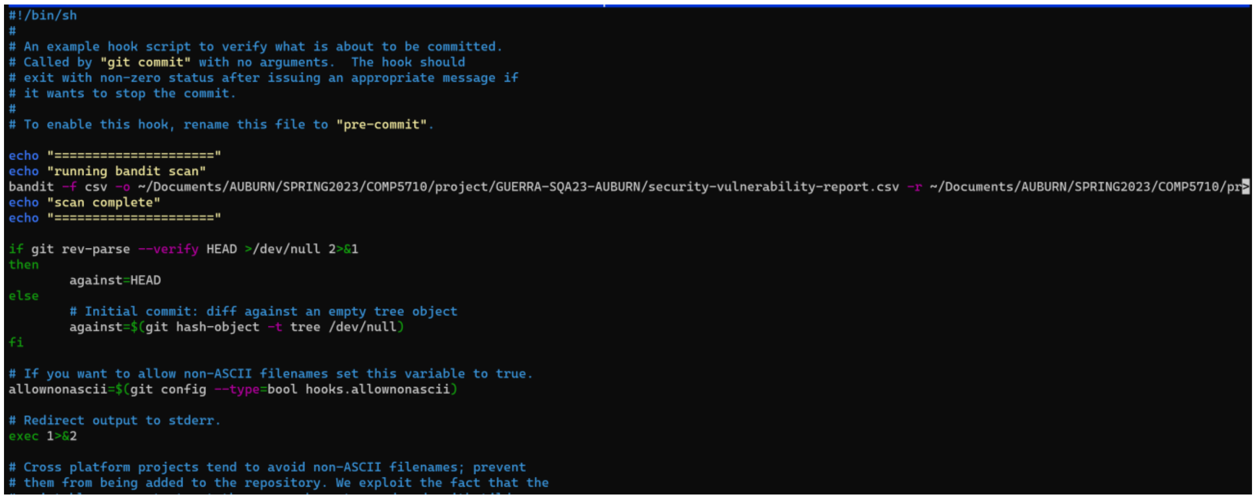
Olivia Guerra
5/1/2023
COMP 5710-001

4A – Git Hooks to Run and Report Security Weaknesses Using Bandit

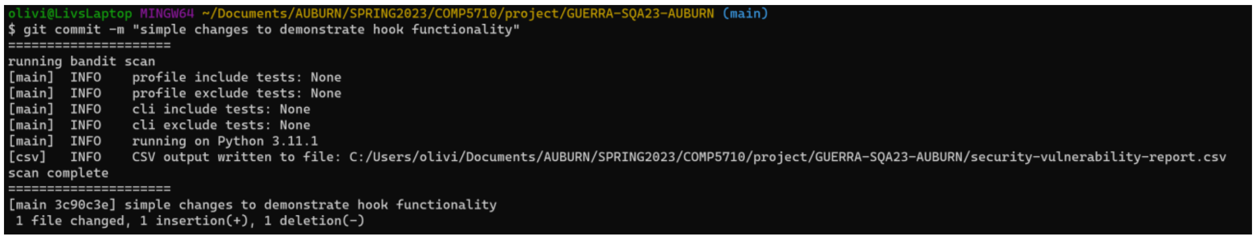
To create the pre-commit GitHook, I first navigated to the .git/hooks/ directory of my repository. I then copied the file pre-commit.sample an renamed the new file pre-commit. I then added the command to scan the files in the repository using bandit to the pre-commit file. The command added was "bandit -f csv -o ~/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/security-vulnerability-report.csv -r ~/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/". Upon committing changes to the repository, the Bandit scan is automatically run and the output is saved into the file security-vulnerability-report.csv at the specified location.

Pre-Commit GitHook



```
#!/bin/sh
#
# An example hook script to verify what is about to be committed.
# Called by "git commit" with no arguments.  The hook should
# exit with non-zero status after issuing an appropriate message if
# it wants to stop the commit.
#
# To enable this hook, rename this file to "pre-commit".

echo "===================="
echo "running bandit scan"
bandit -f csv -o ~/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/security-vulnerability-report.csv -r ~/Documents/AUBURN/SPRING2023/COMP5710/pr
echo "scan complete"
echo "===================="

if git rev-parse --verify HEAD >/dev/null 2>&1
then
        against=HEAD
else
        # Initial commit: diff against an empty tree object
        against=$(git hash-object -t tree /dev/null)
fi

# If you want to allow non-ASCII filenames set this variable to true.
allownonascii=$(git config --type=bool hooks.allownonascii)

# Redirect output to stderr.
exec 1>&2

# Cross platform projects tend to avoid non-ASCII filenames; prevent
# them from being added to the repository. We exploit the fact that the
```

