Toward Tool-Independent Summaries for Symbolic Execution (Artifact)

Frederico Ramos
Instituto Superior Técnico, University of Lisbon, Portugal
INESC-ID Lisbon, Portugal

Nuno Sabino
Instituto Superior Técnico, University of Lisbon, Portugal
Carnegie Mellon University, Pittsburgh, PA, USA
Institute of Telecommunications, Campus de Santiago, Aveiro, Portugal

Pedro Adão
Instituto Superior Técnico, University of Lisbon, Portugal
Institute of Telecommunications, Campus de Santiago, Aveiro, Portugal

David A. Naumann
Stevens Institute of Technology, Hoboken, NJ, USA

José Fragoso Santos
Instituto Superior Técnico, University of Lisbon, Portugal
INESC-ID Lisbon, Portugal

Abstract

The artifact contains the extended versions of the tools *angr* and *AVD* with support for the symbolic reflection API proposed in the paper. Additionally, the artifact contains the source code of SumBoundVerify, our novel tool for the bounded-verification of symbolic summaries for the C programming language. The artifact contains all the scripts and datasets required to obtain the results presented in the paper, including: a library of 67 symbolic summaries implemented using the proposed symbolic reflection API; two symbolic test suites designed to test two open source C libraries; and the source code of the third-party summaries that were validated checked with SumBoundVerify.

2012 ACM Subject Classification Software and its engineering → Software verification and validation; Security and privacy → Formal methods and theory of security

Keywords and phrases Symbolic Execution, Runtime Modelling, Symbolic Summaries

Digital Object Identifier 10.4230/DARTS.9.2.7

Acknowledgements The authors were supported by Fundação para a Ciência e a Tecnologia (UIDB/50008/2020, Instituto de Telecomunicações, and UIDB/50021/2020, INESC-ID multi-annual funding, and PhD grant SFRH/BD/150692/2020), project DIVINA (CMU/TIC/0053/2021), the SmartRetail project (C6632206063-00466847) financed by IAPMEI, the European Commission under grant agreement number 830892 (SPARTA), and the NSF award CNS-1718713.

Related Article Frederico Ramos, Nuno Sabino, Pedro Adão, David A. Naumann, and José Fragoso Santos, "Toward Tool-Independent Summaries for Symbolic Execution", in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 24:1–24:29, 2023.
`https://doi.org/10.4230/LIPIcs.ECOOP.2023.24`

Related Conference 37th European Conference on Object-Oriented Programming (ECOOP 2023), July 17–21, 2023, Seattle, Washington, United States

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2023 Call for Artifacts and the ACM Artifact Review and Badging Policy.

Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Scope

The artifact is composed of two main elements: **(1)** our versions of the tools angr [5] and AVD [3], extended with support for the symbolic reflection API proposed in the paper and **(2)** the source code of SumBoundVerify, our novel tool for the bounded verification of symbolic summaries for the C programming language. This artifact aims to fulfill the requirements for ECOOP's *functional* badge.

Symbolic Reflection API

The paper proposes a new symbolic reflection API for developing tool-independent summaries that can be shared across different symbolic execution tools. The core idea of the paper is that instead of writing symbolic summaries in the programming language used to build each tool, tool developers should implement symbolic summaries directly in C using a shared symbolic reflection API. By implementing the shared API, tool developers gain access to all the symbolic summaries implemented using that API.

To demonstrate how easy it is to implement the proposed symbolic reflection API, we extended two tools with support for it: *angr* [5], a state-of-the-art tool developed at the University of California Santa Barbara, and *AVD* [3], our own symbolic execution tool for C. The two extended tools are included in the artifact along with instructions that explain their usage.

To evaluate the expressivity of our symbolic reflection API, we have developed a comprehensive library of symbolic summaries. This library comprises 67 summaries, encompassing 26 libc functions sourced from three distinct header files: *string.h*, *stdlib.h*, and *stdio.h*. All the developed summaries are included in the artifact.

