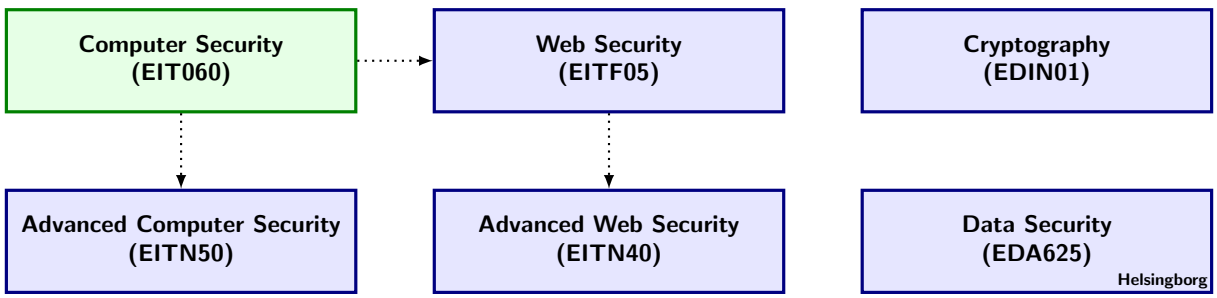
Computer Security 2014

Project 2: Medical Records and Secure Connections

- This project will be done in groups of 4 people.

Learning goals:

- Practicing your technical writing skills.
- Implementing a simple access control scheme.
- Applying knowledge gained in the course by identifying and documenting the strengths and weaknesses of your solution.
- Deeper technical familiarity with SSL.



1 Introduction

The purpose of this project is to study a "real case" problem that requires security mechanisms in its solution. The project aims at going through most of the stages of the development process, including design, implementation, evaluation and documentation. Medical records contain sensitive data and how these records should be handled is regulated by law [1].

Looking at the law, we can see that

- Someone working at a hospital has access to a medical record only if he/she treats the patient or if he/she needs the medical record for some other reason related to his/her work.
- A hospital can allow the patient to access his/her medical record provided that the patient is authenticated.
- An audit log must be kept that logs all access to a medical record. This can help preventing, detecting and reacting to security breaches and other types of unauthorized access to the medical records.
- Socialstyrelsen can decide that a medical record should be destroyed.

Interpreting the law, "Socialstyrelsen" provides more detailed regulations [2], see also [3]. In these documents, among other things, we can find that

- If an open network is used to manage medical records, the data must be transmitted such that unauthorized disclosure of the data is prevented, and strong authentication must be used, i.e., two factor authentication.
- People treating a patient must have enough access so that their work can be carried out safely, but they should not have more access than necessary. Only individual access is allowed, no group access.
- The audit log should contain information about who did what and when.

2 Project Description

You will not implement all aspects of all regulations, but the project will use them as a starting point. Let us consider the following scenario:

The medical records in a hospital are kept in a common database managed by a server. Individuals are allowed access to the database by remote access to the server using an open network.

Individuals are of the following types: patient, nurse, doctor, and government agency. Each patient has one or several medical records. Each record contains the name of the associated nurse and doctor (those who treated the patient), the hospital division (where it took place), and some medical data.

For the sake of this project, we interpret the law and the regulations as follows. Each nurse and doctor is associated with a hospital division. Access to medical records is done according to the following rules.

- A patient is allowed to read his/her own list of records.
- A nurse may read and write to all records associated with him/her, and also read all records associated with the same division.
- A doctor may read and write to all records associated with him/her, and also read all records associated with the same division. In addition, the doctor can create new records for a patient provided that the doctor is treating the patient. When creating the record, the doctor also associates a nurse with the record.
- A government agency is allowed to read and delete all types of records.

This interpretation might seem a bit liberal, as it in some cases gives more access than allowed by the regulations. However, this is nothing compared to reality [4]. At the same time, having access control on medical records for hospital staff comes with problems in itself. This is a tricky subject and a reliable overall solution does not exist. We have a tradeoff between availability and confidentiality and in our solution we will have more confidentiality and less availability.

Your implementation should consider a small example with a few patients, a few nurses, a few doctors, and one government agency. Your do not need to include a database system for your records (if you don't want to), simply store them in some convenient way. You must also make sure that all actions are properly logged.

Access to the records is provided through a public client program (you must consider all implementations as public knowledge in your security evaluation). The server is situated in a physically protected room. Access to this room is only allowed under the supervision of trusted staff. Proper backup is assumed.

The project must be implemented in the Java language. You will need to write a client application, a server application and possibly some additional application. You may reuse the client and server implementations from Project 1. Some form of two factor authentication of individuals is needed and one way to accomplish this is to authenticate users with certificates. This will require them to have both the actual keystore and also the password to the keystore. You are free to use some other two factor authentication method if you want. All certificates must be signed by a CA, as done in Project 1. Define and use a sensible naming convention for your certificate fields. The communication between the client and the server must be encrypted and established using the network standard SSL. The CA certificate should be installed in a truststore on both the server and the client. The CA private key should be used to sign all other certificates. The server should store all records and respond to client requests. The records do not need to be stored encrypted.

3 Documentation

Read this section very carefully. The report can be written in either Swedish or English. The quality of your report is expected to be good in terms of layout, readability, content and so on. Use single spacing and a normal-sized font with serifs! Playing around with larger spacing and fonts just makes your report look weak, do not fall into that trap. Make sure that you have included references to all books, articles, web pages, pictures and such that you have utilized when writing your report. If you have not included a reference, then you are claiming that it is your original work.

The documentation consists of three main parts, in total 8-10 A4 pages.

- The first part should be a short architectural overview that explains how your program is *structured*. Be sure to *illustrate* (yes, draw a picture, it is a requirement to do so, explore your creativity) how keystores and truststores are used, by whom and how, certificate hierarchy, and so on. Elaborate somewhat on your access control scheme.