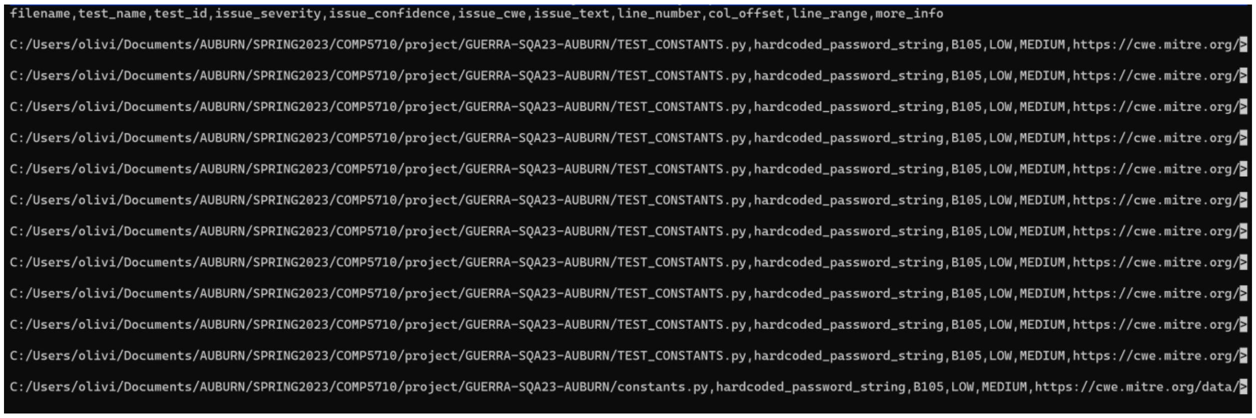
Demonstrating the Execution of bandit when python files are changed and committed



```
olivi@LivsLaptop MINGW64 ~/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN (main)
$ git commit -m "simple changes to demonstrate hook functionality"
====================
running bandit scan
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.11.1
[csv]   INFO    CSV output written to file: C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/security-vulnerability-report.csv
scan complete
====================
[main 3c90c3e] simple changes to demonstrate hook functionality
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Output CSV File

Olivia Guerra
5/1/2023
COMP 5710-001



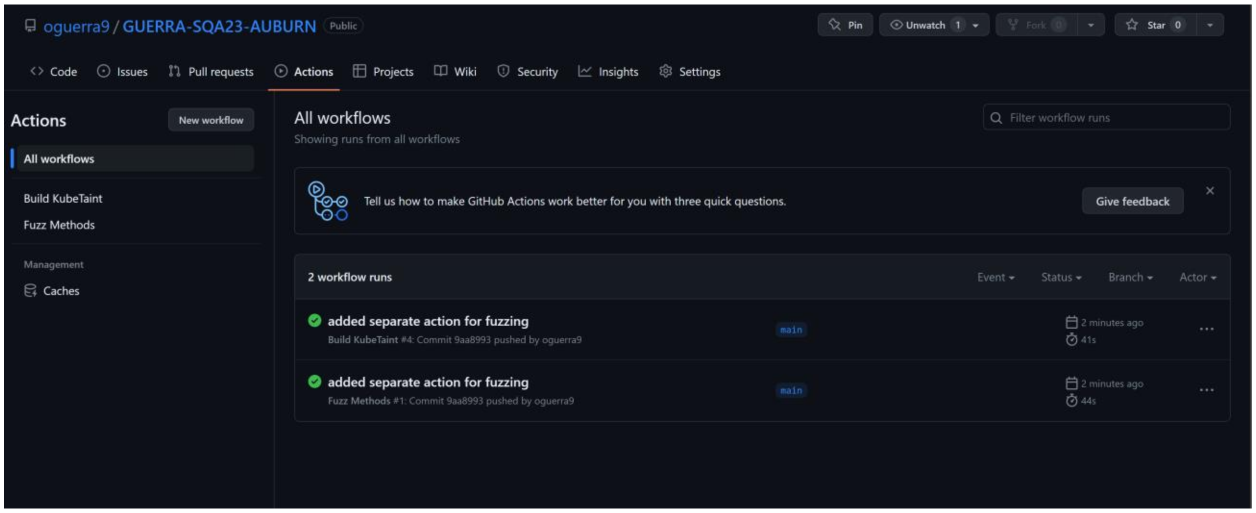
```
filename,test_name,test_id,issue_severity,issue_confidence,issue_cwe,issue_text,line_number,col_offset,line_range,more_info
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/TEST_CONSTANTS.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/
C:/Users/olivi/Documents/AUBURN/SPRING2023/COMP5710/project/GUERRA-SQA23-AUBURN/constants.py,hardcoded_password_string,B105,LOW,MEDIUM,https://cwe.mitre.org/data/
```

4B – Fuzz.py to fuzz 5 methods and run automatically in GitHub Actions

To fuzz 5 methods, I first created the file fuzz.py in the repository. I then added a list of various unexpected inputs that could produce an error or another unexpected result from the methods I chose to fuzz. I then added loops to iterate through the inputs in the list and run each of the 5 methods with the list item as one of their parameters. I then added conditional statements that would print the output if it was unexpected in any way. By doing this, any unexpected input not already accounted for would be printed.

