EIT060
Datasäkerhet
Projekt II

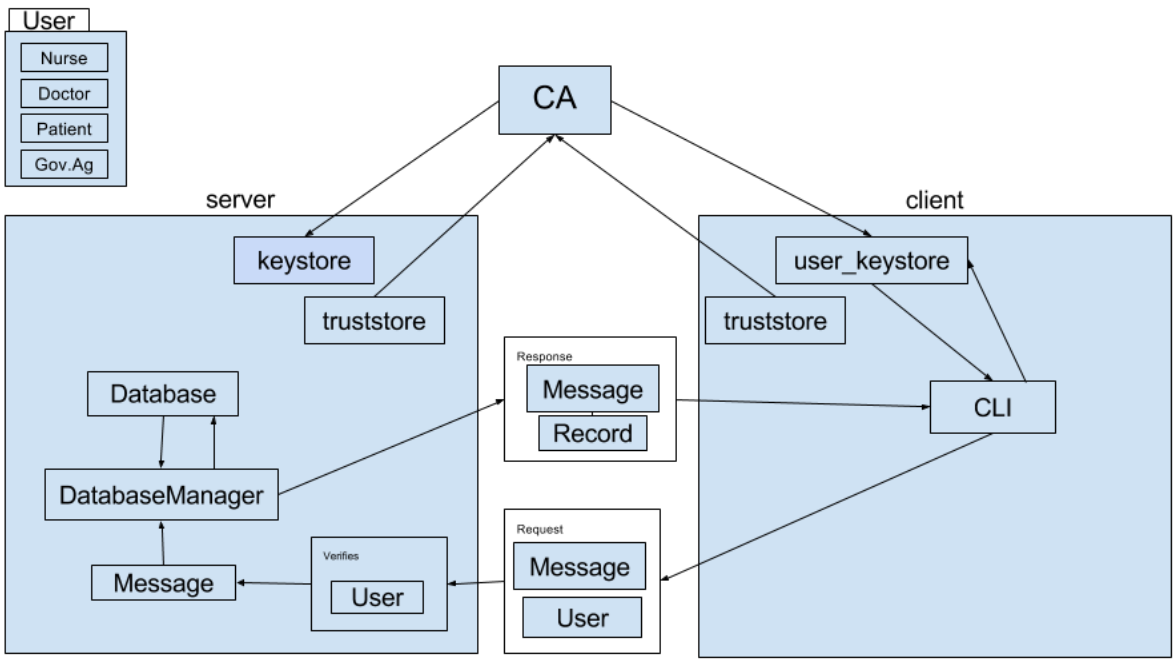
Participants (alphabetical order):

- ***Markus Olsson***
- ***Gustaf Oxenstierna***
- ***David Phung***
- ***Nicolas Salas***

Table of Contents:

High-Level Architectural Overview

Diagram



Project Description

The goal of the project is to provide and manage a database containing medical records in a hospital. The database is stored in a server and to be accessed remotely by users in an open network. This leads to several requirements that need to be solved.

- Only authorized individuals should be allowed to access or read the information from the database.
- These individuals are given rights to perform certain actions based on their roles.
- Unauthorized parties should not be able to read or modify the database as well as the traffic between the server and the client computer.

The focus of our project lies in the security aspects of the system. Therefore this report will mostly discuss topics concerning security problems and our approach to solve them.

Background Information

One way to send messages safely in an open network is to encrypt the messages using the public key of the receiver. The receiver can then decrypt the message using his/her private key.
However we need a way to make sure that the public key we are using actually belongs to the receiver. We achieve this by using digital certificates. Each public key is signed by a chain of authorities, if the chain ends with an authority that we trust then we trust the binding between the public key and subject. The key pair together with the certificate chains are stored in the files called keystores used to provide credentials. Each keystore has a password to protect its integrity. Any attempts to modify the file without knowing the password will make the file invalid for the system, we will use this later in our design.
In our system, all certificates are signed by a common certificate authority (CA) that is trusted by both the client and server application. The length of all certificate chains are two, the CA certificate and user/server certificate. The CA is stored in different files called truststores used to verify credentials.

