Source Code Security Evaluation

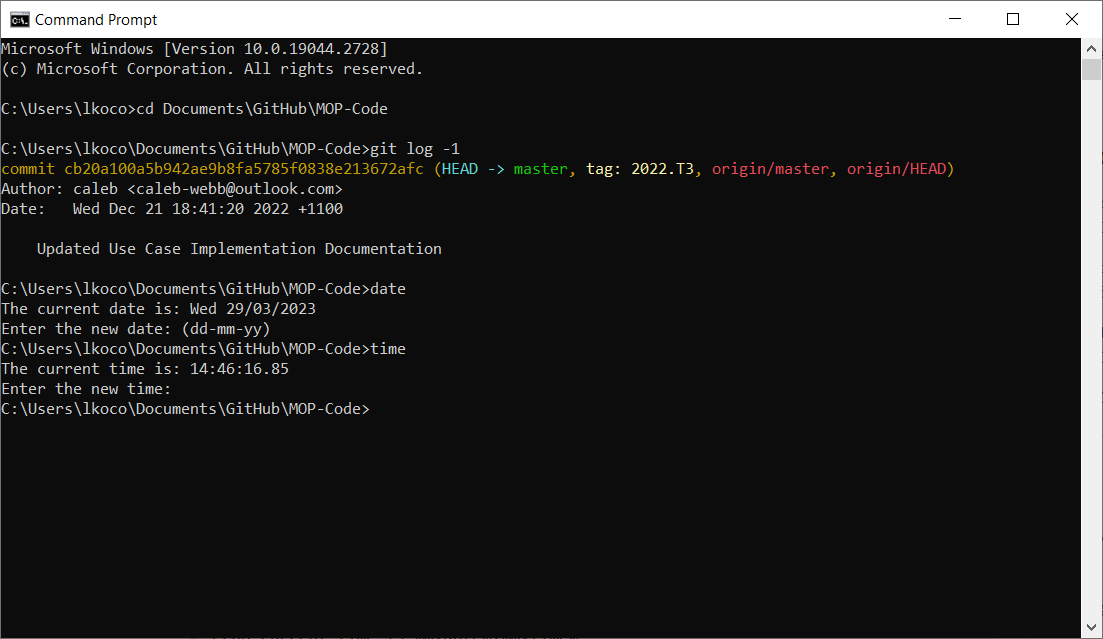
## Executive Summary

Chameleon’s “Melbourne Open Data Playground” team has recognised a need for security measures in the project to work on before its deployment. A lot of security weaknesses can be spotted in human-developed solutions rather than in end-user-facing situations. This report is created to report on the project’s security implementations at time of writing, and the major objective is to inform development members, present and future, of key security fixes to make while the product is still public.

The current problems found during this report were an inadequate content security policy in place for the website, and procedural details exposed by development.

