AUBURN

UNIVERSITY

# Software Quality Assurance (COMP 6710)

**Prof: Akond Rahman, PhD**

**Project Report**
**Date: April 27th, 2023**

**Done By:**
**Topher Daisy -cwd0032**
**Surya Pradeepthi Chakka- szc0238**
**John Chong- jhc0065**

## Table of Contents

**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.

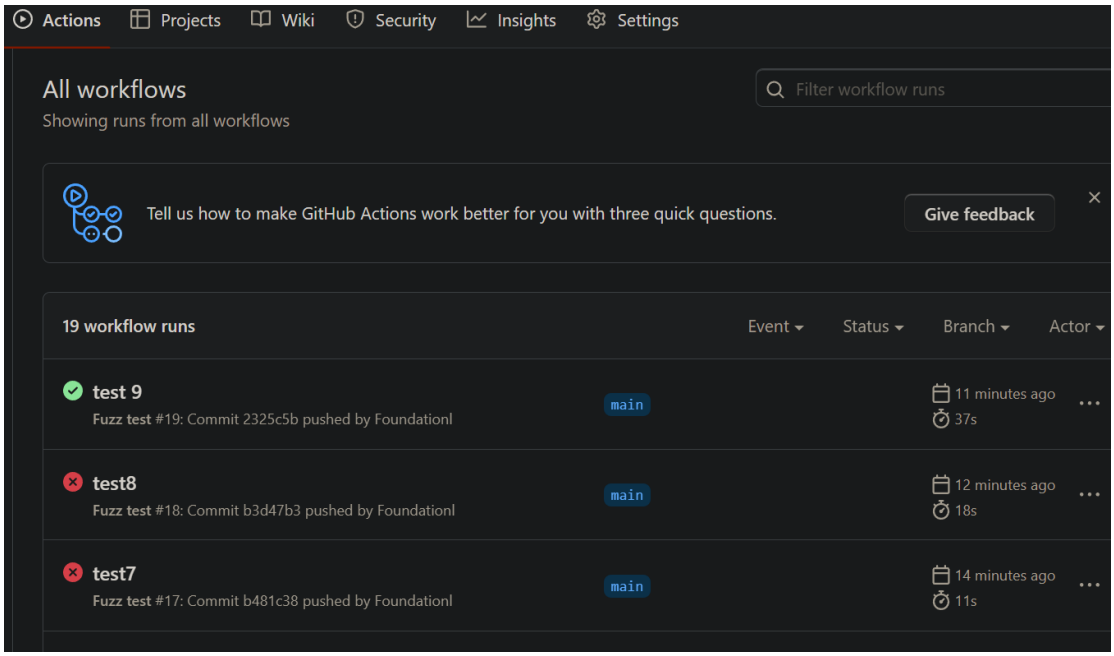
## Project for Software Quality Assurance (CSC 5710/6710)
## Team TSP-SQA2023-Auburn

**T**opher Daisy – I did part 4a- creating a Git Hook that reports the security weaknesses of python files into results.csv. This file includes 17 security weaknesses in the TEST_CONSTANTS.py, constants.py and parser.py files. Inside the csv file you have the information broken down by file name, test name, test id, issue severity, issue confidence, issue cwe, issue text, line number, col offset, end col off set, line range,

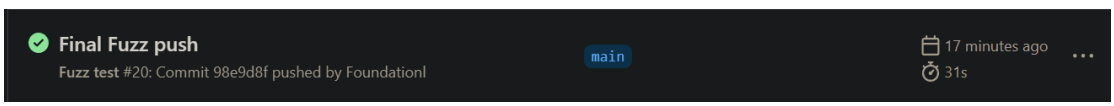
and extra information. These files have passwords and different security issues that need to be addressed. Throughout this part of the project, I learned a lot about bandit and how Git Hooks could be used to analyze security weaknesses or faults. Lastly I put 4a and scanning into a separate folder in order to easily see where the pre-commit and csv files are.

1. **Fuzzing :**

After establishing a Git repository, I proceeded to duplicate it onto my local machine and implemented modifications. Initially, I included a 'main.yml' file within the '.github/workflow/' directory to enable the execution of fuzzing on five specific functions and generate a report when actions such as pushing occur. Following multiple testing and review cycles, I successfully obtained a report through the workflow process.

