COMP 5710 - Project Report
Auburn University, Spring FY23

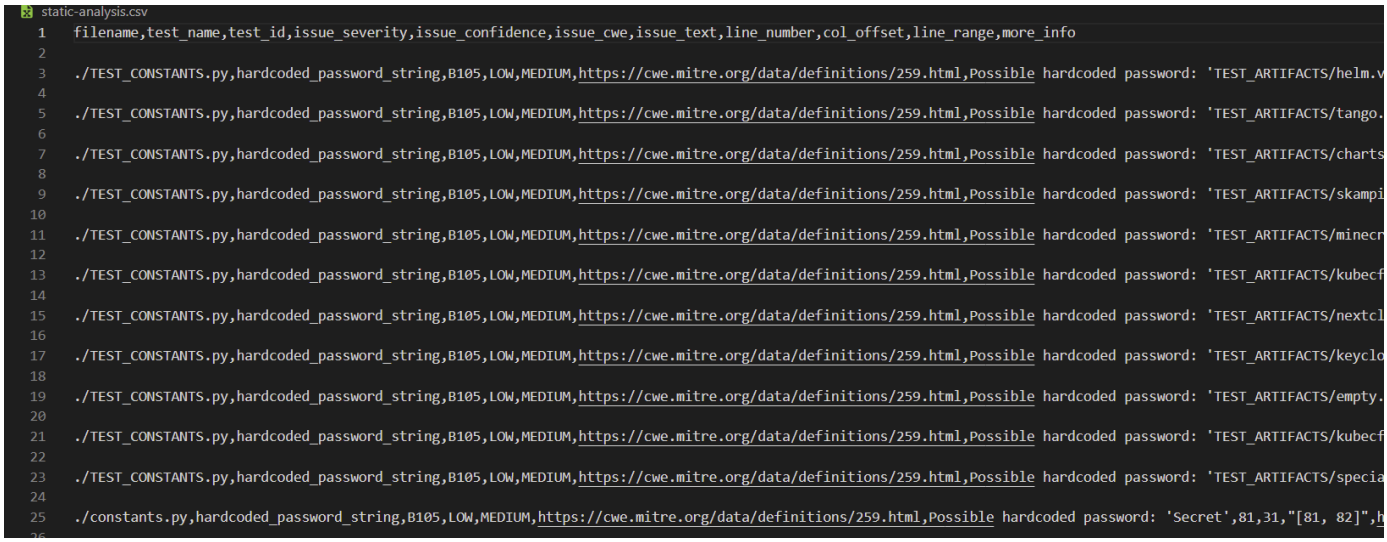
Team Name: SoftwareQualityAssumption
Team members: George Martin, Matthew Jayroe, Jared Prather.

Github URL:
https://github.com/jprather7966/SoftwareQualityAssumption-SQA2023-AUBURN

Objective: "The objective of this project is to integrate software quality assurance activities into an existing Python project" by methods of an integrated Git Hook, Fuzzing, and embedded forensics that output a log-file.

Git Hook:

We created a pre-commit file to run bandit on the repository and format the output in csv and place it in a file called static-analysis.csv. This file is placed in the github repo for the user to place in their .git/hooks folder. This is a sample of the static-analysis.csv file after running the git hook:

