

Mitigating interrupt-driven attacks against enclaved execution

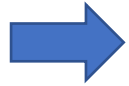
Frank Piessens

Journées Nationales 2023 du GDR Sécurité Informatique

27/06/2023 Paris, France

(This talk reports on a line of work done in collaboration with many people, see references on the slides)

Overview



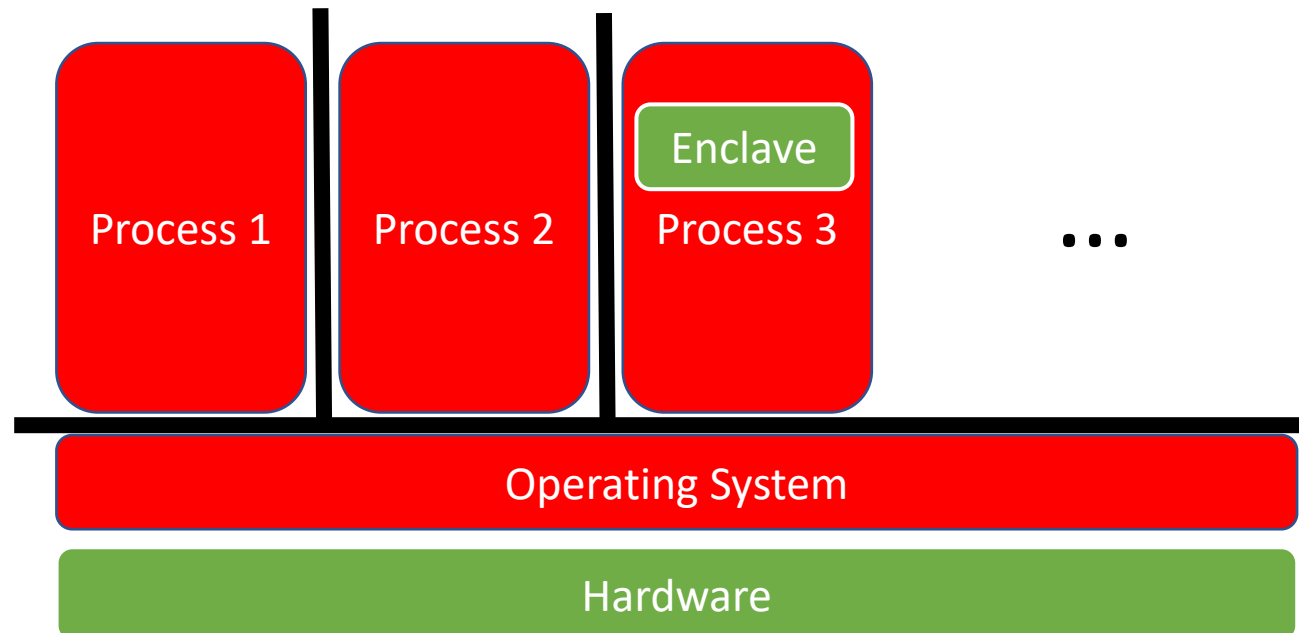
- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
- Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
- A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- Mitigations for embedded microprocessors
- Conclusions

Trusted Execution Environments (TEEs)

- Standard operating systems can not be trusted for all use cases:
 - Too big and too open to provide strong security assurance
 - Tens of millions of lines of operating system code
 - A very open eco-system of applications
 - No shielding from the computation platform provider
 - No protection of “data in use” on a public cloud, no “confidential computing”, ...
- Hence, many computing platforms provide some notion of “trusted execution environment” (TEE)
 - Arm TrustZone
 - Intel SGX, Intel TDX
 - AMD SEV
 - Many research prototypes
 - ...

TEE designs and Enclaved Execution

- Many designs for TEEs exist:
 - Physically separate design, like a smartcard
 - Two-world design, like ARM TrustZone
 - Shielded VM design, like AMD SEV and Intel TDX
 - **Enclaved execution design**, like Intel SGX

