Token-based Authentication

With most of the applications are API based, it’s very crucial to implement security for all clients trying to access data through these services.

The most prominent approach is to authenticate the clients over HTTP using a signed token. Token-based authentication ascertains that every request made to a server is associated with a token, which is then validated by the server for authenticity to further respond to the request.

Users instead of entering their usernames and passwords at each subsequent request, send the token provided by the server to validate it and accordingly take action.

To improve the token-based authentication the following steps can be taken.

* Using strong token values by using an extremely large set of possible values. Make the tokens long enough (at least 16 bytes)
* Transporting the token in a secure channel like using https application.
* Manage the expiration time for the token.
* Re-authencation of the token. It’s very important to re-authenticate the token, especially during the banking transactions. Before submitting solid action require users to re-enter their credentials or ask users to provide some alternative (temporary) PIN numbers or smart IDs.
* Allow the user to have multiple logins at the same time. A simple way to address parallel authentication issues is to store temporary attributes somewhere inside the user’s row in a database or cache. A couple of examples include “logged = true” and “user agent = IE 11.1.1” (or maybe even the active token). Storing key attributes can help make sure that the single account will never be used by two different people at the same time. Obviously, if this happens, someone has to be kicked out.