To automatically run fuzz.py in GitHub actions, I went to the actions tab of the GitHub repository and selected “New Workflow”. I then named the worflow “Fuzz Methods.” I specified that the action should be run when the main branch is pushed or when a pull request is made on the main branch. I then specified more details for the workflow including the steps. The steps include setting up Python, installed dependencies, and then running the file fuzz.py. I then pushed the code to the repository. The next time I pushed changes to the main branch of the repository, the action ran successfully.

Successful workflow run in GitHub Actions following pushing committed changes



All workflows
Showing runs from all workflows

Tell us how to make GitHub Actions work better for you with three quick questions.

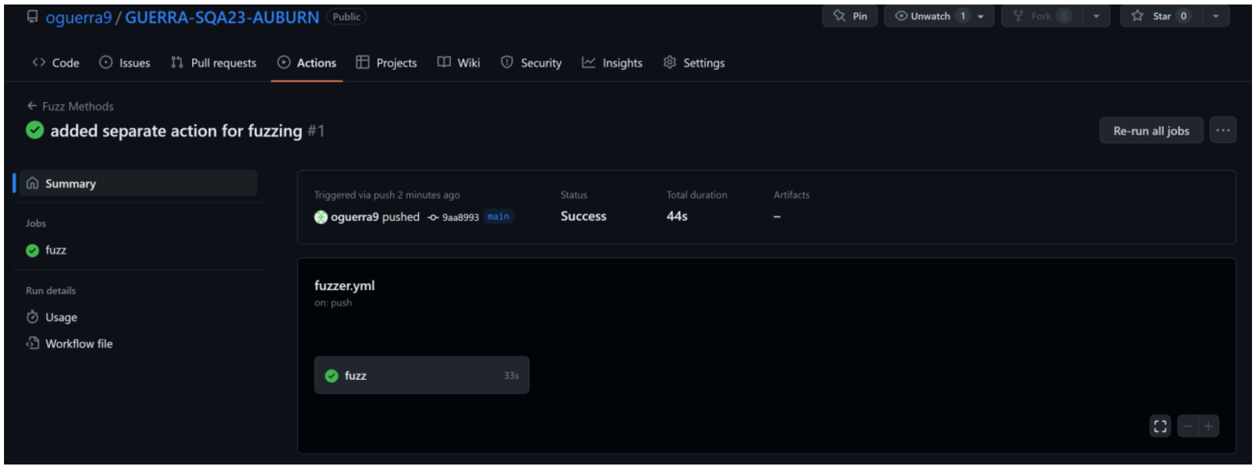
2 workflow runs

added separate action for fuzzing
Build KubeTaint #4: Commit 9aa8993 pushed by oguerra9 — main — 2 minutes ago — 41s

added separate action for fuzzing
Fuzz Methods #1: Commit 9aa8993 pushed by oguerra9 — main — 2 minutes ago — 44s

Successful job

Olivia Guerra
5/1/2023
COMP 5710-001



added separate action for fuzzing #1

Triggered via push 2 minutes ago
oguerra9 pushed 9aa8993 main
Status: Success
Total duration: 44s
Artifacts: –

fuzzer.yml
on: push
fuzz 33s

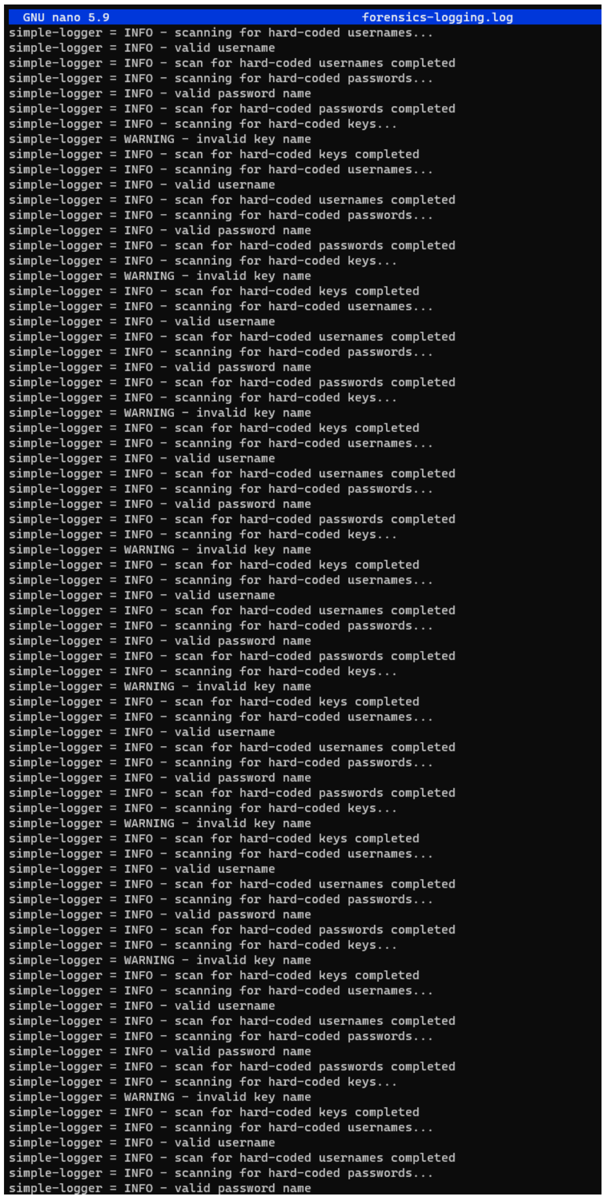
4C – Forensics Integrated with Logging

To integrate forensics, I chose to add logging to five methods. I first added a file with a function to create a logging object. In the files where I used logging, I imported the function from this file instead of having to create a new logging object from scratch each time. The following is a list of functions I modified to integrate forensics and a list of the events logged.

1. scanner.py/isValidUsername()
   - warning log when username is invalid
   - warning log when username is forbidden
   - info log when username is valid
2. scanner.py/isValidPasswordName()
   - warning log when password name is invalid
   - warning log when password name is forbidden
   - info log when password name is valid
3. main.py/main