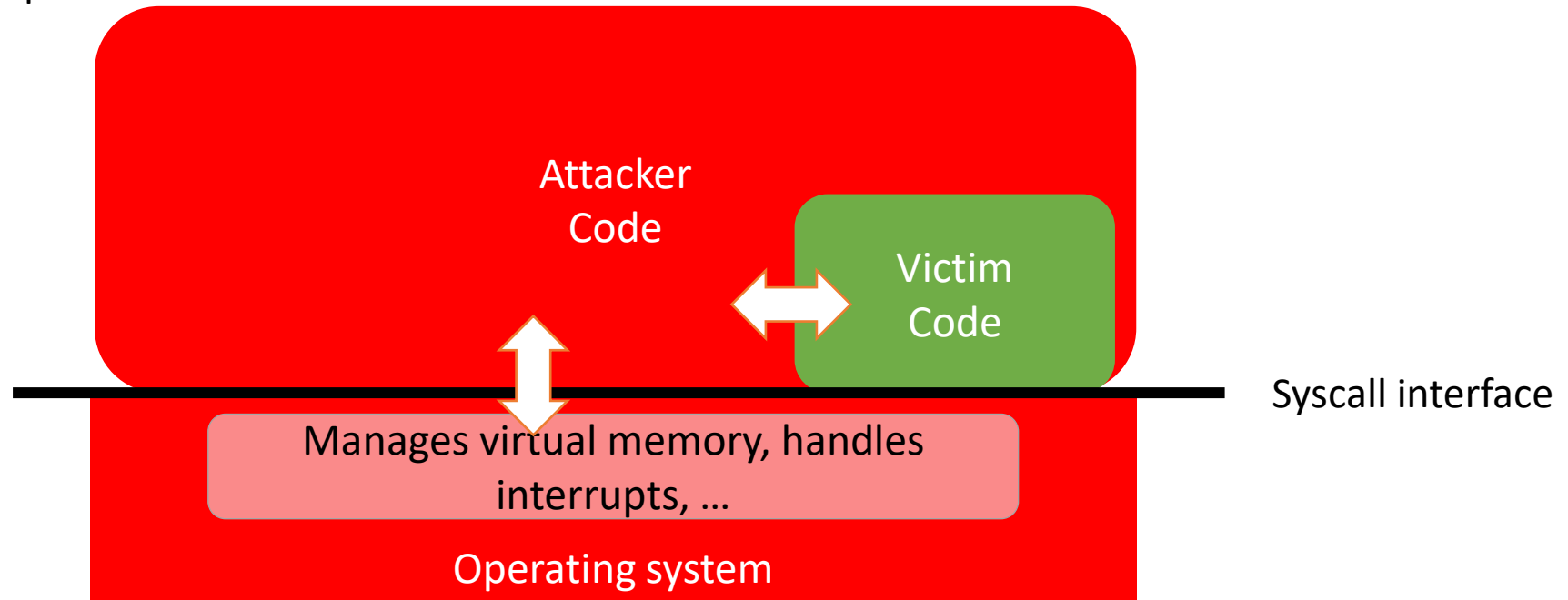


Attacking enclaves

- Enclaves face a very powerful attacker model:
 - both the surrounding process and the system software are attacker-controlled
- Therefore, there is a very rich literature on attack techniques
 - Interface attacks
 - Van Bulck et al., **A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes**, ACM CCS 2019
 - Microarchitectural attacks
 - Side channel attacks, like cache attacks
 - Transient execution attacks, like Spectre and Foreshadow
 - **Controlled channel attacks**

Controlled-channel attacks

- Victim code is shielded from system software, but still relies on system software for resource management
 - Important attack vectors:
 - Page handling
 - Interrupts



Seminal paper introducing controlled channel attacks

- Y. Xu, W. Cui and M. Peinado, **Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems**, IEEE S&P 2015
- Example attack from their paper:

