ETH_zürich_

Certificate Authority Project

1   Background

The (fictional) company iMovies produces independent movies of various kind but with a focus on investigative reporting. Therefore, information exchanged within the company and with informants must be handled confidentially.

To do so, iMovies wants to take its first steps towards PKI-based services. For this reason, a simple certificate authority (CA) should be implemented, with which employees can be provided with digital certificates. These certificates will be used for secure e-mail communication.

2   Assignment

In groups of three students, you are expected to design and implement a CA according to the requirements given below. In a second step, the implementations will be exchanged among the groups and each group should then review another group's CA.

2.1   Functional Requirements

We describe the functional requirement of the CA. Note that the entire functionality must be accessible to a remote client (outside the company's network).

2.1.1   Certificate Issuing Process

The company already maintains a MySQL database in which all employees are listed, along with their personal data as well as a user ID and a password. This database is a legacy system, which cannot be migrated. The CA should verify authorized certificate requests on the basis of this database.

The process of granting certificates should be as follows:

1. The user logs in via a web form by entering his user ID and his password. The user ID and password are verified by consulting the information stored in the database. Alternatively, login using a valid certificate is also possible;

2. The user is shown the user information stored in the database. If required, the user may correct this information and any changes will be applied to the database;

3. A certificate is issued based on the (possibly corrected) user information from Step 2;

4. The user is offered the possibility to download the new certificate, including the corresponding private key, in PKCS#12 format.

2.1.2 Certificate Revocation Process

Employees need the possibility to revoke certificates, for example, when their private key is compromised or lost.

The process of revoking a certificate should be as follows:

1. The affected user authenticates himself to a web application. Authentication can either be certificate-based client authentication over SSL/TLS (if the user still holds the certificate and the corresponding private key) or the user uses his user name and password stored in the database (if the user has lost the certificate or the corresponding private key);

2. After successful authentication, the certificate in question (or all affected certificates of the affected user) will be revoked. Additionally, a new certificate revocation list will be generated and published on the web server. Make sure that login with revoked certificates is not possible.

2.1.3 CA Administrator Interface

Using a dedicated web interface, CA administrators (not necessarily system administrators!) can consult the CA's current state. The interface provides at least the following information:

1. Number of issued certificates;

2. Number of revoked certificates;

3. Current serial number.

CA administrators authenticate themselves with their digital certificate.

2.1.4 Key Backup

A copy of all keys and certificates issued must be stored in an archive. The archive is intended to ensure that encrypted data is still accessible even in the case of loss of an employee's certificate or private key, or even the employee himself.

2.1.5 System Administration and Maintenance

The system should provide appropriate and secure interfaces for remote administration. In addition, an automated back-up solution must be implemented, which includes configuration and logging information.

Note that these interfaces do not need to be especially comfortable or user friendly. It is sufficient to provide suitable and simple interfaces with the help of standard protocols such as SSH and FTP.

2.1.6 Components to Be Provided

Web Server: User interfaces, certificate requests, certificate delivery, revocation requests, etc;

Core CA: Management of user certificates, CA configuration, CA certificates and keys, functionality to issue new certificates, etc;

MySQL Database: Legacy database with user data. The database specification can be found in Sect. 2.4;

Backup: Backup of keys and certificates from the Core CA and of configuration and logging information.

Client: Sample client system that allows one to test the CA's functionality from outside the company's network. The client system should be configured such that all functions can be tested. This includes the configuration of a special certificate to test the administrator interfaces.

Describe exactly how these components are distributed on various computers and exactly what additional components are required to enforce the security measures. The implementation is left up to the project team.

All systems must be built using VirtualBox, which defines the "hardware". The software is freely choosable. However, the operating systems must be Linux variants, and the legacy database requires MySQL.

2.2 Security Requirements

The most important security requirements are:

- Access control with regard to the CA functionality and data, in particular configuration and keys;

- Secrecy and integrity with respect to the private keys in the key backup. Note that the protection of the private keys on users' computers is the responsibility of the individual users;

- Secrecy and integrity with respect to user data;

- Access control on all components.

You are supposed to derive the necessary security measures from a risk analysis. Use the methodology provided in the book.

2.3 Backdoors

You must build two backdoors into your system. Both backdoors should allow remote access to the system(s) and compromise its purpose. The reviewers of your system will later have to search for these backdoors.

Design and implement a first backdoor so that it will be nontrivial but likely for the reviewers to find it. Give your best effort when it comes to the second backdoor! Try to hide it so well that the reviewers will not find it. Do not forget to hide your traces in the end (e.g., shell history).

