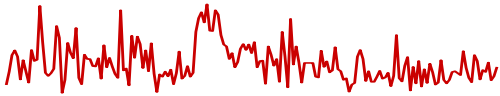


Microarchitectural Side-Channel Attacks for Privileged Software Adversaries

A review & perspective

Jo Van Bulck

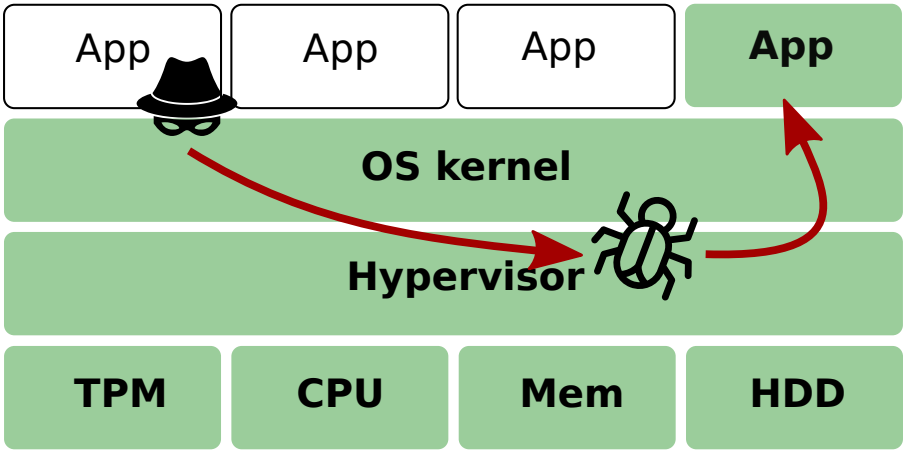
CIF Review, Leuven, October 28, 2021

imec-DistriNet, KU Leuven jo.vanbulck@cs.kuleuven.be

DistriNet

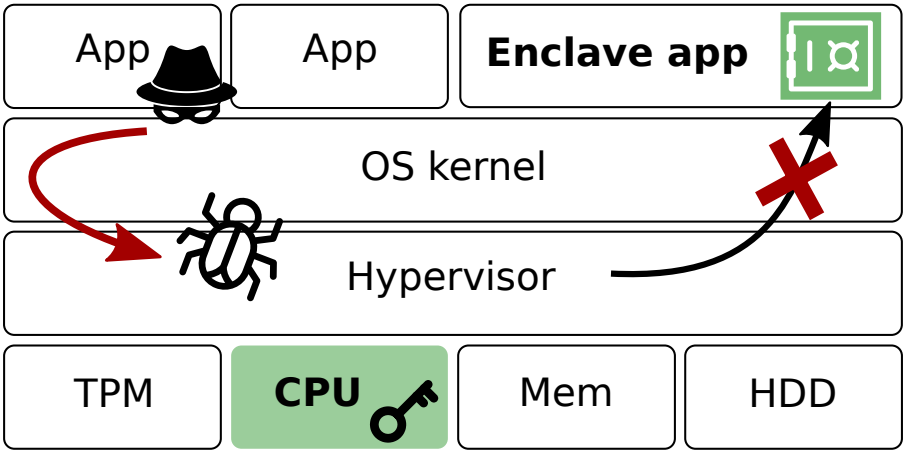
KU LEUVEN

The big picture: Enclaved execution attack surface



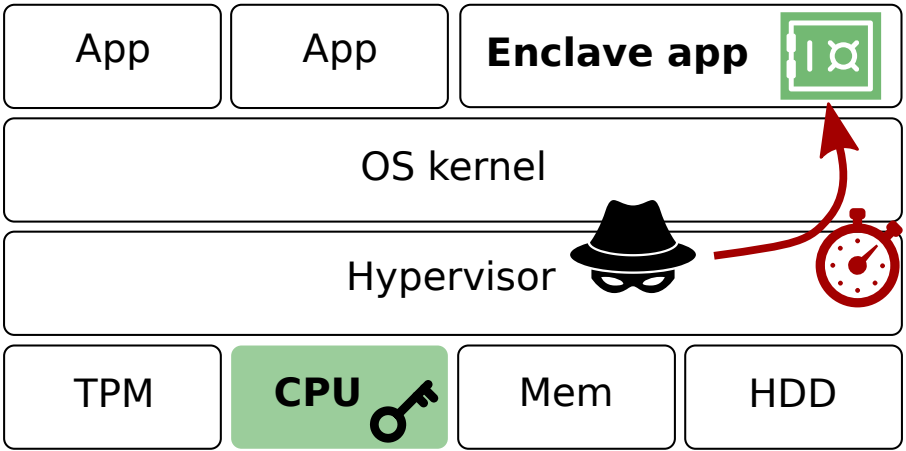
Traditional **layered designs:** large trusted computing base

