Differential Fuzzing on coreutils Using LibAFL

—

Report

Valentin Huber

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

May 24, 2024

Abstract

Hello, World!

Contents

1 Introduction

1.1 Differential Fuzzing

- Fuzzing has been popular and effective at finding bugs.
- Much research has gone into input selection/guiding.
- Not much has been done when it comes to oracles.
- Differential Fuzzing: What it is.

1.2 coreutils

- Short overview
- uutils' version

1.3 LibAFL

- History of AFL/AFL++
- Problems with incompatible forks and how LibAFL attempts to fix them

1.4 Research Questions

1. Which parts of coreutils can be fuzzed? What performance tradeoffs does each part introduce?
2. How can the necessary instrumentation be introduced into coreutils? What are the engineering and performance implications of each option?
3. Can LibAFL feasibly be used to build a system with all logic defined in the answers to the questions above?
4. If yes, how effective is the resulting fuzzer at finding bugs in coreutils? What kind of bug can be found with it?
5. Can the system be expanded to implement differential fuzzing between the different implementations? What changes are necessary?
6. If yes, how effective is the this second fuzzer at finding bugs in coreutils? What kind of bug can be found with it?

2 Background

2.1 coreutils

On Feb. 8, 1990, MacKenzie announced fileutils, a suite of utilities for reading and altering files [1]. A year later, he released textutils (to parse and manipulate text) [2] and shellutils (to write powerful shell scripts) [3]. These three collections were folded into one on Jan. 13, 2003, called the GNU coreutils. [4] `ls`, `cat`, `base64`, `grep`, `env`, or `whoami`: GNU's coreutils are at the basis of how users interact with most Linux distributions on the command line. [5] Because they are so widely used and central to how users interact with their computers, software quality and lack of software defects is especially important to coreutils.

Version 9.5 of the GNU coreutils was released on Mar. 28, 2024 and thus marks the current version as of this report. 106 programs are built per default. [6]

2.1.1 Interface

Users primarily interact with coreutils through the command line or in shell scripts. They take different kinds of inputs, i.e. behave differently based on changes to:

- Data passed to `stdin`, e.g. through Unix pipes
- Command line arguments:
  - Unnamed arguments, either required (such as `cp <source> <destination>`) or optional (such as `ls [directory]`)
  - Flags without any associated data, such as `--help`
  - Flags with associated data, either required (such as `dd if=<input file> of=<output file>`) or optional such as `-name <pattern>` in `find`
- Environment variables, such as `LANG`
- The file system content, such as for `ls`

The output, or effects of invocations fall into the following categories:

- Data written to `stdout`
- Data written to `stderr`
- The exit status of the process
- The signal terminating the process
- Changes to the file system

2.1.2 Alternative Implementations

Since the release of GNU coreutils, multiple alternative implementations were released. Notable among those are BusyBox [7], which aims to provide most of the GNU coreutils with a focus on resource restrictions. It is therefore primarily used in embedded systems [7] or tiny distributions such as Alpine Linux [8].

In the general push towards rewriting software in memory-safe languages, the uutils project [9] maintains a drop-in replacement implementation of the GNU coreutils written in Rust. [10] It does contain all programs, but is still missing certain options. All differences with GNU's coreutils are treated as bugs. It further aims to not only work on Linux, but is also available for MacOS and Windows.

2.1.3 Build System

2.2 LibAFL

2.2.1 Generic Concepts

2.2.2 Usage of Traits

2.2.3 Interface

Shared Memory

CommandExecutor

3 State of the Art

3.1 Fuzzing coreutils

3.1.1 Concolic Execution Frameworks

3.1.2 Other approaches

Sjöbom and Hasselberg use a very simplistic approach: They just run AFL [12] on coreutils. [11] However, their approach has all the drawbacks outlined in section4.1.1.

3.2 Differential Fuzzing

4 Implementation

4.1 Basic Unguided Fuzzer

4.1.1 Feedback and Environment Protection

4.1.2 Custom Input Type

4.2 Optimizations

4.3 Gathering Coverage Information from GNU coreutils

4.3.1 Instrumentation

4.3.2 Dynamic Interface

4.4 Gathering Coverage Information from uutils coreutils

4.5 Differential Fuzzing

4.5.1 Existing Functionality

4.5.2 Custom Extensions

5 Results

5.1 Performance

5.2 Evaluation on base64

6 Discussion

6.1 Research Questions

6.2 Contributions

6.3 Limitations

6.4 Future Work

6.5 Summary

Bibliography

[1] D. J. MacKenzie, *Gnu file utilities release 1.0*, Email to gnu.utils.bug Email List, Feb. 8, 1990. [Online]. Available: `https://groups.google.com/g/gnu.utils.bug/c/CviP42X_hCY/m/YssXFn-JrX4J` (visited on May 22, 2024).

[2] D. J. MacKenzie, *New gnu file and text utilities released*, Email to gnu.utils.bug Email List, Jul. 15, 1991. [Online]. Available: `https://groups.google.com/g/gnu.utils.bug/c/iN5KuoJYRhU/m/V_6oiBAWF0EJ` (visited on May 22, 2024).

[3] D. J. MacKenzie, *Gnu shell programming utilities released*, Email to gnu.utils.bug Email List, Aug. 22, 1991. [Online]. Available: https://groups.google.com/g/gnu.utils.bug/c/xpTRtuFpNQc/m/mRc_7JWZ0BYJ (visited on May 22, 2024).

[4] J. Meyering, *Package renamed to coreutils*, git commit, Jan. 13, 2003. [Online]. Available: https://git.savannah.gnu.org/cgit/coreutils.git/tree/README-package-renamed-to-coreutils (visited on May 22, 2024).

[5] R. Stallman. "Linux and the gnu system." (), [Online]. Available: https://www.gnu.org/gnu/linux-and-gnu.en.html (visited on May 22, 2024).

[6] J. Meyering, P. Brady, B. Voelker, E. Blake, P. Eggert, and A. Gordon, *Gnu coreutils 9.5*, Software Release, Mar. 28, 2024. [Online]. Available: https://ftp.gnu.org/gnu/coreutils/coreutils-9.5.tar.gz (visited on May 22, 2024).

[7] "Busybox: The swiss army knife of embedded linux." (), [Online]. Available: https://www.busybox.net/about.html (visited on May 22, 2024).

[8] "Alpine linux — about." (), [Online]. Available: https://alpinelinux.org/about/ (visited on May 22, 2024).

[9] "Uutils." (), [Online]. Available: https://uutils.github.io/ (visited on May 22, 2024).

[10] "Uutils — coreutils." (), [Online]. Available: https://uutils.github.io/coreutils (visited on May 22, 2024).

[11] A. Sjöbom and A. Hasselberg. "Fuzzing." (Apr. 25, 2019), [Online]. Available: https://github.com/adamhass/fuzzing/ (visited on May 21, 2024).

[12] M. Moroz. "Fuzzing." (Jun. 8, 2021), [Online]. Available: https://github.com/google/AFL (visited on May 21, 2024).