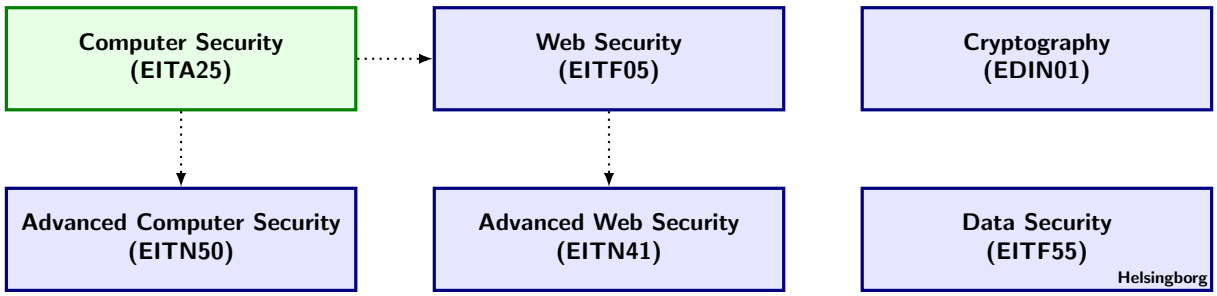
Computer Security 2020

Project 1: Digital Certificates

- This project will be done in groups of 4 people.
- You will not get detailed instructions of exactly how to do each step. Figuring out the exact steps is part of the project.
- The results of this project will be used in Project 2, so document your work.

Learning goals:

- Understanding the concept of keystores and truststores in Java.
- Get hands-on experience handling and creating digital certificates.
- Get acquainted with OpenSSL and keytool.
- Setting up a TLS connection in practice.



1 Introduction

Read everything in this document before you start. The first aim of this project is to learn about certificates, certificate chains, keystores and truststores. You will learn what they are and how they can be created and handled in practice. The second aim of the project is to get you to use these components to set up your own TLS connection.

OpenSSL includes a library with several cryptographic functions. We will use functions related to certificates. You can find the OpenSSL documentation at [1]. We are mostly interested in the OpenSSL commands *req* and *x509*, but it might be useful to look at other commands as well.

We will also use the program keytool, which is a tool included in the Java Development Kit. It is used for managing and creating key pairs and certificates, which are stored in keystores. However, keytool can not be used to sign other certificates. OpenSSL has to be used for this step. You can find the keytool documentation at [2].
In the first part of the project you will

1. create a certificate authority (CA) that can sign other certificates,
2. learn about certificate chains, keystores and truststores,
3. create a client-side keystore and truststore,
4. create a server-side keystore and truststore.

In the second part of the project you will

1. run existing code for setting up a TLS connection,
2. modify the given code to use your certificates.

2 Project Instructions

2.1 Certificates, keystores and truststores

You first need to download and install OpenSSL and keytool. OpenSSL is included in most Linux distributions. If you use Windows, you can find it at [3].

Read about certificates in the course book, and learn what keystores and truststores are (search online). Then perform the following steps.

1. Create a X.509 CA certificate using OpenSSL. Set the *commonName* (CN) field of the certificate to CA for clarity. Make sure to save the private key of the CA in a file. **Question A:** What does the OpenSSL switch -CAcreateserial do?
2. Use keytool to create a truststore that contains the CA certificate (and nothing else). Name the file clienttruststore (no file extension) and set the password to password.
3. Use keytool to create an end-user keypair that is stored in a (client-side) keystore. Use your full names and STIL identities for the CN field[1] for the certificate; X/X/X/X, where

   X = <FULL NAME> (<STIL-id>)

   for each respective group member. Name the file clientkeystore (no file extension) and set the password to password.
4. Use keytool to create a Certificate Signing Request (CSR) for the keys created in the previous step. A CSR can be seen as a blueprint of a certificate that has not been signed yet.
5. Use OpenSSL to let your CA (from step 1) sign the CSR.

[1] CN is short for Common Name. To fill this out, keytool will ask you for "first and last name".

6. Import the certificate chain into your keystore. Note that you should import the CA certificate before you import the signed certificate.

7. Use keytool to verify that a certificate chain has been established. Your keystore should contain TWO entries;

   a) one CA certificate,

   b) one certificate chain of **length 2**, consisting of the CA certificate and the signed certificate.

   The command "keytool -list -v ..." is appropriate.

8. Both keytool and OpenSSL typically generate X.509 version 3 CA certificates. But the certificates that you have created for your server and client are most likely X.509 version 1 certificates. **Question B:** How can you tell keytool to generate a CSR for an X.509 version 3 client certificate (at step 4), or tell OpenSSL to force generation of a version 3 certificate (at step 5)? **Question C:** What are extensions and what can they contain?

9. Follow steps 3 through 7 above to create a server-side keystore. Name the file serverkeystore (no file extension) and set the password to password. Use CN=MyServer for the server certificate.