We evaluate the performance of the summaries developed using our API by comparing it against the performance of both native summaries and reference implementations. In order to carry out this experiment, we designed two symbolic test suites for two open source C libraries: **(1)** an *HashMap* library [6], which provides an implementation of a standard array-based hash table, and **(2)** a *Dynamic Strings* library [4], which provides an implementation of heap-allocated strings. This experiment was designed to focus on libc usage with both libraries making intensive use of libc string processing functions. The source code of both symbolic test suites is included in the artifact together with the scripts for automating their execution and the source code of the targeted libraries.

SumBoundVerify

In addition to the symbolic reflection API, the paper describes SumBoundVerify, a new tool for the bounded verification of symbolic summaries. SumBoundVerify works by comparing the symbolic states resulting from the symbolic execution of the summary to be verified against the states resulting from the execution of its reference implementation. SumBoundVerify classifies summaries as: **(1)** *over-approximating* if the traces modelled by the summary *contain* the traces of the corresponding reference implementation; **(2)** *under-approximating* if the traces modelled by the summary *are contained* in the traces of the corresponding reference implementation; or **(3)** *unsound* if the summary is neither over- nor under-approximating. The source code of SumBoundVerify is included in the artifact along with instructions that explain its usage and a range of illustrative examples.

To evaluate the effectiveness of SumBoundVerify, we conducted a thorough bug-finding analysis on a number of summaries used by three prominent symbolic execution tools: *angr* [5], *Binsec* [1], and *Manticore* [2]. Furthermore, we used SumBoundVerify to verify the correctness of part of our own summaries. This experiment revealed a total of 24 bugs in third-party tools and 13 bugs in our summaries. The artifact includes the source code of SumBoundVerify as well as the code of all the summaries that were checked against it and the corresponding reference implementations.

2 Content

The artifact package includes:
- the source code of *AVD* and *angr* including the extension of both tools with support for our symbolic reflection API;
- the source code of SumBoundVerify;
- the source code of our tool-independent summaries for libc functions;
- the source code of the third-party summaries that we validated with SumBoundVerify;
- the source of the *C-HashMap* and *Dynamic Strings* libraries together with their corresponding symbolic test suites.

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). In addition, the artifact is also available at: `https://doi.org/10.6084/m9.figshare.21696386.v1`.

4 Tested platforms

To ensure the reproducibility of our experiments, we have made the artifact available as a compressed *Docker* image (`.tgz`) that can be run on any platform supporting the *Docker Engine*. This portable format enables easy deployment and execution of our experiment environment.

For accurate reproduction of our performance experiments, it is recommended to run the Docker image on hardware with the following specifications:

- CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
- RAM: 32GiB
- Disk Space: 200GiB

These hardware specifications were used during our experimental setup and will help ensure optimal compatibility when reproducing the results of the paper.

5 License

The *HashMap* [6] library is licensed under the MIT license while the *Dynamic Strings* [4] library and *angr* [5] are both licensed under the BSD 2-Clause simplified licence. Our C-implemented summaries, *AVD* and SumBoundVerify are licensed under the Apache license.

6 MD5 sum of the artifact

2d124f1174690bd2451636860b4b804b

7 Size of the artifact

1.33 GB

References

1. R. David, S. Bardin, T. D. Ta, L. Mounier, J. Feist, M. Potet, and J. Marion. Binsec/se: A dynamic symbolic execution toolkit for binary-level analysis. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 653–656, 2016. doi:10.1109/SANER.2016.43.
2. Mark Mossberg, Felipe Manzano, Eric Hennenfent, Alex Groce, Gustavo Grieco, Josselin Feist, Trent Brunson, and Artem Dinaburg. Manticore: A user-friendly symbolic execution framework for binaries and smart contracts. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1186–1189, 2019. doi:10.1109/ASE.2019.00133.
3. Nuno Sabino. Automatic vulnerability detection: Using compressed execution traces to guide symbolic execution. Master's thesis, Instituto Superior Técnico, November 2019.
4. Salvatore Sanfilippo. Simple dynamic strings [online]. 2015. Accessed: 29th June 2023. URL: https://github.com/antirez/sds.
5. Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel, and G. Vigna. SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 138–157, 2016. doi:10.1109/SP.2016.17.
6. Richard Wiedenhöft. C Hash map [online]. 2014. Accessed: 29th June 2023. URL: https://gist.github.com/Richard-W/9568649.