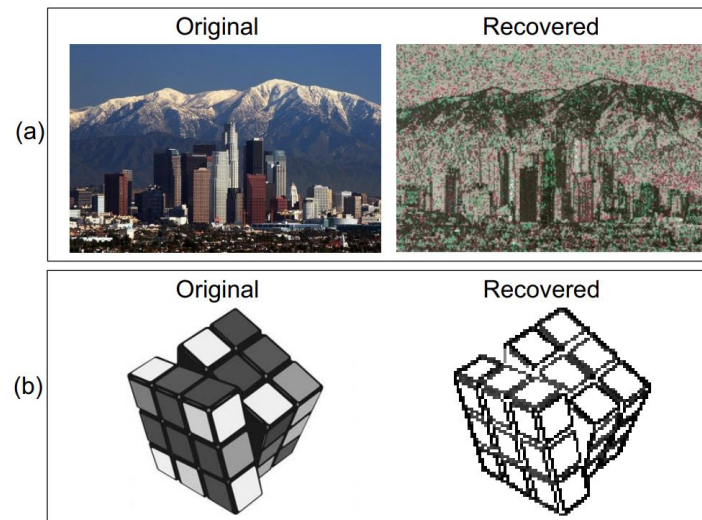



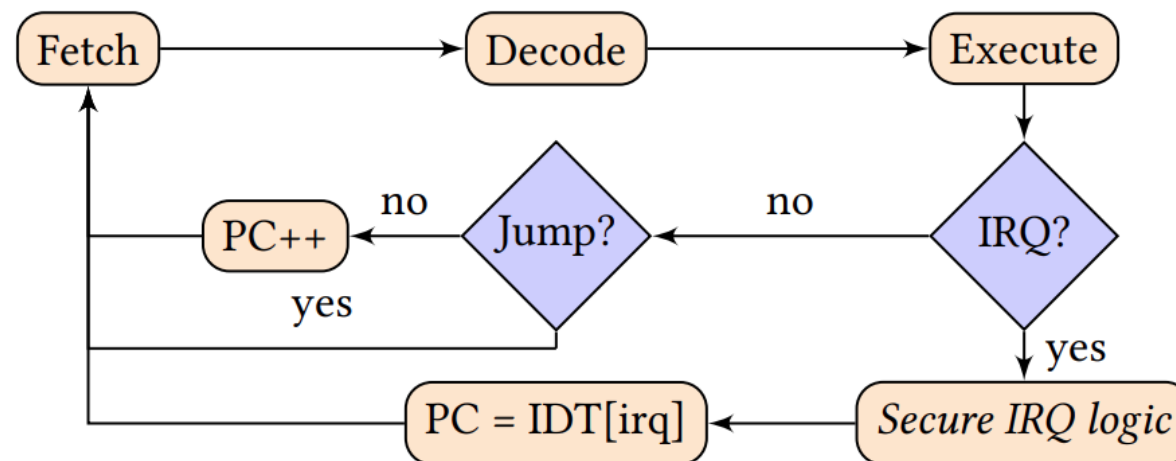
Fig. 11: A small sample of the images we used to test the libjpeg attack.

Overview

- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
-  • Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
- A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- Mitigations for embedded microprocessors
- Conclusions

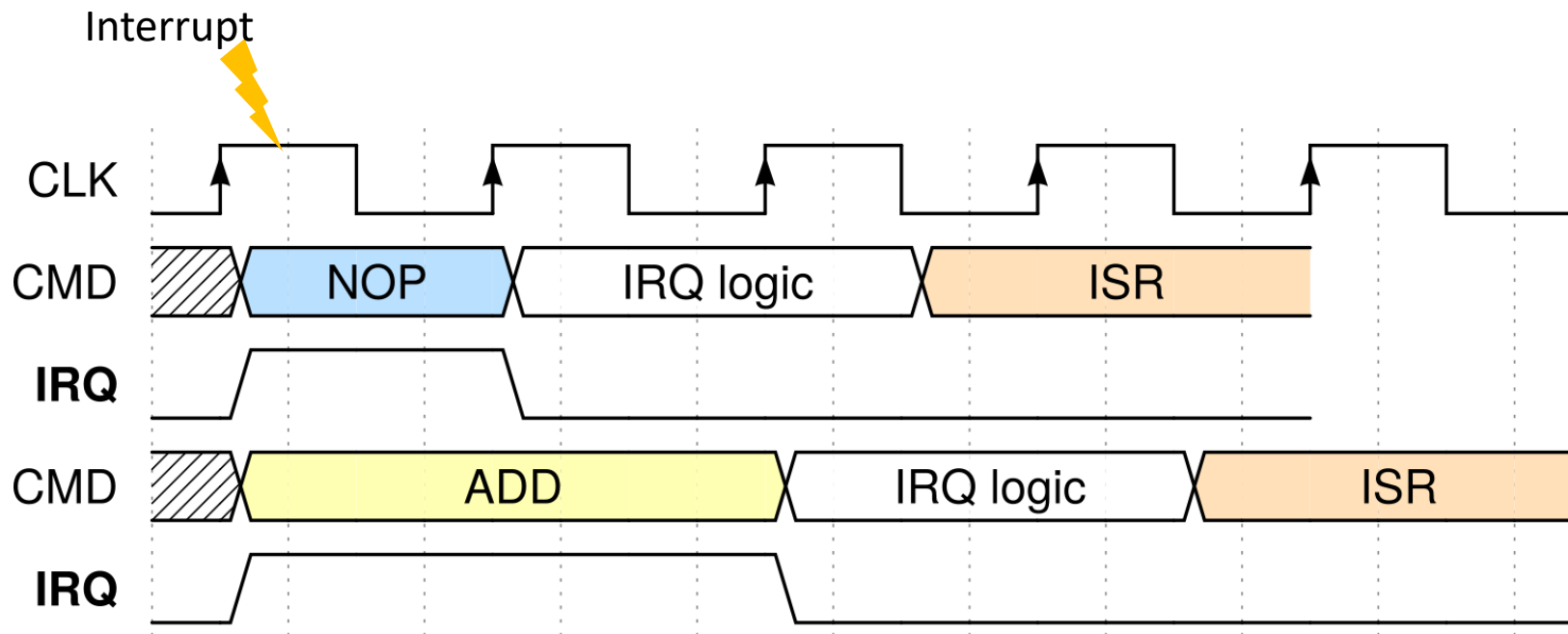
Interrupts and interrupt-driven attacks

- Most processors support **interrupts**: a hardware mechanism to handle events that occur asynchronously to the current instruction stream
- Enclaves can also be interrupted, and are **interrupt-unaware** (but the hardware will save and clear registers that might contain sensitive information)
- This enables a powerful controlled channel attack, first described in:
 - Jo Van Bulck, Frank Piessens, Raoul Strackx, **Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic**, ACM CCS 2018



On embedded enclaves

- Interrupts can be timed cycle-accurately and timing is deterministic
 - Hence, a deterministic controlled channel to leak information on control flow