   - info log when dataframe is created
4. parser.py/checkWeirdYAML()
   - info log when invalid YAML is found
5. scanner.py/isValidKey()
   - info log when key name is validated
   - warning log when key name is invalid
   - warning log when key is not a string
6. scanner.py/scanUserName()
   - info log when scan begins
   - info log when secret username is found hard-coded
   - info log when scan for secret usernames is completed
7. scanner.py/scanPasswords()
   - info log when scan begins

   - info log when secret password is found hard-coded
   - info log when scan for secret passwords is completed
8. scanner.py/scanKeys()
   - info log when scan begins
   - info log when secret key is found hard-coded
   - info log when scan for hard-coded keys is completed

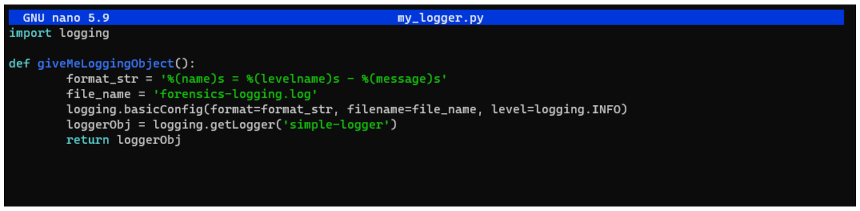
log output [partial screenshot; full log file is in repository]:



```
  GNU nano 5.9                        forensics-logging.log
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
simple-logger = INFO - scan for hard-coded passwords completed
simple-logger = INFO - scanning for hard-coded keys...
simple-logger = WARNING - invalid key name
simple-logger = INFO - scan for hard-coded keys completed
simple-logger = INFO - scanning for hard-coded usernames...
simple-logger = INFO - valid username
simple-logger = INFO - scan for hard-coded usernames completed
simple-logger = INFO - scanning for hard-coded passwords...
simple-logger = INFO - valid password name
```

my_logger.py files used to make logger for use throughout repository

Olivia Guerra
5/1/2023
COMP 5710-001



```
  GNU nano 5.9                                          my_logger.py
import logging

def giveMeLoggingObject():
        format_str = '%(name)s = %(levelname)s - %(message)s'
        file_name = 'forensics-logging.log'
        logging.basicConfig(format=format_str, filename=file_name, level=logging.INFO)
        loggerObj = logging.getLogger('simple-logger')
        return loggerObj
```