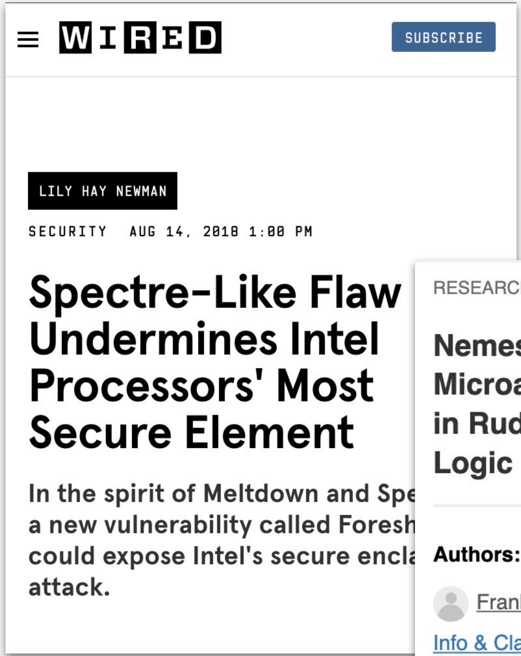
Exceptions Prove the Rule

Investigating and Resolving Residual Side Channels in Provably Secure Interrupt Handling

Matteo Busi, Pierpaolo Degano, Riccardo Focardi, Letterio Galletta, Flaminia Luccio, Frank Piessens, and Jo Van Bulck

PAVeTrust @ FM'24 - Milan, 9th Sept. 2024

Side-channel attacks on TEEs make computers, IoT, automotive, home appliances less secure



WIRED

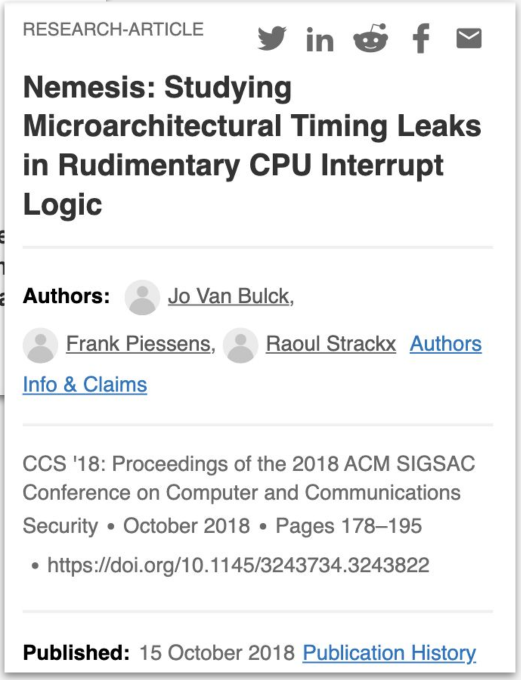
SUBSCRIBE

LILY HAY NEWMAN

SECURITY AUG 14, 2018 1:00 PM

Spectre-Like Flaw Undermines Intel Processors' Most Secure Element

In the spirit of Meltdown and Spe
a new vulnerability called Foresh
could expose Intel's secure encla
attack.



RESEARCH-ARTICLE

Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic

Authors: Jo Van Bulck, Frank Piessens, Raoul Strackx Authors Info & Claims

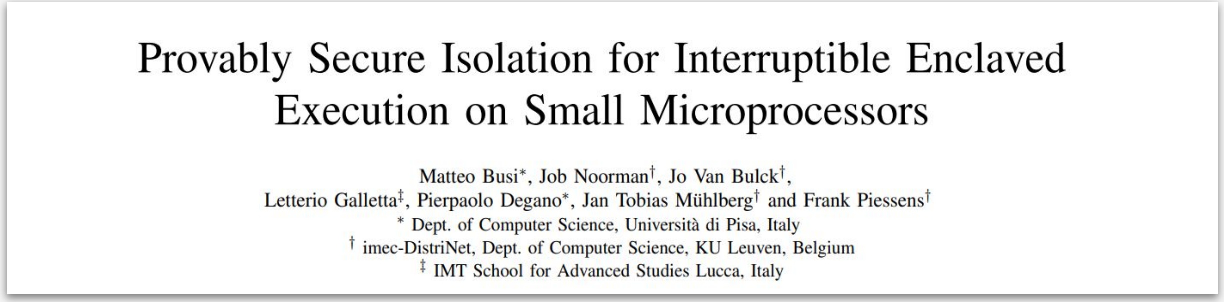
CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security • October 2018 • Pages 178–195

• https://doi.org/10.1145/3243734.3243822

Published: 15 October 2018 Publication History

A (short) revealing story

- **Sancus:** an embedded architecture with enclaves designed at KU Leuven
  - Enclaves are **trusted-execution environments (TEEs)**: separate areas of the processor providing protection to data and code
- **Sancus$_V$**
  - Proves that it is possible to implement interrupts in Sancus enclaves securely
  - **Big** manual effort in writing the model and doing all the proofs!



