Compiler Fuzzing via Guided Value Mutation - Progress Report

PIUS KRIEMLER* and JONAS DEGELO*, ETH Zürich, Switzerland

ACM Reference Format:
Pius Kriemler and Jonas Degelo. 2023. Compiler Fuzzing via Guided Value Mutation - Progress Report. In . ACM, New York, NY, USA, 2 pages.

1 PROGRESS OVERVIEW

This progress report outlines the advances in developing a pipeline for guided value mutation. The current pipeline commences by automatically preprocessing a set of around 150 seed programs from the c-testsuite [1]. The preprocessing stage involves running the gcc preprocessor and filtering incompatible seed files. In a later stage, the seed files are parsed using pycparser [2] to identify all constants. However, for this step, pycparser requires the removal of all compiler directives, which is why the preprocessing step is recommended by the pycparser documentation.

The preprocessed seed files are parsed using pycparser to create an abstract syntax tree (AST). Constants are then easily identified and modified by manipulating this AST. The mutated code is written back into a C source file, and before it is compiled and compared, it is first compiled with sanitizer compiler flags and run. This helps to identify any invalid programs, which are by definition, invalid mutations. Although the response of this validity check could be utilized to improve the efficiency of finding valid mutations in the future, we postponed it's implementation, depending on the performance of the planned Bayesian Optimizer. The mutations itself are currently random and in the case of an invalid mutation we retry.

The results of the mutation process, including all mutation attempts and their outcomes, are saved. These statistics can be used to create visualizations and obtain further insights, thus enhancing the overall usability of the program. The stage to automatically do this after every run is currently in the making.

In conclusion, we have a working version of the naive approach where we mutate values uniformly at random. We have found two mutations where the two compiler versions produce a different number of assembly instructions. In one case the difference is extremely minor with only one instruction. The other case is more interesting as the difference is 16 instructions.

2 STATUS AND OUTLOOK

We are currently on track with our initial timeline, as we completed the pipeline and random value mutation. We will proceed as planned, starting with a Bayesian Optimizer that hopefully guides the value mutation to large binary differences more efficiently. As already mentioned, we are also working on making the program more user-friendly. We already parameterized the program to a large extend, which enables running experiments more quickly, and we are working on providing better visualisations of the results. Besides that, we will run more extensive experiments with the naive version.

*Both authors contributed equally to this research.

Manuscript submitted to ACM

REFERENCES

[1] [n. d.]. c-testsuite. https://github.com/c-testsuite/c-testsuite Accessed: 2023-04-28.
[2] [n. d.]. pycparser. https://github.com/eliben/pycparser Accessed: 2023-04-28.