Technonomicon API Documentation

Olivia Trewin

May 6, 2018
Version 0.1

1 Introduction

The Technonomicon is a reverse-engineering grimoire. It allows the public to view and create records of reverse engineering efforts and results. These records are organized to make browsing easier and facilitate teamwork on complex reverse-engineering problems.

1.1 High-level Overview

Interface with the Technonomicon is done using a REST API, which uses JWT tokens for authentication. Users are identified with a username, use their email for password resets, and have a user-specified password which can be used to receive an API token. All interactions with the API are stateless, in keeping with REST practices.

1.2 Purpose of this Document

This document intends to describe every possible interaction with the REST API, as well as provide some useful example pseudocode. To see usage information for reference bindings for various languages, consult the binding repository or, if present, the package's man page.

2 User Creation and Authentication

These calls are for creating users and retrieving authentication tokens.

2.1 /api/register

- HTTP Verb: POST
- Authorization Required: No
- Parameters:
  - username - String - The username of the user to register.
  - email - String - The email of the user to register.
  - password - String - The password of the user to register.
- Response:
  - success - Boolean - True if registration succeeded, false otherwise.
  - errors - String[] - A list of errors that caused the registration to fail. Undefined if the registration succeeded.

2.1.1 Description

Attempts to register a new user with the specified username, email, and password. If this succeeds, it will send a validation email to the specified address. This should be followed with a call to /api/validate once the end user receives the validation token.

2.2 /api/validate

- HTTP Verb: POST
- Authorization Required: No
- Parameters:
  - username - String - The username of the user to register.
  - emailToken - String - The validation token the user received in their email.
- Response:
  - success - Boolean - True if registration succeeded, false otherwise.
  - errors - String[] - A list of errors that caused the registration to fail. Undefined if the registration succeeded.

2.2.1 Description

Attempts to validate the user's email address. After this call succeeds, the user is allowed to receive API tokens and call further functions.

2.3 /api/token

- HTTP Verb: POST
- Authorization Required: No
- Parameters:
  - username - String - The username of the user to get a token for.
  - password - String - The user's password.
- Response:
  - On success: A valid JWT token.
  - On failure: A 400 Bad Request response.

2.3.1 Description

Authenticates a user and produces a JWT token that can be used to validate subsequent API calls. To use this token on functions which require authorization, add an HTTP header called "Authorization" with the value of "Bearer", a space, and the received JWT token to all requests. By default, this token is valid for 1 day, after which the caller must call /api/token again to receive a new token.

3 Data Model Manipulation

These calls allow the retrieval and alteration of data model objects, like users, posts, tags, and events.

3.1 /api/user

- HTTP Verb: GET
- Authorization Required: No
- Parameters:
  - id - GUID - The unique ID of the user to get information for.
- Response:
  - userId - GUID - The unique ID of the specified user.
  - username - String - The specified user's username.
  - email - String - The specified user's email.

3.1.1 Description

Retrieve information about a user using a GUID, including their username and email.