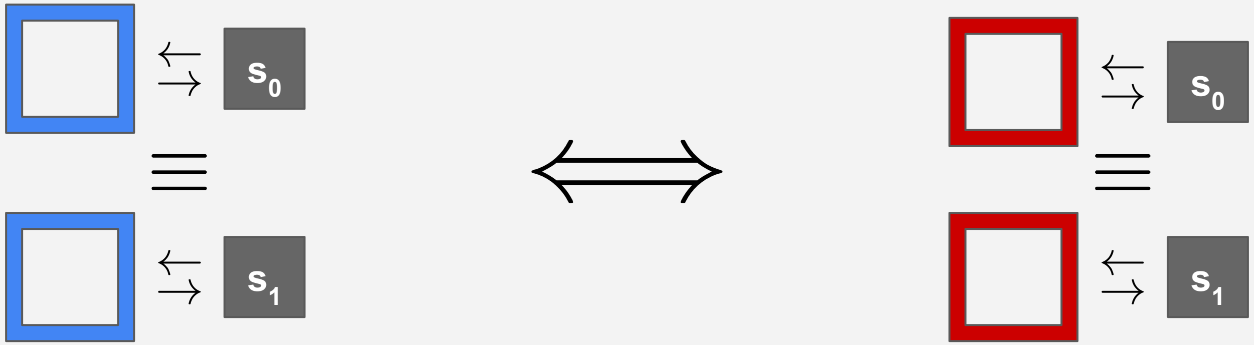
Provably Secure Isolation for Interruptible Enclaved Execution on Small Microprocessors

Matteo Busi[*], Job Noorman[†], Jo Van Bulck[†],
Letterio Galletta[‡], Pierpaolo Degano[*], Jan Tobias Mühlberg[†] and Frank Piessens[†]
[*] Dept. of Computer Science, Università di Pisa, Italy
[†] imec-DistriNet, Dept. of Computer Science, KU Leuven, Belgium
[‡] IMT School for Advanced Studies Lucca, Italy

[CSF'20]

Sancus_V

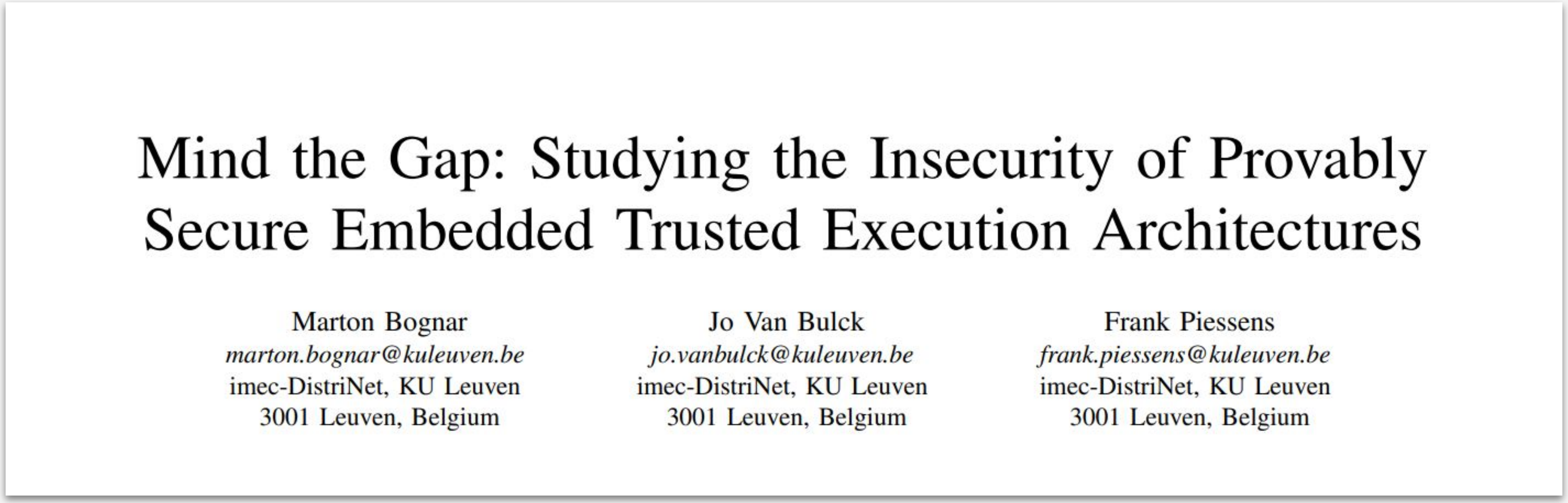
Sancus is secure **without** interrupts iff it is secure **after adding** them