The big picture: Enclaved execution attack surface



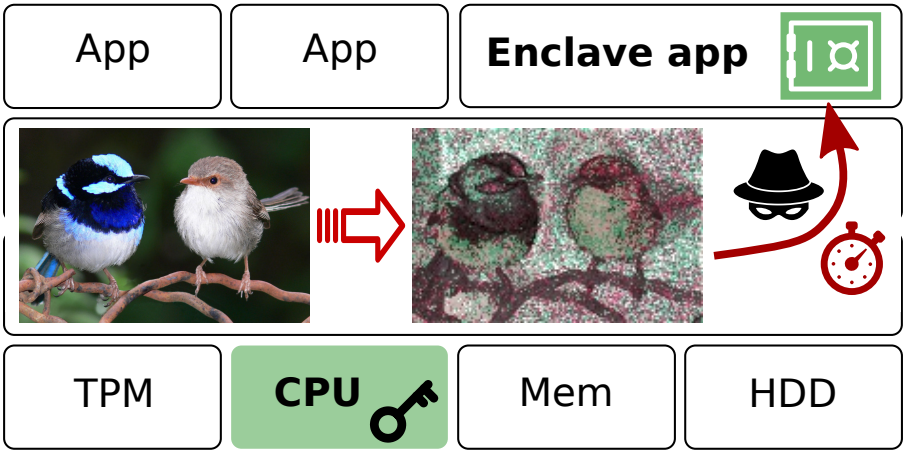
Intel SGX promise: hardware-level **isolation and attestation**

The big picture: Privileged side-channel attacks



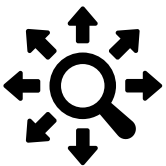
Game-changer: Untrusted OS → new class of powerful side channels!

The big picture: Privileged side-channel attacks



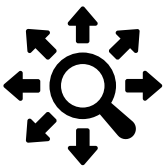
Xu et al. "Controlled-channel attacks: Deterministic side channels for untrusted operating systems", IEEE S&P 2015

Research agenda: Understanding privileged side-channel attacks



1. **Which** novel privileged side channels exist?
2. **How** well can they be exploited in practice?
3. **What** can be leaked?

Research agenda: Understanding privileged side-channel attacks



1. **Which** novel privileged side channels exist?
   → We uncover previously unknown attack avenues

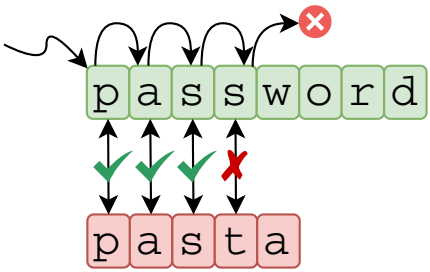
2. **How** well can they be exploited in practice?
   → We develop new techniques and practical attack frameworks

3. **What** can be leaked?
   → We recover metadata *and* data

Idea 1: Privileged interrupts for side-channel amplification

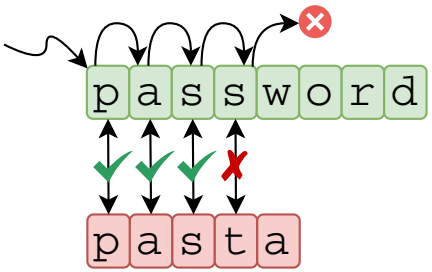
Case study: Comparing a secret password

```
1 void check_pwd(char *input)
2 {
3     for (int i=0; i < PWD_LEN; i++)
4       if (input[i] != pwd[i])
5           return 0;
6
7     return 1;
8 }
```



