AUBURN

UNIVERSITY

SAMUEL GINN

COLLEGE OF ENGINEERING

Software Quality Assurance Final Project
COMP 5710

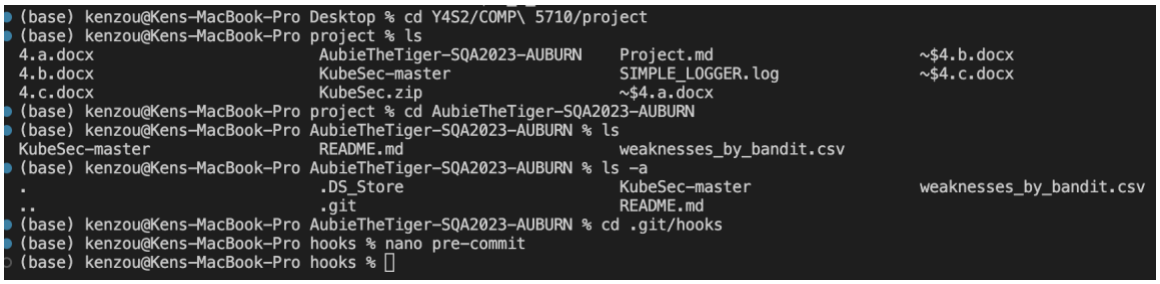
Team: AubieTheTiger

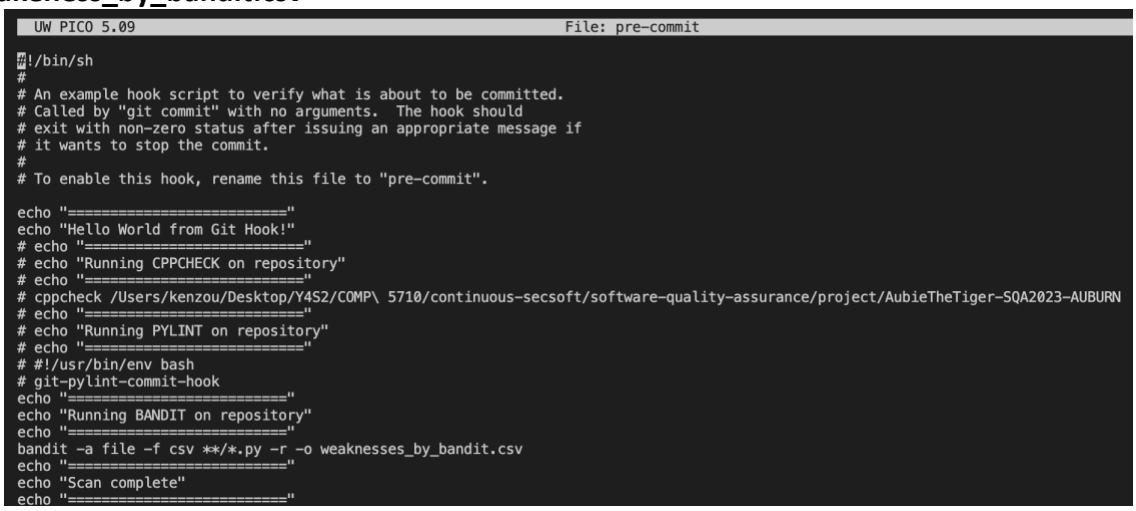
By:
Isabelle Cochran
Rebekah Harrison
Courtney Lee
Ken Zou

Department of Computer Science and Software Engineering
Samuel Ginn College of Engineering, Auburn University