Client Application

The client is a simple console program in which the users can login by entering his/her username and password. The program then opens a connection to the server and handles all message encrypting, decrypting and exchanging.

Server Application

In the lowest level of the server we have a data structure representing different types of users: patient, nurse, doctor, government agency. Each type of user is represented by a class and inherit from the superclass “User”. Each user has a name used for identification and nurses and doctors belong to a division. We'll use these classes to determine access rights for each user. The access rights for each type of users is given in table 1.
Each record has a patient, a nurse, a doctor and a division associated with it. It also has a unique identifier and some data in the form of text.

In the next level we have our database. We have chosen to implement our own database using a linked list containing records. The list is saved in form a binary file. Records are not encrypted when saved since the server drives are well protected. The database class supports all necessary methods such as search after a certain field, add, edit, or remove a record. However it does not check to verify access rights, instead we do it in class “DatabaseManager” which serves as our access control manager.

“DatabaseManager”: Handles all access requests to the database. If the subject does not have the permission to perform a certain request, an exception is thrown, the request is discarded and a message is sent back to the client.

Authentication and Identification

To authenticate an user, we have chosen to let each person have a certificate stored in a keystore. In other words each keystore contains only one certificate chain belong to one user. In order to gain access, the user needs to have the keystore file in the current computer and know the password to the keystore.

This is our approach to achieve 2-factor authentication. An outsider trying to login as user will have to first obtain the keystore file, which we assume to be protected by access control in the client computer. Although it is not completely impossible for a devoted attacker to gain access to this file it does add an extra layer of security to the system.

In order for the client to authenticate the server, the user who is trying to login must know the password to the truststore storing the CA. If there is only one truststore then the password to this file must be known by all users. Therefore we let each user have their own copy of the truststore file and the password to this file is the same as the password to their keystore. As a result each user will need a total of two files available in the actual computer to be able to login.

We also name the keystore and truststore files after the usernames to avoid having to search for the correct file.

Once the connection between the client and the server has been established, the server needs information about the user to able to determine access rights. We achieve this by storing this information in the CN of the client certificate. The format is as followed:

CN = username ROLE division.

Role	*Read*	*Write*	*Create*	*Delete*
Nurse	*division, self*	*self*	-	-
Doctor	*division, self*	*self*	*self*	-
Patient	*self*	-	-	-
GovernmentAgency	*all*	-	-	*all*

Table 1. Access control scheme

Ethical Discussion

Confidentiality and Availability

When the customer places an order it is assumed that the developer will focus on the confidentiality of the product but shouldn't they focus on its availability as well?
Confidentiality is protecting your secret information from being viewed by unauthorized users. Availability is making sure authorized users always can access that information. For example a person has lost blood and arrives at a hospital. The doctors or nurses need to give him new blood and must access his records to see his blood type. In that case it is more important that they can access that information than it is to make sure that other people can not.
Confidentiality has to do with people doing something wrong where as availability is about the system doing something wrong. It is more important that the system works because you can be sure that it works and that it is failsafe but you can never be 100% sure about human beings. If a system is secure enough and works properly a lot of the would-be wrong doers never get to do something wrong because the system will not let them do that. At the same time it protects against people even thinking of committing a "crime" because they know that the system that they work with is very secure so people without permission just do not get into the system at all.
The responsibilities of an engineer should be to strive to be ethical without doing something that goes against what the customer wants but sometimes going that extra mile. For example the institution ordering the system probably doesn't know as much about cybersecurity as the implementer does. If the Engineer sees a big flaw in the system that malicious people could use to their advantage the codewriter should implement stricter security measures while at the same time having good communication with the buyer so that the customer always knows which changes will be implemented.