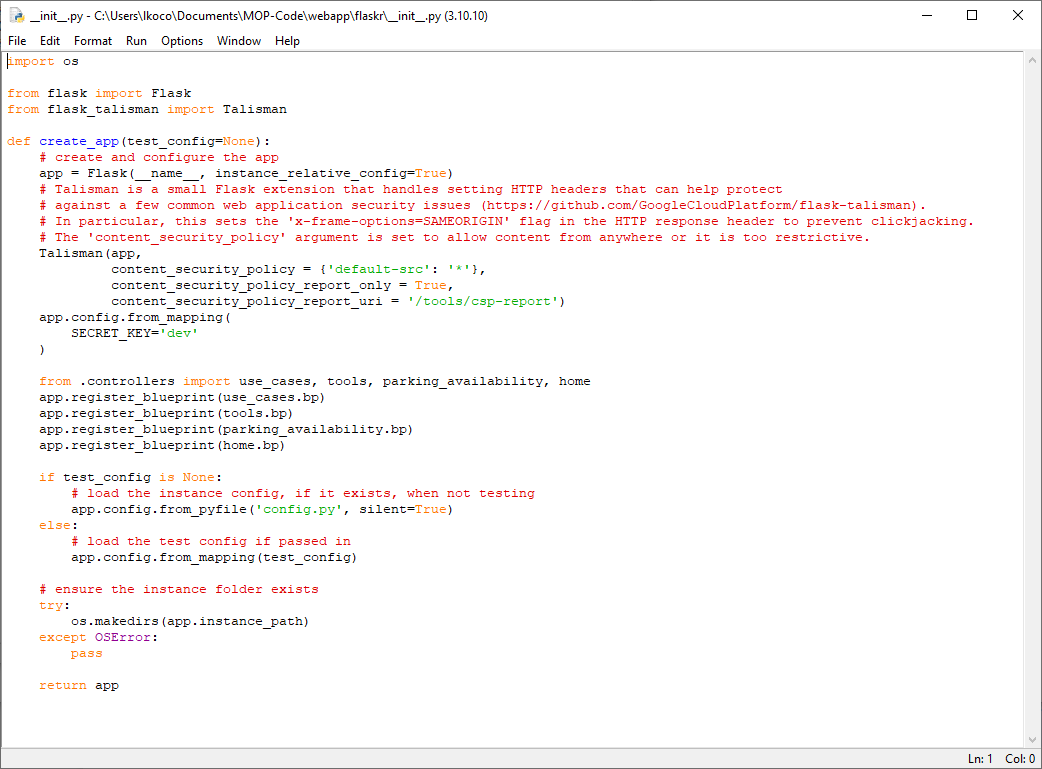
## Current file version

The project has been distributed within the team through GitHub, where the project and its contents are parts of a snapshot system to track changes, implement merges of project progression and rolling back any broken features. Each “commit” that tracks progress is identified with a hash using SHA-1, which is good enough in identifying and comparing a single commit against the others. This is the current commit ID that is being tested against in case the project overall, or any files in specific, changes during this assessment.