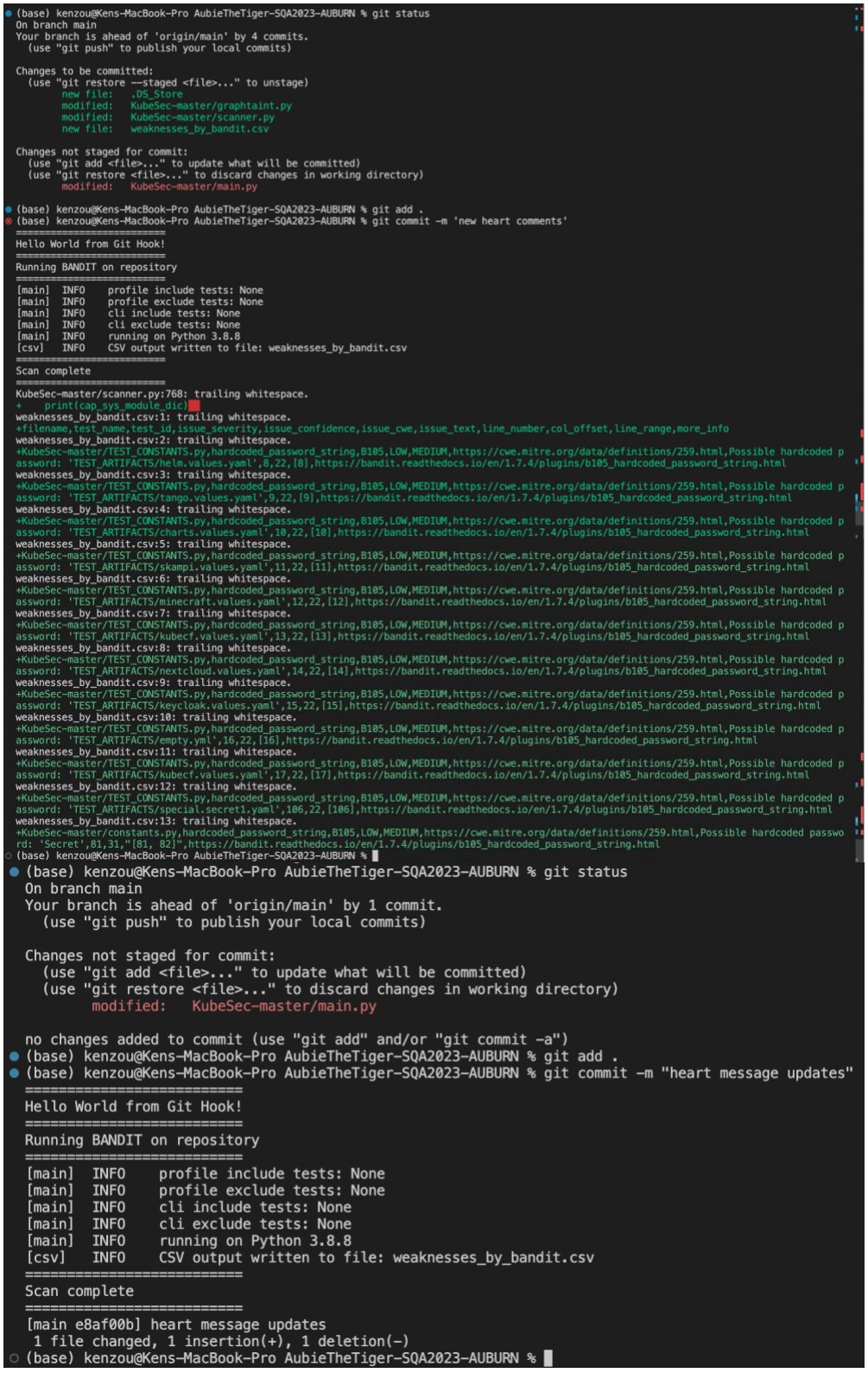
Auburn, Alabama
May 1, 2023

1. Unpack the project `KubeSec.zip`. (1%)
    - Our team was able to successfully download and unpack the KubeSec.zip file from the professor's GitHub repository: https://github.com/paser-group/continuous-secsoft/tree/master/software-quality-assurance/project
2. Upload project as a GitHub repo on `github.com`. Format of the repo name is `TEAMNAME-SQA2023-AUBURN` (2%)
    - After downloading the KubeSec repository, we were able to create the GitHub repository AubieTheTiger-SQA2023-AUBURN: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN
3. In your project repo create `README.md` listing your team name and team members. (2%)
    - In our AubieTheTiger repository, we were able to create a 'README' file that includes our team names, team members, and details about the project: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN/blob/main/README.md
4. Apply the following activities related to software quality assurance:

