Logo

Description automatically generated

**Software Quality Assurance (COMP 6710)**

**Prof: Akond Rahman, PhD**

**Project Report**

**Date: April 27th, 2023**

**Done By:**

**Logan Smith - lws0022**

**Satya Sai Teja Charla - szc2039**

**Table of Contents**

|  |  |
| --- | --- |
| Description | Page Numbers |
| Summary | 1 |
| Problem Description | 2 |
| cpmFsys project | 2-9 |
| Output | 10 |
| Design Implementation | 11-12 |
| Conclusion | 12 |
| Reference | 12 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Summary:**

The objective of this project is to integrate software quality assurance activities into an existing Python project. Whatever we learned from our workshops will be integrated in the project by apply the following activities related to software quality assurance:

1. Create a Git Hook that will run and report all security weaknesses in the project in a CSV file whenever a Python file is changed and committed.
2. Create a fuzz.py file that will automatically fuzz 5 Python methods of your choice. Report any bugs you discovered by the fuzz.py file. fuzz.py will be automatically executed from GitHub actions.
3. Integrate forensics by modifying 5 Python methods of your choice.

**Project for Software Quality Assurance (CSC 5710/6710)**

**1. Static Analysis:**

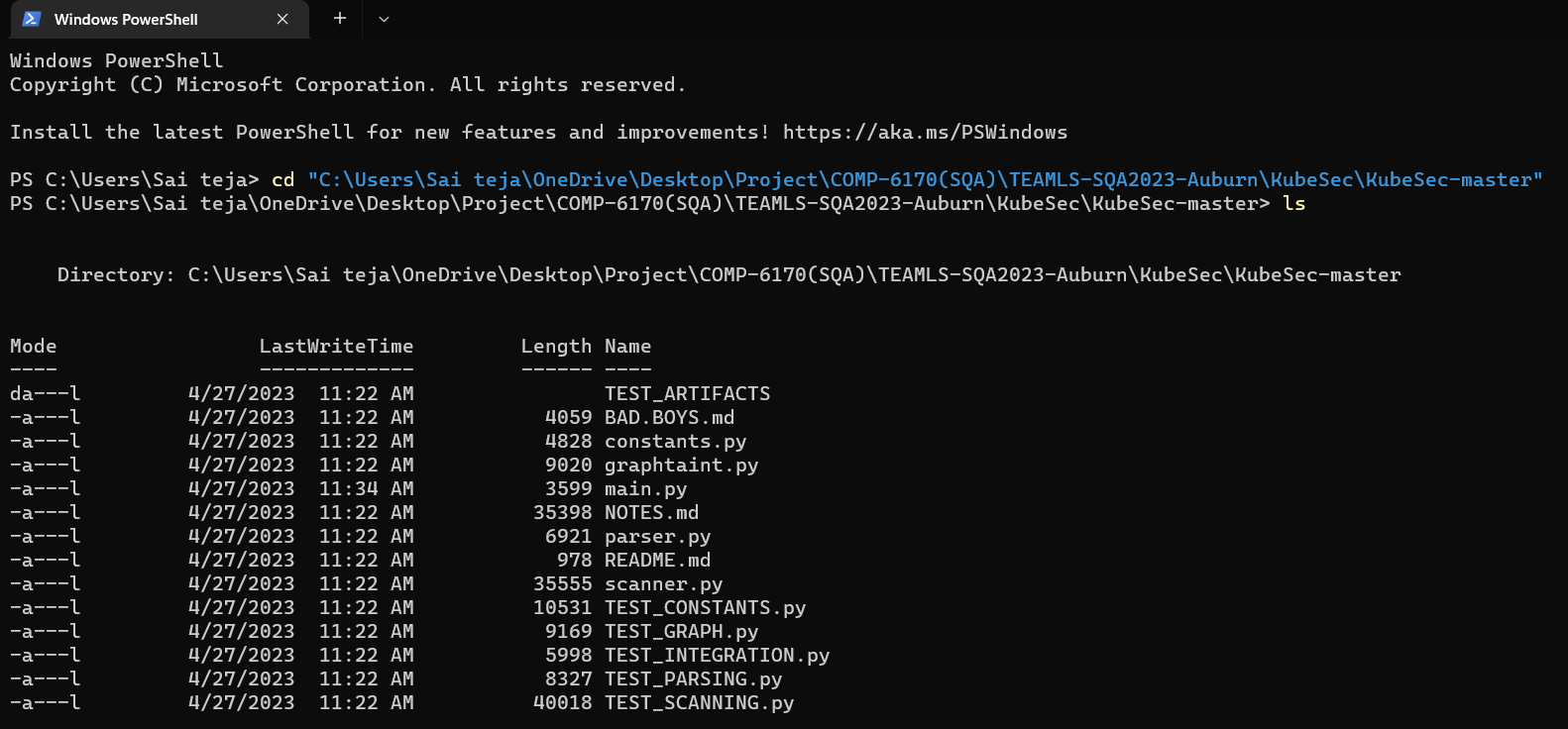
By Sai Teja szc0239

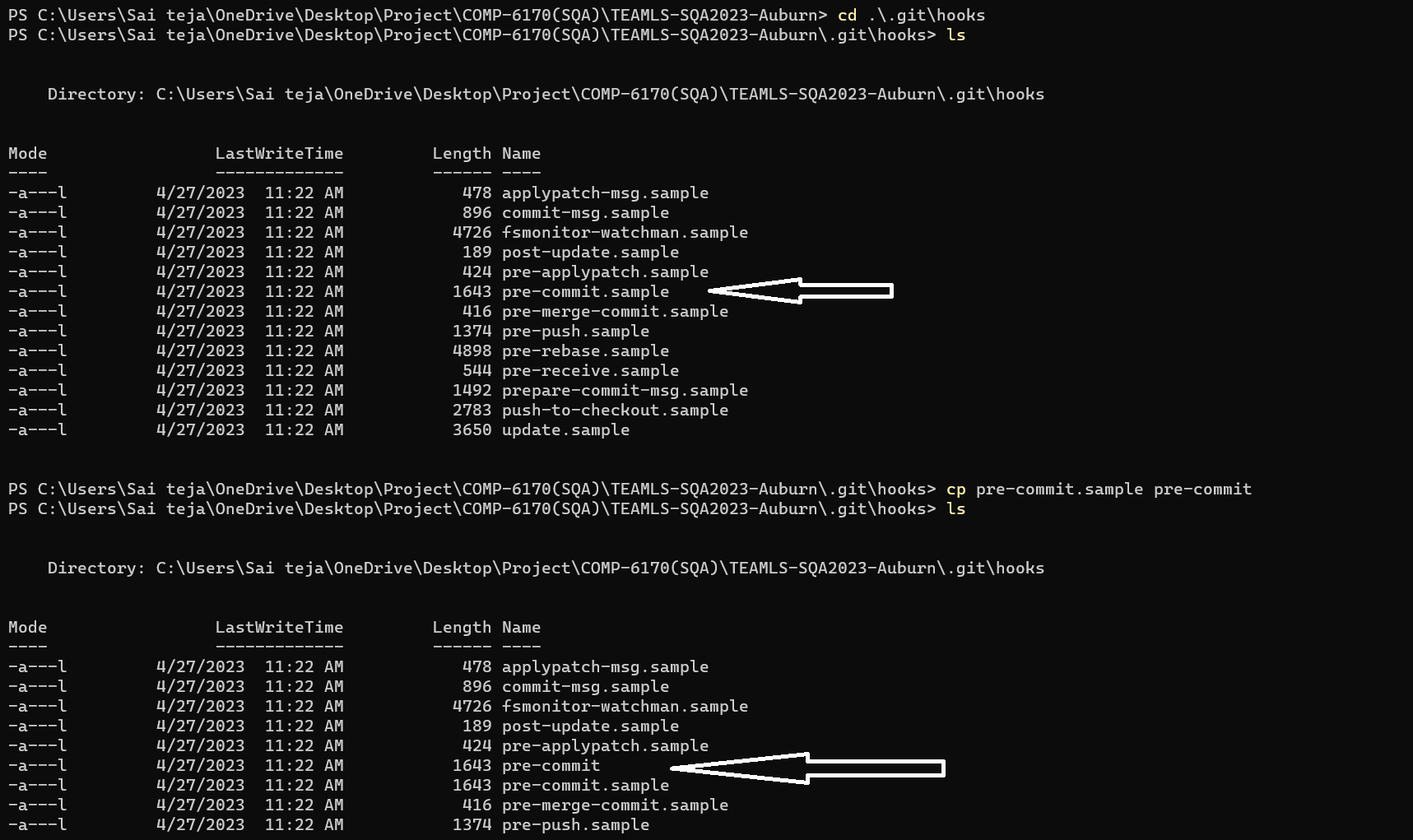
After creating a git-repo, I’ve cloned the repository onto my machine and made some changes to