2.4 Database Specification

The database contains the following information:

```
mysql> select * from users;
+-----+-------------+-------------+----------------+----[...]---+
| uid | lastname    | firstname   | email          | pwd        |
+-----+-------------+-------------+----------------+----[...]---+
| db  | Basin       | David       | db@imovies.ch  | 8d0[...]05 |
| fu  | Fuerst      | Andreas     | fu@imovies.ch  | 6e5[...]c7 |
| ms  | Schlaepfer  | Michael     | ms@imovies.ch  | 4d7[...]75 |
| a3  | Anderson    | Andres Alan | and@imovies.ch | 523[...]e6 |
+-----+-------------+-------------+----------------+----[...]---+
4 rows in set (0.03 sec)
```

The password entries in column "pwd" correspond to the SHA1-checksums of the original passwords. The plaintext passwords are:

UserID	Password
db	D15Licz6
fu	KramBamBuli
ms	MidbSvlJ
a3	Astrid

The checksums are generated, for example, as follows:

```
bob@bob:~$ echo -n KramBamBuli | openssl sha1
```

The existence as well as the content of the database are mandatory. For the implementation of the CA, the database thus has to exist and must contain the above table.

To simplify this, a corresponding dump of the database can be found on the course's web page (imovies_users.dump). The database is then created as follows:

```
echo 'create database imovies' | mysql -uroot -p<password>
```

and the table users is imported using the command

```
mysql -uroot -p<password> imovies < imovies_users.dump
```

3 Review

After the implementation of the infrastructure described above, access to it will be given to another team that will review it. Simultaneously, your team will review another team's project.

The review should be conducted both as a "black-box" and a "white-box" review. On one hand, you should examine the system from outside through appropriate scans and tests. On the other hand, the other team will provide you with all necessary information for access to all components that you must review, so that you can analyze the entire system from the inside.

You should decide on an appropriate "review computer" to conduct the black-box analysis from. The choice is yours. You could, for example, use the virtual machine mallet, develop tools yourself, or use freely available collections such as Kali Linux (`www.kali.org`).

4 Written Reports

4.1 Structure

You will write reports both on your own system and on the system that you review. Templates specifying the structure of these reports are provided on the course web page. Follow the instructions in the templates.

4.2 Writing Guidelines

Please observe the following points when writing your reports:

- Adhere to the given templates.

- Refer to the security principles in the book for justifying your choices.

- Terminology: secure = confidential + authentic. Be clear about which properties you are writing.

- Refer to the source document of your risk definitions if appropriate.

- For the risk evaluation, formulate the threats in active, not passive, voice: who (threat source) does what (bad things)?

- Use a spell checker before hand-in!

4.3 Evaluation Criteria

Both the form and the content of your two reports will be evaluated. The content will be more strongly weighted than the form. We attach special importance to the following points:

- The reports fulfill the instructions given in the templates, are structured according to the templates and are complete.

- The reports are written in a precise, correct, and clear way. *Brevity is preferable to verbosity*. Extracts from configuration files and lists of executed commands, for example, are not desired.

- The risk analysis corresponds to the project description and provides an adequate catalog of countermeasures. All decisions are justified.

- All discussions are objective.

- The system built reflects what is described in the report, and the functionality described in the requirements must be entirely present. However interfaces must only be functional, and their graphical attractiveness is irrelevant.

- The review of the other team's system is careful and systematic.

We provide detailed evaluation criteria on the course's webpage.

5 Timeline

Timeline	Milestones
30.09.2016	Project start, register teams.
24.10.2016	Submit the first rough system description and risk analysis, including a graphical sketch of the application architecture, a rough overview of the system with the most important components and information flows, the most important assets, and 3-5 of the worst risks/vulnerable areas.
	Submit by noon to `sprenger@inf.ethz.ch`.
28.10.2016	You get feedback to your rough system description and risk analysis (by email).
24.11.2016	Hand-in of complete system description and risk analysis, including a description of the security measures (max 30 pages). By e-mail to `sprenger@inf.ethz.ch`.
	Hand-in of the VMs on provided USB stick.
	Hand-over of your system to review group. Start of the reviews.
15.12.2016	End of reviews. Submit the system review (max 18 pages) by email to `sprenger@inf.ethz.ch`.
	Every group is expected to give a short (20 minutes) presentation about their system and their review highlights today.