- 4.a. 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. (30%)
    1. Create a new pre-commit file that has permissions of 777 so that all groups and all users get access
        - 

    2. Add bandit command to the pre-commit file. "**bandit -a file -f csv **/*.py -r -o weakeness_by_bandit.csv**"
        - 

    3. The following is the terminal output when we commit a change with git



4. The following is a brief overview of the csv file created. The actual csv file can be viewed in the GitHub repository: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN

filename	test_name	test_id	issue_severit	issue_confid	issue_cwe	issue_text	line_number	col_offset	line_range	more_info
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible hard	8	22	[8]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	9	22	[9]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	10	22	[10]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	11	22	[11]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	12	22	[12]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	13	22	[13]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	14	22	[14]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	15	22	[15]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	16	22	[16]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	17	22	[17]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	106	22	[106]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-ma	hardcoded_p	B105	LOW	MEDIUM	https://cwe.	Possible harc	81	31	[81, 82]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html

- All the issues were just possible hardcoded passwords with low issue severity and medium issue confidence.

LESSONS LEARNED: The Git Hook is triggered when you commit something using git. We have effectively used a Git Hook to trigger a status analysis through Bandit which analyzed all the python files for any suspected vulnerabilities. Through performing this task, we showed how we can use two tools, Git Hooks and the static analysis tool, Bandit, to automatically run scans on a repository. We are doing software quality assurance without having to take the time to do it manually.

- 4.b. 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. (30%)

1. A fuzz.py file was created that automatically fuzzes five Python methods in the sanner.py file.

KubeSec-master > fuzz.py > ...

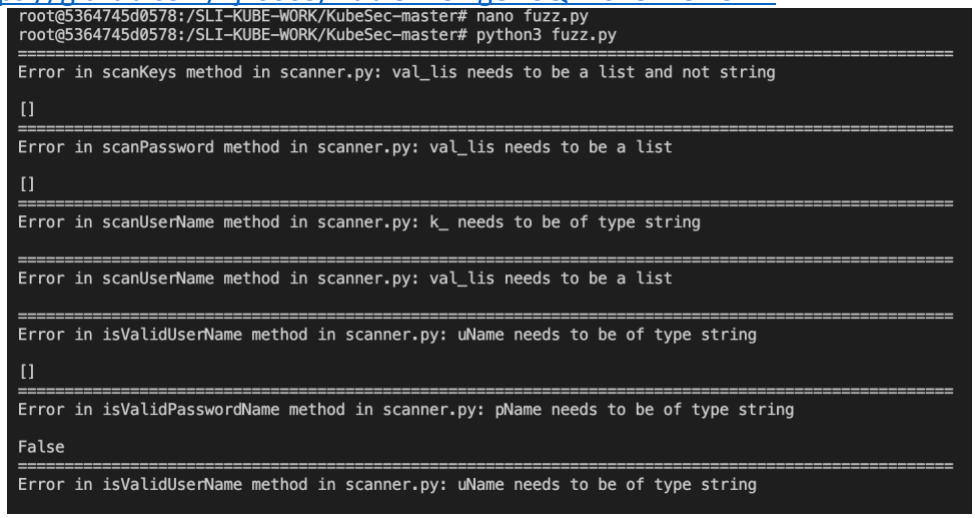
```python
import scanner

def fuzzValues():
    # testing scanner methods
    print (scanner.scanKeys("hello", "hello2"))
    print (scanner.scanPasswords("pass123", "hello2"))
    print (scanner.scanUserName(None, "hello2"))
    print (scanner.isValidPasswordName(None))
    print (scanner.isValidUserName(12345))

def simpleFuzzer():
    fuzzValues()


if __name__=='__main__':
    simpleFuzzer()
```

2. In the scanner.py file, we modified 5 methods so that it would catch any invalid inputs that might be passed through.

```python
def scanUserName(k_ , val_lis ):
    hard_coded_unames = []
    if not isinstance(k_, str):
        print('='*100)
        print ("Error in scanUserName method in scanner.py: k_ needs to be of type string\n")
    if not isinstance(val_lis, list):
        print('='*100)
        print ("Error in scanUserName method in scanner.py: val_lis needs to be a list\n")
```

