

Modeling and Validating Complex Fault Injection Protections on Binary Code

Yanis Sellami
CEA, List

Sébastien Bardin
CEA, List

yanis.sellami@cea.fr, sebastien.bardin@cea.fr

Automated program analysis is a research topic that tries to design tools and analyses to detect bugs in programs. Existing techniques typically target memory access vulnerabilities (*e.g.* buffer overflows) and numerical vulnerabilities (*e.g.* zero division) that can corrupt program data or crash the execution. Yet, even when one can prove that a program contains no such bug, an attacker can still target the hardware on which the program is running. Examples of such attacks include fault injection attacks, where the attacker leverages a physical perturbation of the hardware to modify the execution of the program and bypass existing security protections.

Automated detection and verification analyses for fault injection vulnerabilities are expensive, as each fault increases the number of possible executions we have to consider. The binary code analysis platform BINSEC [1] developed at CEA offers a state of the art analysis for fault attacks detection [2], which supports a variety of fault effects such as value modification, test inversions and instruction skips. Still, there remains considerations in the fault attacks that are not yet modeled within BINSEC, but are required to evaluate some of the countermeasures that are designed to protect against fault injection attacks. First, some protections make use of both software and hardware mechanisms, such as supplementary instructions, in order to reduce their overhead [3]. Second, some countermeasures protect against attack scenarios in which the fault budget is limited, usually by the number of possible faults, but also through more complex requirements such as the absence of value replay, a minimal distance between faults, etc [4].

The goal of this project is to extend the capabilities of the fault injection analysis of BINSEC to model these mechanisms in order to verify existing protections, starting with [3]. This will require understanding the fault attacks limitations and hardware/software protections considered, deciding how to represent them in a binary program analysis, implementing them, and validating them through rigorous experimental evaluations.

Logistics. The project will be co-supervised by Yanis Sellami and Sébastien Bardin, Nano-INNOV.

Candidate. To candidate or obtain additional information, please contact the supervisors by email.

References

- [1] “Plateforme binsec.” <https://binsec.github.io/>.
- [2] S. Ducousso, S. Bardin, and M.-L. Potet, “Adversarial reachability for program-level security analysis,” in *Programming Languages and Systems* (T. Wies, ed.), (Cham), pp. 59–89, Springer Nature Switzerland, 2023.
- [3] S. Michelland, C. Deleuze, and L. Gonnord, “From low-level fault modeling (of a pipeline attack) to a proven hardening scheme,” in *Proceedings of the 33rd ACM SIGPLAN International Conference on Compiler Construction*, CC 2024, (New York, NY, USA), p. 174–185, Association for Computing Machinery, 2024.
- [4] C. Averill, I. Buzzetti, A. Bellon, and K. Hamlen, “Lapse: Automatic, formal fault-tolerant correctness proofs for native code,” BAR 2026.