WFH Project Report

Will Humphlett, Keith Fuller Henderson, Hari Prabhanjan L

December 1, 2022

1. Security Weaknesses

A pre-commit hook was created to run bandit on all python files in the repository in an attempt to detect vulnerabilities. For versioning purposes, it is placed in the root of the repository, but to be used, copy it into .git/hooks. It outputs vulnerabilities.csv to record findings. Below is an example:

```
filename,test_name,test_id,issue_severity,issue_confidence,issue_cwe,issue_text...
generation/probability_based_label_perturbation.py,blacklist,B311,LOW,HIGH...
label_perturbation_attack/probability_based_label_perturbation.py,blacklist...
select_repos/dev_count.py,blacklist,B404,LOW,HIGH...
select_repos/dev_count.py,start_process_with_partial_path,B607,LOW,HIGH...
select_repos/dev_count.py,subprocess_without_shell_equals_true,B603,LOW,HIGH...
```

2. Fuzzing

Five methods are fuzzed in fuzz.py located in the root of the repository. The lessons learned from each are as follows.

(a) generateUnitTest - Performs no argument type checking and doesn't gracefully fail on a bad filename

(b) euc_dist - Performs no argument type checking and returns garbage if type correct garbage is passed in. Checks on expected input should instead be performed.

(c) predict - Performs no argument type checking and requires a "self" parameter despite the fact that the method is not defined in a class.

(d) call_loss - Performs no argument type checking and doesn't check for valid filenames, should fail gracefully.

(e) call_prob - Performs no argument type checking and doesn't check for valid filenames, should fail gracefully. In addition, method requires I and g arguments that have no effect on the output.

Below is an excerpt of the output of fuzz.py:

```
FUZZ: generateUnitTest FAILED
Traceback (most recent call last):
  File "fuzz.py", line 14, in fuzz
    result = method(*args)
  File "/home/whumphlett/personal/WFH-SQA2022-AUBURN/generation/main.py", ...
```

```
    file_name = "../../output/attack_unit_test/test_attack_" + algo + ".py"
TypeError: can only concatenate str (not "NoneType") to str
FUZZ: generateUnitTest FAILED
Traceback (most recent call last):
  File "fuzz.py", line 14, in fuzz
    result = method(*args)
  File "/home/whumphlett/personal/WFH-SQA2022-AUBURN/generation/main.py", ...
    file_name = "../../output/attack_unit_test/test_attack_" + algo + ".py"
TypeError: can only concatenate str (not "int") to str
FUZZ: generateUnitTest FAILED
Traceback (most recent call last):
  File "fuzz.py", line 14, in fuzz
    result = method(*args)
  File "/home/whumphlett/personal/WFH-SQA2022-AUBURN/generation/main.py", ...
    file_name = "../../output/attack_unit_test/test_attack_" + algo + ".py"
TypeError: can only concatenate str (not "float") to str
FUZZ: generateUnitTest FAILED
```

3. Forensics

Logging has been added to the same five methods that were fuzzed. It includes both info whenever the method is invoked as well as the args used, and, should the method fail, the error that was encountered. Below is an excerpt of the log.

```
INFO:generation:generateUnitTest(None, None)
ERROR:generation:generateUnitTest(None, None) FAILURE can only concatenate str ...
INFO:generation:generateUnitTest(1, 2)
ERROR:generation:generateUnitTest(1, 2) FAILURE can only concatenate str ...
INFO:generation:generateUnitTest(1.0, 2.0)
ERROR:generation:generateUnitTest(1.0, 2.0) FAILURE can only concatenate str ...
INFO:generation:generateUnitTest([], {})
ERROR:generation:generateUnitTest([], {}) FAILURE can only concatenate str ...
INFO:generation:generateUnitTest(bad-filename, random)
ERROR:generation:generateUnitTest(bad-filename, random) FAILURE [Errno 2] ...
INFO:label_pert/knn:euc_dist(None, None)
ERROR:label_pert/knn:euc_dist(None, None) FAILURE unsupported operand type(s) ...
INFO:label_pert/knn:euc_dist(bad, args)
ERROR:label_pert/knn:euc_dist(bad, args) FAILURE unsupported operand type(s) ...
INFO:label_pert/knn:euc_dist([], {})
ERROR:label_pert/knn:euc_dist([], {}) FAILURE unsupported operand type(s) ...
INFO:label_pert/knn:euc_dist(inf, inf)
INFO:label_pert/knn:euc_dist(-inf, inf)
INFO:label_pert/knn:euc_dist(1j, 1)
INFO:label_pert/knn:euc_dist(nan, nan)
INFO:label_pert/knn:predict(None, 0)
```

```
ERROR:label_pert/knn:predict(None, 0) FAILURE object of type 'int' has no len()
INFO:label_pert/knn:predict(None, 1.0)
ERROR:label_pert/knn:predict(None, 1.0) FAILURE object of type 'float' has no len()
INFO:label_pert/knn:predict(None, bad-iterable)
ERROR:label_pert/knn:predict(None, bad-iterable) FAILURE 'NoneType' object has ...
INFO:label_pert/knn:predict(None, [None, None, None])
ERROR:label_pert/knn:predict(None, [None, None, None]) FAILURE 'NoneType' object ...
INFO:label_pert/knn:predict(None, [])
```

4. Discussion

Each activity aided in the detection of vulnerabilities or made it easier to diagnose them. Developing these tools and integrating them into a CI/CD pipeline helps to build more resilient code over time.