3. Docker is then run so that we can run **fuzz.py**

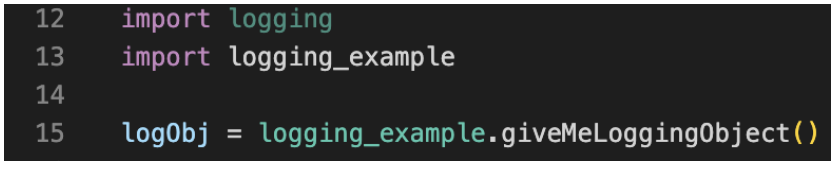
```
- docker rm $(docker ps -a -f status=exited -f status=created -q)
- docker rmi -f $(docker images -a -q)
- docker pull akondrahman/sli-kube
- docker images -a
- docker run --rm -it akondrahman/sli-kube bash
- cd SLI-KUBE-WORK/KubeSec-master/
- python3 fuzz.py
```

4. Before running "**python3 fuzz.py**", it is important to make sure that all new changes have been made to the docker version of the project too.
5. After running the fuzz.py file, the terminal gives several errors where all the invalid inputs were caught. To look more into the fuzz.py file and the modified scanner.py methods, the two files can be viewed in the repository here: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN
   - 

LESSONS LEARNED: In this part, we were able to use the source code to do a form of white box testing where we were able to generate test cases to check for invalid inputs. We can find bugs in methods where an invalid input might have otherwise been passed through causing a crash. We want to find these bugs early in a development cycle so that the bugs do not become amplified later down the road. In this implementation, we were able to develop a fuzzer file that fuzzes the scanner.py file to catch bugs early on.

- 4.c. Integrate forensics by modifying 5 Python methods of your choice. (30%)

1. A logging_example.py file was created to define a function that gives a logging object.
   - 

2. In scanner, we defined what a logObj was and then imported logging_example so we can use the method inside.
   - 

3. Throughout the scanner.py file, we added several logging statements in many of the methods in the scanner.py file.

```
if(checkVal):
    dict_as_list = parser.loadMultiYAML( script_path )

# ------------------------------------------------------------------------------
    # Log check for load
    logObj.info('Log Check for loadMulti over Priviledges')
# ------------------------------------------------------------------------------
```

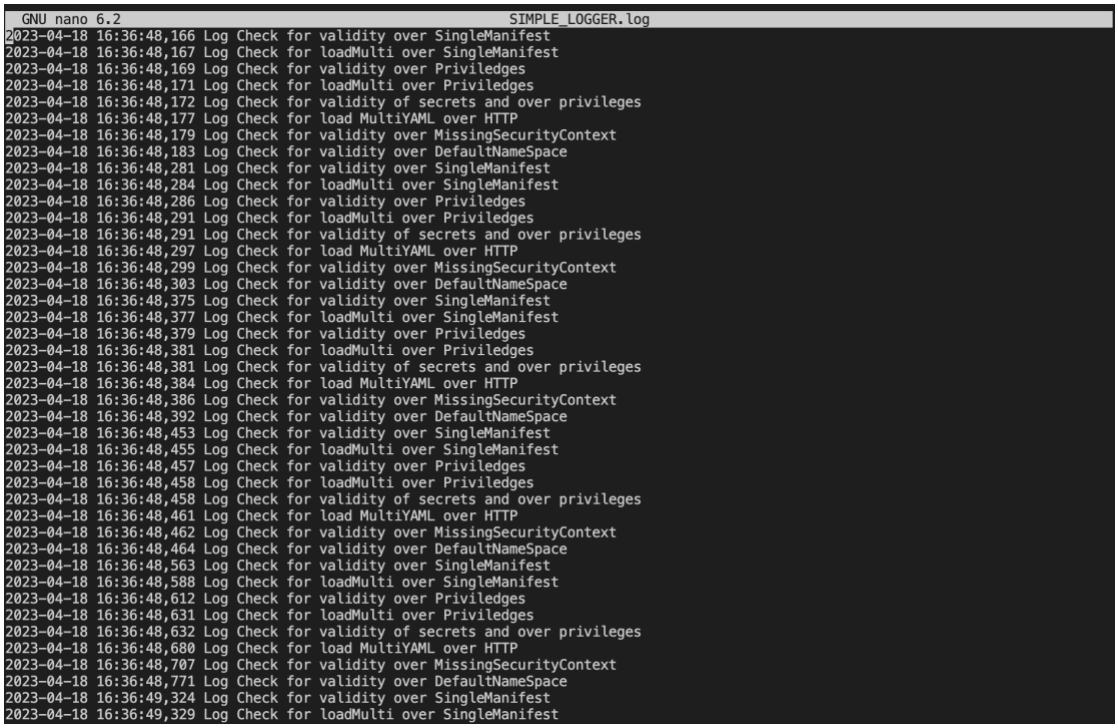
4. Docker is then run so that we can run **main.py** to get the logging file

```
- docker rm $(docker ps -a -f status=exited -f status=created -q)
- docker rmi -f $(docker images -a -q)
- docker pull akondrahman/sli-kube
- docker images -a
- docker run --rm -it akondrahman/sli-kube bash
- cd SLI-KUBE-WORK/KubeSec-master/
- python3 main.py
```