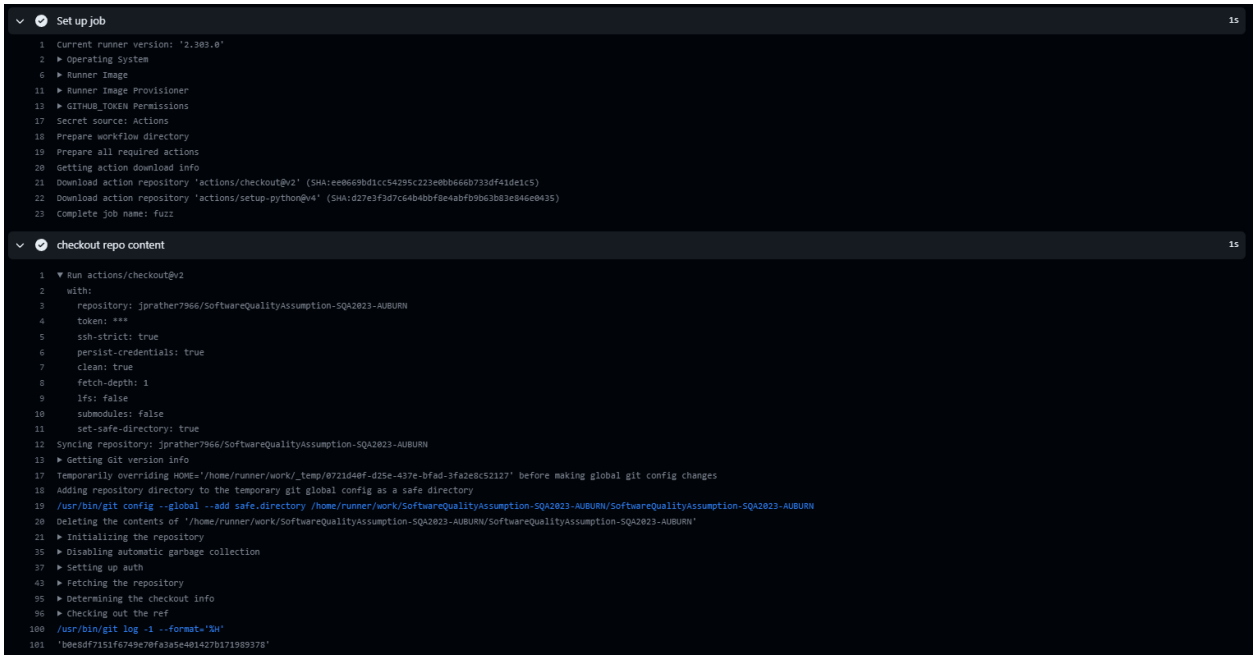

static-analysis.csv

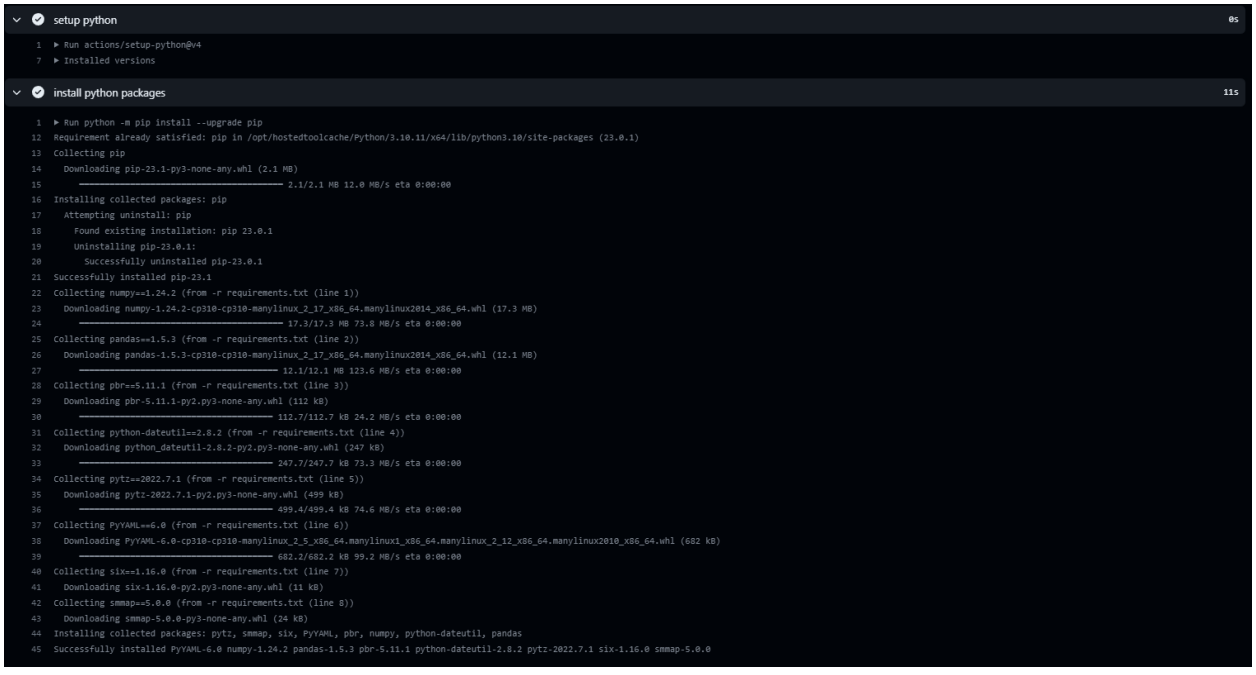
```
filename,test_name,test_id,issue_severity,issue_confidence,issue_cwe,issue_text,line_number,col_offset,line_range,more_info

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/helm.v

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/tango.

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/charts

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/skampi

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/minecr

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/kubecf

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/nextcl

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/keyclo

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/empty.

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/kubecf

./TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'TEST_ARTIFACTS/specia

./constants.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/definitions/259.html,Possible hardcoded password: 'Secret',81,31,"[81, 82]",h
```