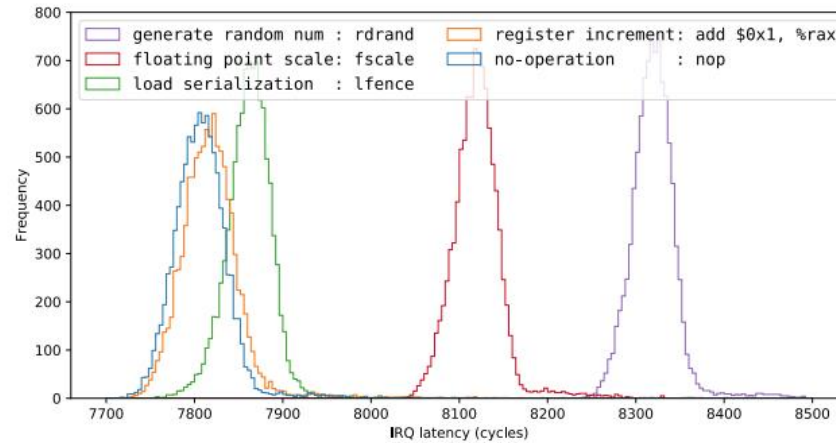


```
...  
if secret {  
  NOP; NOP // 2 x 1 cycle  
}  
else {  
  ADD @R5+,R6 // 2 cycles  
}  
...
```

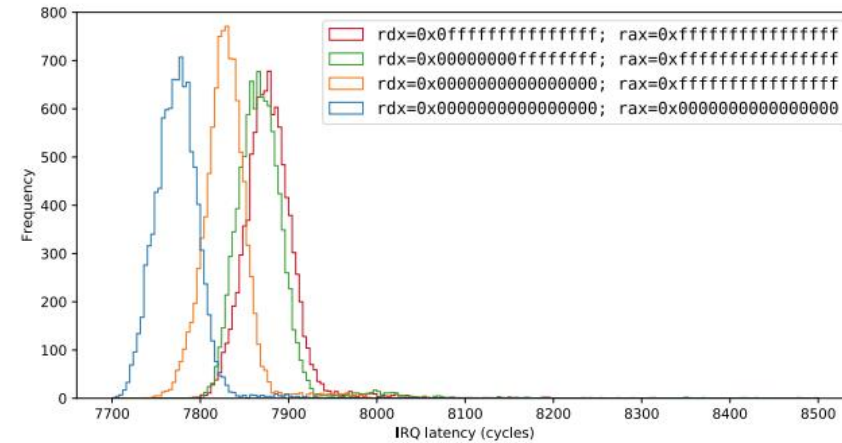
On cloud enclaves

- Interrupt-driven attacks seem much harder:
 - Interrupts can **not** be scheduled cycle-accurately
 - Execution time of instructions is non-deterministic and noisy
- Yet, for Intel SGX enclaves, it has been shown to be a very useful attack primitive
 - The SGX-Step attack framework:
 - <https://github.com/jovanbulck/sgx-step>
 - Used in 30+ follow-up projects that build attacks using SGX-Step
 - Attacks include:
 - Interrupt latency measurements
 - Interrupt counting (single-stepping)
 - Side-channel measurements with high temporal resolution
 - Zero stepping

