KU LEUVEN

Towards Secure Speculation for the Constant-Time Policy

Short Talk @ 2022 SILM workshop

June 6th 2022

Work in Progress

Lesly-Ann Daniel
KU Leuven

Marton Bognar
KU Leuven

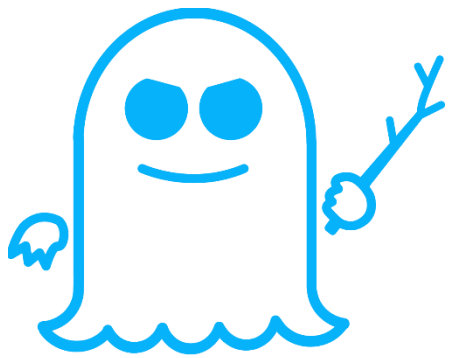
Job Noorman
KU Leuven

Sébastien Bardin
CEA List

Tamara Rezk
INRIA

Frank Piessens
KU Leuven

Spectre Attacks & Hardware-Software Contracts



Hardware-Software Contracts for Secure Speculation

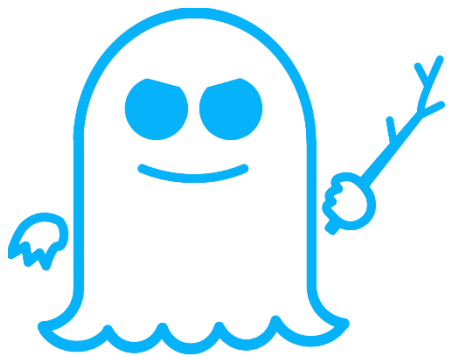
Marco Guarnieri*, Boris Köpf†, Jan Reineke‡, and Pepe Vila*
*IMDEA Software Institute †Microsoft Research ‡Saarland University

Formally reason about defenses & Enable hardware-software co-design

Foundational Framework

- Secure software design, verification and compilation
- *Formally express guarantees of hardware defenses*

Spectre Attacks & Hardware-Software Contracts



Hardware-Software Contracts for Secure Speculation

Marco Guarnieri*, Boris Köpf†, Jan Reineke‡, and Pepe Vila*
*IMDEA Software Institute †Microsoft Research ‡Saarland University

Formally reason about defenses & Enable hardware-software co-design

Foundational Framework

☹ No hardware defense studied in the paper enables secure speculation for constant-time policy!

Secure Speculation for Constant-Time?

Constant-time Programming

Protection against (non-transient) microarchitectural attacks

- Used in many cryptographic implementations
- No secret-dependent control flow & memory accesses

Constant-Time in the Spectre Era

- Speculative semantics for software defenses & verification
  → Hard to reason about & accommodate new speculation mechanisms?

- Hardware defense: disable speculation
  → Not acceptable

Secure Speculation for Constant-Time?

Constant-time Programming

Protection against (non-transient) microarchitectural attacks

- Used in many cryptographic implementations
- No secret-dependent control flow & memory accesses

Secure Speculation for Constant-Time:
Efficient hardware defense → off-the-shelf constant-time programs do not leak secrets

Secure Speculation for Constant-Time via Hardware Secret-Tracking

Hardware Secret-Tracking (HST)

- Inform hardware of what is secret
- Track secret taint in hardware
- Hardware do not leak tainted values during speculation

ConTExT: A Generic Approach for Mitigating Spectre

Michael Schwarz[1], Moritz Lipp[1], Claudio Canella[1], Robert Schilling[1,2], Florian Kargl[1], Daniel Gruss[1]
[1]Graz University of Technology [2]Know-Center GmbH

SpectreGuard: An Efficient Data-centric Defense Mechanism against Spectre Attacks

Jacob Fustos, University of Kansas
Farzad Farshchi, University of Kansas
Heechul Yun, University of Kansas

Speculative Privacy Tracking (SPT): Leaking Information From Speculative Execution Without Compromising Privacy

Rutvik Choudhary, UIUC, USA
Jiyong Yu, UIUC, USA
Christopher W. Fletcher, UIUC, USA
Adam Morrison, Tel Aviv University, Israel

Secure Speculation for Constant-Time via Hardware Secret-Tracking

Hardware Secret-Tracking (HST)

- Inform hardware of what is secret
- Track secret taint in hardware
- Hardware do not leak tainted values during speculation

ConTExT: A Generic Approach for Mitigating

Mechanism

Technical implementation details & evaluation
But still no end-to-end formal security guarantee
for constant-time programs

Rutvik Choudhary
UIUC, USA

Jiyong Yu
UIUC, USA

Christopher W. Fletcher
UIUC, USA

Adam Morrison
Tel Aviv University, Israel

What we propose

- Formal framework for hardware secret-tracking
  - Wide range of speculation mechanisms
  - Generalizes prior HST mechanisms
- Proof that CT programs do not leak secrets during speculations
  - All Spectre variants + LVI
  - Allows for *declassification*
- Implementation in a RISC-V microarchitecture
  - First synthesizable implementation
  - Evaluation of the hardware costs

Future Work

- Hardware-software contract?

$\{\cdot\}_{\mathrm{HST}} \not\vdash [\![\cdot]\!]^{\mathrm{seq}}_{\mathrm{ct}}$

→ Declassification?
→ Policy-aware contract?

- Compiler-support?
  - → Separate secret from public memory
  - → Ensure no unintentional declassification
- Validating our RISC-V implementation
  - → Contract-based CPU testing (e.g. Revizor, Scam-V)?
  - → Hardware-fuzzing?
  - → Model checking?

Future Work

- Hardware-software contract?

$\{\cdot\}_{\mathrm{HST}} \not\vdash [\![\cdot]\!]^{\mathrm{seq}}_{\mathrm{ct}}$

→ Declassification?
→ Policy-aware contract?

- Compiler-support?
  - → Separate secret from public memory
  - → Ensure no unintentional declassification
- Validating our RISC-V implementation

Thanks for your attention
Any question, feedback, suggestion is welcome ☺

Credit

Icons made by **Freepik**
from www.flaticon.com

Hard work icon created by
monkik – Flaticon
www.flaticon.com/free-icons/hard-work

Diamond icons created by
Vectors Market – Flaticon
www.flaticon.com/free-icons/diamond