COMP 5710 - Project Report
Auburn University, Spring FY23

Fuzzing:

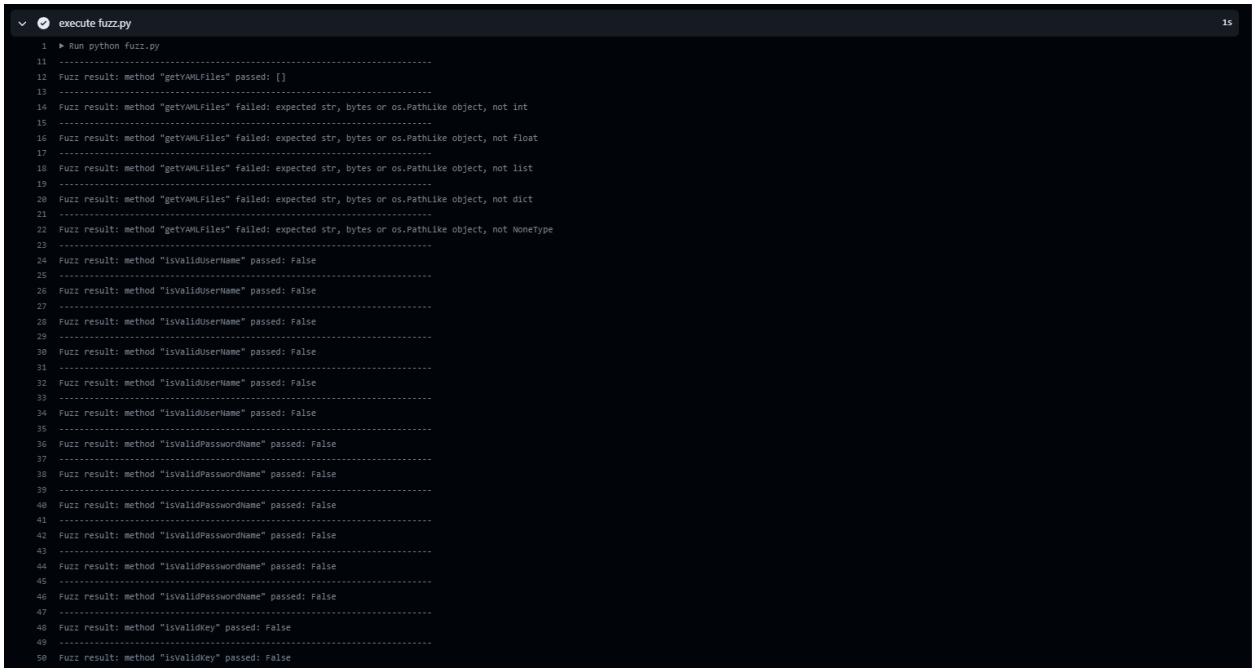
The fuzz.py file was created to generate arguments for 5 different methods found throughout KubeSec, then use said arguments to fuzz each selected method. The fuzz.py file runs automatically when new code is pushed to the repository using fuzz.yml, a simple workflow action file. The fuzz results can easily be found in the Actions section of the repository on GitHub. Screenshots are shown below.