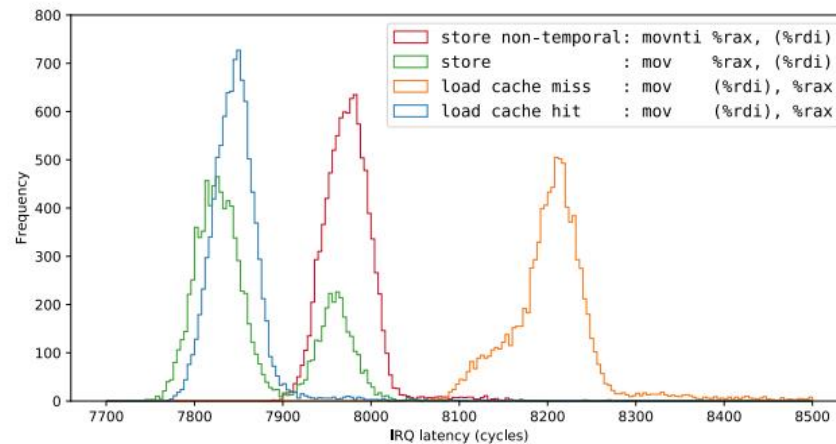
Interrupt latency attack



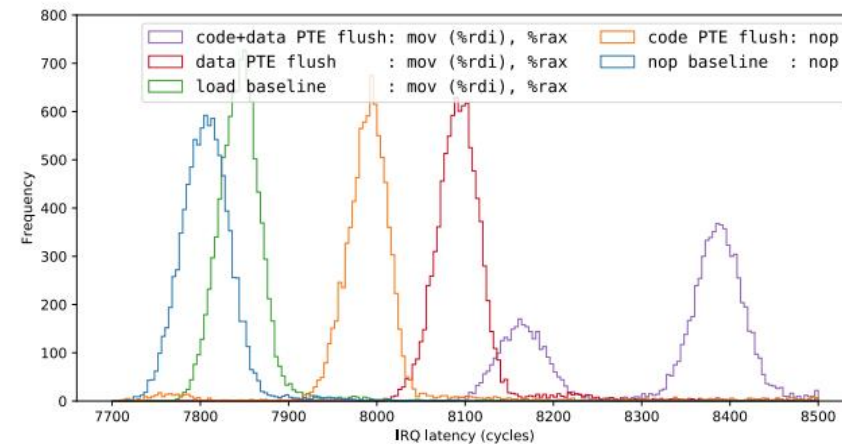
(a) IRQ latency distributions for selected x86 instructions.



(b) Data-dependent IRQ latencies for the x86 div instruction.



(c) Increased IRQ latencies from enclave private memory caching conditions.



(d) Increased IRQ latencies from unprotected PTE data cache misses.

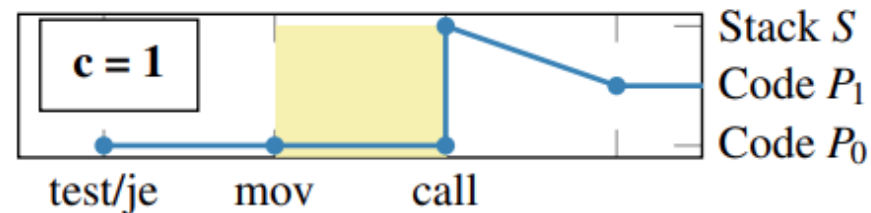
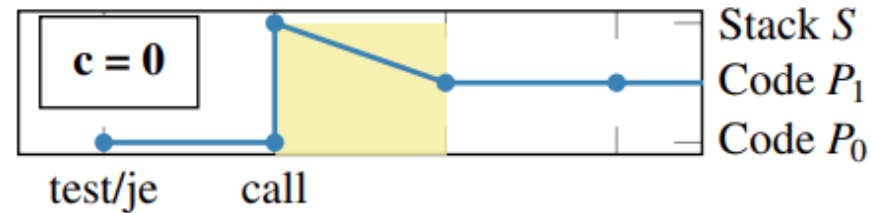
Figure 6: SGX microbenchmarks: IRQ latency distribution timing variability based on (a) enclave instruction type, (b) secret input operands, (c) enclave private memory caching conditions, and (d) untrusted address translation data cache misses.

Interrupt counting attack

- Even if the latency signal is too noisy, just **counting** of instructions by single-stepping through frequent interrupts leaks control flow
 - Daniel Moghimi, Jo Van Bulck, Nadia Heninger, Frank Piessens, Berk Sunar, **CopyCat: Controlled Instruction-Level Attacks on Enclaves**, Usenix Security 2020

```
if (c == 0) { r = add(r, d); } else { r = add(r, s); }
```

```
test %eax,%eax
je 1f
mov %edx,%esi
1:
call add
mov %eax,-0xc(%rbp)
```



Amplification of other side-channels

- Frequently interrupting the enclave, after each step (or even multiple times at the same execution point – zero-stepping), makes it possible to do other microarchitectural side channel measurements at maximal temporal resolution
 - Cache attacks
 - Branch predictor attacks
 - Power attacks
 - ...