(This is a full-abstraction result)

CARBER
VAC
YOU SURE ABOUT THAT?

We forgot about the gap!



Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures

Marton Bognar
marton.bognar@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

Jo Van Bulck
jo.vanbulck@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

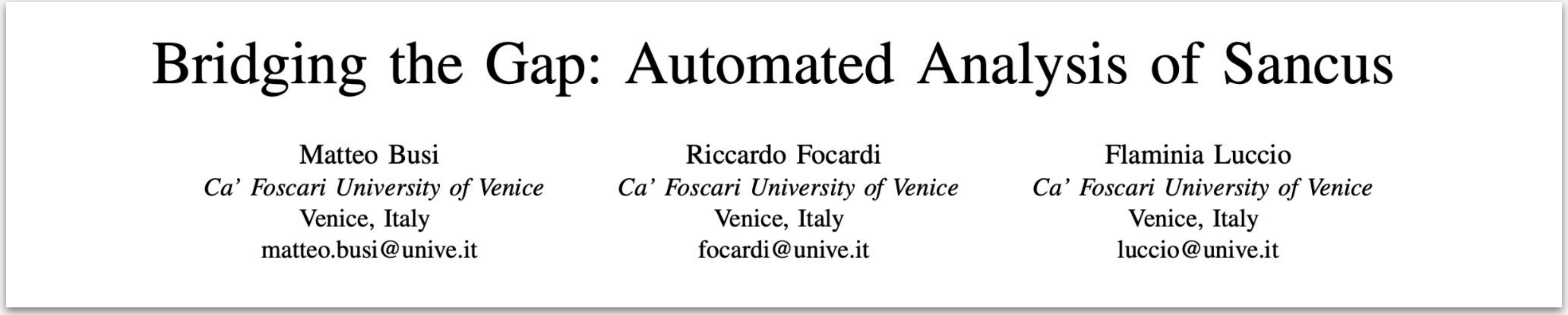
Frank Piessens
frank.piessens@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

[S&P'22]

How to bridge the gap?

ALVIE - https://github.com/matteobusi/alvie

- (Semi-)automated tool for analysing Sancus
- Three phases
  a. Specify attacker and victim capabilities
  b. **Automatically** build a formal model of the attacker/victim interaction on Sancus
  c. Look for side-channels on the model


Bridging the Gap: Automated Analysis of Sancus

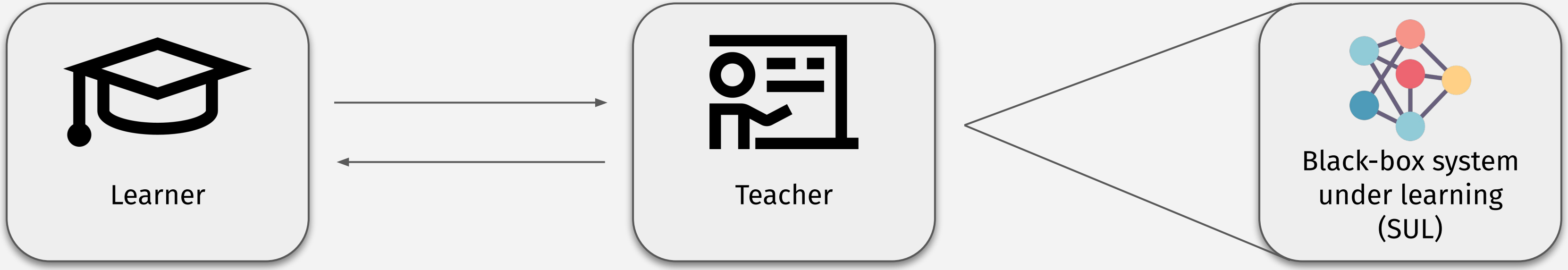
Matteo Busi
Ca' Foscari University of Venice
Venice, Italy
matteo.busi@unive.it

