CPSFuzz–ArtifactEvaluation–ReadMe

Sanaz sheikhi

February 2022

1 Introduction

This document is a user manual for the CPSFuzz project [?]. In this project a fuzz testing methodology has been designed and developed to generate test cases for Cyber Physical Systems where software interacts with physical world and advances complexity where traditional testing methodologies such as code coverage or branch coverage are not promising in this area. Our methodology is called CPSFuzz and for evaluation we compared it a baseline random input generator, and two fuzz testing tools called Atheris and Hypothesis. Following sections, are guidance thorough reproducing the results in the paper by running the four tools.

2 Requirements

To run different tools in this project you need to have the following software installed:

- python3.9 or higher version
- Atheris `https://github.com/google/atheris`
- Hypothesis `https://github.com/HypothesisWorks/hypothesis`
- F110th project `https://github.com/stanleybak/f1tenth_gym`

3 Case Study

We apply our methodologies and other ones to the F110th project which is head to head autonomous racing simulation. To get familiar with the project and how to set up the environment, please reference to `https://github.com/stanleybak/f1tenth_gym`

4 Running the tools

To provide the data for analysis you should run each tool separately. We have provided the required setting for each tool to run on top of the F110th project in separate python script. You just need to run the following commands in separate command-lines.

- python3.9 CPSFuzzer.py
- python3.9 AtherisFuzzer.py
- python3.9 HypothesisFuzzer.py
- python3.9 RandomFuzzer.py

The data required for cpsfuzz score analysis and DBScan will automatically be stored in the following files in the current directory. You just need to check if the files have proper read/write permissions:

- cpsfuzz_data, atheris_data, hypothesis_data, random_data
- cpsfuzz_crash_cmd , atheris_crash_cmd, hypothesis_crash_cmd, random_crash_cmd

Atheris crashes based on its code coverage logic. If you didn't get what you need run it again.

5 CPSFuzz Score Analysis

After running any/all of the aforementioned fuzzing tools and having all the data ready you can run the following command to get total cpsfuzz score, the score over time plot and maximum coverage of objective space for the chosen fuzzing tool.

- python3.9 score_analysis.py

These results are corresponding to results in the evaluation section of the paper respectively including the total cps fuzz score, the score progress thorough time plot, maximum state space coverage plot. However, it is worth noting that the results published in the paper have been gathered and averaged thorough 5000 time frames, equivalent to 6 hours of running each tool.

So, here to demo the performance and functionality of the CPSFUzz tool we set the runtime of each fuzzer to be 500 time frames and stored the results in the aforementioned files. So, if you don't want to go thorough running process, you can easily leave the files untouched and just run the score_analysis.py program.

If you would like to change the number of running cycles you can change this number which is hard coded in the fuzzer.py and CPSFuzzer.py programs. But pay attention that for small number of time frames CPSFuzz performs no better or even worse than other fuzzers due to computation runtime.

6 DBScan Coverage metric

To generate the DBScan coverage plots, make sure you have the crash data inputs in the *_crash_cmd files and go to the following directory:

- cd /DBS

Run one of the following commands to get the respective DBScan diagram corresponding to Fig.4 in the paper:

- python3.9 fuzz_test_smooth_blocking.py ../cpsfuzz_crash_cmd
- python3.9 fuzz_test_smooth_blocking.py ../atheris_crash_cmd
- python3.9 fuzz_test_smooth_blocking.py ../hypothesis_crash_cmd
- python3.9 fuzz_test_smooth_blocking.py ../random_crash_cmd

Running any of these command generates and saves a .pkl file in *DBS/cache* directory and you can later run the db_scan_result.py tool on this file to reproduce the DBScan diagram without running the above commands which running them can take hours depending on the number of test cases. On the other hand if you want a fresh new execution you should delete the pkl file from the DBS/cache directory.

- python3.9 db_scan_result.py cache/root_SmoothBlockingDriver_ran_0.pkl

7 Project Files

There exist a bunch of file being automatically filled, stored and restored by the tools and you need to do nothing except checking their permissions:

- seeds: initial data inputs (seeds) used as corpus of the CPSFuzz.
- cpsfuzz_data, atheris_data, hypothesis_data, random_data : hold the fuzz testing results of each tool and are maintained by each tool. They are being used by the score analysis program later.
- cpsfuzz_crash_cmd , atheris_crash_cmd, hypothesis_crash_cmd, random_crash_cmd : hold the test case input led to crash during fuzzing. They are used for DBScan metric

However, there may exist slow-unit-* and crash-* files that are artifacts of the Atheris too and you can remove them. Just pay close attention not to delete other files with similar names. It is worth mentioning that Atheris which is a newly developed tool crashes a lot because of some code coverage metric it follows and we don't care about. In this case we repeatedly run this tool until we get desired results.

References