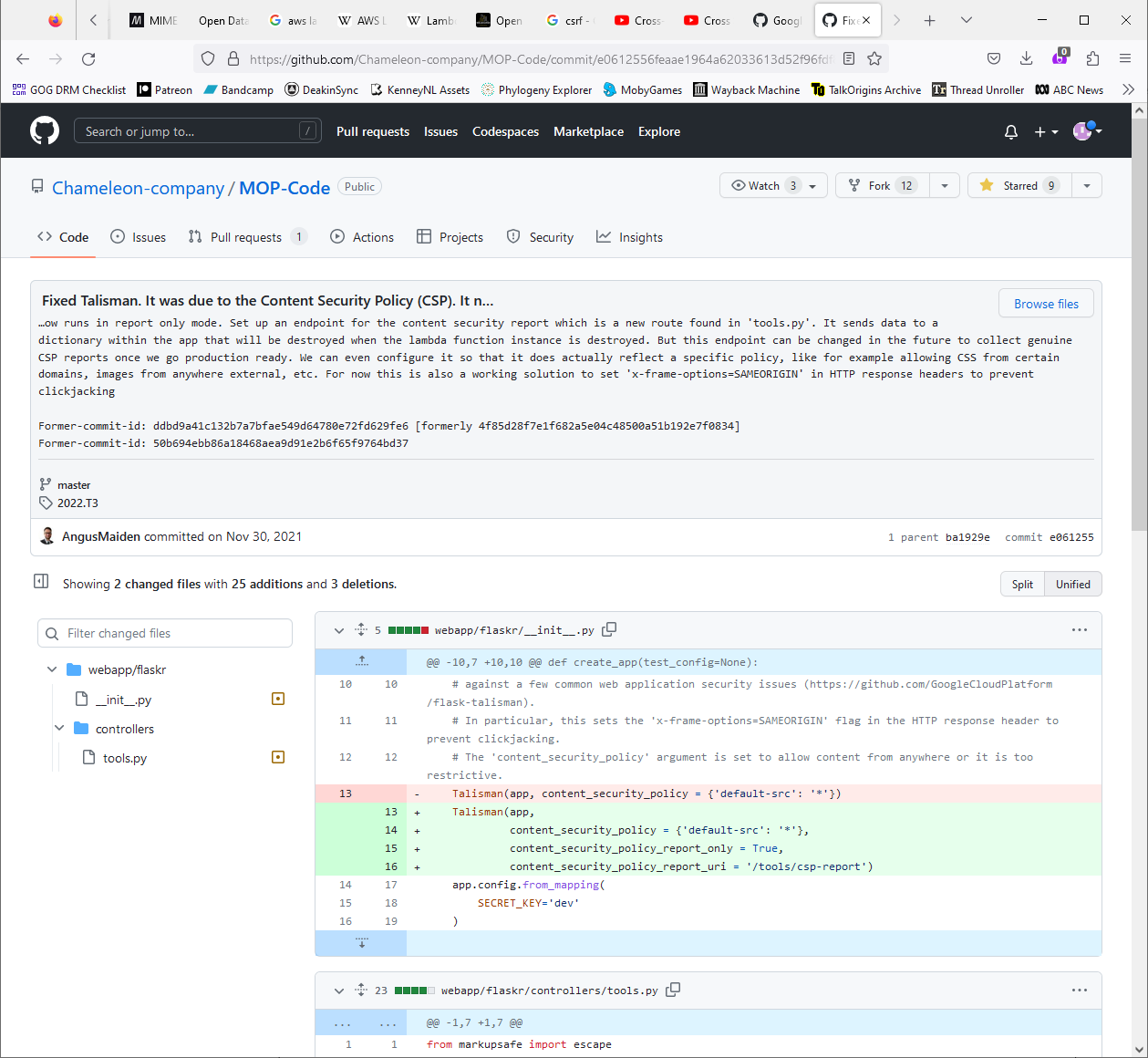


Lax Content Security Policy

Within the core of the project is the main initialisation file that sets up the Talisman procedures that sets up most of the security measures. Documentation made by the developers indicate that it is initialised to have an inflexibly strong security, but the implementation indicates how some features have been opted out where one key concern is the content security policy (CSP).  
  
The author has noted that the default content security policy was too restrictive, but that can be alleviated in concretely is expected to change, while sites that host dynamic content are named and included in the content security report. The strictness of the default CSP is already noted in the Talisman documentation as “the only setting that you should reasonably change”. This policy needs to be amended before any public-facing processes are implemented (such as the existing but non-functional Contact page), as such a lax CSP is prone to some basic web attacks against the project’s infrastructure or the users.

