Linear Code Extraction

Lightweight Countermeasures Against Original Attacks on a RISC-V Core

Théophile Gousselot*, Olivier Thomas†, Jean-Max Dutertre*, Olivier Potin*, Jean-Baptiste Rigaud*
*Mines Saint-Etienne, CEA, Leti, Centre CMP, F - 13541 Gardanne France
theophile.gousselot@emse.fr dutertre@emse.fr olivier.potin@emse.fr rigaud@emse.fr
†Texplained olivier@texplained.com

Linear Code Extraction Background

Linear Code Extraction (LCE):

- Invasive attack, introduced in [1]
- Extracting completely a code from a memory.

LCE involves to microprobe:

- The instruction bus (**eavesdropping** ◀) [2]
- Internal nodes of the core to induce linear execution (**forcing** ◀)

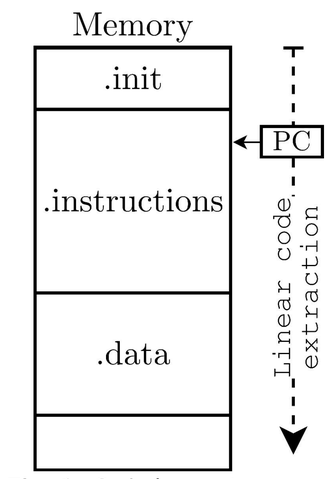

Fig. 1. Code layout in memory.

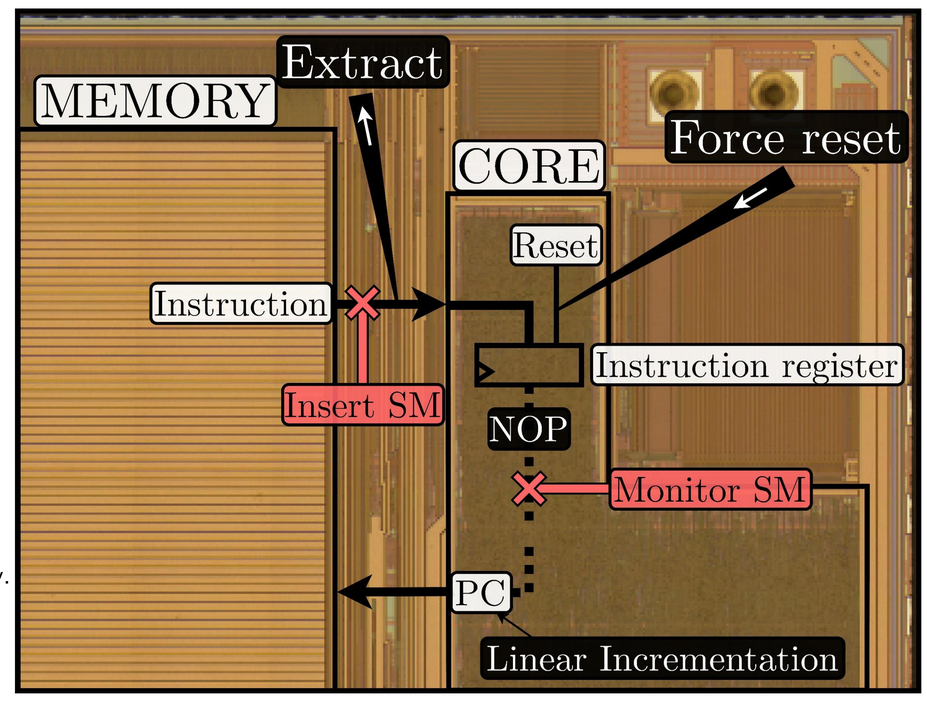

Fig. 2. LCE operation on an Integrated Circuit overview.