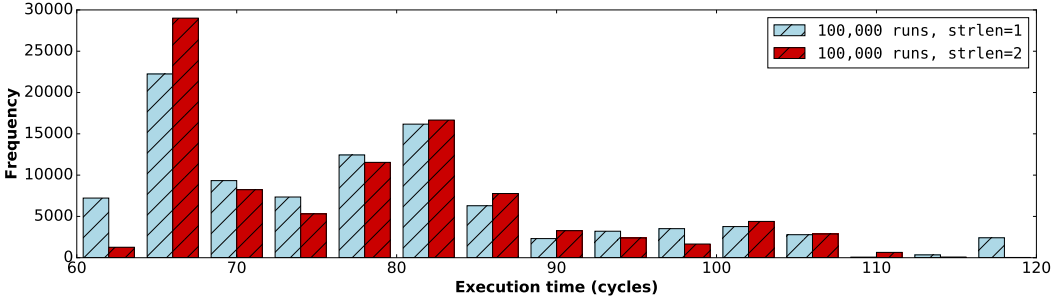
Case study: Comparing a secret password

```
1 void check_pwd(char *input)
2 {
3     for (int i=0; i < PWD_LEN; i++)
4         if (input[i] != pwd[i])
5             return 0;
6
7     return 1;
8 }
```



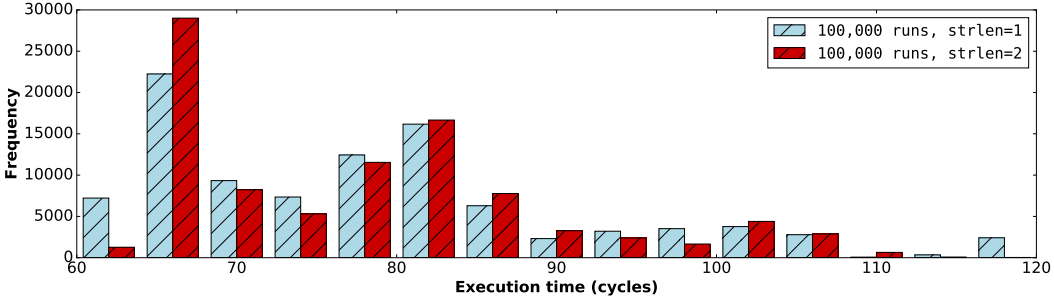
Overall execution time reveals correctness of individual password bytes!

Building the side-channel oracle with execution timing?



Building the side-channel oracle with execution timing?

Too noisy: modern x86 processors are lightning fast...

