For static analysis we implemented bandit as a git hook in “pre-commit”. This operation is only executed when a file is altered and committed. When a file is committed, bandit analyzes all files within the project for security weaknesses and outputs them to “security_report.csv”

For fuzzing, we chose 5 methods that require specific data types and are public. To fuzz generateUnitTest, we passed it an improper filename and received a “No Such file or directory” error. For checkAlgoNames we passed it an integer when it expected a string and received “int object is not iterable”. For checkForLibraryImport we passed a list with a few different data types and received “list object has no \attribute ‘body’”. Run_expirement was given the same list and we received “No such file or directory”. Finally, we provided run_label_perturbation a completely incorrect model name and received “No such file or directory”. While the project requirements state that fuzz.py must be executed automatically via github actions, I had difficulty targeting the file from the workflows directory and was unable to correctly execute it.

For forensics, we implemented logging for all methods in the python_parser.py file in the generation subdirectory. We logged the file name, function name, time, and the input and outputs of the function. Logging the outputs allows someone to compare the actual output with an expected output and ensure the function is working properly, logging the inputs allows someone to ensure that no erroneous inputs are being passed either by accident or maliciously.

Git hook for security analysis (using bandit)

```
$ git commit -m "finalizing fuzz"
[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 3.10.6
[csv]   INFO    CSV output written to file: security_report.csv
```

filename	test_name	test_id	issue_sev	issue_con	issue_cwe	issue_text	line_num	col_offset	line_range	more_info
./generati	blacklist	B311	LOW	HIGH	https://cw	Standard p	28	40	[28]	https://bandit.readthedocs.io/en/1.7.4/blacklists/blacklist_calls.html#b311-random
./label_pe	blacklist	B311	LOW	HIGH	https://cw	Standard p	28	40	[28]	https://bandit.readthedocs.io/en/1.7.4/blacklists/blacklist_calls.html#b311-random
./select_r	blacklist	B404	LOW	HIGH	https://cw	Consider p	7	0	[7]	https://bandit.readthedocs.io/en/1.7.4/blacklists/blacklist_imports.html#b404-import-subprocess
./select_r	start_proc	B607	LOW	HIGH	https://cw	Starting a	26	24	[26]	https://bandit.readthedocs.io/en/1.7.4/plugins/b607_start_process_with_partial_path.html
./select_r	subproces	B603	LOW	HIGH	https://cw	subproces	26	24	[26]	https://bandit.readthedocs.io/en/1.7.4/plugins/b603_subprocess_without_shell_equals_true.html

Execution of “fuzz.py”

client.cpp | server.cpp | fuzz.py | py_parser.py | main.py — generation\identify_algo... | attack_model.py | label_perturbation_main.py | main.py — generation

C:\Users\aweso\OneDrive\Documents\College\Software Quality Assurance\Project\SQAProject\generation\fuzz.py

```python
import main
import py_parser
import label_perturbation_main

if __name__ == '__main__':

    #Fuzzing generateUnitTest. Expects a valid algorithm and attack type. We give it a valid attack type, but we give it 'invalidAlgo'
    try:
        main.generateUnitTest('invalidAlgo', 'random') #passing an invalid filename for the algorithm
    except Exception as exc:
        print(exc)

    #Fuzzing checkAlgoNames. Expects a function list, instead we give it 43
    try:
        py_parser.checkAlgoNames(43) #Inputs something that should not work, but does not return an error
    except Exception as exc:
        print(exc)

    # Fuzzing checkForLibraryImport. Expects a valid model name, instead we give it notWhatItNeeds (which is a list of different data types)
    try:
        notWhatItNeeds = [12, "willit work", 'n']
        py_parser.checkForLibraryImport(notWhatItNeeds)
    except Exception as exc:
        print(exc)

    # Fuzzing run_experiment. Expects a valid model name, instead we give it notWhatItNeeds (which is a list of different data types)
    try:
        notWhatItNeeds = [12, "willit work", 'n']
        label_perturbation_main.run_experiment(notWhatItNeeds)
    except Exception as exc:
        print(exc)


    # Fuzzing run_label_perturbation. Expects a valid model name, instead we give it 'george'
    try:
        label_perturbation_main.run_label_perturbation('george')
    except Exception as exc:
        print(exc)

```

```
[Errno 2] No such file or directory: '../../output/attack_unit_test/test_attack_invalidAlgo.py'
'int' object is not iterable
'list' object has no attribute 'body'
[Errno 2] No such file or directory: 'data//IST_MIR.csv'
george
Started at: 2022-12-01 22:44:51
Change:  0.2
****************************************************************************************************
Initial Experiment
[Errno 2] No such file or directory: 'data//IST_MIR.csv'
[Finished in 5.9s]
```