**‘./git/hooks/pre-commit.sample’** file by copying the contents in that file and created a new file named ‘**pre-commit’** after that I’ve modified the main.py file in the same repository such that to see the effects of the modified **pre-commit** hook.

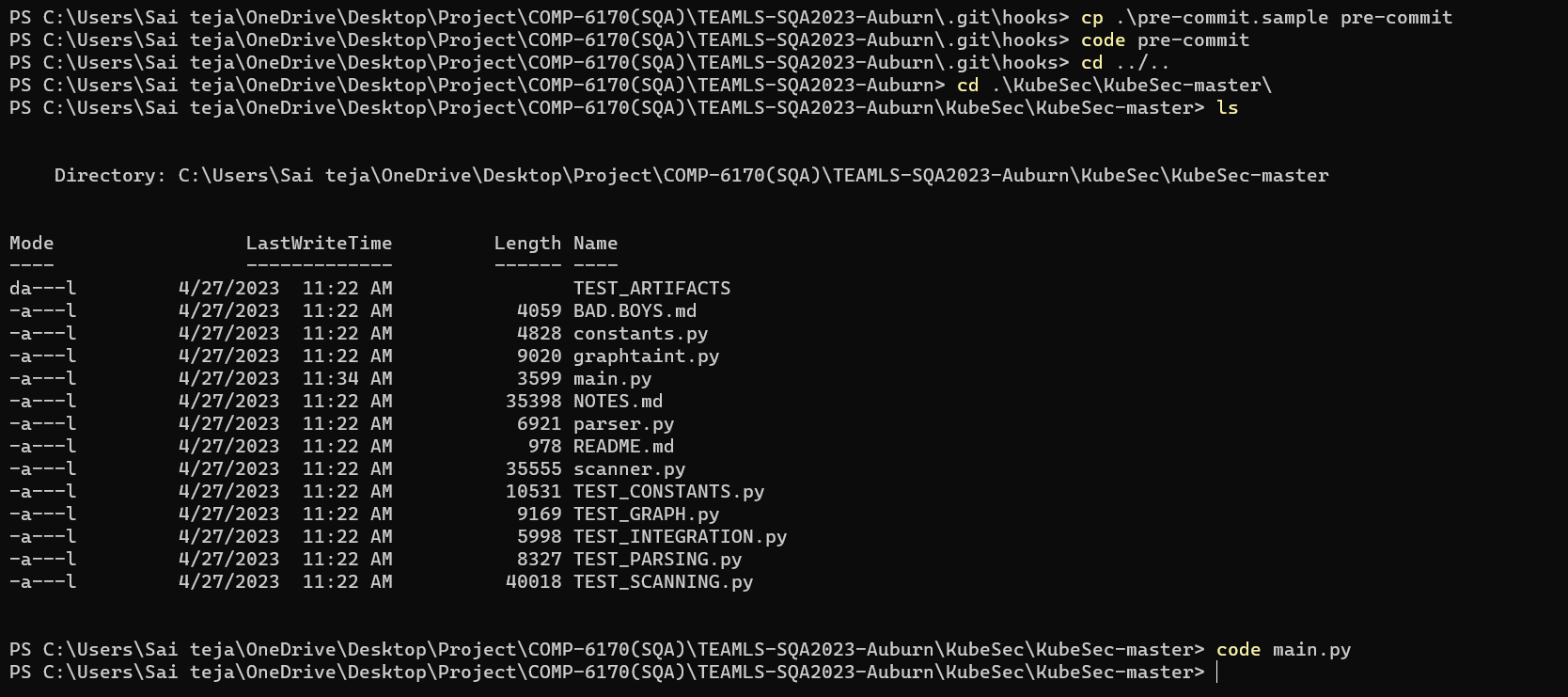
Finally, I ran bandit -r command to see any security weaknesses in the provided file and recorded the output.