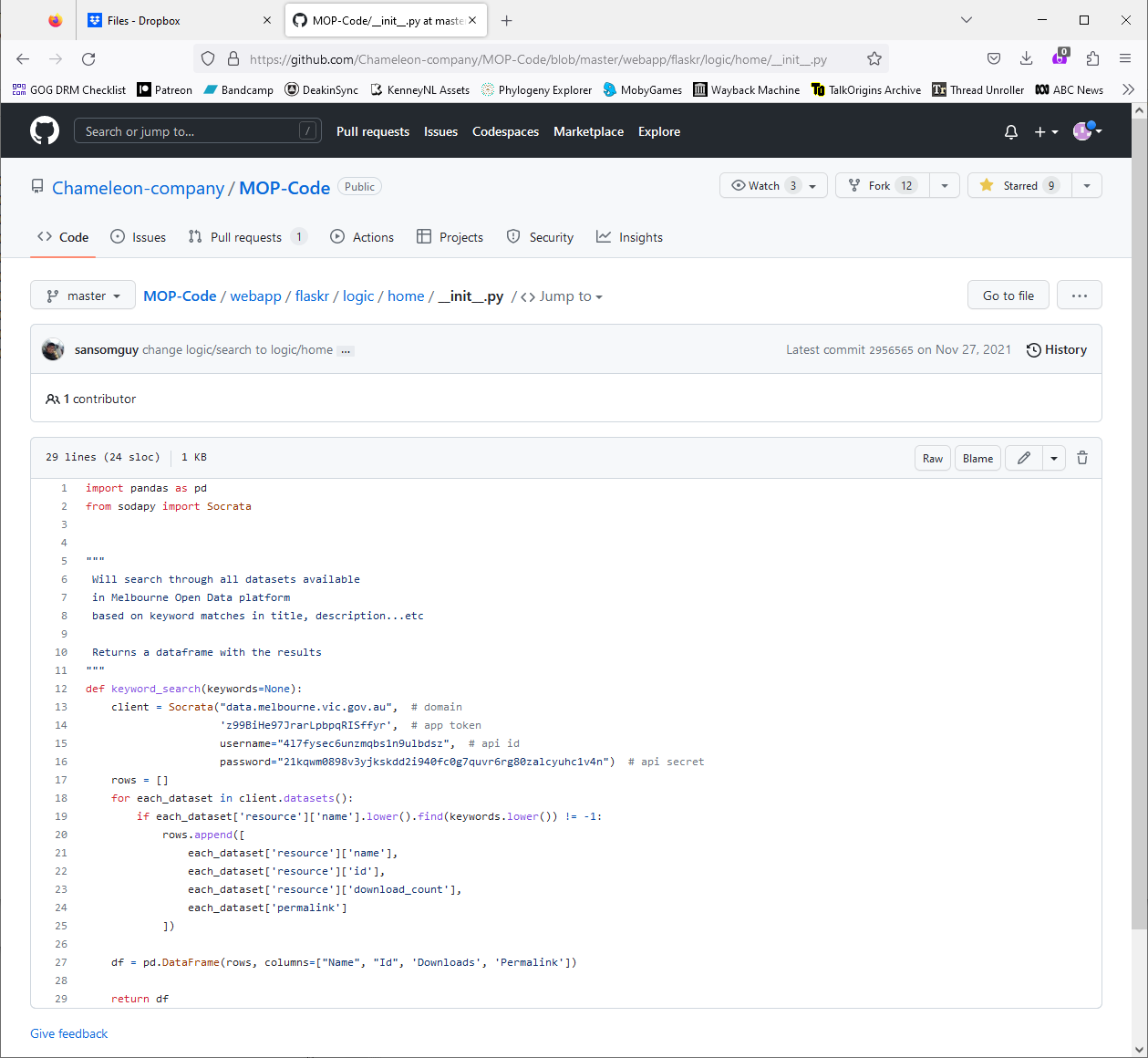


One file has noted it has faced issues with Talisman during production because of the strict content security policy in T3 of 2021, where it may have had issues of working during production and put in place a plan to modify for when product launches. I recommend this changing already instead of at the end while the product is already public. This submission also mentions that it now allows “X-Frame-Option = Same-Origin” so clickjacking attacks are prevented, but Talisman already does this by default.



