Unique Program Execution Checking:

Formal Security Guarantees for RISC-V Systems

Alex Wezel[1], Lucas Deutschmann[1], Tobias Jauch[1],
Dino Mehmedagić[1], Johannes Müller[1], Mohamed Ali[1],
Anna Lena Duque Antón[1], Philipp Schmitz[1],
Mohammad Rahmani Fadiheh[2], Dominik Stoffel[1], Wolfgang Kunz[1]

[1]RPTU Kaiserslautern-Landau, Germany
[2]Stanford University, Stanford, CA, USA

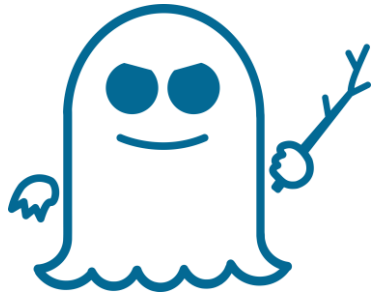
RISC-V Summit Europe, Munich, 27th June 2024

RP TU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Stanford University

Motivation

- Discovery of Meltdown and Spectre

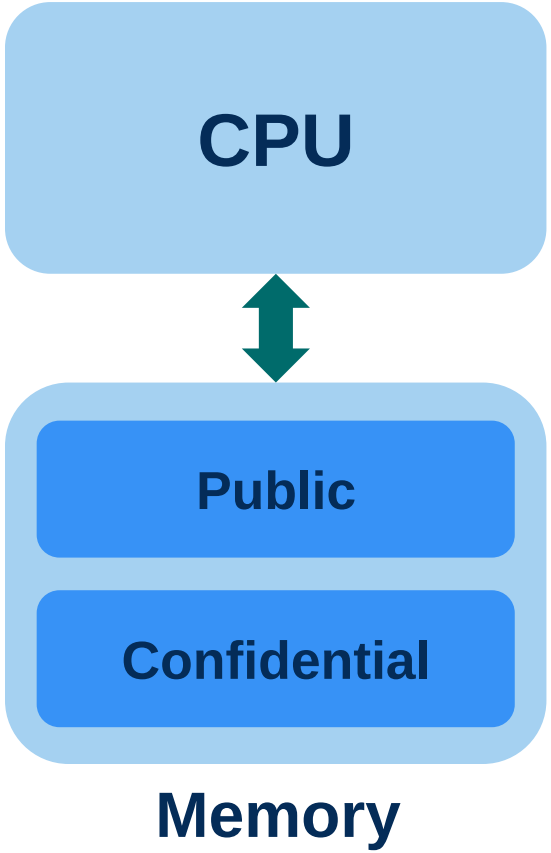


- Industry and academia found various vulnerabilities

- Need for exhaustive security guarantees

Confidentiality: Spectre Attack

- Attacker mistrains branch predictor to make the victim access confidential data transiently
- Secret data is encoded in the cache („microarchitectural footprint“) and extracted via a Flush and Reload attack
- Confidentiality is violated because the confidential data interferes with the execution of the attacker program



Unique Program Execution Checking (UPEC)

UPEC exhaustively detects all propagations of information from or to critical locations in a given RTL design:

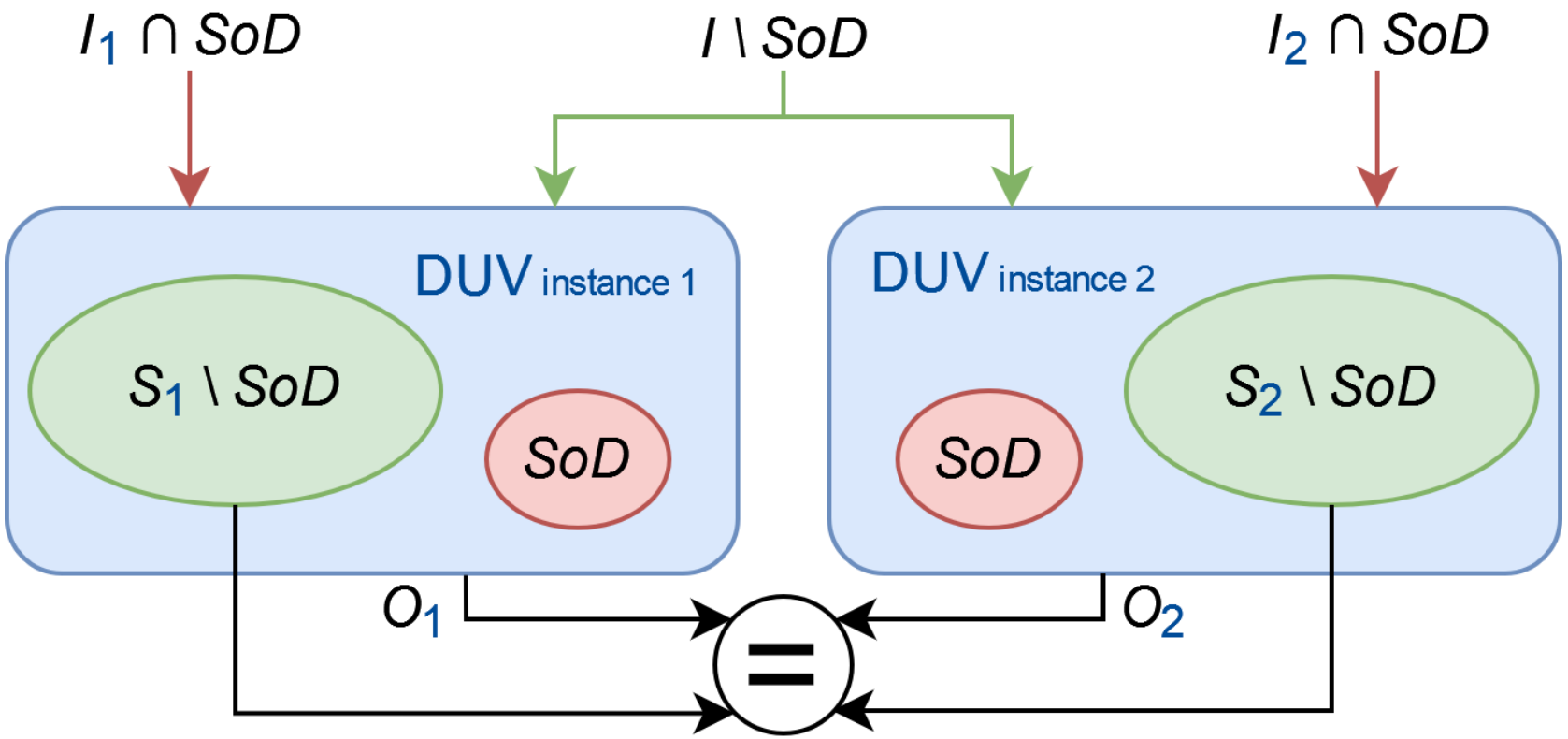
- Possible leakage of confidential information
- Malicious interference with protected data (integrity)