Riccardo Focardi
Ca' Foscari University of Venice
Venice, Italy
focardi@unive.it

Flaminia Luccio
Ca' Foscari University of Venice
Venice, Italy
luccio@unive.it

[CSF'24]

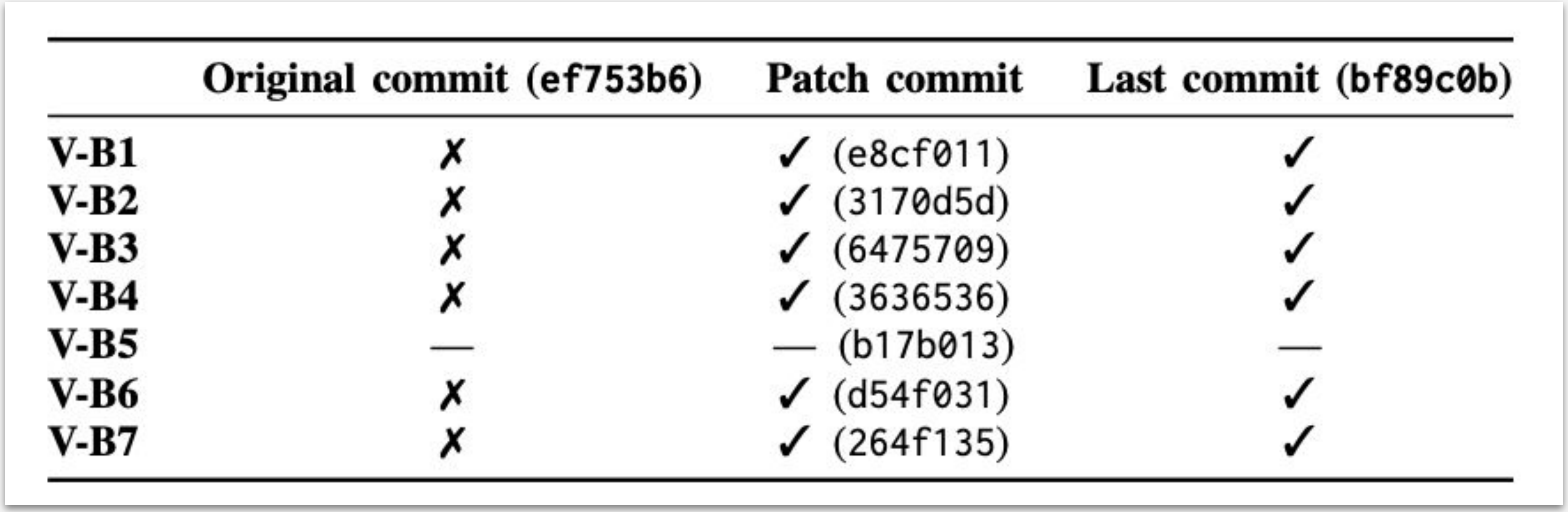
ALVIE: learning the model



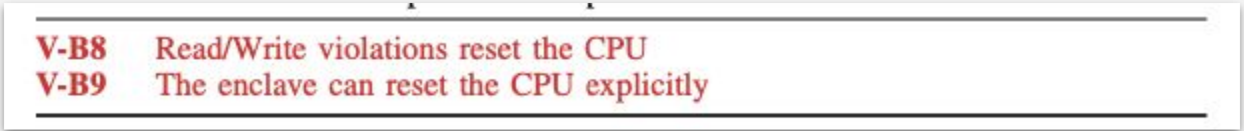
- **SUL (Sancus):** is an unknown DFA that we want to discover (call it *S*)
- **Learner:** tries to discover the unknown DFA
- **Teacher**: called a Minimally Adequate Teacher *[Angluin, 1987]*
  - **Answers** queries from the learner about the SUL
    - `member(s)` *iff* `s` *accepted by S*
    - `equiv(H)` *tells if* `H` *accepts the same language as S, or return a counterexample*