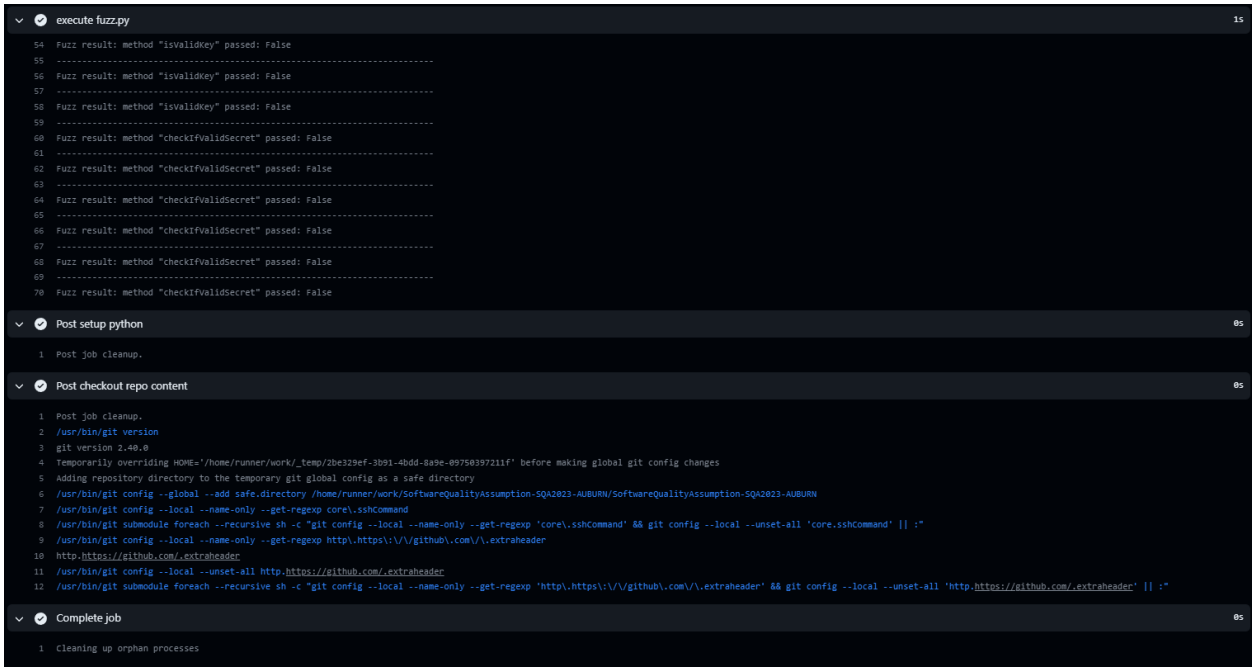


The above screenshots show the setup of the GitHub action.

COMP 5710 - Project Report
Auburn University, Spring FY23



The above screenshot shows some of the output from running fuzz.py (the full output can be seen in the Actions section of the GitHub repository).



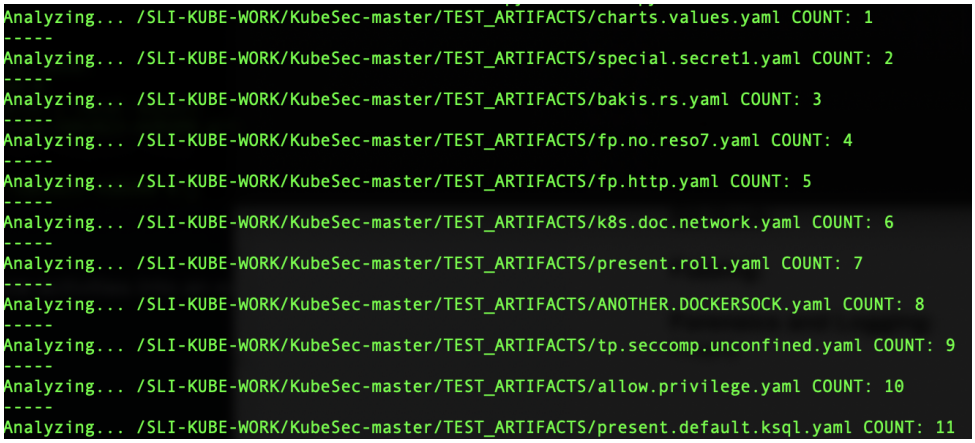
The above screenshot shows the post job cleanup of the GitHub action.