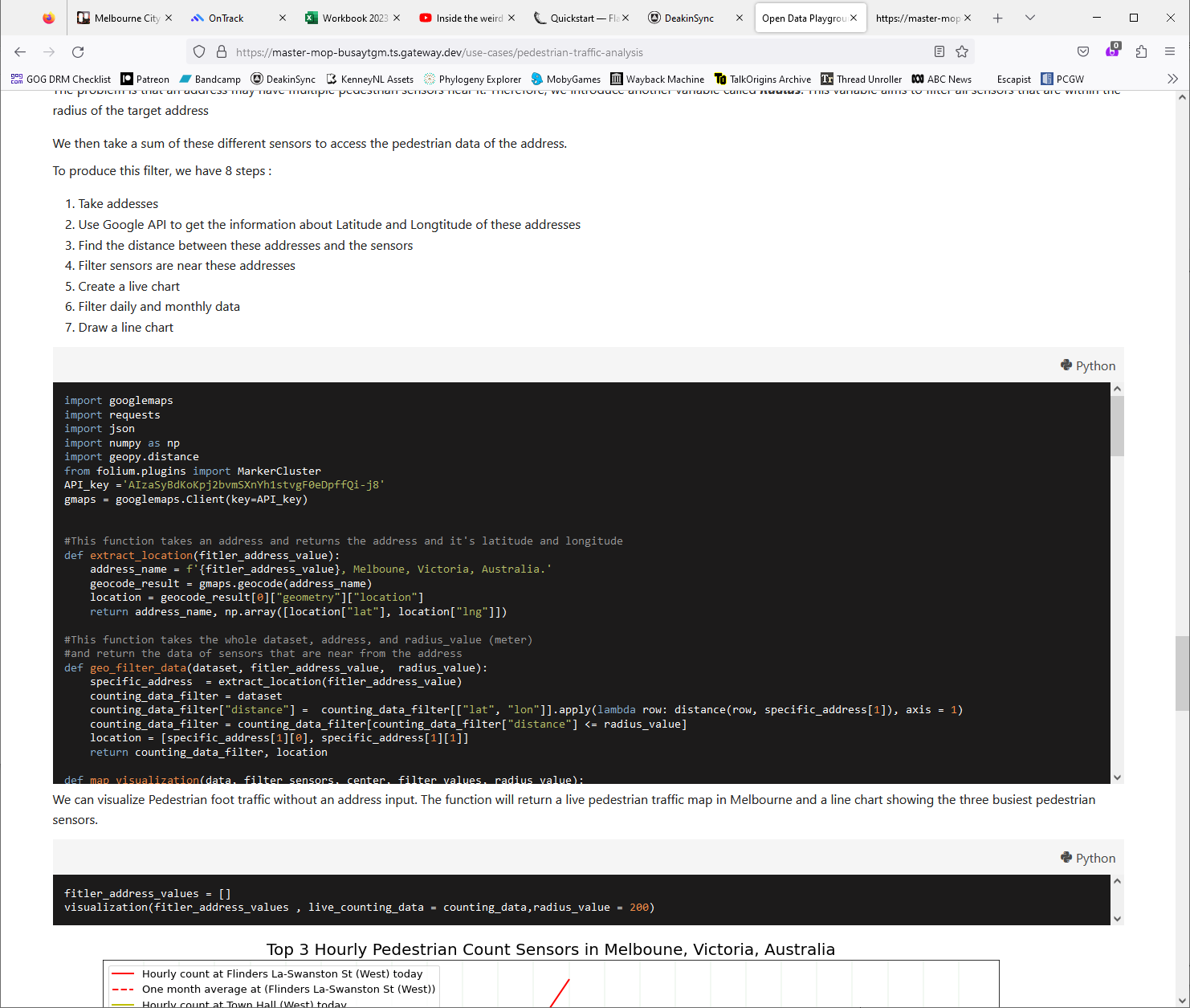
## Credential Leaks

Within one of the Python scripts that’s publicly viewable on the company repository is how the project’s API credentials are written in plaintext. Initially I thought these would be details that can be used to log in to the Melbourne Open Data, and thankfully they are not; the consequence of them being used to login to the project’s Open Data account would be a lockout that breaks the project’s continuity, but using these details doesn’t sign into any account so such threat is void.



Regardless of the impact that this technique could have caused, it reveals how credential data is not being hidden by developers that attackers don’t even have to try in finding. Not hiding key data like this might happen later in the project and these would be intel attackers could gather in deploying attacks against the project.

Right now, notebooks that process data and visualise solutions within the project are shown within the website as a teaching tool, but they also don’t obfuscate their API keys within the scripts they show. These techniques within the notebooks were discovered by team leader Angie who was reading over how the previous notebooks had worked and was disappointed in how members hadn’t obfuscated data like this.



This has already led to Angie bringing up in a mentoring session (the main objective of that was to mentor what to do with Melbourne Open Data’s new API being deployed which didn’t exist before) about sourcing API keys in a separate file and obfuscating it so they aren’t leaked. Going forward I would recommend a small mentoring session, perhaps as part of up-skilling new members, in obfuscating credentials like this in their project submissions.