SUL Image from www.flaticon.com

AVLIE vs. Sancus$_V$



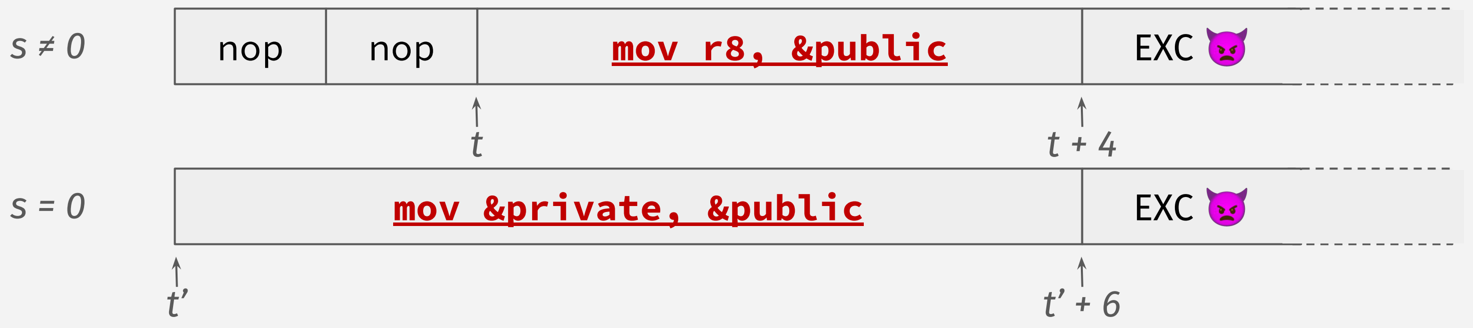
	Original commit (ef753b6)	Patch commit	Last commit (bf89c0b)
V-B1	✗	✓ (e8cf011)	✓
V-B2	✗	✓ (3170d5d)	✓
V-B3	✗	✓ (6475709)	✓
V-B4	✗	✓ (3636536)	✓
V-B5	—	— (b17b013)	—
V-B6	✗	✓ (d54f031)	✓
V-B7	✗	✓ (264f135)	✓



V-B8	Read/Write violations reset the CPU
V-B9	The enclave can reset the CPU explicitly