Our contributions

Threat: assessing a RISC-V core vulnerability to LCE

➔ Defining 3 types of LCE attack paths for linear execution:

- Freezing a CI[2] (e.g., *addi t1,t1,-260*)
- Editing bits of the incoming instructions to turn DI[1] into CI[2]
- Tampering with the Program Counter (PC)

Introducing 3 lightweight countermeasures to detect LCE:

- Security Marker Monitoring: (▭) in Fig. 2
- Discontinuity Instruction Monitoring
- Linear Addressing Monitoring

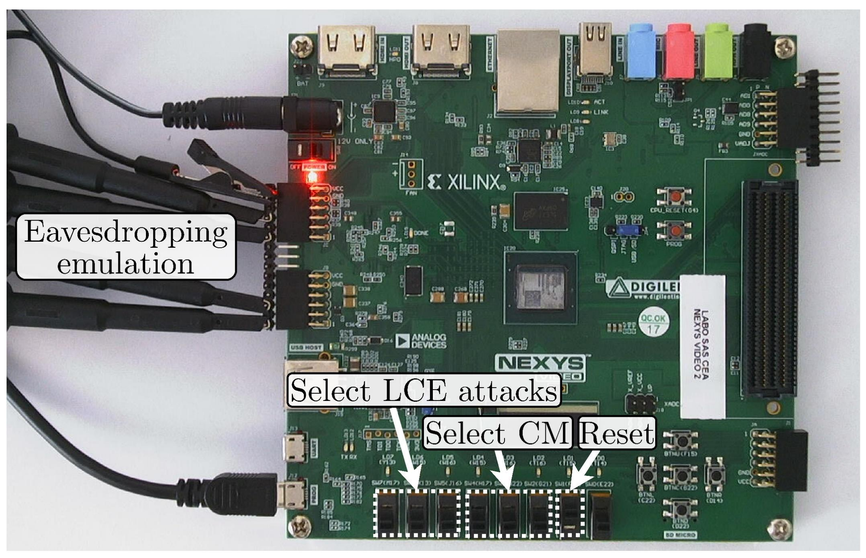

Fig. 3. Setup of the board with probes connected to the oscilloscope.

Experimental Setup

https://github.com/theophile-gousselot/linear_code_extraction

Functional simulation framework:

- cv32e40p RISC-V core
- Embench benchmarks
- All LCE attack paths/countermeasures

FPGA-based demonstration: (Artix 7 on a Nexys Video)

- Observe core behaviors for many LCE attack paths/countermeasures
  - ➔ Emulate microprobing:
    - Eavesdropping: route bus to FPGA outputs
    - Forcing: multiplexers placed on target nodes

Demonstration results

Three contexts of execution:

- **Nominal:** extraction without linear execution branch (*bgtu*) is executed in a loop
  - ➔ **Incomplete and disorder extraction**
- **LCE:** branch (*bgtu*) not executed
  - ➔ **Complete extraction**
- **CM:** LCE on a core protected
  - ➔ **Neutralize the LCE**

```
...
0x8c: addi   t1,t1,-260
0x90: sw     zero,0(t0)
0x94: addi   t0,t0,4
0x98: bgtu   t1,t0,-8
0x9c: auipc  sp,34070528
0xa0: addi   sp,sp,744
0xa4: j      2578
...
```

