EIT060 - Projekt 2

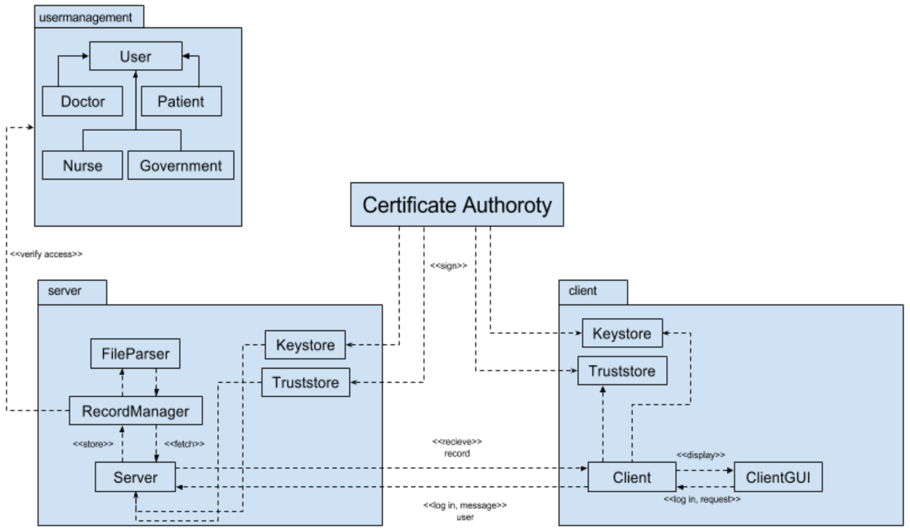
Simon Thoulouis, Bert Kurqberg

February 2016

1 Introduction

In this project a security mechanism has been designed for use by a hospital and government agency. The computer programs has been implemented in Java and uses an authentication service based on certificates as well as basic editing features. The certificates were created using the tool OpenSSL. The program is intended to be used for accessing and editing medical records where permission to read, edit or delete records are determined by the security clearance of the users and a control access scheme. The report aims to explain and detail the security aspects of the service.

2 High-level architectural overview

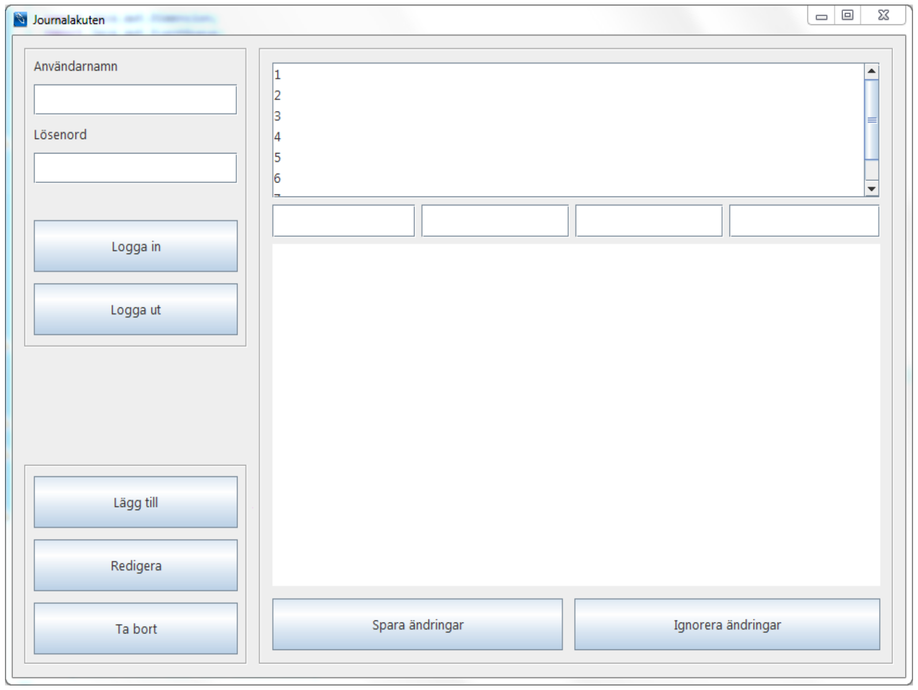


2.1 Digital certificates

The design goal includes to create a system for authentication of users and establishment of a communication link. A scheme needs to be implemented that ensures the mutual trust of identities of two communicating parties; a server and a user. This is achieved by use of public key certificates. A certificate authority (CA) acts as a trusted third party for the two communicating parties. The CA signs both the certificate belonging to the server hosting the medical records and any user that is allowed to access the server. The two parties can then communicate without distrusting the alleged identity of each other.