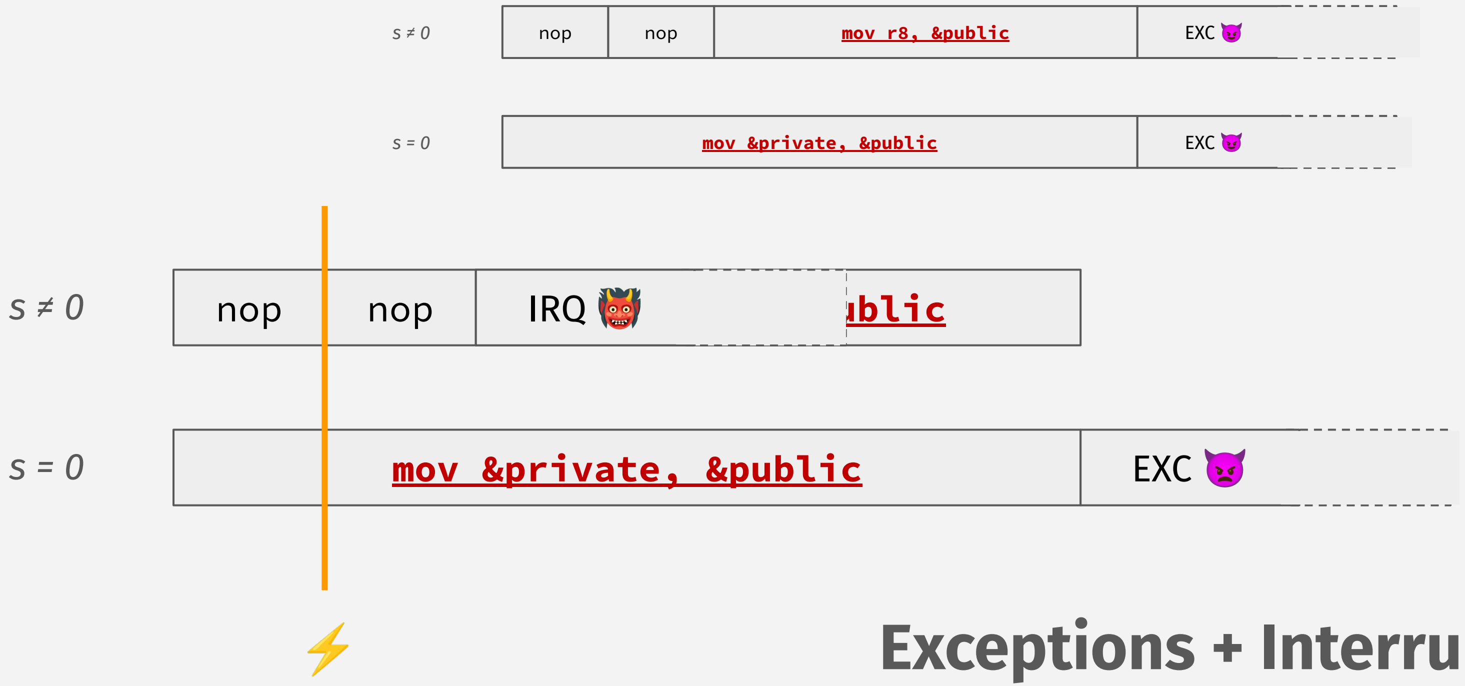
Let’s focus on V-B8

- Upon exception, the CPU executes an **attacker-defined** exception handler
    - If offending instruction *i* starts at cycle *t*, exception handler starts at *t+cycles(i)*
    - Is this a problem?



Exceptions alone won’t leak *s***!**

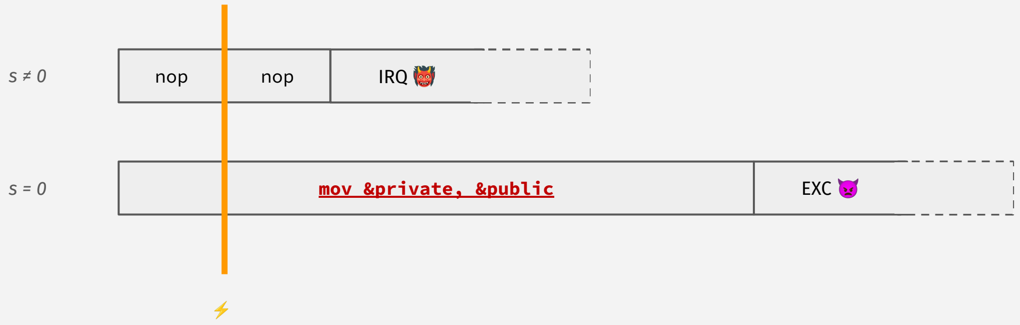
What about interrupts?



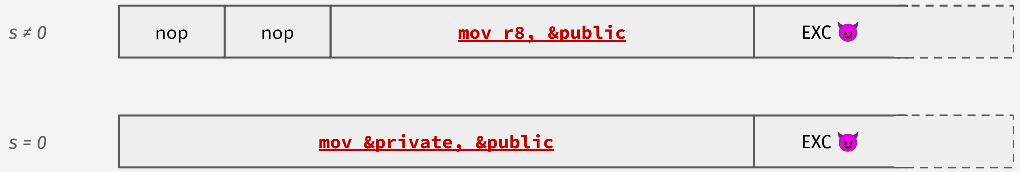
Exceptions + Interrupts leak *s*!

What does it mean for the model?

There exists an **insecure** execution in the interruptible Sancus...



...which was secure in the non-interruptible Sancus*



* To be precise we should prove that this attack has no counterpart in the non-interruptible Sancus.

Recovering full abstraction

- **Minimal change:** make the non-interruptible execution **insecure**!

(CPU-Violation-PM)

$$\frac{\mathcal{B} \neq \langle \bot, \bot, t_{pad} \rangle \qquad i, \mathcal{R}, pc_{old}, \mathcal{B} \nvdash_{mac} \mathsf{OK}}{\mathcal{D} \vdash \langle \delta, t, t_a, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle \to \mathrm{EXC}_{\langle \delta, \boxed{t+cycles(i)}, t_a, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle}} \quad i = decode(\mathcal{M}, \mathcal{R}[\mathsf{pc}]) \neq \bot$$

This "controls" the time in the model, must be changed...