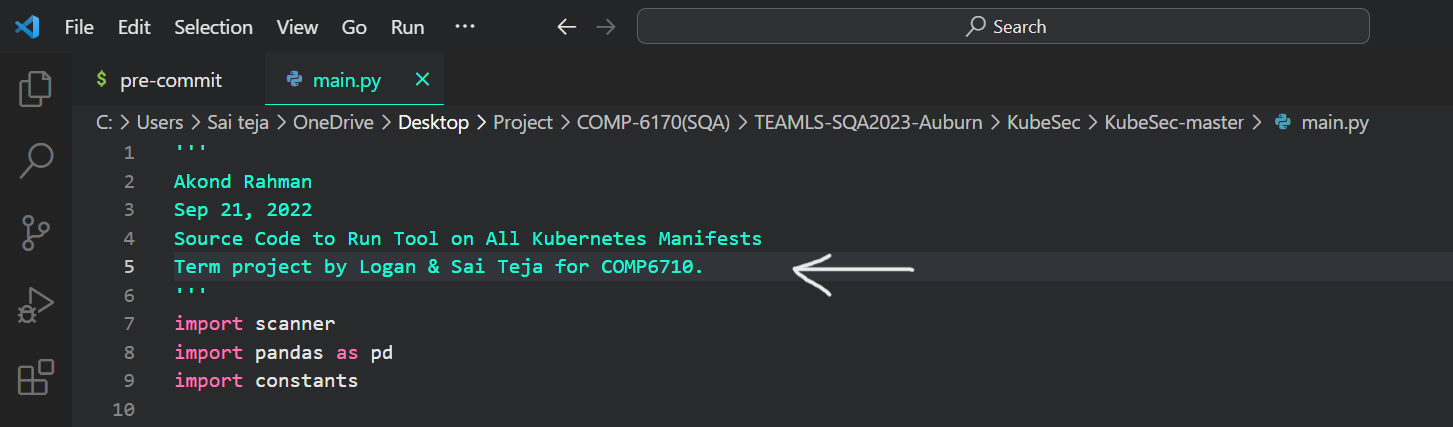
Down below are the screenshot taken during the execution of this segment:





