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

Thomas Chamelot¹, Damien Couroussé¹, Karine Heydemann²

¹Université Grenoble Alpes, CEA, LIST F-38000 Grenoble, France, firstname.lastname@cea.fr
²Sorbonne Université, CNRS, LIP6 F-75005, Paris, France, firstname.lastname@lip6.fr

This work was partially funded by the French National Research Agency (ANR) under grant agreement ANR-18-CE39-0003

CONTENTS

Fault Injection Attacks
An attacker performs a fault injection attack by using power or clock glitch, EM pulse or laser beam to perturb an integrated circuit.

Required Security Properties

- Code integrity
- Control-flow integrity
- Data integrity

Problem

It has been shown that some vulnerabilities exist at the microarchitectural level [1].

Contributions

Protecting the Pipeline Control Path

New security property: Execution integrity

- SCI-FI combines code and control-flow integrity properties with execution integrity
- SCI-FI achieves execution integrity by protecting the pipeline’s control signals

Principles

1. Data-independent control signals outputted by Decode are gathered into a so-called pipeline state Σ
2. The CCFI module enforces code and control-flow integrity and execution integrity for Decode and Execute stages
   (a) Computes signature from current pipeline state and previous signature
      \[ S_i = f(Σ_i, S_{i-1}) \]
   (b) Updates signature to generate collision for instructions with multiple predecessors after a taken branch
      \[ S_i' = u(S_i, patch) \]
   (c) Verifies runtime signatures against reference signatures located after dedicated control-flow instructions
3. The CSI module enforces execution integrity
   (a) Duplicate signals from the pipeline state
   (b) Checks duplicated signals between pipeline stages
4. A dedicated tool generates reference signatures and patches at compile time

SCI-FI dedicated instructions

- Verification instructions load a reference signature immediately following in the program memory, and trigger the signature verification: scifi.beq, scifi.bne, scifi.jal, scifi.jalr
- Load patch instructions fetch a patch value into the CCFI module: scifi.ldp

ARCHITECTURE

Experimental Evaluation

RISC-V RV32I CV32E40P

ASIC implementation 28-FDSOI @ 400MHz
- CRC32: +6.5%
- CBC-MAC Prince: +23.8%

Software evaluation using LLVM 12 with Newlib on Embench-IOT
- Code size: +29.4%
- Execution time: +18.4%

CONCLUSION

Security Properties

Code, control-flow and execution integrity and additionally code authenticity with CBC-MAC as the signature function

Overheads

Similar to existing state-of-the-art counter-measures for code and control-flow integrity

Future Work

Support for more complex architectures and more complex software (OOP, OS, ...)
You can learn more about SCI-FI in [2]!

BIBLIOGRAPHY