Access Control Scheme

The only way to implement an access control of this scale on such important information is in a firm way. There can be no downsizing in any of the three departments (confidentiality, integrity and availability) if one wishes for the system to be secure. Important things to think about while implementing it may be:

- A logical system
- Easy to understand system
- Responsive and a self explanatory GUI

The people who use these kinds of systems may not be tech savvy and will be stressed a lot of times, it is important to assist them in any way possible. When implementing such a system in real life a big part of the final security will be because of monetary issues. When talking about confidentiality one can implement the program once and that technology stands but new ways of hacking are invented all the time and upgrading or updating one's system may be necessary but expensive. The integrity of a system is maybe the most commonly heard problem nowadays because the institutions do not always see the need to invest money or time into keeping logs of how the information is used and thusly information may be changed or tampered with. without any track of changes being made everyone with access to the logs are a risk to the information if it is not backed up in its original state somewhere else. Lastly, the availability seems to be the one that is easiest to keep track of since it's just regulations on how the personnel may access the logs or even if they have access to said logs.

Original access control scheme:

Role	***Read***	***Write***	***Create***	***Delete***
Nurse	*division, self*	*self*	-	-
Doctor	*division, self*	*self*	*self*	-
Patient	*self*	-	-	-
GovernmentAgency	*all*	-	-	*all*

New access control scheme:

Role	***Read***	***Write***	***Create***	***Delete***
Nurse	*division, self*	*division, self*	-	-
Doctor	*division, self*	*division, self*	*division, self*	*division, self*
Patient	*self*	-	-	-
GovernmentAgency	*all*	-	-	*all*

Comparison:
The main difference between the old and new access control scheme is based on more trust for nurses and doctors so as to distribute the workload between more capable hands in the institution. The hope is for the integrity to stay the same because, although more workers are able to write on files, the trust for those workers is also greater. it doesn't mean that it will work but it's a tryout for future research. The major change is to the confidentiality, it's much lower. We have given more people both the possibility to both write and create files. This distributes the workload so that not only certain individuals can carry certain duties. More people have access to more information so that the nurses get a higher role in the hospital when talking about taking care of logs and records. The availability might be somewhat compromised because of a bigger amount of workers using the hardware and straining it.

Original system:

- Pros
    - Restricted access to sensitive material
    - High integrity: log and record accuracy is high
        - At least if high-record accuracy is a direct result of restricted access to them
    - High confidentiality: vital information can only be accessed by a privileged few

New system:

- Pros
    - Nurses can do more work that is relevant for them when taking care of patients
    - Low confidentiality: in stressful or dire situations more workers can assist with a broader multitude of tasks because they have the required information to do so
    - high integrity (?): with lower integrity and broader access the maintenance of records and logs could be a thing of the past. More people accessing them means also that more people can detect if something is wrong

Ethical Considerations

There are certain ethical considerations that need to be weighed when using or designing systems such as this, for example:

When handling the ordering and administration of such a system it's always important to be careful during the work process so as to not mix different documents or delete something. When working at that height in the information cycle you get disconnected with what the documents actually represent and people maybe aren't as careful as they should. For them it's just files on a computer and a little mistake maybe deletes or changes a log but it is always important to remember that some people lives may be on the line because of these documents.

When designing the system it is of outmost importance to go through all different security flaws the system might have and failsafe it against any kind of incursion. It is important for the designer to do its part since malicious partners might want to access sensitive information if a system is fragile. More than that one can not do. It is up the institution to declare which security measures one would like to have and then up to implementers to do the best job possible.

The most difficult question to discuss is how the hospital staff should use the system. They are the ones that come into contact with the patients the most and will have the most benefits of using the information. The two points that will come most into play here will be a patient's health contra the institution's privacy policies. There could be contradicting methods of taking care of a patient between two doctors but only one with the access to the records or just a nurse that can not help in a dire situation because they can not access the system in time or at all because of privacy restrictions.

Security Evaluation of the Design

Man in the Middle

In SSL man in the middle attacks requires the attacker to have a valid certificate. Since it is easy to get a certificate signed by the CA (you get one if you are a patient), an attacker might attempt an attack by signing a certificate with their own patient certificate. They may then pretend to be both a user and the server. This will not work because the client and server programs check the length of the certificate chains.

Buffer Overflow

The program is written in Java so buffer overflow attacks are not possible.