(CPU-Violation-PM)

$$\frac{\mathcal{B} \neq \langle \bot, \bot, t_{pad} \rangle \qquad i, \mathcal{R}, pc_{old}, \mathcal{B} \nvdash_{mac} \mathsf{OK}}{\mathcal{D} \vdash \langle \delta, t, t_a, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle \to \mathrm{EXC}_{\langle \delta, \boxed{t}, t_a, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle}} \quad i = decode(\mathcal{M}, \mathcal{R}[\mathsf{pc}]) \neq \bot$$

Implementing the fix

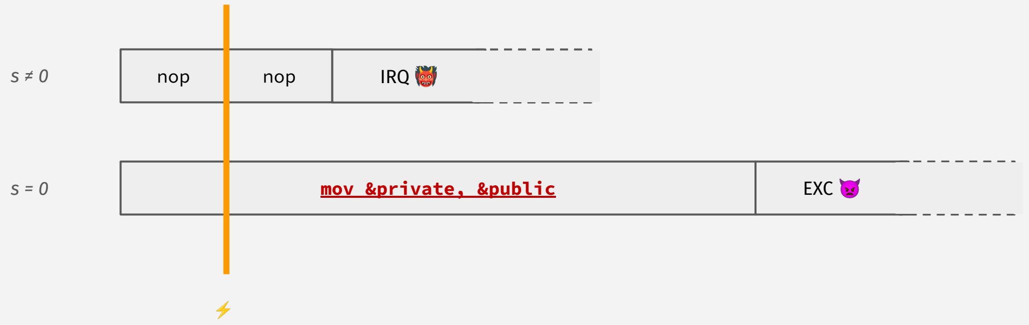
- **New rule:** the CPU must detect exceptions **before** execution
  - Requires non-trivial changes in the Sancus$_V$ implementation!
- **Solution:**
  - Make the time between the start of the offending instruction and the start of the exception state is a constant (e.g., `MAX_TIME`)

(CPU-Violation-PM)

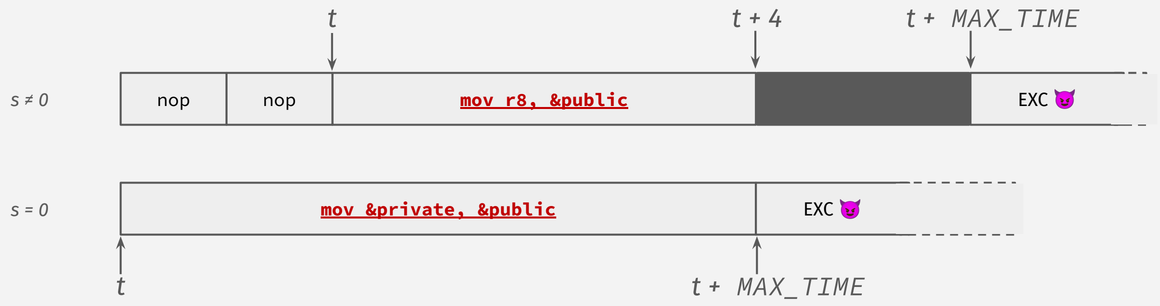
$$\frac{\mathcal{B} \neq \langle \bot, \bot, t_{pad} \rangle \qquad i, \mathcal{R}, pc_{old}, \mathcal{B} \nvdash_{mac} \mathsf{OK}}{\mathcal{D} \vdash \langle \delta, t, t_a, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle \rightarrow \mathrm{EXC}_{\langle \delta, t + \boxed{\mathtt{MAX\_TIME}}, \mathcal{M}, \mathcal{R}, pc_{old}, \mathcal{B} \rangle}} \quad i = decode(\mathcal{M}, \mathcal{R}[\mathsf{pc}]) \neq \bot$$

The fix at work

The **insecure** execution in the interruptible Sancus...



...was already **insecure** in the non-interruptible Sancus*



* This is not a proof! For that we need to rework part of the original development.

Conclusions

- We identified a novel full abstraction breach in Sancus$_V$
  - Similar attack tactics were previously used to attack Intel SGX
- We proposed a minimal fix to the model and implementation to recover full abstraction
- Take aways:
  - Formal models are **important**
  - The gap between the model and the implementation **must** be as small as possible
  - Models should be developed with tools support
    - e.g., ALVIE for automated model extraction and verification
    - Proof assistants for model development and proof mechanization

THE

END

Exceptions Prove the Rule

Investigating and Resolving Residual Side Channels in Provably Secure Interrupt Handling

Matteo Busi, Pierpaolo Degano, Riccardo Focardi, Letterio Galletta, Flaminia Luccio, Frank Piessens, and Jo Van Bulck

PAVeTrust @ FM’24 - Milan, 9th Sept. 2024