System Description and Risk Analysis

Cyrill Krähenbühl Silvan Egli Lukas Bischofberger

Contents

1 System Characterization

1.1 System Architecture

Our system is a certificate authority. A user can retrieve his certificates, create new certificates and revoke certificates.

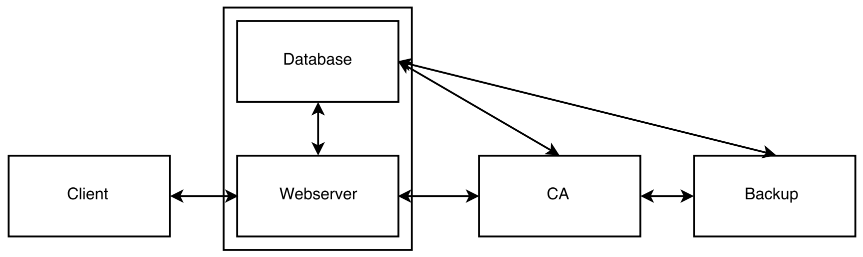

Figure 1: System overview

The architecture is shown in figure 1. All components except the client are in the same local network and are separated from the client by a firewall [siegli: are we using a dedicated machine (like a reverse proxy) or do we use a local firewalls on each machine ? in case of seperate machine I would put it into figure 1]. A client can only communicate with the system by connecting to the webserver. The webserver authenticates the user, manages the database and retrieves, creates or revokes certificates through the certificate authority (CA). The CA handles all certificate related operation. The backup server periodically backs up all data on the database and the CA.

1.2 Components

We divide our system into four major components.

- Client: The client side of the system is implemented using the javascript framework Angular [1].
- Webserver & Database: The webserver and the database run on the same Ubuntu system. For the webserver we use nginx [2] and for the database mysql. The web application is implemented using the python framework Django [3].
- CA server: The CA server runs on a Ubuntu system. All certificate related operations are implemented using openssl [4]. The CA stores all certificates and their corresponding keys.

[1]https://angularjs.org/
[2]https://www.nginx.com/
[3]https://www.djangoproject.com/
[4]https://www.openssl.org/

- Backup server: The backup server runs on a Ubuntu system and uses rsync to synchronize certificates, keys and revocation lists from the CA and user informations from the database.

1.3 Information Flows

- Client authentication: The client sends his username and password (or certificate) by using the client application to the webserver. The webserver (nginx) passes this information to the backend application (django) which then checks the username and password with help of the mysql database. If the user sent a certificate, this is checked by nginx. It must be ensured, that the certificate is valid and has not been revoked. Django then creates a JWT and returns it to the client application.

- Client certificate creation: The client checks in his view if he has already generated a key pair. Assuming he has, he sends a new certificate name to the webserver. Django requests a certificate from the CA server based on information from the database. The user is given the possibility to update his user information if he wants to. The CA server then creates the certificate based on the already present key pair and stores them safely. It returns the locations of these assets to django, which stores it in the database. Also it returns the message of the successful creation of the certificate to the client and offers the possibility to download the certificate, including the private key, in PKCS #12 format.

- Certificate revocation: The client sends a revocation request to the webserver. Django asks the CA server to revoke the certificate, which then adds it to the revocation list. The CA server responds with this updated revocation list.

2 Risk Analysis and Security Measures

2.1 Assets

Our project does not have physical assets. [siegli: maybe we later have to make the assumption that the system is running on real machines. but for the moment i'm okay with that (let's see what they say ;-)] All the servers and networks are virtualized and thus will be listed as virtual assets. We will distinguish between software assets and information assets.

Software includes our virtual servers, including the software running on them. Further we count the connectivity of the different components as a logical asset as the network is also virtualized.

- Frontend server: The frontend server runs a webserver and a database server. These are running an nginx and a mysql service on Ubuntu. Further our application is a django (python framework) system which delivers the frontend (Angular) application. The packages and frameworks used are automatically updated with the newest security patches. The django application is updated by the system administrators.

- CA server: The backend server system consists of a bunch of bash scripts as well as the openssl package, both running on Ubuntu. The packages are automatically updated with the newest security patches.
- Backup server: The backup server runs rsync on Ubuntu, it is also automatically updated with the latest security patches.
- Availability of service / connectivity of the components: We rely on the virtualized network which is configured externally to work consistently [siegli: why consistently ?] for our application to function.

Our information assets include the following:

- Certificates: This includes our own server and CA certificates as well as all the user generated certificates. Whereas the secrecy of the certificates is not critical, the security of the private keys is crucial.
- User data: User data includes the users names, emails but also passwords. The security of this information is also critical to our application.

Further assets are the people working with the product. This includes the engineers building the platform and preserving the technical knowledge but also spans to the customers (employees of iMovie) using the product, wherein also originates the customer relation.

2.2 Risks and Vulnerabilities

- Weak passwords:
  - Users: Weak passwords of employees let advisories issue and revoking certificates in their names.
  - System Administrator: Weak passwords of system administrators allow advisories access to the system with unbound possibilities.
- Security Awareness and Knowledge of Employees: If the employees do not know how to request certificates and how to use them to sign/encrypt messages. This also includes the the protection of the private keys.
- Unavailability of CA: Threat actions like DOS attacks can make the CA unavailable. This results in making signing/verification of messages impossible as the public key of the CA can not be fetched.
- Misconfiguration: The system administrator might expose the system to the outside by opening unintended ports for example.

- Threat sources: The iMovie company has a focus on investigative reporting. Depending on the subject being reported the threat source might include organized crime or governmental agency.
- Information leakage: if confidential data (e.g. about a movie) is leaked customers might loose confidence