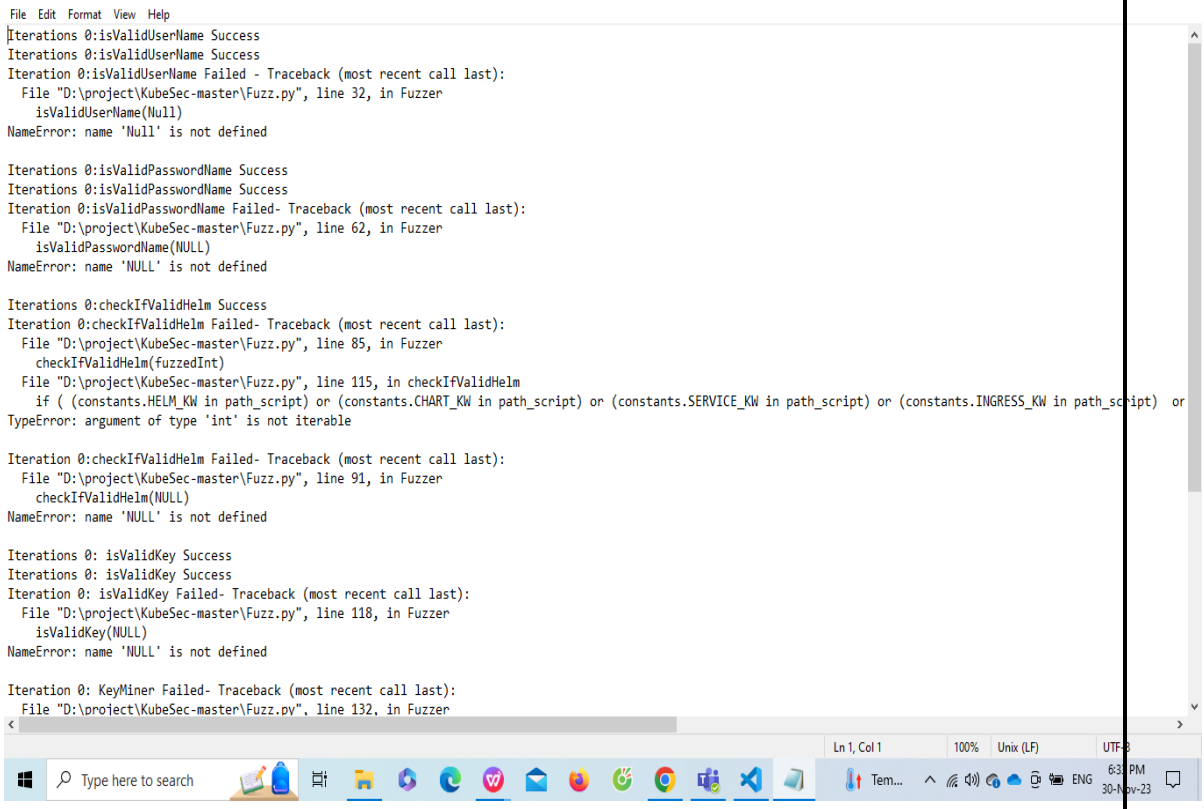



I was prepared to configure my Fuzzing function, a process that involved identifying five methods within the zip for testing purposes. These selected methods were {Class Scanner [Functions: isValidUserName, isValidPasswordName, isValidKey], Class Parser [Functions: keyMiner, checkIfValidHelm]}. The chosen inputs consisted of a randomly generated integer, a randomly generated string of fixed size, and NULL. The Fuzz.py script would then evaluate these five methods with the specified inputs. The fuzzing function was designed to produce a report highlighting both successful and unsuccessful tests, with details on the errors associated with any failures. Upon pushing to GitHub, users could navigate to the Actions tab, locate the latest commit under workflows, and access the report to reference the final functioning Fuzz push.