  Your main challenge here is to use your technical writing ability to explain technical details in a way that is easily absorbed and not so easily misunderstood. Keep your description at a high structural level. Only include details if they are important from a security perspective. Do *not* include a user guide or user instructions for your program!

  The intended audience for this part should be a student in your program that has not taken the Computer Security course. Upon reading your description, the reader should understand the structure and main points of your program.

  Approximately 2-3 pages are expected for this part.

- The second part is a security evaluation of the design. In the security evaluation, you must enumerate all possible and impossible attacks on the system. For each attack, briefly explain the attack itself, and then summarize the protection that your solution offers. If none (e.g., because the attack is unrealistic or not applicable), this must also be stated. You should be able to list at least 20 different attacks, threats or issues. *As only a few examples*, you should consider the following:

- What if an attacker attempts a spoofing attack, writing his/her own client, and tricks users to use this client instead?
- Man-in-the-middle attacks.
- Eavesdropping communication.
- Fake certificates.
- The human factor. What is required by the users in order to avoid security breaches. Would it be necessary to educate the users?
- How should the passwords be chosen?

In this part of your report, include a little something on cipher suite selection. Which cipher suite is chosen when you run your program? Dissect the cipher suite identifier and explain its different parts. Are there good and bad cipher suites? How can you control the choice of cipher suites in your programs?

Approximately 3-4 pages are expected for this part.

- The third part should describe one of the SSL attacks or weaknesses. Use the paper *Lessons Learned From Previous SSL/TLS Attacks, A Brief Chronology Of Attacks And Weaknesses* [5] as a starting point, and utilize the reference section as a source for further information. Choose the attack(s) or weakness(es) that you find most interesting.

  Explain your choice. Provide background information and clearly illustrate or describe what the security problem is. How was the problem fixed, or is it still a problem? Past, current and future security impact? What can be learned from your chosen weakness(es)?

  Approximately 2-3 pages are expected for this part.

4 Getting Approved on Project 2

To be accepted on the project you have to submit the report (uncompressed) and source code (compression ok). The presentation is divided into two parts, a demonstration of your program and a discussion regarding the report. All reports must be sent with email to Paul Stankovski (paul@eit.lth.se). Write 'EIT060 P2' (without quotes) in the subject line.

Suggest a time for presentation when sending in your report. All group members must be present on the presentation. There are two alternatives when demonstrating your program. Either you prepare a computer in the basement to demonstrate your program on, or you bring your own laptop.

4.1 What You Should Think About

The focus of this course is on computer security, and we use programming as a tool to explore security topics. We do not require you to write a program that is spotless in all regards, but we do require that you think extensively about all security aspects that you can find, both technical and nontechnical. In your program, functionality is more important than design. We will not be selling your solution to investors, so you do not need to spend time making your program look nice. In fact, a GUI is *optional*.

5 Resources

- If you do not have it already, the Java SDK package can be downloaded at [6].
- Java security documentation can be found at [7]. The most important part for you is the JSSE documentation which can be found at [8]. This is a rather large document but you will learn tons of stuff by reading it and it makes the project much easier. Spending time reading the JSSE documentation will probably save you lots of time in the end.
- There is sample code in the JSSE documentation. You should be most interested in `ClassServer.java`, `ClassFileServer.java` and `SSLSocketClientWithClientAuth`.

- Java SE 7 API specification can be found at [9].
- The java language specification can be found at [10]

6 Hints

Here are some useful hints.
Hint 1: Take advantage of what you learned in Project 1 when creating keystores and truststores for different end-users. The keytool parameter `-dname` can be useful to minimize tedious and error-prone human interaction when generating certificates.
Hint 2: Before you begin coding, draw a design sketch to show which keystores and truststores your design will be using and where. This sketch can also be used for your report.
Hint 3: When you have established the connection, the server might want to read the client certificate to check which user it is. Check your source code to see how this was done in Project 1.
Hint 4: Distribute the work within the group. Do not sit 4 people in front of the computer.
Hint 5: The JSSE documentation can be tough to read, but it is a *very* good source of information with explanations and examples. Consulting it may save you lots of time.
Hint 6: When starting a Java program, a truststore can be specified by setting the system properties javax.net.ssl.trustStore and javax.net.ssl.trustStorePassword as
`java -Djavax.net.ssl.trustStore=theTruststore -Djavax.net.ssl.trustStorePassword=passphrase prog`
but it can also be set in the source code. If you reuse Project 1, check your source code to see how it is done there.

References

[1] Patientdatalag (2008:355) Url: `https://lagen.nu/2008:355`

[2] SOSFS 2008:14 Informationshantering och journalföring i hälso- och sjukvården (SOSFS 2008:14) Url: `http://www.socialstyrelsen.se/sosfs/2008-14`

[3] Ansvar fr informationssäkerhet - Styrning av behörigheter. Url: `http://www.socialstyrelsen.se/regelverk/handbocker/handbokominformationshanteringochjournalforing/25`

[4] Datainspektionen. Bristande åtkomstkontroll på Karolinska sjukhuset, 2009-04-03. Url: `http://www.datainspektionen.se/sv/press/nyhetsarkiv/2009/bristande-atkomstkontroll-pa-karolinska-sjukhuset/`

[5] Lessons Learned From Previous SSL/TLS Attacks – A Brief Chronology Of Attacks And Weaknesses, `http://eprint.iacr.org/2013/049.pdf`.

[6] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

[7] `http://download.oracle.com/javase/7/docs/technotes/guides/security/index.html`

[8] `http://download.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html`

[9] `http://download.oracle.com/javase/7/docs/api/`

[10] `http://java.sun.com/docs/books/jls/index.html`