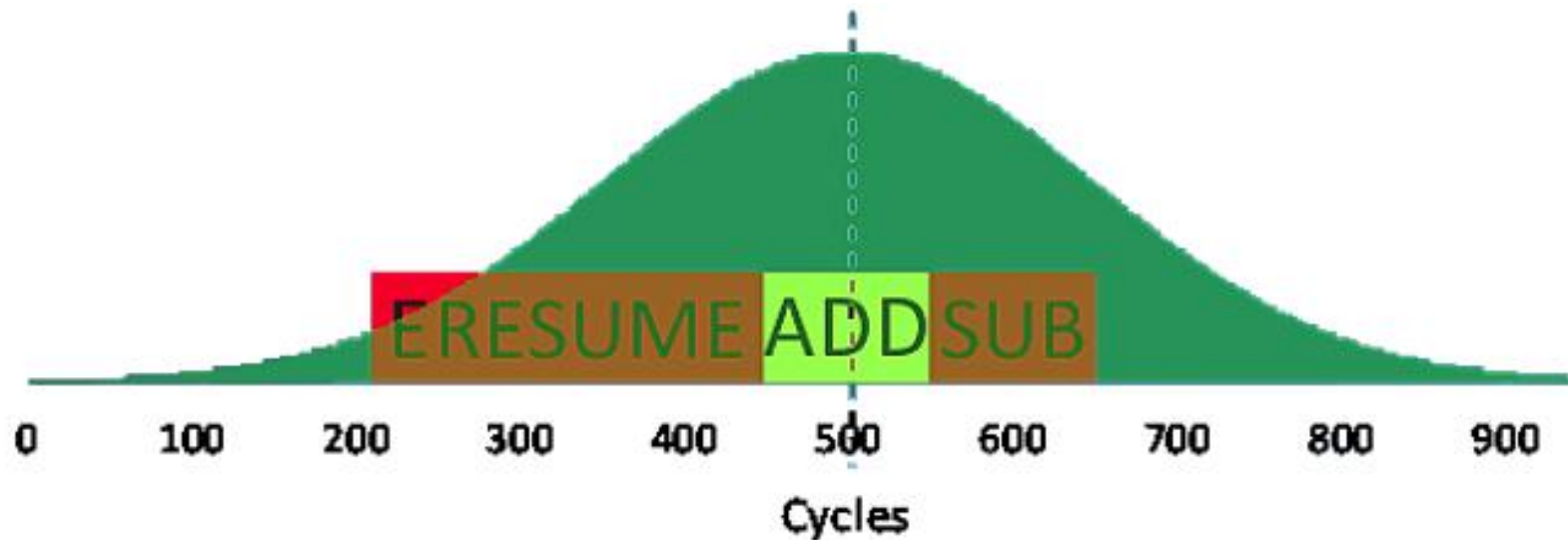
Overview

- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
- Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
- A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- Mitigations for embedded microprocessors
- Conclusions

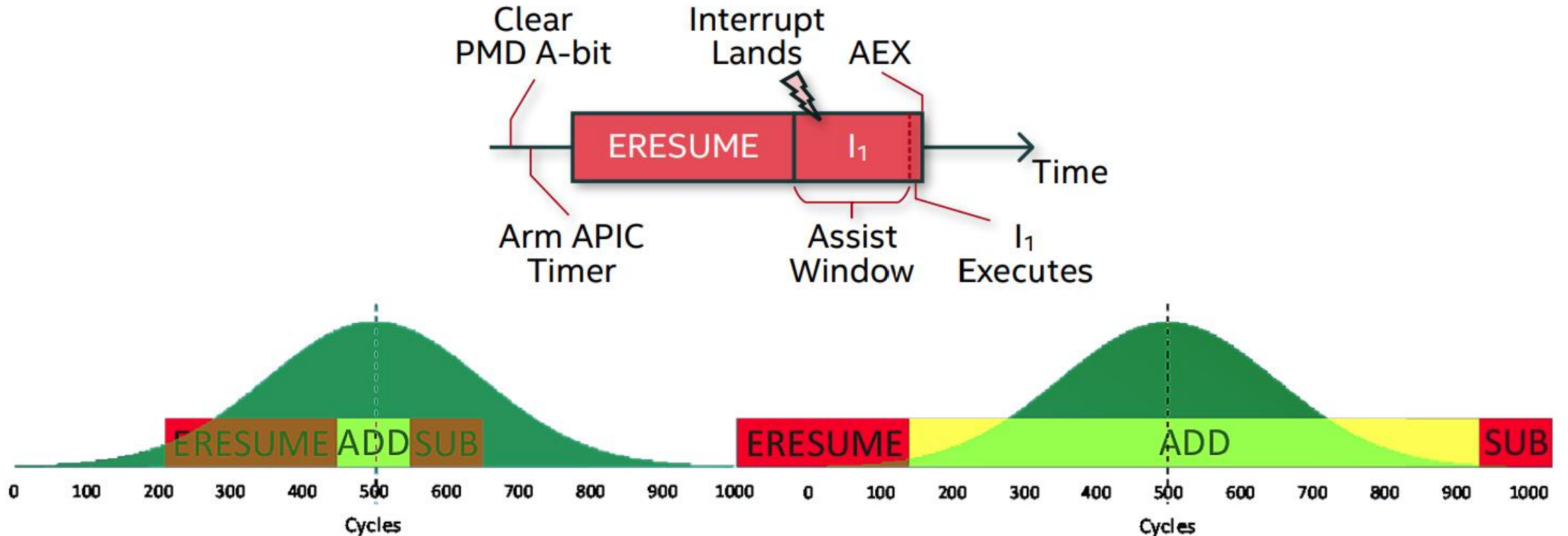


Analysis of the attack on Intel SGX


- It is **surprising** that interrupt-driven attacks work so well on Intel SGX
 - Scheduling of interrupts is noisy
 - Instruction to resume enclave after interrupt is complex with unpredictable execution time



