Greeting App

Santiago Gil

May, 2018

Greeting App

This is a demo of the Greeting App. It consists of a user-facing web application and a backend API.

As of this writing it is just a demo, and should **not** be used in a production environment.

Next we will describe the basic details of each component of the Greeting app.

Dependencies

All dependencies are exclusively under open source licenses. All of them have been used as-is in this demo, without special modifications.

Web Application

The web application is powered by `Flask`, a Python framework.

The frontend consists of static HTML5 files and makes use of Bootstrap CSS library.

Implementation Details

User Credentials

The users credentials are stored in an SQLite database. The information stored consists of `(username, password)` tuples, where the password is hashed and salted using the `bcrypt` algorithm.

We currently allow only usernames consisting of alphanumeric characters up to a length of 30 characters.

Communication with the Backend

When a user requests his or her greeting, the web application issues a `GET` request to the backend to fetch the greeting message that should be shown to the user. That request consists of a signed *token.*

The webapp owns a pair of RSA keys [1]. In order for the backend to validate that the request came from the webapp, the web application signs every token with its private key.

The token is a string consisting of three fields separated by colons: `'username:timestamp:signature'`, where:

- `username` is the user's chosen username,
- `timestamp` is an UNIX timestamp (integer number), and
- `signature` is a url-safe Base64 encoded value that represents the RSA signature of the token's first portion, `'username:timestamp'`, using the web application's private key, SHA256 as a hashing function, and PKCS #1 1.5 padding.

The timestamp is used as an enforcing mechanism for the token's expiration time (it is only valid for 30 minutes after being issued).

Backend API

The backend API is implemented as a single Java class.

It consists of an `HTTPServer` that listens on port 8080 and responds to requests to the `/greet` path.

Requests

More specifically, it only handles requests of the form `/greet?token=T`, where `T` is a token in the format described above.

When a well-formed request is received, the backend first proceeds to verify the signature portion of the token. (For that it needs a copy of the web application's public key.)

If the signature is valid, then the timestamp is checked to ensure that the token was issued not longer than 30 minutes from the current time. Since the timestamp field of the token is signed, we can trust that it was issued by the web application, and has not been tampered.

If both checks are successful, the API returns a personalized greeting message that uses the `username` value.

[1] They are generated by utility script `rsa-keys.py`.