Fig. 4. Running code

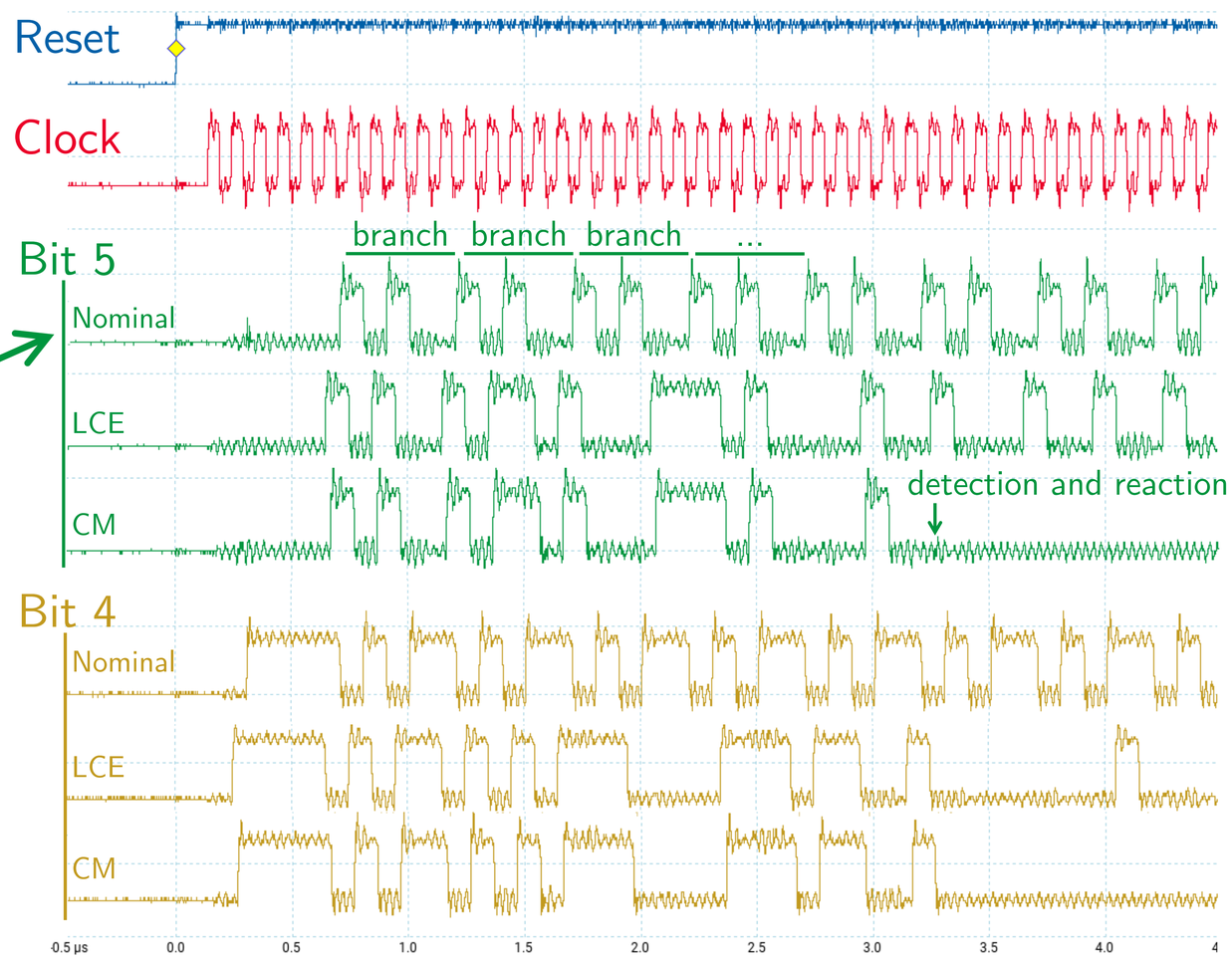

Fig. 5. Bits of the instruction bus extracted on the oscilloscope.

Conclusion

LCE:

- RISC-V processors are vulnerable to LCE
- LCEs performed on simulation and FPGA-emulation.

Countermeasures:

- Detect the three types of LCE attack paths
- Area overhead between 0.52% and 1.47% of the cv32e40p
- Clock cycle overhead can be null or kept below 1%

[1]**Discontinuity Instruction (DI)** refers to every instruction able to break a sequential execution (e.g., jump, branch).

[2]**Continuity Instruction (CI)** refers to every other instruction.

[1] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors." Smartcard, vol. 99, pp. 9–20
[2] C. Helfmeier et al., "Breaking and entering through the silicon," in ACM CCS, 2013, pp. 733–744.

The ARSENE project was funded by the "France 2030" government investment plan managed by the French National Research Agency, under the reference ANR-22-PECY-0004.
This research was also funded by the European Union under grant agreement no. 101070008, Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.