10. Create a server-side truststore. Name the file servertruststore (no file extension) and set the password to password. **Question D:** Is it possible to just make a copy of the client-side truststore, why or why not?

11. There are in total six passwords used in your server and client programs (do not count CA passwords). The two truststores use one password each, and the two keystores use two passwords each. **Question E:** What is the purpose of each password? That is, what does each password protect?

2.2 TLS connection

Now that you have created certificates, your next goal is to set up a TLS connection that uses these certificates. On the course homepage you can find sample Java code bundled with some keystores and truststores. You will be using this code as a starting point for your server and client implementations. Do the following.

1. Compile the provided server and client code samples. The server and client can be compiled with

   ```
   javac server.java client.java
   ```

2. Run both server and client with the provided key- and truststores. The server and client can be run as

   ```
   java server 9876
   java client localhost 9876
   ```

   The server handles several connecting clients. If you are not familiar with threading, you do not need to worry about it.

3. The server responds to text messages sent from the client. **Question F:** What does the server answer?

4. Both server and client display the *subject* field of the connecting party's certificate. Check the code to see how this is done. Modify the code to also print out the *issuer* field of the certificate on a separate line just below the subject field printout.

5. Print the serial number of the certificate just below the issuer field printout.

6. Get a TLS connection up and running using the key- and truststores that you have generated. Modify the code if you need to.

7. **Question G:** What is the purpose of `setNeedClientAuth(true)` in the server?

8. **Question H:** Show printouts, written directly in your submission pdf, from both server and client (we know that this is not really a question, but you get the point).

3 Helpful Hints

Hint 1: To avoid rewriting many commands, it can be a good idea to write all steps in a batch-file or shell script. Then you can also reuse the batch file for the second project.
Hint 2: Do not use the openssl ca command, it is very likely to get you into trouble. Use other openssl commands instead.
Hint 3: It is possible to use a keystore as a truststore. However, you are to employ separate usage to promote program structure.
Hint 4: If you run into problems involving a non-existent srl-file, you should look into the openssl command x509 option -CAcreateserial.
Hint 5: When signing a client certificate, the openssl command x509 option -signkey is not what you want to use, go for options -CA and -CAkey instead.
Hint 6: At step 7b, if your keystore contains three entries or if your certificate chain is of length 1 instead of 2, verify that your usage of aliases is correct. You need to use the same alias for the genkey, certreq and import commands (for the same certificate/keypair). Otherwise, keytool will not be able to match the import with the request that you previously made. The keytool option -trustcacerts may also be of interest.
Hint 7: It is a good idea to check what is in the keystore/truststore between the different steps so that it aligns with your expectations.
Hint 8: Before you start, make sure that you understand what you are going to do. Then, make sure that everyone in the group understands this.

4 Getting Approved on Project 1

You get approved on Project 1 by submitting clear but concise answers to the questions posed in the instructions above. Write your answers in LaTeXor .doc document, and send the answer in a pdf format. Do not forget to include printouts from your server and client to show that you have successfully set up a TLS connection that uses your own certificates.

Last but not least, attach your *uncompressed* keystores and truststores, both client- and server-side. These files must be named as specified above, with all passwords set to password.

Now **verify** that your submission complies with the stated requirements. In particular, verify that your certificates and certificate chains are correct, and that they can be used to successfully set up a TLS connection. Verify that the CN-fields of your certificates are compliant. This is necessary for passing the assignment! Sending in non-compliant submissions is a waste of our time, both yours and mine. Do not forget to check for updates, corrections or clarifications on the course homepage.

Please use the subject line 'EITA25 P1' (without quotes) for all email correspondence associated with this project. Email your submission to Syafiq (syafiq_al.atiiq@eit.lth.se). Syafiq will reply, hopefully to let you know that your work has been approved.

In parallel, upload your pdf to Urkund, syafiq_al.atiiq.lu@analys.urkund.se for plagiarism check. Only one student per group must do this.

As congestion is likely around deadline, do not be alarmed if the response time is not up to your convenience level. Submissions are handled in the order they are received, so send it in early if you are fastidious.

Suggestions on improvements of the project itself or the project description are appreciated.

5 Getting Help

You are encouraged to discuss the problems between groups, but the answers and the commands should be developed within each group. You may ask Syafiq if you are not able to generate valid keystores/certificates, but you must know what the problem is. Just sending your failed commands asking "something is wrong, can you find the problem?", is not allowed. Understand the problem by looking at the certificates and the keystore entries first. Then be specific when you ask for help.

Good luck!

References

[1] OpenSSL documentation, `https://www.openssl.org/docs/`, last accessed on 2019-01-24.

[2] Java keytool documentation, `http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`, last accessed on 2019-01-24.

[3] OpenSSL for Windows, `http://www.slproweb.com/products/Win32OpenSSL.html`, last accessed on 2019-01-24.