SCI-FI: Control Signal, Code, and Control Flow Integrity against Fault Injection Attacks

Thomas Chamelot[1], Damien Couroussé[1], Karine Heydemann[2]

[1]: CEA, List [2]: Sorbonne Univesité, LIP6
Presenter: Thomas Chamelot, thomas.chamelot@cea.fr

Fault injection attacks are known to be able to tamper with the code and the control flow of a program. Several counter-measures have been proposed to thwart such attacks [1,2,4,5]. However, recent work highlights that some vulnerabilities exist in the microarchitecture [3]. As a consequence, the control signals involved in the whole pipelined execution of instructions inside the processor also needs to be protected. Such so-called execution integrity is not covered by state-of-the-art approaches.

We present SCI-FI, a counter-measure against fault injection attacks that guarantees simultaneously code integrity, control-flow integrity and execution integrity. SCI-FI is a mixed hardware and software counter-measure. It combines sequentially two techniques: a signature-based approach and a duplication-based one. Code integrity and control flow integrity are ensured by the signature-based approach, which needs compiler support as well as additional custom instructions. The duplication-based approach guarantees execution integrity until the end of the execution pipeline. The security level provided by SCI-FI highly depends on the signature function as well as the size of the reference signatures. SCI-FI can be implemented with several signature functions, as the properties of the signature function imply a trade-off between security (e.g., number of bit flips that can be detected) and silicon area overhead. It may also impact code size and code slowdown. We also illustrate how signature constructs based on cryptography can also support other security properties, such as authentication.

This poster will cover work already published [6, 7] regarding SCI-FI and its implementation in a RISC-V core, and in addition we will describe how SCI-FI handles indirect branches regarding the CFI protection. We will present evaluation results regarding the overheads in terms of silicon area, code size and execution time. These results show that our countermeasure is competitive regarding existing code and control-flow integrity approaches, while also providing control signal integrity. To the best of our knowledge, our countermeasure is the first to cover fault injections targeting the processor microarchitecture.

[1] J.-L. Danger et al., "Processor Anchor to Increase the Robustness Against Fault Injection and Cyber Attacks," in Constructive Side-Channel Analysis and Secure Design, vol. 12244, G. M. Bertoni and F. Regazzoni, Eds. Cham: Springer International Publishing, 2021, pp. 254–274.

[2] O. Savry, M. El-Majihi, and T. Hiscock, "Confidaent: Control FLow protection with Instruction and Data Authenticated Encryption," in 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia, Aug. 2020, pp. 246–253, doi: 10.1109/DSD51259.2020.00048.

[3] J. Laurent, V. Beroulle, C. Deleuze, F. Pebay-Peyroula, and A. Papadimitriou, "Cross-layer analysis of software fault models and countermeasures against hardware fault attacks in a RISC-V processor," Microprocessors and Microsystems, vol. 71, p. 102862, Nov. 2019, doi: 10.1016/j.micpro.2019.102862.

[4] M. Werner, T. Unterluggauer, D. Schaffenrath, and S. Mangard, "Sponge-Based Control-Flow Protection for IoT Devices," arXiv:1802.06691 [cs], Feb. 2018, Accessed: Dec. 03, 2019. [Online]. Available: http://arxiv.org/abs/1802.06691.

[5] R. de Clercq et al., "SOFIA: Software and control flow integrity architecture," in 2016 Design, Automation Test in Europe Conference Exhibition (DATE), Mar. 2016, pp. 1172–1177.

[6] T. Chamelot, D. Couroussé, and K. Heydeman, "SCI-FI – Control Signal, Code, and Control Flow Integrity against Fault Injection Attacks," https://jaif.io/2021, Paris, 2021.

[7] T. Chamelot, D. Couroussé, and K. Heydeman, "SCI-FI – Control Signal, Code, and Control Flow Integrity against Fault Injection Attacks," in DATE, 2022.