# Team 1 Project Report

Trace Hereford, Maurice Kenon, Cheyenne Ashou

December 1, 2022

## Security Weaknesses

A pre-commit file was created in the .git/hooks folder to run bandit against all files within TestOrchestrator4ML-main. After a commit, bandit will automatically CSV file called security_weaknesses.csv in the root directory, containing all security vulnerabilities found. The following band it command was used: **bandit -r TestOrchestrator4ML-main -f csv -o security_weaknesses.csv**. A copy of the output can be found at https://github.com/cheyenne-ashou/TEAM1-SQA2022-AUBURN.

Lessons learned: I learned the usefulness of using hooks during program development. It allows you to run tests, check for vulnerabilities, etc. prior to every commit. This is can improve development time and is very simple to write.

## Forensics

Using the forensics method of logging taught in class, a **logger.py** file was created and 5 python methods were modified to incorporate this forensics method. The logger file is inside of the detection folder but is imported throughout.
The modified methods are:

- **main.py (detection)**
- **main.py (generation)**
- **detect_test.py (main)**
- **label_perturbation_main (generation)**
- **attack_model.py (generation)**

Results of the logging store in a PROJECT.LOG file, but as we had troubles running the files as intended, the PROJECT.LOG file does not contain all the logs. The Professor has indicated to us that this is acceptable.

Lessons learned: Forensics are an important way to ensure delivery of quality code. This method of loggins allows the user to ensure everything is working as intended and get feedback from the methods they choose to implement without having to use print statements. This makes it much easier to keep up with as the log can be created and added to rather than having to view the print statements after running the files.

## Fuzzer

The way fuzzing was implemented in this project was by providing bad data as inputs to methods. Five methods were chosen and were imported into a fuzz.py file where the test functions were written

Lessons learned from this is that fuzzing can be a very useful tool when it comes to testing the security of your code. It also is a good way to test out what types of input works for your different functions.