3b - Authorisation

## Learning Outcomes

On completion of this lab you will have:

* Implemented two types of API authentication

## Organisation

Please complete the exercises individually.

## Grading

This worksheet is worth up to 5% of your overall module grade.

**Note**: You must attend and sign in at 10 labs in order to obtain full credit for your submitted worksheets. You may work on this worksheet during lab 5 with instructor assistance.

## Submission

The deadline for submission is Sunday Mar 19, 2017 @23:59 through Webcourses.

## Demonstration

You will demonstrate your solution to the lab instructor during the lab 8 session.

## Requirements

For this lab you will need to

* Review the related module lecture material on Webcourses (lectures 14-16)

## Resources

You are free to research whatever you need to solve the problems in this lab. Some recommended resources include:

* <https://jwt.io/>
* <https://github.com/auth0/node-jsonwebtoken>
* <https://www.postgresql.org/docs/current/static/pgcrypto.html>
* <https://www.wolfe.id.au/2012/10/20/what-is-hmac-authentication-and-why-is-it-useful/>

## Problem Sets

The following platform-independent tasks can be solved on Windows, Mac local Linux or Cloud Linux as you prefer

Extend the Courts solution you developed providing two separate API authentication mechanisms.

|  |  |  |
| --- | --- | --- |
| **1** | Implement a users table having a username and hashed password fields. Use the postgresql crypt() and gen\_salt() functions to implement the password hashing | 10 Marks |
| **2** | Implement a JWT-secured version of the API based on the users table from the previous step. Your solution will implement the following API extensions   * A (pre-authentication) login API call which accepts a username and password and returns (if successful) a JWT with a set of claims. The claims should include, minimally, the user id and an expiry timestamp * A mechanism to verify client tokens as bearer tokens in a HTTP Authorization header field * Authentication should be applied, minimally, to any API calls which update the courts systems models; Token validation should be performed on all API calls * Assume the client has a priori knowledge of the password   If authenticated or validated, the API return code should be in the 2xx range, otherwise 401. | 40 Marks |
| **3** | Extend the users table or add another apikeys table to include an access key (160 bits) and secret key (320 bits) | 10 Marks |
| **4** | Implement a Hash-based message authentication scheme to secure the API. In your solution you should include the following API message contents as part of the hashed/signed component:   * Message body (if any) * Access key (prepended or appended as you choose) * Query parameters (if any)   If authenticated, the API return code should be in the 2xx range, otherwise 401. | 40 Marks |