SQL Injection (or Similar)

Since we store data in a binary file with our own software used to access it a SQL injection would have little effect. We do generate database access commands directly from user commands, instead we map a few known commands to specific functions in our code; as such, an injection attack is unlikely to be successful.

Replay Attack

The SSL handshake uses random numbers, so a replay attack will not work.

Brute Force Attack / Dictionary Attack

Generally not feasible unless attacker has access to a certificate, see Two factor authentication for a more indepth explanation.

Two Factor Authentication

We have implemented two factor authentication in the sense that a user needs both a) a valid certificate and b) a username and a password. The username is used to associate a user to a specific certificate and the password is required for the user to be able to access the certificate. Since hospital records contain sensitive information it is very important that this information stays secret and is not accessible by outsiders. In our system this holds true to some extent; an attacker which has only the

username and password for a user will not be authenticated by the system. However, if the attacker were to gain access to the certificate of a user, the attacker would still need to crack the password. This may be feasible in our system since there is no limit enforced on how many times one may try to access the system. The program does however shut down if wrong credentials are entered. This is weak protection against brute force since the attacker could simply use the keytool to try and guess the password programmatically. Since the server only checks the certificate provided by the client an attacker with access to a certificate (having successfully guessed the password to the keystore containing the certificate) would be able to gain unauthorized access to the system.

Cipher Suite Selection

"TLS_DHE_DSS_WITH_AES_128_CBC_SHA256" is the default cipher suite. For encryption, AES is used with 128 bit key. Cipher block chaining (CBC) is utilized to remove redundancy in ciphertext. The hash algorithm used is SHA-2 with a 256 bit output. Consulting the "openssl ciphers -v" yields the following information. The key exchange algorithm used is Diffie-Hellman, the DHE string is used to signify this. DSS denotes the signature algorithm used (DSA) which is standardized as Digital Signature Standard (DSS).

In the list of ciphers generated from the command "openssl ciphers -v" some ciphers are prepended with the "EXP" string which means they are meant for export (the US have historically limited the key size on exported cryptosystems) which means these cipher suites are not secure. Some of them use weak encryption algorithms (such as DES [1], RC2 [2], RC4 [3]) and sometimes they also use inappropriate hash functions (such as MD5 [4]). There are however many strong ciphers to choose from, one need only assert oneself that the chosen hash- and encryption algorithms for the given cipher suite are considered safe to use.

To set a custom cipher suite there is a function called setEnabledCipherSuites(String[] ciphers) that can be used. Calling this method on the SSLSocket will make the session use the specified cipher suite, however, this must be done before a handshake is performed or the default cipher suite will be utilized instead. One can also use specify the enabled cipher suites via the SSLEngine class, analogous to SSLSocket [5] with one caveat; the specified cipher suite must be listed by a call to getEnabledCipherSuites() on the SSLEngine [6] class. Furthermore, the SSL protocol provides a way to change cipher suite mid session using the Change Cipher Spec Protocol. To change cipher suite in a java SSL socket, the startHandshake() method is called on the SSLSocket [5]. This will start a handshaking sequence where the client and server can negotiate new cipher suites and/or keys to be used.

References

[1] "Data Encryption Standard - Wikipedia, the free encyclopedia." 2011. 25 Feb. 2016 <https://en.wikipedia.org/wiki/Data_Encryption_Standard>
[2] "RC2 - Wikipedia, the free encyclopedia." 2011. 25 Feb. 2016 <https://en.wikipedia.org/wiki/RC2>
[3] "RC4 - Wikipedia, the free encyclopedia." 2011. 25 Feb. 2016 <https://en.wikipedia.org/wiki/RC4>
[4] "MD5 - Wikipedia, the free encyclopedia." 2011. 25 Feb. 2016 <https://en.wikipedia.org/wiki/MD5>
[5] "SSLSocket (Java Platform SE 7) - Oracle Documentation." 2012. 25 Feb. 2016 <https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLSocket.html>
[6] "SSLEngine (Java Platform SE 7) - Oracle Documentation." 2012. 25 Feb. 2016 <https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLEngine.html>