SymSan: Time and Space Efficient Concolic Execution via Dynamic Data-flow Analysis

Ju Chen, *UC Riverside;* Wookhyun Han, *KAIST;* Mingjun Yin, Haochen Zeng, and Chengyu Song, *UC Riverside;* Byoungyoung Lee, *Seoul National University;* Heng Yin, *UC Riverside;* Insik Shin, *KAIST*

https://www.usenix.org/conference/usenixsecurity22/presentation/chen-ju

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

A Artifact Appendix

A.1 Abstract

We have packed all the required software environments for reproducing our experiment results in a single docker image. The readers will rebuild the docker image and run the provided scripts to collect the results on time consumption, memory consumption, and code coverage. The collected results should match the records presented in the paper. To reproduce the results, the readers should have a server with at least 16GB RAM and 512GB storage space. Due to hardware discrepancies, the readers may observe some minor variations.

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

- **Run-time environment:** Linux, Docker
- **Hardware:** At least 16GB main memory and at least 512GB free disk space.
- **Metrics:** Execution time, memory consumption, and code coverage
- **Output:** Execution time and memory consumption records and raw generated inputs.
- **How much disk space required (approximately)?:** 512GB
- **How much time is needed to prepare workflow (approximately)?:** About 4 hours
- **How much time is needed to complete experiments (approximately)?:** About 7-14 days. The actual time needed depends on the hardware configurations.
- **Publicly available (explicitly provide evolving version reference)?:** Yes. We have made our code and scripts to reproduce the results in the paper publicly available at Github. The stable link pointing to the Git commit is https://github.com/R-Fuzz/fastgen/commit/01d31bc6bb42ee3535bb3aa8a0f88d345e9cb23d
- **Code licenses (if publicly available)?:** 3-clause BSD license
- **Data licenses (if publicly available)?:** 3-clause BSD license
- **Archived (explicitly provide DOI or stable reference)?:** Yes. Our code and scripts are archived in the Github. The stable link pointing to the Git Commit is https://github.com/R-Fuzz/fastgen/commit/01d31bc6bb42ee3535bb3aa8a0f88d345e9cb23d

A.3 Description

A.3.1 How to access

We have open-sourced all our code at https://github.com/R-Fuzz/fastgen/commit/01d31bc6bb42ee3535bb3aa8a0f88d345e9cb23d. We have also included the instructions to build the experiments environment and reproduce our results in the same repository.

A.3.2 Hardware dependencies

N/A

A.3.3 Software dependencies

Docker

A.3.4 Data sets

N/A

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

We have provided a docker image which contains the complete environment for reproducing the experiments results.
To build the docker image, just run:

```
$ docker build -t usenix .
```

The details are under the "installation" section of the *README* file in the provided repository.

A.5 Experiment workflow

We have described the instructions for reproducing the results in the *README* file in the provided repository. A reader can follow the instructions, run the scripts and collect all the experiment results.

A.6 Evaluation and expected results

The main claim of the paper is that our system SymSan can significantly speed up the concolic execution and reduce the memory consumption. To support the claims, we conducted the experiments described in sections 5.2.1, 5.2.3, and 5.3 with the results presented in Table 2, Figure 4, Figure 5, and Figure 6 respectively.

By following the instructions in the *README*, a reader should be able to collect the results in Table 2, Figures 4,5, and 6. Specifically,

- To reproduce the results in paper's section 5.2.1, follow instructions in section 3.1.1 (nbench) of the *README*. The results in this section show that our system SymSan has a smaller instrumentation overhead than SymCC and SymQEMU.
- To reproduce the results in paper's section 5.2.3, follow instructions in section 3.1.2 (CGC) and section 3.1.3 (Real-world applications) of the *README*. The results in this section show that our system SymSan has a smaller performance overhead than SymCC and SymQEMU.

- To reproduce the results in paper's section 5.3, follow the instructions in section 3.2 (Memory consumption without solving) of the *README*. The results in this section show that SymSan has a smaller memory overhead than SymCC.

Because we perform all our evaluations on a server with an Intel(R) Xeon(R) E5-2683 v4 @ 2.10GHz (40MB cache) and 512GB of RAM, running Ubuntu 16.04 with Linux 4.4.0 64-bit, we recommend our readers hire a server with similar configurations for minimal variations.

The variations of performance numbers depend on the following factors: 1. The CPU frequency and size of CPU caches 2. The memory size and bandwidth, and 3. Whether or not the external storage is a hard disk or solid-state driver (for program loading). The maximum variations in absolute numbers should not exceed 100%. The variations for the relative numbers (SymSan's speed-up over SymCC) should not exceed 50%.

A.7 Version

Based on the LaTeX template for Artifact Evaluation V20220119.