is loaded when the print server starts, i.e. the policy must not be hardcoded into the program.

The second task is to identify roles and define a role hierarchy and permissions for each role, so that the access control policy outlined above can be implemented. The third task is to develop a second prototype, based on the prototype developed in the authentication lab, which enforces the access control policy using Role Based Access Control, i.e. based on the role hierarchy and permissions defined in Task 2. The role hierarchy must be specified in one or more external policy files, or the same form of external media used in Task 1, that is/are loaded when the print server starts.

Now consider the situation where Bob leaves the company and George takes over the responsibilities as service technician. At the same time, two new employees are hired: Henry, who should be granted the privileges of an ordinary user, and Ida who is a power user and should be given the same privileges as Cecilia.

The third task is to implement the necessary changes in the access control policy specifications of the two prototypes developed in this lab, so that they reflect the organisational changes in the company. The experience gained from these modifications will allow you to compare the management support for the two policy enforcement mechanisms, i.e. the flexibility and facility with which organisational changes can be reflected in the access control policy. This comparison must highlight the strengths and weaknesses of each implementation and discuss the expressive power and the ease of management supported by the two policy specification abstractions, e.g., which organisational changes are easily captured in both implementations and which are more easily specified in one implementation than in the other.

Note that depending on the choices you made to implement sessions in the first part, it is possible to use them to record and implement some aspects of access control. However, security implications of these choices must be discussed if you choose to use them.

Evaluation

This lab is an integral part of the course, which means that the report will be evaluated and contribute to your final grade. All positive contributions count, so it is always better to hand-in something than nothing, even if you are not personally satisfied by your results. The report should document your work, as defined above, and follow the normal structure of a report, and we recommend that you use the following structure:

1. **Introduction** (max 2 pages): The introduction should provide a general introduction to the problem of authentication and access control 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 5 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), password verification and session management. 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.

3. **Access Control Lists** (max 2 pages) This section should provide a short overview of the implementation of the access control lists and the motivation behind all non-trivial design choices.

4. **Role Based Access Control** (max 3 pages including diagrams): This section should document the results of the *role mining* process performed in in Task 2 and provide a short overview of the implementation of the role based access control mechanism implemented in Task 3 along with the motivation behind all non-trivial design choices. In particular, it must describe the syntax used to specify the RBAC policy.

5. **Changes in the policy** (max 2 page): This section documents the reflections and discussions of the final task.

6. **Evaluation** (max 4 pages): This section should document that the requirements defined in the previous sections 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 method name may be written to a logfile every time a service is invoked. This section should also document that the prototype enforces the access control policies defined in this assignment; both ACL and RBAC and both before and after the changes. The evaluation should provide a simple summary of which of the requirements are satisfied and which are not.

7. **Conclusion** (max 2 pages): 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 the Evaluation page). The conclusions may also include a brief outline of future work.

The full report for this part should be limited to a maximum of 20 pages, excluding the source code. Source code for this lab should be included in separate zip-archives.

Although the maximum limit for the report length is 20 pages, it is expected that most reports will be shorter.

Useful Links

- Java RMI tutorial
- this short video on YouTube
- Role Based Access Control