Running KLEE on GNU coreutils

Valentin Huber

Institute of Applied Information Technology
Zürich University of Applied Sciences ZHAW
contact@valentinhuber.me

January 24, 2024

Contents

1 Introduction

KLEE [1] is an open source, symbolic execution based, advanced fuzzing platform. It was introduced in the seminal paper titled "KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs" in 2008. In their article, Cadar *et al.* present their work and evaluate it on a diverse set of programs. The most prominent of those is the GNU coreutils suite, in which ten fatal errors were found.

Ever since then, KLEE has not only matured as a fuzzer, it has also been used extensively as a platform for other researchers to build on top of, as I have discovered in [3]. As an introduction to the practical side of fuzzing, I attempted to answer the following questions about KLEE:

1. Reproducing the original paper (see Section 2)
   (a) Can the current version of KLEE be run on coreutils version 6.10 (as tested in the original paper)?
   (b) Can the same metrics as measured in the original paper still be measured?
   (c) How do the measured metrics compare to what was published 15 years ago?
2. Examining the statistical distribution of results over different fuzzing times (see Section 3)
   (a) How does the non-determinism in KLEE influence the variance in the results between different test runs?
   (b) How do different testing timeouts influence results?
3. Testing more recent versions of coreutils (see Section 4)
   (a) What needs to change in the test setup to test more recent versions of coreutils?
   (b) How do the results from testing different versions of coreutils differ?

All experiments were run on a virtuallized server with the following specs: AMD EPYC 7713 64C 225W 2.0GHz Processor, 1 TiB RAM, 2x 25GiB/s Ethernet.

2 Reproducing the Original Paper

I'm basing my experiment setup on the original paper [2], the FAQs in the project's documentaition [4] and the tutorial on testing coreutils version 6.11 [5].

2.1 Project Setup

2.1.1 Building coreutils 6.10 on a Current Version of Ubuntu

2.1.2 Using an Old Version of Ubuntu

2.1.3 LLVM

2.1.4 Running KLEE

2.2 Analyzing the Results

2.2.1 Gathered Metrics

2.2.2 Comparison to the Original Paper

3 Analysis of the Results

3.1 Metrics Distribution

3.2 Influence of Testing Timeout

4 Testing More Recent Versions of coreutils

4.1 Differences in Testing Setup

4.2 Findings

5 Discussion

5.1 Research Questions

5.2 Produced Artifacts

5.3 Future Work

Bibliography

[1] "KLEE symbolic execution engine." (2024), [Online]. Available: `https://klee.github.io` (visited on Jan. 24, 2024).

[2] C. Cadar, D. Dunbar, and D. Engler, "Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08, San Diego, California: USENIX Association, 2008, pp. 209–224.

[3] V. Huber, "Challenges and mitigation strategies in symbolic execution based fuzzing through the lens of survey papers," Dec. 2023.

[4] "OSDI'08 coreutils experiments." (2024), [Online]. Available: `https://klee.github.io/docs/coreutils-experiments/` (visited on Jan. 24, 2024).

[5] "Tutorial on how to use KLEE to test GNU coreutils." (2024), [Online]. Available: `https://klee.github.io/tutorials/testing-coreutils/` (visited on Jan. 24, 2024).