5. Before running "**python3 main.py**", it is important to make sure that all new changes have been made to the docker version of the project too.
6. While running the main.py file, the terminal gives outputs analysis of the TEST_ARTIFACTS. Part of the analysis can be viewed below:

```
root@5364745d0578:/SLI-KUBE-WORK/KubeSec-master# python3 main.py
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/sample-nfs-server.yaml COUNT: 1
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.nspace2.yaml COUNT: 2
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.nspace3.yaml COUNT: 3
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/allow.privilege.yaml COUNT: 4
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/dataimage.airflowimage.manifests.deployment.yaml COUNT: 5
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.no.reso1.yaml COUNT: 6
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/tp.seccomp.unconfined.yaml COUNT: 7
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.no.reso5.yaml COUNT: 8
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/helm.values.yaml COUNT: 9
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/present.varnish.yaml COUNT: 10
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/absent.default1.yaml COUNT: 11
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.glance.pv.yaml COUNT: 12
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.no.reso10.yaml COUNT: 13
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/nginx.deployment.result.yaml COUNT: 14
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/tp.nsp.dflt.yaml COUNT: 15
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.http.yaml COUNT: 16
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/absent.prome.yaml COUNT: 17
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.nspace1.yaml COUNT: 18
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/nginx.present.yaml COUNT: 19
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.no.reso8.yaml COUNT: 20
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/present.deployment.default.yaml COUNT: 21
------
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/tp.host.net2.yaml COUNT: 22
```

7. After the analysis is done, a **SIMPLE_LOGGER.log** file is created with all the logging statements included. Part of the file can be viewed below:



8. To look more into the full **SIMPLE_LOGGER.log** file and the altered scanner.py methods, the two files can be viewed in the repository here: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN

LESSONS LEARNED: We were able to append logging statements in several methods inside the scanner.py file. Everything is handled by the logger, and nothing is handled by the console in terms of logging information. We put logging statements around areas that might be susceptible to forms of security issues. This way, we can determine if there are places where something is not behaving as it should be and even find the timestamp of when it happened which can become very useful in tracking down potential issues.

5. Report your activities and lessons learned (5%)
   - All activities have been reported in this report and the GitHub repository provides all modified and newly created files. The GitHub repository can be accessed here: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN

Deliverables

Task	Information	Status
A repo hosted on GitHub. Name of the repo will be TEAMNAME-SQA2022-AUBURN	Repository can be found here: https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN with the name of the repo being AubieTheTiger-SQA2023-AUBURN.	✔
Full completion of all activities as recorded on the GitHub repository	See all modified files along with newly added files in the AubieTheTiger-SQA2023-AUBURN repository.	✔
Report describing what activities your performed and what you have learned	See report above for all activities performed and lessons learned for each major activity.	✔
Logs/screenshots that show execution of forensics, fuzzing, and static analysis	See report above for logs and screenshots of execution of static analysis, fuzzing, and forensics.	✔