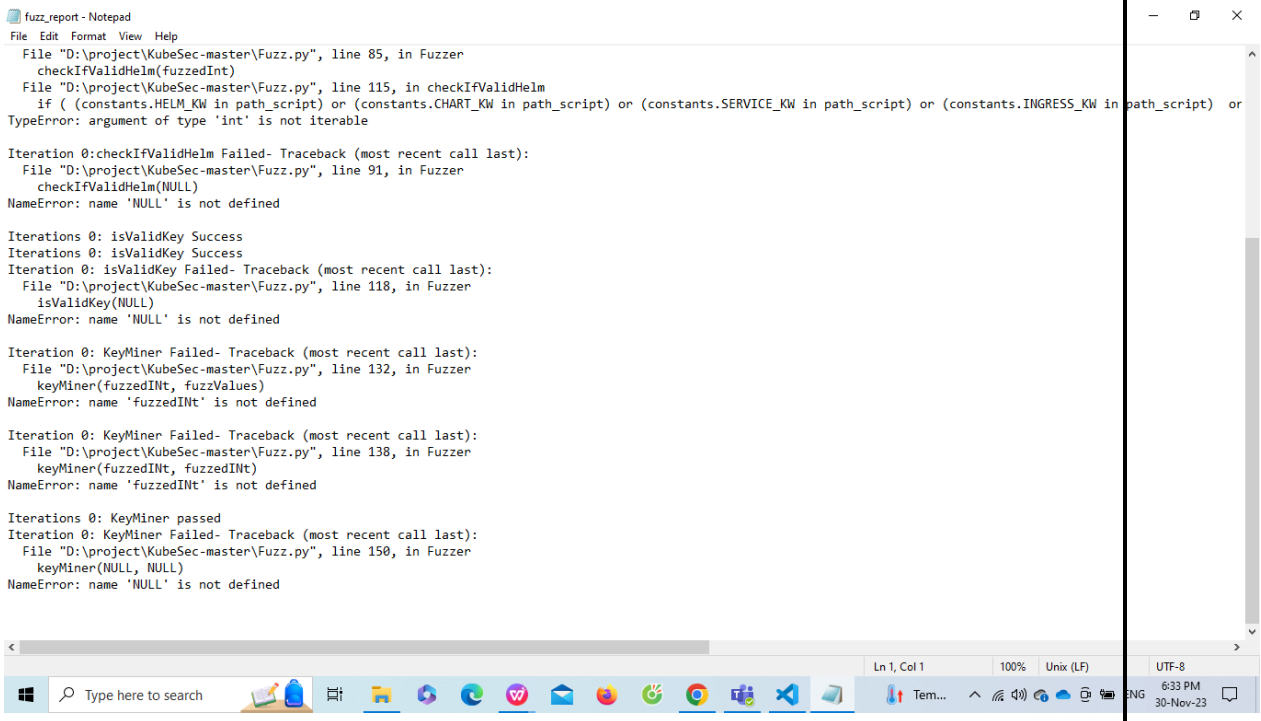

In this workflow you will be able to see the fuzz report which prints the tests after first iteration.

# Fuzz.py Output



```
File  Edit  Format  View  Help
Iterations 0:isValidUserName Success
Iterations 0:isValidUserName Success
Iteration 0:isValidUserName Failed - Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 32, in Fuzzer
    isValidUserName(Null)
NameError: name 'Null' is not defined

Iterations 0:isValidPasswordName Success
Iterations 0:isValidPasswordName Success
Iteration 0:isValidPasswordName Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 62, in Fuzzer
    isValidPasswordName(NULL)
NameError: name 'NULL' is not defined

Iterations 0:checkIfValidHelm Success
Iteration 0:checkIfValidHelm Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 85, in Fuzzer
    checkIfValidHelm(fuzzedInt)
  File "D:\project\KubeSec-master\Fuzz.py", line 115, in checkIfValidHelm
    if ( (constants.HELM_KW in path_script) or (constants.CHART_KW in path_script) or (constants.SERVICE_KW in path_script) or (constants.INGRESS_KW in path_script)  or
TypeError: argument of type 'int' is not iterable

Iteration 0:checkIfValidHelm Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 91, in Fuzzer
    checkIfValidHelm(NULL)
NameError: name 'NULL' is not defined

Iterations 0: isValidKey Success
Iterations 0: isValidKey Success
Iteration 0: isValidKey Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 118, in Fuzzer
    isValidKey(NULL)
NameError: name 'NULL' is not defined

Iteration 0: KeyMiner Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 132, in Fuzzer
```





```
fuzz_report - Notepad
File  Edit  Format  View  Help
  File "D:\project\KubeSec-master\Fuzz.py", line 85, in Fuzzer
    checkIfValidHelm(fuzzedInt)
  File "D:\project\KubeSec-master\Fuzz.py", line 115, in checkIfValidHelm
    if ( (constants.HELM_KW in path_script) or (constants.CHART_KW in path_script) or (constants.SERVICE_KW in path_script) or (constants.INGRESS_KW in path_script)  or
TypeError: argument of type 'int' is not iterable

Iteration 0:checkIfValidHelm Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 91, in Fuzzer
    checkIfValidHelm(NULL)
NameError: name 'NULL' is not defined

Iterations 0: isValidKey Success
Iterations 0: isValidKey Success
Iteration 0: isValidKey Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 118, in Fuzzer
    isValidKey(NULL)
NameError: name 'NULL' is not defined

Iteration 0: KeyMiner Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 132, in Fuzzer
    keyMiner(fuzzedINt, fuzzValues)
NameError: name 'fuzzedINt' is not defined

Iteration 0: KeyMiner Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 138, in Fuzzer
    keyMiner(fuzzedINt, fuzzedINt)
NameError: name 'fuzzedINt' is not defined

Iterations 0: KeyMiner passed
Iteration 0: KeyMiner Failed- Traceback (most recent call last):
  File "D:\project\KubeSec-master\Fuzz.py", line 150, in Fuzzer
    keyMiner(NULL, NULL)
NameError: name 'NULL' is not defined
```

**John Chong:** I was tasked to do part 4c, which deals with integrating forensics inside the project's functions. I initialized and created a logging object and used the files scanner.py and parser.py to find 5 methods to add forensics to where files were being read and data was being accessed. Throughout this project, I learned that logging is crucial because it tracks what files were accessed or updated. It then keeps a log about the history of those updates. This is important because it allows everyone to be able to see what has changed and helps determine who was responsible for these changes or what functions are called throughout the run.

Scanner.py
I made logging changes to these following functions inside scanner.py.
ScanForOverPrivelages
runScanner

parser.py
I made logging changes to these following functions inside parser.py
Show_line_for_paths
LoadMultiYAML
checkParseError