Security Analysis

CentroGeo Interactive Dashboard, Team 27

Secure Token Generation

In order to facilitate authentication in our system we were required to implement a secure way to generate tokens that would be shared among the users that have logged into the system providing a way for the server to differentiate between clients that are authenticated and that are not.

Each secure token consists of 20 randomly generated “secure” bytes. Secure bytes in this context means that the bytes were generated using a “SecureRandom” class which would make it difficult for an attacker to generate a valid secure token since this pseudo-random function lacks predictability. The bytes are encoded into a string and are then used as tokens.

Use of prepared statements

All of the sql statements that contain users input make use of prepared statements making it not possible for an attacker to attempt an sql injection as the input of the user is isolated from the sql query in the case of an sql injection a statement will consider the injected code as a parameter in the “where” clause likely returning no matches for the submitted sql code.

Use Scrypt to encode and verify the password and stored in the database

All of the user’s passwords are first encoded by using methods of encode() and match() from SCryptPasswordEncoder. This method uses the SCrypt algorithm to hash password.Since it is a kind of slow hashing way to encode password, so it is very difficult to revert back to the original raw password in a short time even if the database is compromised.

Authenticating the users

Once the user enters correct credentials, the backend generates the token mentioned above, and assigns it to an http only cookie, isolating it from the Javascript. The cookie is then sent to the user through the header. For all future requests the browser will automatically send that cookie back to the server. The cookie is only valid for 24 hours, when it expires the user has to log in again.

All the important methods have been annotated with @secured, meaning that when the method is called, it first has to go through the authentication filter, which looks at the header and extracts the authentication cookie. If valid, the method is called normally, otherwise it aborts the action.