



# 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

```
(base) kenzo@Kens-MacBook-Pro Desktop % cd Y4S2/COMP\ 5710/project
(base) kenzo@Kens-MacBook-Pro project % ls
4.a.docx      AubieTheTiger-SQA2023-AUBURN  Project.md      ~$4.b.docx
4.b.docx      KubeSec-master                SIMPLE_LOGGER.log ~$4.c.docx
4.c.docx      KubeSec.zip                   ~$4.a.docx
(base) kenzo@Kens-MacBook-Pro project % cd AubieTheTiger-SQA2023-AUBURN
(base) kenzo@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % ls
KubeSec-master  README.md  weaknesses_by_bandit.csv
(base) kenzo@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % ls -a
.               .DS_Store      KubeSec-master  weaknesses_by_bandit.csv
..              .git           README.md
(base) kenzo@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % cd .git/hooks
(base) kenzo@Kens-MacBook-Pro hooks % nano pre-commit
(base) kenzo@Kens-MacBook-Pro hooks %
```

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

```
UW PICO 5.09 File: pre-commit
#!/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 "Hello World from Git Hook!"
# echo "====="
# echo "Running CPPCHECK on repository"
# echo "====="
# cppcheck /Users/kenzo/Desktop/Y4S2/COMP\ 5710/continuous-secsoft/software-quality-assurance/project/AubieTheTiger-SQA2023-AUBURN
# echo "====="
# echo "Running PYLINT on repository"
# echo "====="
#!/usr/bin/env bash
# git-pylint-commit-hook
echo "====="
echo "Running BANDIT on repository"
echo "====="
bandit -a file -f csv **/*.py -r -o weaknesses_by_bandit.csv
echo "====="
echo "Scan complete"
echo "====="
```

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

```

(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .DS_Store
    modified:   KubeSec-master/graptaint.py
    modified:   KubeSec-master/scanner.py
    new file:   weaknesses_by_bandit.csv

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   KubeSec-master/main.py

(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git add .
(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git commit -m 'new heart comments'

Hello World from Git Hook!

Running BANDIT on repository

[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 3.8.8
[csv] INFO     CSV output written to file: weaknesses_by_bandit.csv

Scan complete

KubeSec-master/scanner.py:768: trailing whitespace.
+ print(cmp.sys.module_id)
weaknesses_by_bandit.csv:1: trailing whitespace.
+ filename, test_name, test_id, issue_severity, issue_confidence, issue_cwe, issue_text, line_number, col_offset, line_range, more_info
weaknesses_by_bandit.csv:2: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/hela.values.yaml', 8, 22, [8], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:3: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/tango.values.yaml', 9, 22, [9], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:4: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/charts.values.yaml', 10, 22, [10], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:5: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/skampi.values.yaml', 11, 22, [11], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:6: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/minecraft.values.yaml', 12, 22, [12], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:7: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/kubecf.values.yaml', 13, 22, [13], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:8: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/nextcloud.values.yaml', 14, 22, [14], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:9: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/keycloak.values.yaml', 15, 22, [15], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:10: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/empty.yaml', 16, 22, [16], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:11: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/kubecf.values.yaml', 17, 22, [17], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:12: trailing whitespace.
+KubeSec-master/TEST_CONSTANTS.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded p
assword: 'TEST_ARTIFACTS/special.secret1.yaml', 106, 22, [106], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
weaknesses_by_bandit.csv:13: trailing whitespace.
+KubeSec-master/constants.py, hardcoded_password_string, B105, LOW, MEDIUM, https://cwe.mitre.org/data/definitions/259.html, Possible hardcoded passwo
rd: 'Secret', 81, 31, ["[81, 82]"], https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html

(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN %
(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   KubeSec-master/main.py

no changes added to commit (use "git add" and/or "git commit -a")
(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git add .
(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN % git commit -m "heart message updates"

Hello World from Git Hook!

Running BANDIT on repository

[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 3.8.8
[csv] INFO     CSV output written to file: weaknesses_by_bandit.csv

Scan complete

[main e8af00b] heart message updates
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) kenzou@Kens-MacBook-Pro AubieTheTiger-SQA2023-AUBURN %

```

- 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_severity	issue_confid	issue_cwe	issue_text	line_number	col_offset	line_range	more_info
KubeSec-master	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-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	9	22	[9]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	10	22	[10]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	11	22	[11]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	12	22	[12]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	13	22	[13]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	14	22	[14]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	15	22	[15]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	16	22	[16]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	17	22	[17]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	106	22	[106]	https://bandit.readthedocs.io/en/1.7.4/plugins/b105_hardcoded_password_string.html
KubeSec-master	hardcoded_p	B105	LOW	MEDIUM	https://cwe	Possible hard	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 scanner.py file.

```
KubeSec-master > fuzz.py > ...
1  import scanner
2
3  def fuzzValues():
4      # testing scanner methods
5      print (scanner.scanKeys("hello", "hello2"))
6      print (scanner.scanPasswords("pass123", "hello2"))
7      print (scanner.scanUserName(None, "hello2"))
8      print (scanner.isValidPasswordName(None))
9      print (scanner.isValidUserName(12345))
10
11 def simpleFuzzer():
12     fuzzValues()
13
14
15 if __name__ == '__main__':
16     simpleFuzzer()
```