2.2 Client application

The client program is implemented with a graphical user interface (GUI) that allows the user to attempt to authenticate themselves by logging in and to read, edit, delete or create records based on the clearance associated with the user. The login process is based on mutual authentication where a communication link with the medical record server is attempted to be started using certificates.



2.3 Server application

The networking model is based on the award winning, top of the line server - client application used in project1.

Each medical record has a doctor, nurse, patient and division associated with it and is saved on the server as a txt file. Since the server is assumed to be stored in a secure environment encryption has been deemed unnecessary. **FileParser** reads a file containing a medical record and forwards it to **RecordManager**. **RecordManager** creates a **FileParser** for every record on the server and stores them in a collection. **RecordManager** handles requests for medical records and has the necessary methods for searching, creating and editing them. It is also responsible for enforcing access control.

Each user type (Doctor, Nurse, Patient and Government) is represented by its own class which inherits the abstract class **User**. **User** contains user's credentials as well as user's access rights which **RecordManager** uses for access control. The users' rights can be further examined in section 3.3.

2.4 Access control and Authentication

Each user has a certificate stored in a keystore and a truststore which is used by the client and server for authentication. Thus in order to log into a client computer a user needs to have a keystore, a truststore and know the password to them. The keystore is used to authenticate the user locally and the truststore with the server. For simplicity's sake both stores have the same password.

This admittedly increases the administrative overhead but makes the system much more secure since only authorized clients can access the records. Essentially this setup functions as 2-factor authentication since any attacker needs not only user's credentials but also a correct certificate which is stored locally.

3 Ethical discussion

3.1 Confidentiality and availability

If a secure system is designed with confidentiality as the prime focus, the product might very well be of a high standard, where any unauthorized user is prevented from access to the protected medical records. When a program is designed with a focus on confidentiality it is with the intent of protecting information about hospital patients. If, however, the hospital's prime objective is to ensure the well being of patients it could be argued that the availability of the medical records are of higher importance than keeping records secure from unwanted viewers. A system designed with availability first and foremost in mind values the importance of hospital workers to be able to access any relevant information about patients, whenever and wherever that might happen to be. A situation in a hospital environment might be imagined, where an available system might very well save lives where as a confidential system would keep records safe from prying eyes albeit at the expense of the patient's health.

3.2 Ethical responsibility

When it comes to the moral responsibility of the engineer one would hope that they or any company that they work for has the disposition to inform their customers of potential design flaws of the product that they have requested. The responsibility should extend to informing customers of alternatives, explaining potential ramifications of their choices and thus verifying what security aspects are of actual importance to the customer. In reality there may exist several ethical problems where what is important to one person might not coincide with the opinion of another. A doctor may be inclined to request a security system where the quality of overall security, availability or integrity are knowingly sacrificed for economical reasons; An engineer might deliver a product that includes all requested design features though ultimately being substandard, for reasons such as moral laziness or greed. Ultimately, what should be valued for the hospital requesting the security system, the engineer designing it and the hospital staff using it should be the welfare of everyone else. As the scenario being discussed is one where lives are at stake one could argue that the highest maxim when any aspects are weighed against each other should be the continued vitality of the people most likely to lose it, the patients.

3.3 Access control scheme for real hospital environment

As previously discussed it is highly important for the well-being of patients that authorized nurses and doctors can access their files when it is essential they be able to do so. The original access control scheme described in this project already has a satisfactory degree of confidentiality. The records are protected from being read outside of the division with exception of the government agency allowed to do so. This should mean that the confidentiality is not too high to impede with the work being conducted within a division, but high enough to ensure security from outside of the division. The writing rights however seem too restrictive for a real hospital. It seems that any number of situations could turn up where a doctor or nurse should be able to replace any other at a moments notice. This might require them to inherit the rights associated with the person they replaced as well. For this reason we think it prudent to extend the writing rights of the nurses and doctors to the full division. Such a system of access rights would place a higher importance of a well functioning log system such that all versions of the records would be available to read with corresponding information about who edited what, time stamps e.t.c.

