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.

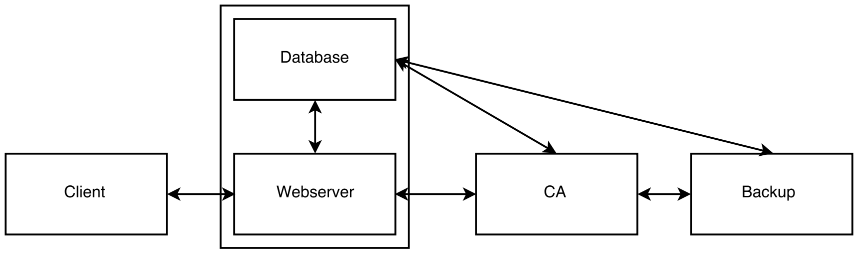

Figure 1: System overview

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. 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 can divide our system into four major components.

- Client: The client side of the system is implemented using the Angular framework[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://angular.io/
[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

Here we present some of the most important 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 It must be ensured, that the certificate is valid and has not been revoked.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 in P12 format as well as the private key.
- Certificate revocation: The client sends a revocation request to the webserver. Django asks the CA server to revoke the certificate, which also adds it to the revocation list. The CA server responds with updated this updated revocation list.

2 Risk Analysis and Security Measures

2.1 Assets

Our project does not have physical assets. All the servers and networks are virtualized and thous will be listed as virtual assets. So 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. This are a nginx and a mysql service running 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 developers.

- CA server: The backend server system consists of a bunch of bash script 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 newest security patches.
- Availability of service / connectivity of the components: We rely on the virtualized network which is configured externally to work consistently for our application to function.

Then we also have information as assets, this includes 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.

Our further assets are the people working with the product. This includes the engineers building the platform, preserving the technical knowledge. But also spans to the customers using the product, wherein also originates the customer relation.

2.2 Threat Sources

threats ——

- key leakage - user data leakage - compromise of webserver, ca server. backup server - unavailability - misbehaviour of trusted entities - sysadmin access to system

vulnerabilites ————-

- misconfiguration of used software - bad passwords - badly implemented software (eg. access control) - zero days in used software - lack of security awareness of employees - sys admin access privileges