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

```
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>

```
root@5364745d0578:/SLI-KUBE-WORK/KubeSec-master# nano fuzz.py
root@5364745d0578:/SLI-KUBE-WORK/KubeSec-master# python3 fuzz.py
=====
Error in scanKeys method in scanner.py: val_lis needs to be a list and not string
[]
=====
Error in scanPassword method in scanner.py: val_lis needs to be a list
[]
=====
Error in scanUserName method in scanner.py: k_ needs to be of type string
=====
Error in scanUserName method in scanner.py: val_lis needs to be a list
=====
Error in isValidUserName method in scanner.py: uName needs to be of type string
[]
=====
Error in isValidPasswordName method in scanner.py: pName needs to be of type string
False
=====
Error in isValidUserName method in scanner.py: uName needs to be of type string
```

**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.

```
KubeSec-master > logging_example.py > ...
1  import logging
2
3
4  def giveMeLoggingObject():
5      format_str = '%(asctime)s %(message)s'
6      file_name = 'SIMPLE_LOGGER.log'
7      logging.basicConfig(format=format_str, filename=file_name, level=logging.INFO)
8      loggerObj = logging.getLogger('simple-logger')
9      return loggerObj
```

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

```
12  import logging
13  import logging_example
14
15  logObj = logging_example.giveMeLoggingObject()
```



- 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
```

- 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.
- 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.namespace2.yaml COUNT: 2
Analyzing... /SLI-KUBE-WORK/KubeSec-master/TEST_ARTIFACTS/fp.namespace3.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.namespace1.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:

```

GNU nano 6.2 SIMPLE_LOGGER.log
2023-04-18 16:36:48,166 Log Check for validity over SingleManifest
2023-04-18 16:36:48,167 Log Check for loadMulti over SingleManifest
2023-04-18 16:36:48,169 Log Check for validity over Privileges
2023-04-18 16:36:48,171 Log Check for loadMulti over Privileges
2023-04-18 16:36:48,172 Log Check for validity of secrets and over privileges
2023-04-18 16:36:48,177 Log Check for load MultiYAML over HTTP
2023-04-18 16:36:48,179 Log Check for validity over MissingSecurityContext
2023-04-18 16:36:48,183 Log Check for validity over DefaultNameSpace
2023-04-18 16:36:48,281 Log Check for validity over SingleManifest
2023-04-18 16:36:48,284 Log Check for loadMulti over SingleManifest
2023-04-18 16:36:48,286 Log Check for validity over Privileges
2023-04-18 16:36:48,291 Log Check for loadMulti over Privileges
2023-04-18 16:36:48,291 Log Check for validity of secrets and over privileges
2023-04-18 16:36:48,297 Log Check for load MultiYAML over HTTP
2023-04-18 16:36:48,299 Log Check for validity over MissingSecurityContext
2023-04-18 16:36:48,303 Log Check for validity over DefaultNameSpace
2023-04-18 16:36:48,375 Log Check for validity over SingleManifest
2023-04-18 16:36:48,377 Log Check for loadMulti over SingleManifest
2023-04-18 16:36:48,379 Log Check for validity over Privileges
2023-04-18 16:36:48,381 Log Check for loadMulti over Privileges
2023-04-18 16:36:48,381 Log Check for validity of secrets and over privileges
2023-04-18 16:36:48,384 Log Check for load MultiYAML over HTTP
2023-04-18 16:36:48,386 Log Check for validity over MissingSecurityContext
2023-04-18 16:36:48,392 Log Check for validity over DefaultNameSpace
2023-04-18 16:36:48,453 Log Check for validity over SingleManifest
2023-04-18 16:36:48,455 Log Check for loadMulti over SingleManifest
2023-04-18 16:36:48,457 Log Check for validity over Privileges
2023-04-18 16:36:48,458 Log Check for loadMulti over Privileges
2023-04-18 16:36:48,458 Log Check for validity of secrets and over privileges
2023-04-18 16:36:48,461 Log Check for load MultiYAML over HTTP
2023-04-18 16:36:48,462 Log Check for validity over MissingSecurityContext
2023-04-18 16:36:48,464 Log Check for validity over DefaultNameSpace
2023-04-18 16:36:48,563 Log Check for validity over SingleManifest
2023-04-18 16:36:48,588 Log Check for loadMulti over SingleManifest
2023-04-18 16:36:48,612 Log Check for validity over Privileges
2023-04-18 16:36:48,631 Log Check for loadMulti over Privileges
2023-04-18 16:36:48,632 Log Check for validity of secrets and over privileges
2023-04-18 16:36:48,680 Log Check for load MultiYAML over HTTP
2023-04-18 16:36:48,707 Log Check for validity over MissingSecurityContext
2023-04-18 16:36:48,771 Log Check for validity over DefaultNameSpace
2023-04-18 16:36:49,324 Log Check for validity over SingleManifest
2023-04-18 16:36:49,329 Log Check for loadMulti over SingleManifest

```

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: <a href="https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN">https://github.com/kjz0005/AubieTheTiger-SQA2023-AUBURN</a> 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.	✓