UPEC proves that a system executes uniquely w.r.t. the signals of interest called Source of Discrepancy (SoD)

Advantages of UPEC

- ✓ Exhaustive detection of security vulnerabilities
- ✓ Independent of functional correctness of the DUV
- ✓ Scalable even for large designs (OOO cores, whole SoCs)
- ✓ Adaptable to different threat models

Computational Model



DUV: Design under Verification

SoD: Source of Discrepancy

I: Inputs

O: Outputs

S: Microarchitectural State Variables

Generic UPEC Property

assume:

at t: $S_1 \setminus SoD == S_2 \setminus SoD;$

during$[t, t+k]$: $I_1 \setminus SoD == I_2 \setminus SoD;$

at t: *threat_model();*

prove:

during$[t, t+k]$: $S_1 \setminus SoD == S_2 \setminus SoD;$

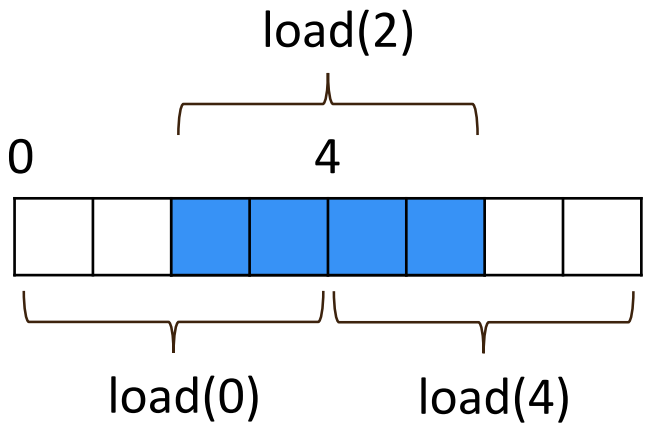
during$[t, t+k]$: $O_1 == O_2;$

UPEC-driven Design of Secure Systems

- Set up the miter (SoD, threat model-dependent constraints)
- Run the property check and inspect counterexamples
  - Harmless propagation → refine SoD
  - Propagation violating security → apply appropriate mitigation
- Repeat until no more counterexamples appear
- DUV is guaranteed to be secure w.r.t. the threat model

UPEC for Data-Independent Timing (DIT)

- UPEC-DIT detects data-dependent timing in accelerators and processors
  - Example Ibex: features a DIT mode enabled by a corresponding CSR
  - UPEC-DIT revealed a timing dependency, even in DIT mode: misaligned addresses lead to an additional memory access



Number of memory accesses is different from an aligned access!

Paper:

Case Studies

Security Target	DUV	Detected Vulnerabilities	Reference
Transient-Execution-Attacks	*BOOM*	*Multiple Spectre variants, Meltdown*	
Functional Security Bugs in SoCs	*Pulpissimo*	*Confused deputy attack using an accelerator ignoring PMP*	
Operation Integrity in SoCs	*OpenTitan*	*Denial-of-Service attack using an untrusted IP*	

Summary

- UPEC is a scalable methodology for exhaustively detecting malicious information flows
- Case studies show the versatility of UPEC and its easy adaptibility to different threat models
- UPEC enables to provide a hardware root of trust for higher levels of the system stack

Thank you for your attention!

Many thanks to many collaborators!

Mohamed Ali, Jörg Bormann, Lucas Deutschmann, Anna Lena Duque Antón, Wolfgang Ecker, Mohammad Rahmani Fadiheh, Jason M. Fung, Bo-Yuan Huang, Tobias Jauch, Wolfgang Kunz, John Matthews, Johannes Müller, Dino Mehmedagić, Subhasish Mitra, Sayak Ray, Philipp Schmitz, Stian Gerlach Sørensen, Dominik Stoffel

The reported research was supported in part by **BMBF** ZuSe (Scale4Edge), 16ME0122K-16ME0140+16ME0465, in part by **DFG** SPP Nano Security, KU 1051/11-2, in part by **Intel** Corporation (Scalable Assurance), and in part by **Siemens EDA**

Questions?

Contact me at:
alex.wezel@rptu.de