C++ const and Immutability: An Empirical Study of Writes-Through-const (Artifact)

Jon Eyolfson[1] and Patrick Lam[2]

1 **University of Waterloo**
Waterloo, ON, Canada
jeyolfso@uwaterloo.ca
2 **University of Waterloo**
Waterloo ON, Canada
patrick.lam@uwaterloo.ca

Abstract

This artifact is based on `ConstSanitizer`, a dynamic program analysis tool that detects deep immutability violations through `const` qualifiers. Our tool instruments any code compiled by `clang` with the `-fsanitizer=const` flag. Our implementation includes both instrumentation of LLVM code and a runtime library to support our analysis. The provided package includes our tool and all experiments used in our companion paper. Instructions are also provided.

1998 ACM Subject Classification D.3.3 Language Constructs and Features
Keywords and phrases empirical study, dynamic analysis, immutability
Digital Object Identifier 10.4230/DARTS.2.1.3
Related Article Jon Eyolfson and Patrick Lam, "C++ const and Immutability: An Empirical Study of Writes-Through-const", in Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP 2016), LIPIcs, Vol. 56, pp. 8:1–8:25, 2016.
http://dx.doi.org/10.4230/LIPIcs.ECOOP.2016.8
Related Conference 30th European Conference on Object-Oriented Programming (ECOOP 2016), July 18–22, 2016, Rome, Italy

1 Scope

This artifact includes the full source code for our tool, `ConstSanitizer`, and supports the repeatability of our experiments in the companion paper. We encourage users to reproduce our results and use the tool in other software projects.

2 Content

The artifact package includes:

- a Virtual Machine image with everything included
- detailed instructions for using the artifact and for rebuilding it from scratch, provided as an `index.html` file.

To simplify repeatabiliy of our experiments, we provide a QEMU and VirtualBox disk image containing a minimal GNU/Linux distribution with all required sources for the tool and experiments. The image is based on Arch Linux, up-to-date on 2016-04-22. For reference, the Linux kernel version is 4.4.3. This image includes a pre-compiled version of our tool available through the default `clang` and `clang++` commands. The instructions outline how to build our tool from scratch, if desired, however it is time consuming. All of the experiments, including their sources and dependencies, are pre-installed so an internet connection is not required.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of the artifact is available on GitHub: `https://github.com/eyolfson/research-2016-ecoop-artifact`. The `README` in this repository provides the current links to all artifact packages and sources. The sources linked in this `README` are the modified LLVM code for the tool. This repository also contains the scripts for running all experiments.

4 Tested platforms

The artifact is known to work on any platform running QEMU or VirtualBox. We highly recommend CPUs with hardware virtualization and at least 2 GiB of memory for the Virtual Machine. The scripts themselves depend on Arch Linux, or any other variant that has the `pacman` command.

The we originally developed and tested the tool on a machine running Arch Linux, up-to-date at the time of publication. The machine is an Intel i7-3930K with 32 GiB of RAM, and a 256 GB SSD.

5 License

LLVM Modifications. University of Illinois/NCSA Open Source License (`https://opensource.org/licenses/UoI-NCSA.php`)
Scripts. GPL-3.0 (`http://www.gnu.org/licenses/gpl-3.0.en.html`)

6 MD5 sum of the artifact

`0d30b05ad520209348fd38a3662e45f5` (QEMU)
`6d5a88c36de2cec6fa50bd74d601cea9` (VDI)

7 Size of the artifact

1.5 GB