Why single-stepping works so well



Overview

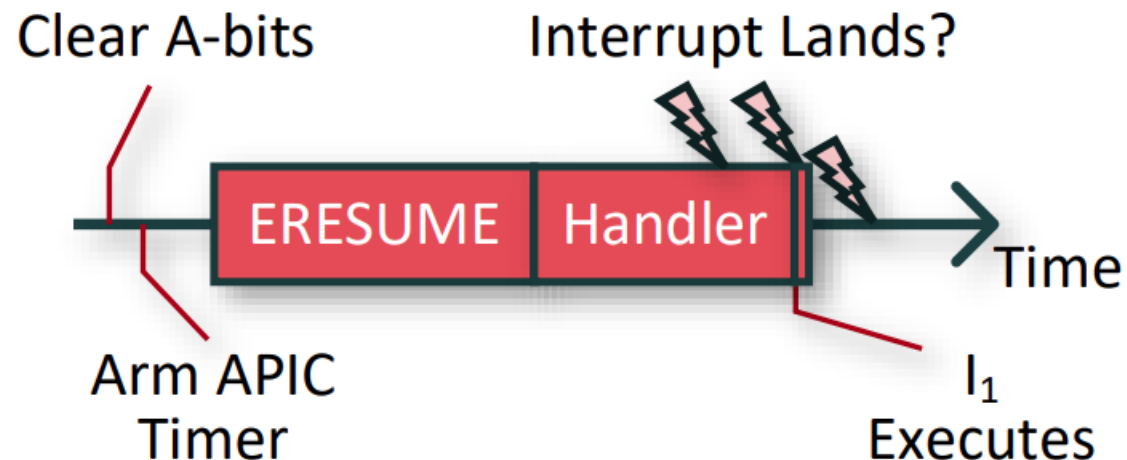
- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
- Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
-  • A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- Mitigations for embedded microprocessors
- Conclusions

A hardware/software codesign for Intel SGX

- Intel will be rolling out a principled but probabilistic mitigation later this year:
 - S. Constable, J. Van Bulck, X. Cheng, Y. Xiao, C. Xing, I. Alexandrovich, T. Kim, F. Piessens, M. Vij, M. Silberstein. **AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves**, Usenix Security 2023
- Objectives of the mitigation:
 - Obfuscated forward progress: **no (reliable) single-stepping**
 - Never **increase** the amount of information leaked
 - In addition, it must be **compatible** with existing software and **practical**

The key idea of the mitigation

- On the hardware side: the AEX-Notify ISA extension
 - Allows an enclave to “opt in” to (guaranteed) interrupt notifications
 - After an interrupt, before resuming, the enclave can run an interrupt handler
- On the software side: a software handler that minimizes execution time of the first enclave instruction



Responsibilities of the handler

- Determine what the next instruction is
 - Implemented as a **constant-time** instruction disassembler
- Verify page-table permissions
- Atomically prefetch the working set of the next instruction
- Randomly insert a small delay with 50% probability

Evaluation of effectiveness

Table 1: Single-stepping success rates for different stimuli.

Adversary Action(s)	Single-Step Hit Rate
None	0.042
Clear PTE A-bit	0.107
L1 contention (page/set)	0.118/0.112
L2 contention (page/different set/matching set)	0.114/0.126/0.223
L3 contention (same/separate/all cores)	0.141/0.030/0.104

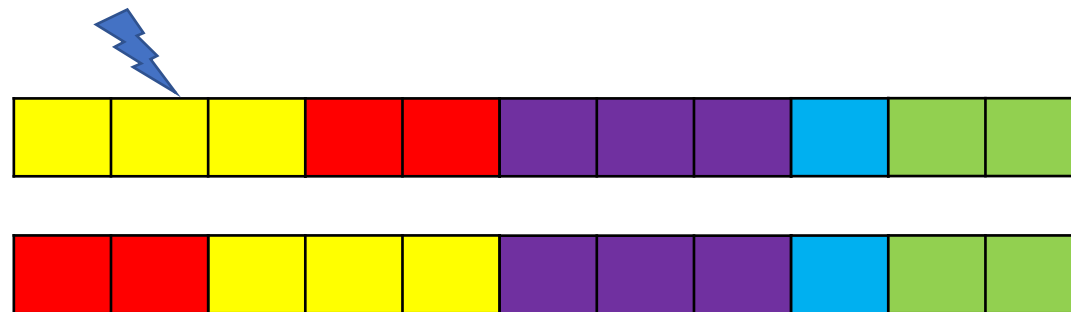
NOTE: attacks require several consecutive single-step successes!