COMP 5710 - Project Report
Auburn University, Spring FY23

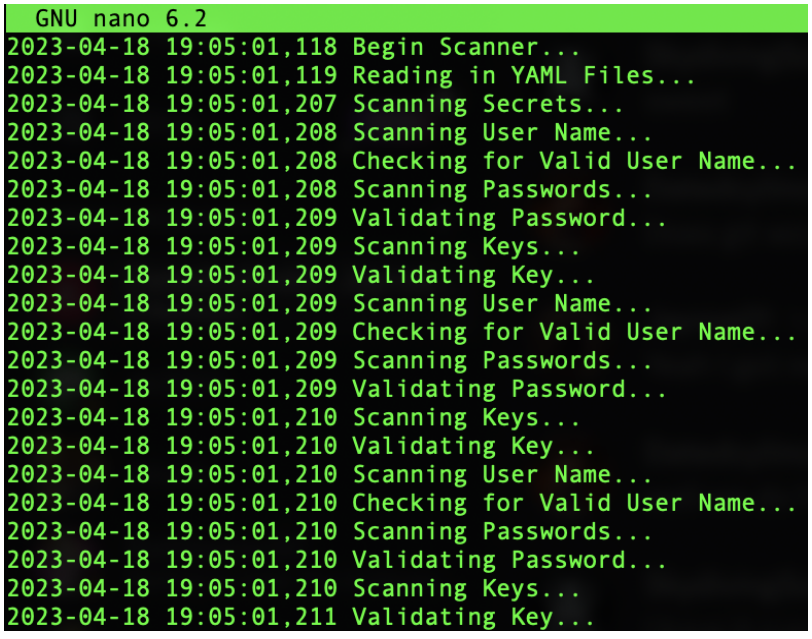
Forensics and Logging:

A simpleLogger.py file was integrated similar to that seen in a previous workshop. This file was imported into scanner.py and a loggerObject instantiated. In front of key methods which would read potentially sensitive (or malicious) data from user files, logs were taken with descriptive outputs for each action. When a new YAML file was read this would appear in the log, when a username, password, key, or secret was scanned, they too would appear in the log. Our team chose to include more than the minimum 5 logging statements, which resulted in a sizeable output file named “SIMPLE-LOGGER.log”.

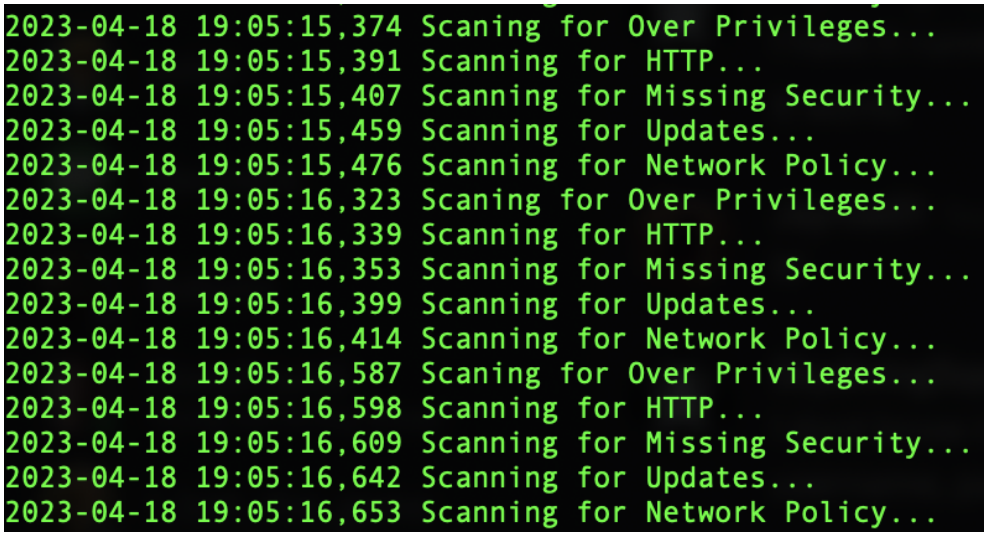
Screen shots are shown below.



Above: an image of when python3 main.py is running. The program iterates through the YAML files and updates the count. In total there were 56 read yaml files.



From the SIMPLE-LOGGER.log file, date and timestamps accompany each logged item from the yaml files. When the scanner.py class is first called it is logged as “Begin Scanner…”. From there each event or key methods are logged.



Other methods were logged such as “Scaning for Over Privileges…” (sic). The logs here indicate that security aspects were executed.

Lessons Learned:

In this project we first learned on how to use git hooks to perform static analysis on our code so we can see potential vulnerabilities before we fully commit to them. Next, we utizlied github actions to fuzz 5 methods in our project so we can be able to find any critical problems that can potentially cause if we provide random inputs. Finally we place a ton of logging statements in front of important methods so we can know when sensitive data is utilized and where certain issues can arise.