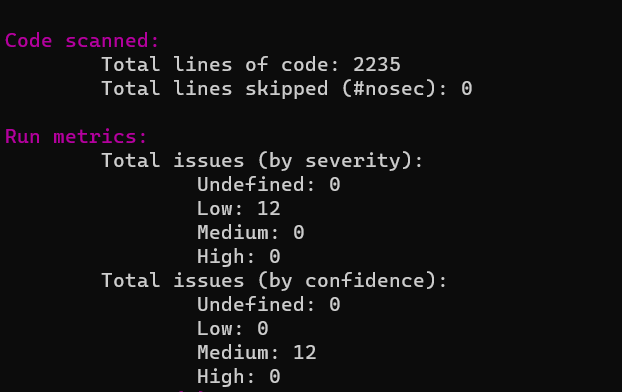
Modifying pre-commit file



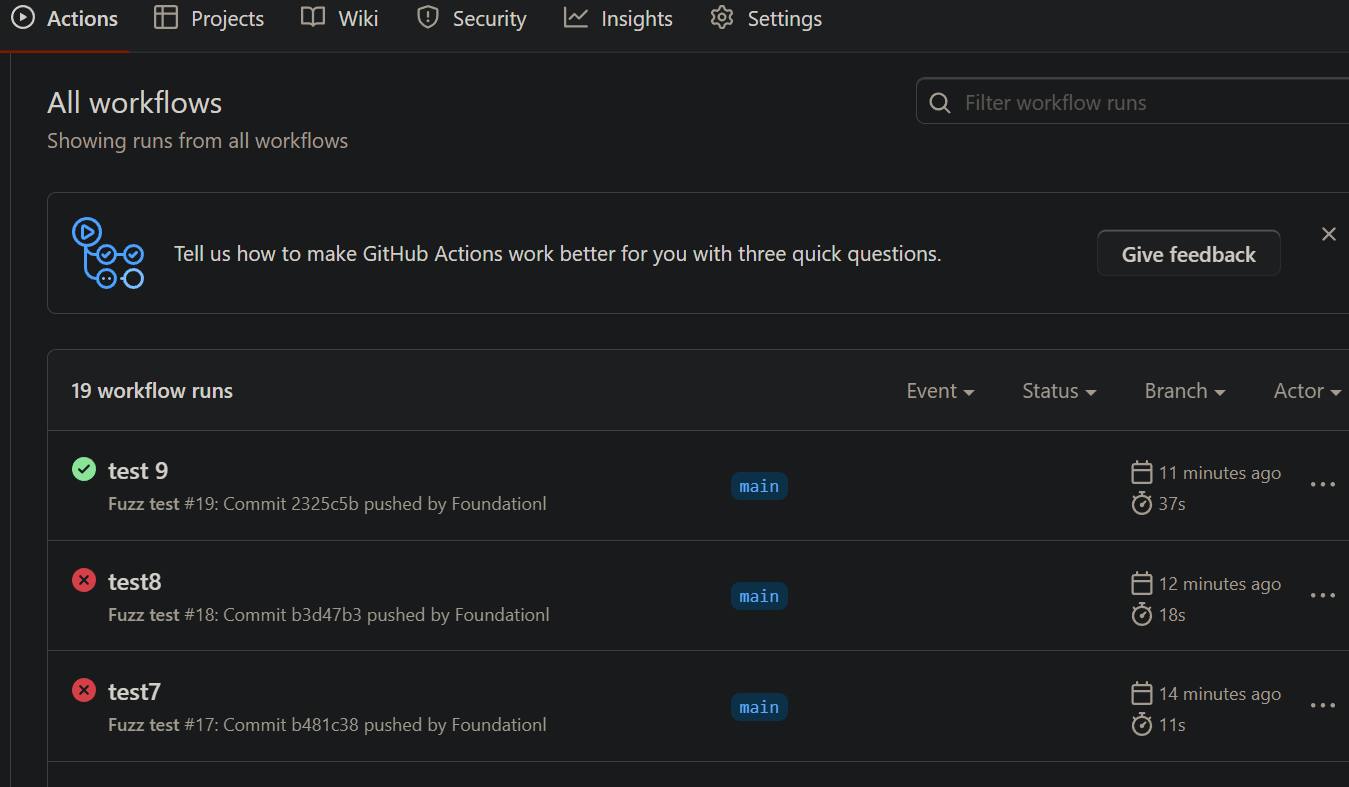
Git status


**Bandit output:**

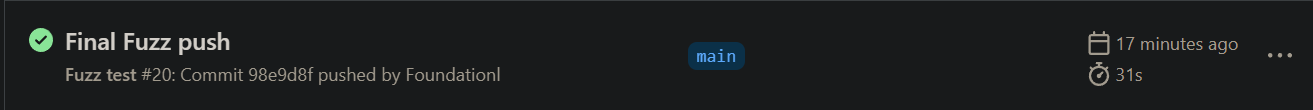


**1. Fuzzing:**

After creating a git-repo, I’ve cloned the repository onto my machine and made some changes. First, I had to add a **‘.github/workflow/main.yml’** file so that on actions like pushing, the fuzzing will run on the 5 chosen functions and print a report. After several test and review iterations I was able to print a report from workflows.



I now was ready to set up my Fuzzing function. This involved finding 5 methods throughout the zip to test. These chosen methods where {Class Scanner [Function isValidUserName, isValidPasswordName, isValidKey], Class Parser [Function keyMiner, checkIfValidHelm]}. The inputs chosen where a random generated int, a random generated string of fixed size, and NULL. The Fuzz.py will test those 5 methods with these inputs. The fuzzing function would then print successful and unsuccessful tests in a report. The unsuccessful test will also display the error associated with the failure. When pushing to GitHub you can go to actions and find under workflows the latest commit and view report, for reference the final working Fuzz push.



In this workflow you will be able to see the fuzz report which prints the tests after first iteration.

**Fuzz.py output:**

Graphical user interface, text, application, email

Description automatically generated

A picture containing text

Description automatically generated