Overview

- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
- Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
- A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- ➔ • Mitigations for embedded microprocessors
- Conclusions

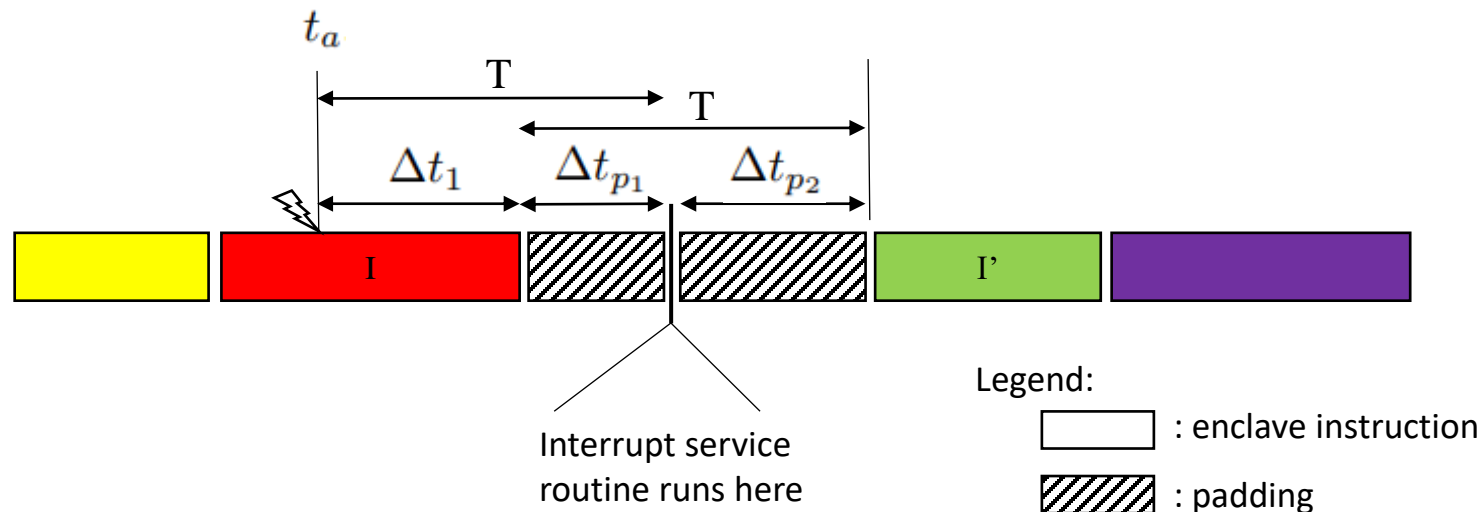
Full mitigation on embedded processors

- Instruction execution time is deterministic, and attacker can schedule interrupts cycle-accurately
- But no other software-exploitable side-channels are available
- IDEA: can we **pad** interrupt handling time?
 - Not secure! Attacker can measure:
 - Interrupt latency time
 - “Resume-to-end” time
 - Maximal number of interrupts (interrupt counting)



A provably secure design

- Busi et al. “Provably secure isolation for interruptible enclaved execution on small microprocessors” (IEEE CSF 2020) proposes:
 - **Pre- and post-padding** during interrupt handling such that:
 - Interrupt latency is constant (T)
 - Resume-to-end-time does not change on interrupt



Overview

- Introduction
 - Trusted Execution Environments (TEEs) and enclaved execution
 - Attacking enclaves
- Interrupt-driven attacks against enclaves
 - Example attacks against embedded and cloud enclaves
 - Analysis of the precise interruption of Intel SGX enclaves
- A HW/SW codesign to mitigate interrupt-driven attacks against SGX
- Mitigations for embedded microprocessors
- Conclusions



Conclusions

- Interrupt-driven attacks are a powerful class of attacks against enclaved execution
 - See: <https://github.com/jovanbulck/sgx-step>
- Attacks and defenses look different for low-end or high-end microprocessors
- Intel is rolling out a hardware-software co-designed mitigation that successfully thwarts single-stepping through interrupts
 - The hardware part (AEX-Notify ISA extension) makes enclaves interrupt aware, and can be used for other mitigations
 - The software part provides an ingenious interrupt handler that atomically prefetches the working set of the next enclave instruction

References

- Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D. Garcia, and Frank Piessens. **A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes**. ACM CCS 2019
- Y. Xu, W. Cui and M. Peinado, **Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems**, IEEE S&P 2015
- Jo Van Bulck, Frank Piessens, Raoul Strackx, **Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic**, ACM CCS 2018
- Daniel Moghimi, Jo Van Bulck, Nadia Heninger, Frank Piessens, Berk Sunar, **CopyCat: Controlled Instruction-Level Attacks on Enclaves**, Usenix Security 2020
- S. Constable, J. Van Bulck, X. Cheng, Y. Xiao, C. Xing, I. Alexandrovich, T. Kim, F. Piessens, M. Vij, M. Silberstein. **AEX-Notify: Thwarting Precise Single-Stepping Attacks through Interrupt Awareness for Intel SGX Enclaves**, Usenix Security 2023
- Matteo Busi, Job Noorman, Jo Van Bulck, Letterio Galletta, Pierpaolo Degano, Jan Tobias Mühlberg, Frank Piessens, **Provably secure isolation for interruptible enclaved execution on small microprocessors**. IEEE CSF 2020
- Matteo Busi, Job Noorman, Jo Van Bulck, Letterio Galletta, Pierpaolo Degano, Jan Tobias Mühlberg, and Frank Piessens. **Securing Interruptible Enclaved Execution on Small Microprocessors**. ACM TOPLAS 2021