Concolic Execution for WebAssembly (Artifact)

Filipe Marques
Instituto Superior Técnico, University of Lisbon, Portugal
INESC-ID Lisbon, Portugal

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

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

Pedro Adão
Instituto Superior Técnico, University of Lisbon, Portugal
Instituto de Telecomunicações, Aveiro, Portugal

Abstract

This artifact contains the implementation of WASP, a symbolic execution engine for Wasm, and WASP-C, a symbolic execution framework for testing C programs built using WASP. WASP works directly on Wasm code and was built on top of a standard-compliant Wasm reference implementation [4]. WASP was thoroughly evaluated: it was used to symbolically test a generic data-structure library and the Amazon Encryption SDK for C, demonstrating that it can find bugs and generate high-coverage testing inputs for real-world C applications; WASP was further tested against the Test-Comp benchmark, obtaining results comparable to well-established symbolic execution and testing tools for C.

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

Keywords and phrases Concolic Testing, WebAssembly, Test-Generation, Testing C Programs

Digital Object Identifier 10.4230/DARTS.8.2.20

Acknowledgements The authors were supported by national funds through Fundação para a Ciência e a Tecnologia (UIDB/50008/2020, Instituto de Telecomunicações, and UIDB/50021/2020, INESC-ID multi-annual funding), projects INFOCOS (PTDC/CCI-COM/32378/2017) and DIVINA (CMU/T-IC/0053/2021), and by the European Commission under grant agreement number 830892 (SPARTA).

Related Article Filipe Marques, José Fragoso Santos, Nuno Santos, and Pedro Adão, "Concolic Execution for WebAssembly", in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 11:1–11:29, 2022.
https://doi.org/10.4230/LIPIcs.ECOOP.2022.11

Related Conference 36th European Conference on Object-Oriented Programming (ECOOP 2022), June 6–10, 2022, Berlin, Germany

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2022 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Scope

This artifact contains a distribution of the WebAssembly Symbolic Processor (WASP), a novel concolic execution engine for testing Wasm (version 1.0) modules. WASP follows the so-called *concolic discipline* [3, 6], combining concrete execution with symbolic execution and exploring one execution path at a time. We implemented WASP by instrumenting the Wasm reference interpreter developed by Haas et al. [4]. This approach opens up the possibility for a range of optimisations

in the context of concolic execution. WASP comes with the two optimisations discussed in the related paper: application of algebraic simplifications to byte-level symbolic expressions generated by memory interactions and shortcut restarts for failed assumption statements. The artifact additionally includes WASP-C, a new symbolic execution framework for testing C programs built on top of WASP, and four symbolic test suites for the evaluation of symbolic execution tools for Wasm. These test suites comprise a stand-alone Wasm B-tree data structure inspired by the one described by Watt et al. [8] and the Wasm compilation of: **(i)** Collections-C [5], a widely-used generic data structure library for C, **(ii)** the Test-Comp [1] benchmarks, and **(iii)** the Amazon Encryption SDK [7].

In summary, the provided artifact supports the following claims of the paper:

1. WASP outperforms Manticore, its only competitor tool, in the analysis of our stand-alone Wasm B-tree data structure [8].
2. WASP-C outperforms Gillian-C [2] in the testing of the Collections-C [5] symbolic benchmarks, finding three bugs, including a new bug that Gillian-C did not detect.[1]
3. WASP-C obtained results comparable to well-established symbolic execution and testing tools for C when tested against the Test-Comp [1] benchmark suite.
4. The optimisations described in the related paper are essential to WASP's performance.
5. WASP-C is able to symbolically test the Amazon Encryption SDK [7], generating a high-coverage test suite for that library.

2 Content

The artifact package includes:

- the source code of Gillian, WASP, and WASP-C;
- the benchmarks on which we evaluate our tools:
  - a stand-alone Wasm B-tree implementation;
  - Collections-C;
  - Test-Comp;
  - AWS Amazon Encryption SDK for C.
- a REAME file describing the project structure, with instructions for running WASP and WASP-C, and reproducing the experimental results.

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://github.com/wasp-platform/wasp/pkgs/container/wasp`.

4 Tested platforms

The artifact was developed, tested, and packaged using Ubuntu 20.04 LTS. The recommended VM hardware requirements are:

- 32GiB of RAM
- 60GiB of disk space
- CPU $\geq$ Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
- Cores ≥ 8

[1] Discovered bug: `https://github.com/srdja/Collections-C/issues/147`.

5 License

WASP, WASP-C, the original Wasm reference interpreter by Haas et al. [4], and the AWS Encryption SDK for C are licensed under the Apache License. Both Test-Comp and Collections-C are licensed under the GNU Lesser General Public License v3.

6 MD5 sum of the artifact

18c15a089a044dcc0b55742c364d6f5a

7 Size of the artifact

916 MiB

References

1 Dirk Beyer. International Competition on Software Testing (Test-Comp). In *Tools and Algorithms for the Construction and Analysis of Systems*, 2019.

2 José Fragoso Santos, Petar Maksimović, Sacha-Élie Ayoun, and Philippa Gardner. Gillian, Part I: A Multi-Language Platform for Symbolic Execution. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020.

3 Patrice Godefroid, Nils Klarlund, and Koushik Sen. Dart: Directed automated random testing. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2005.

4 Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with WebAssembly. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2017.

5 Srđan Panić. Collections-C [online]. Accessed 5th-July-2021. URL: `https://github.com/srdja/Collections-C` [cited 5th July 2021].

6 Koushik Sen, Darko Marinov, and Gul Agha. CUTE: A Concolic Unit Testing Engine for C. *ACM SIGSOFT Software Engineering Notes*, 2005.

7 Amazon Web Services. AWS Encryption SDK for C. `https://github.com/aws/aws-encryption-sdk-c`. Accessed: 2021-09-08.

8 Conrad Watt, Petar Maksimovic, Neelakantan R. Krishnaswami, and Philippa Gardner. A program logic for first-order encapsulated WebAssembly. In Alastair F. Donaldson, editor, *European Conference on Object-Oriented Programming*, 2019.