

Automated Binary Code Analysis: Efficient Memory Reasoning via Generalized Writes

Frédéric Recoules and Sébastien Bardin

CEA, List

Site de NanoInnov (Palaiseau, Plateau de Saclay)

`frederic.recoules@cea.fr` `sebastien.bardin@cea.fr`

`https://binsec.github.io/`

Context. Symbolic execution [2, 1] is a powerful technique for automated program analysis, bug detection, and vulnerability assessment. The open-source BINSEC [5, 11] platform, developed at CEA, leverages symbolic execution for binary-level analysis, enabling applications such as bug characterization [9], side-channel leakage detection [3], and fault injection evaluation [6]. However, symbolic execution suffers from two major scalability issues: path explosion and constraint bloating. These issues are especially acute for binary code analysis, where memory operations like `memset` or `memcpy` generate large numbers of constraints that are computationally expensive for SMT solvers to handle [10].

The problem. Current symbolic memory representations fail to efficiently model generalized write operations (e.g., `memset`, `memcpy`) without resorting to explicit quantifiers or costly expansions.

For example, modeling `memset(ptr, val, cnt)` requires constraints such as $\forall i; ptr \leq i < ptr + cnt \implies mem[i] = val$, which are computationally prohibitive for SMT solvers. While existing approaches like the Array Property Fragment [4, 12] provide partial solutions, they lack seamless integration into symbolic execution frameworks like BINSEC and may introduce extra overhead.

Goal. This internship aims to improve BINSEC’s symbolic memory representation—inspired by [7] — by introducing a new representation for block operations like `memset` and `memcpy` in BINSEC’s intermediate language. The goal is to lower constraint complexity or size without sacrificing analysis precision.

The intern will design and implement this new representation for block operations, integrate it with optimizations such as read-over-write simplification [8], and explore SMT solver alternatives like Uninterpreted Functions. Finally, the candidate will build a representative benchmark to evaluate the performance gains on real-world memory operations.

Expected Outcomes. The internship will deliver a prototype implementation of the generalized write operation in BINSEC’s intermediate representation, fully integrated into the constraint solver and simplification pipeline. A representative benchmark suite will also be developed to quantify performance improvements in symbolic execution, comparing the new approach against existing methods for handling memory operations like `memset` and `memcpy`.

Candidate profile. Experience with functional programming languages with a keen interest in algorithmic problem-solving and the formal aspects of computer science, including logic, automata theory and compilation/compiler techniques. Familiarity with binary analysis or reverse engineering is a plus.

Supervision. The internship will be co-supervised by Frédéric Recoules and Sébastien Bardin. There will be a regular weekly follow-up meeting, either on site at Nano-Innov (Plateau de Saclay) or by videoconference.

References

- [1] Roberto Baldoni, Emilio Coppa, Daniele Cono D’Elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 2018.
- [2] Cristian Cadar and Koushik Sen. Symbolic execution for software testing: Three decades later. *Commun. ACM*, 2013.
- [3] Lesly-Ann Daniel, Sébastien Bardin, and Tamara Rezk. Binsec/rel: Symbolic binary analyzer for security with applications to constant-time and secret-erasure. *ACM Trans. Priv. Secur.*, 2023.
- [4] Leonardo de Moura and Nikolaj Bjørner. Generalized, efficient array decision procedures. In *2009 FMCAD*, 2009.
- [5] Adel Djoudi and Sébastien Bardin. Binsec: Binary code analysis with low-level regions. In *TACAS*, 2015.
- [6] Soline Ducouso, Sébastien Bardin, and Marie-Laure Potet. Adversarial reachability for program-level security analysis. In *ESOP*, 2023.
- [7] Stephan Falke, Florian Merz, and Carsten Sinz. Extending the theory of arrays: `memset`, `memcpy`, and beyond. In *VSTTE*, 2014.
- [8] Benjamin Farinier, Robin David, Sébastien Bardin, and Matthieu Lemerre. Arrays made simpler: An efficient, scalable and thorough preprocessing. In *LPAR*, 2018.
- [9] Guillaume Girol, Benjamin Farinier, and Sébastien Bardin. Not all bugs are created equal, but robust reachability can tell the difference. In *CAV’21*.
- [10] Shelly Grossman, John Toman, Alexander Bakst, Sameer Arora, Mooly Sagiv, and Chandrakana Nandi. Practical verification of smart contracts using memory splitting. (OOPSLA2), 2024.
- [11] CEA List. Binsec. <https://binsec.github.io/>.
- [12] Mathias Preiner, Aina Niemetz, and Armin Biere. Better lemmas with lambda extraction. In *FMCAD*, 2015.