Jenny: Securing Syscalls for PKU-based Memory Isolation Systems

David Schrammel, Samuel Weiser, Richard Sadek,
and Stefan Mangard, *Graz University of Technology*

https://www.usenix.org/conference/usenixsecurity22/presentation/schrammel

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

The artifacts available at https://github.com/IAIK/Jenny contain the source code of the prototype described in the paper including any instructions and scripts to reproduce the figures from the paper. Please see the included `README.md` for further instructions.

A.2 Artifact check-list (meta-information)

- **Program:** We use the existing tools `nginx` and `lmbench` for our benchmarks. Both are included as submodules within our artifact repository and compiled from source.
- **Run-time environment:** The artifacts require Ubuntu 20.04 with Linux 5.4.0 and root permissions. Detailed setup instructions are provided in the `README.md`.
- **Hardware:** A CPU with Memory Protection Keys (MPK) (e.g., Intel Xeon Scalable) is required.
- **Output:** The artifacts include scripts to generate all benchmark-related figures from the paper.
- **How much disk space required (approximately):** 10GB
- **How much time is needed to prepare workflow (approximately):** 1–4 hours
- **How much time is needed to complete experiments (approximately):** approx. 1 day
- **Publicly available:** https://github.com/IAIK/Jenny

A.3 Description

A.3.1 How to access

The artifacts are available at https://github.com/IAIK/Jenny/tree/39bb0c696ce3c178e9593b7dbc034b2447ba2d00. Instructions on how to clone and use them, as well as any required hardware and software dependencies are detailed in the `README.txt` within this repository.

A.4 Installation

The artifacts are built and installed separately for the different benchmarks. See instructions below.

A.5 Evaluation and expected results

When the current working directory is `code/OurLib`, then the microbenchmarks can be run with `make bench-x86`. This creates a new subdirectory called `benchmarks/output_syscalls_{datetime}` for the results. There, `tc_getpid.pdf` and `tc_open.pdf` should be created, which were used in the paper as Figure 4 and 5.

For the application benchmarks, first our library has to be recompiled using `make app-bench-x86`.

Then, within the `benchmarks` directory, the three commands `./run_all.sh applications`, `./run_all.sh nginx`, and `./run_all.sh lmbench` will place the results in the `benchmarks/output_{datetime}`. The resulting pdf files `appbench_single.pdf`, `lmbench_numbers.pdf`, `appbench_nginx_single.pdf`, and `output_true_initialization_overhead.pdf` correspond to Figures 6, 7, 9, and 10 from the paper.

Note, that the resulting numbers will be slightly different compared to the paper since they are dependent on the exact CPU model, CPU frequency, kernel version, compiler versions, virtualization, etc.