Table 1: Original access control scheme

User	Read	Write	Create	Delete
Patient	self			
Nurse	self,division	self		
Doctor	self,division	self	self	
Gov. Agency	all			all

Table 2: Suggested access control scheme

User	Read	Write	Create	Delete
Patient	self			
Nurse	self,division	self,division		
Doctor	self,division	self,division	self	
Gov. Agency	all			all

The main difference between the two access control scheme is the level of integrity. The suggested new scheme has a lower level of integrity with the added benefit of a higher degree of adaptability of the workers at the hospital. They can now react to situations in which they would want to edit a patients medical records while the doctor and nurse associated with the patient are unavailable. While this negatively affects the integrity of the situation hopefully this could be complemented with a well designed log system such that it does not have too much of a negative effect on the security of the system.

4 Security evaluation

What follows is an evaluation of the system's level of protection against common attacks.

4.1 Basic assumptions of security

In the project the server is assumed to be physically protected. Furthermore the network used for authentication and communication between clients and server are assumed to be protected from outside sources. This effectively limits the vulnerability to attacks such as denial of service attacks. If the integrity of the file system of the client computers is secure spoofing attacks are prevented.

4.2 Infeasible attacks

The SQL injection attack will not work since the system is fully implemented in Java without use of SQL databases. Similarly the choice of programming language protects from buffer overflow attacks that exploit a lack of checks on array bounds in other languages.

4.3 Brute-force and dictionary attacks

In a brute force attack a person tries to log in by systematically testing all possible passwords. The brute dictionary attack is a variant where the passwords tried are words from one or several dictionaries and/or combinations and perturbations of the same words. The system is vulnerable to these types of attacks since no protection against the attacks have been implemented. A possible alteration to reduce the risk of the attacks being successful is implementing a lock-out mechanism. This mechanism would prevent a single user from attempting to log in after a number of failed attempts. Additional security could be achieved by requiring that the user passwords have certain properties. Passwords that do not contain words are protected from dictionary attacks while a longer password reduces the effectiveness of brute-force attacks.

4.4 Man-in-the-middle attack

In a man-in-the-middle (MitM) attack the attacker would attempt to intercept traffic between the client and server to eavesdrop on or alter the data sent. To achieve this the attacker would need to have access to a certificate chain containing the CA certificate and a certificate signed by the CA. Assuming that access to the local certificate files are protected and that the certificate authority can indeed be trusted the protection against MitM-attacks should be satisfactory. Additional security against MitM-attacks can be achieved by validating host names [1].

4.5 Cipher suites

TSL and SSl protocols support a number of cipher suites that define the encryption algorithms used to establish a secure connection. This includes the key exchange algorithm, authentication algorithm, symmetric encryption algorithm and hashing algorithm. The default cipher suite used is in this project is "TLS_DHE_DSS_WITH_AES_128_CBC_SHA256". The name includes all information about the algorithms used. It is a TLS cipher using the DHE, or Diffie-Hellman based, key exchange algorithm. The authentication algorithm specified is the Digital Signature Standard, DSS. Messages are encrypted using the symmetric encryption algorithm AES with a 128 bit key. CBC details a block cipher mode of operation utilized to increase security. Finally, the hash function used for integrity and authentication checks is SHA-2 with 256 bit output[2] [3]. The security of the communication then depends on the strength of the algorithms specified in the cipher suite used. The algorithms used here are all considered safe but a number of weak algorithms are available to chose from as e.g. the insecure encryption algorithm DES. If one would like to specify another cipher suite to be used one could, before initiating a socket handshake, call the SSLsocket method: *setEnabledCipherSuites(String[] suites).*

4.6 Security design

The security features of the program consists of a two-factor authentication of users and a secure connection established using mutual authentication between server and client. The login authentication requires both a valid user name and password, and a corresponding, pre-created, certificate.

The two-factor authentication provides an additional level of security where access to a certificate alone does not allow for access to the server. The certificates are stored in separate keystores which provides a certain level of protection to the certificates. In the event that the security of one keystore is compromised, not all certificates are gained access to. Additionally, if a users password is compromised, remote access to the server is still prevented as long as the corresponding certificate is still protected.

The confidentiality and integrity of the communication between server and client are ensured to a high degree through the use of SSLsocket communication established with mutual authentication of certificates.

References

[1] *MSC00-J. Use SSLSocket rather than Socket for secure data exchange*, 2016.
Available from: https://www.securecoding.cert.org/ [26 February 2016].

[2] *DES Modes of Operation*, 1980.
Available from: http://csrc.nist.gov/publications/fips/fips81/fips81.htm [26 February 2016].

[3] *TLS and SSl cipher suites*, 2009.
Available from: http://www.thesprawl.org/research/tls-and-ssl-cipher-suites/ [26 February 2016].