

## BIMM-SQA2023-Project Report

Brayden Simpson, Ian Larson, Matthew Ippolito, Melvin Moreno

### Activities:

#### Git Hook

We created a Git Hook that runs and reports all security weaknesses in the project into a CSV file whenever a Python file is changed and committed. We did this by creating a **.pre-commit-config.yaml** file and configuring our security vulnerability tool **Bandit** to work on commit and output to the *output.csv* file.

Detailed Instructions:

#### 1. Modify any file

#### 2. Git add the files

#### 3. Git commit

- Bandit runs recursively on all files. If you want to work only on the file you committed, change the **.pre-commit-config.yaml** file and remove the **--recursive** flag

- Because bandit failed the git hook, you cannot commit or push changes. To get past this, add the --no-verify flag at the end of your commit.

#### 4. Get results

Filename	test_name	test_id	issue_severity	issue_confidence	issue_cwe	issue_text	line_number	col_offset	end_col_offset	line_range	more_info
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	8	22	55 [8]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	9	22	56 [9]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	10	22	57 [10]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	11	22	57 [11]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	12	22	60 [12]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	13	22	57 [13]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	14	22	60 [14]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	15	22	59 [15]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	16	22	48 [16]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	17	22	57 [17]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\TEST_CONSTANTS.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	108	22	59 [108]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>
.\Muheles-master\constants.py	hardcoded_password_string	#305	LOW	MEDIUM	https://cwe	Possible hardc	81	11	39 [81]		<a href="https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html">https://bandit.readthedocs.io/en/0.0.0/plugins/b005_hardcoded_password_string.html</a>

### Fuzzing

We created a 'fuzz.py' file that will automatically fuzz 5 Python methods of our choice. 'fuzz.py' will be automatically executed from GitHub actions. We reported the bugs we discovered.

Detailed Activities:

1. We selected the 5 methods to perform fuzzing on:
  - Graphtaint.py: getYAMLFFiles
  - Graphtaint.py: getSHFiles
  - Scanner.py: isValidUsername
  - Scanner.py: isValidPassword
  - Scanner.py: isValidKey
2. Created the 'fuzz.py' file to send random values to these methods to see how they would react.

```
import traceback
from graptaint import getYAMLFFiles, getSHFiles
from scanner import isValidUserName, isValidPasswordName, isValidKey

def fuzz():
    func_names = [
        getYAMLFFiles,
        getSHFiles,
        isValidUserName,
        isValidPasswordName,
        isValidKey
    ]

    func_args = [
        ["---saa", 422422, False, None],
        ["@@@#@#$%", "[][]", True, None],
        ["/user/Documents", "admin_domain", 21421421512421, None],
        ["password", False, None, "_hash"],
        [None, True, "mykey", "adminkey", 2141424]
    ]

    index = 0

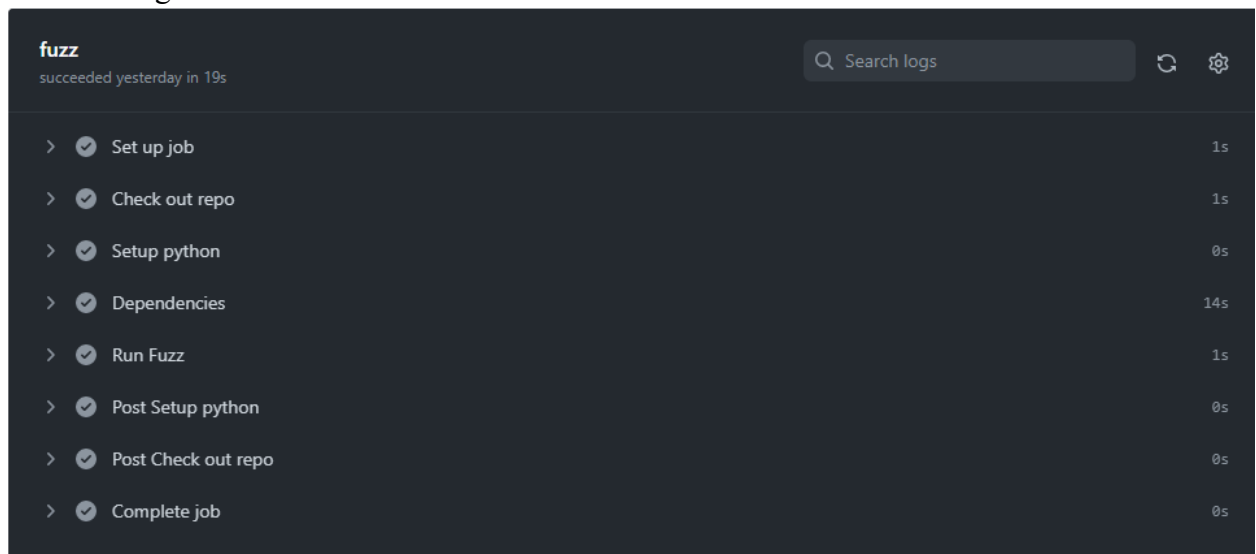
    for func in func_names:
        args = func_args[index]
        for arg in args:
            try:
                result = func(arg)
                if(result != None):
                    print("Result returned is " + str(result) + " for arguments " + str(arg))
            except Exception as e:
                print(str(func.__name__) + " has an issues with arguments " + str(arg))
```

3. Setup the workflow to run 'fuzz.py' automatically using 'fuzz.yaml'

```
1  name: On Push, Fuzz
2  on: push
3  jobs:
4    fuzz:
5      name: fuzz
6      runs-on: ubuntu-latest
7      steps:
8        - name: Check out repo
9          uses: actions/checkout@v2
10       - name: Setup python
11         uses: actions/setup-python@v2
12         with:
13           python-version: 3.8
14       - name: Dependencies
15         run: |
16           python -m pip install --upgrade pip
17           pip install pandas
18           pip install numpy
19           pip install pyyaml
20       - name: Run Fuzz
21         run: python KubeSec-master/fuzz.py
22       shell: sh
```

- The workflow will run every time a push is made.

4. Tested doing our own commits.



## Forensics

We integrated forensics by modifying 5 Python methods of our choice.

Detailed Activities:

1. Selected 5 methods for logging based on what methods we would be fuzzing.
2. Added 'logger.py' function and modified it to take relevant information from fuzzing.
3. Inserted logging statements into each selected method.
4. Logged at the information level to catch all events up to and including errors from fuzzing.

## Lessons Learned

By implementing various tools we learned throughout the course and workshops, we were able to learn significantly more about software engineering and software quality assurance tools.

### *Git Tools*

- Setting up a GitHub repository and using it within a team environment
- Commands like git push, git pull, git status, etc.
- Git Hooks
- Git flags
- Git actions & debugging with Git actions

### *Fuzzing and Forensics*

- How to select methods for fuzzing and logging.
- Different loggings levels and what each should catch.
- Structure of a logging file and how to incorporate it into a larger project.
- Being able to look at a project that already has significant work on it and knowing/understanding what each file does and how it works with others.
- How workflows work

### *Other Tools*

- YAML file configuration was a large part of our project. We learned how to add flags and arguments to get our desired outputs.
- We read documentation on Bandit to properly understand how to get our desired output and how to run it within our project.
- Formatting within files like YAML were critical and ensuring our project ran properly
- Working with a team in a GitHub repository and how to commit, push, and pull changes