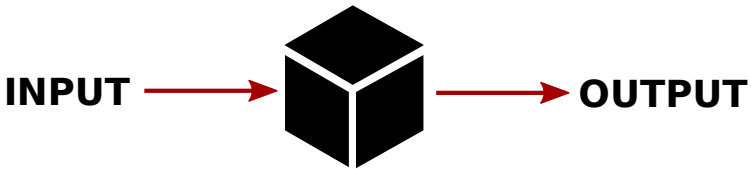


SGX-Step: Executing enclaves one instruction at a time

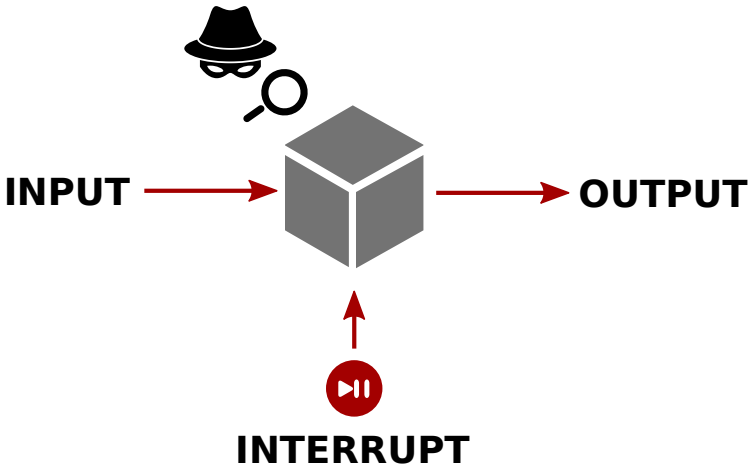
https://github.com/jovanbulck/sgx-step

Unwatch 27 | Star 312 | Fork 63

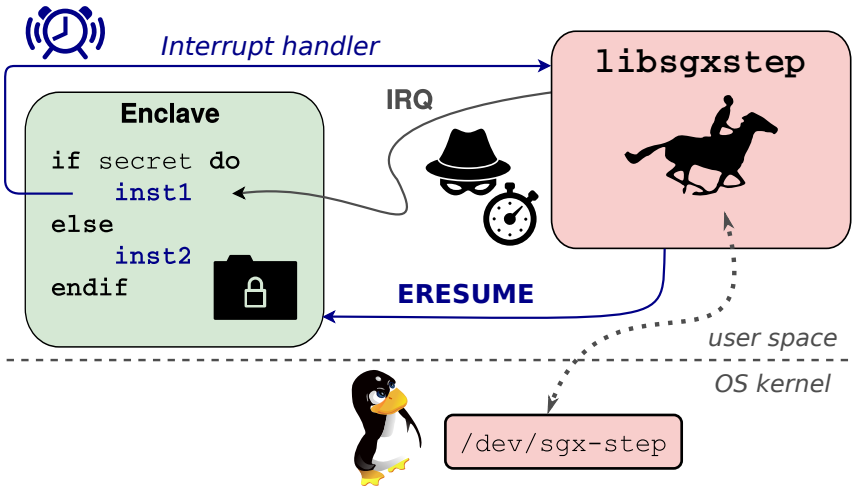
SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time



Van Bulck et al. "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control", SysTEX 2017.

Demo: Building a deterministic password oracle with SGX-Step

```
[idt.c] DTR.base=0xfffffe0000000000/size=4095 (256 entries)
[idt.c] established user space IDT mapping at 0x7f7ff8e9a000
[idt.c] installed asm IRQ handler at 10:0x56312d19b000
[idt.c] IDT[ 45] @0x7f7ff8e9a2d0 = 0x56312d19b000 (seg sel 0x10); p=1; dpl=3; type=14; ist=0
[file.c] reading buffer from '/dev/cpu/1/msr' (size=8)
[apic.c] established local memory mapping for APIC_BASE=0xfee00000 at 0x7f7ff8e99000
[apic.c] APIC_ID=2000000; LVTT=400ec; TDCR=0
[apic.c] APIC timer one-shot mode with division 2 (lvtt=2d/tdcr=0)

--------------------------------------------------------------------------------
[main.c] recovering password length
--------------------------------------------------------------------------------

[attacker] steps=15; guess='******'
[attacker] found pwd len = 6

--------------------------------------------------------------------------------
[main.c] recovering password bytes
--------------------------------------------------------------------------------

[attacker] steps=35; guess='SECRET' --> SUCCESS

[apic.c] Restored APIC_LVTT=400ec/TDCR=0)
[file.c] writing buffer to '/dev/cpu/1/msr' (size=8)
[main.c] all done; counted 2260/2183 IRQs (AEP/IDT)
jo@breuer:~/sgx-step-demo$
```

Van Bulck et al. "A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes", CCS 2019.

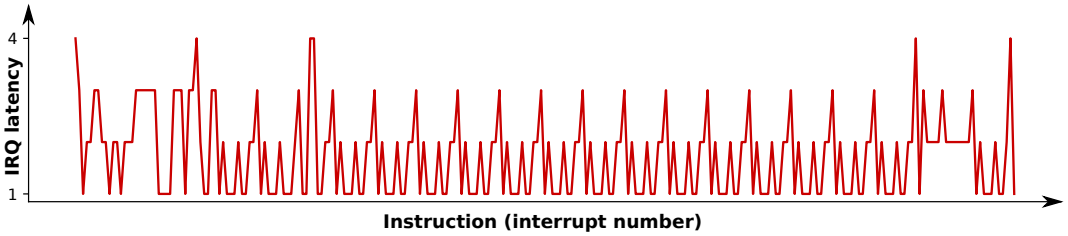
SGX-Step: Enabling a new line of high-precision enclave attacks

Yr	Attack	Temporal resolution	APIC		PTE			Desc		Drv
			IRQ	IPI	#PF	A/D	PPN	GDT	IDT	
'15	Ctrl channel	~ Page	○	○	●	○	○	○	●	✓
'16	AsyncShock	~ Page	○	○	●	○	○	○	○	–
'17	CacheZoom	✗ > 1	●	○	○	○	○	○	○	✓
'17	Hahnel et al.	✗ 0 - > 1	●	○	○	○	○	○	●	✓
'17	BranchShadow	✗ 5 - 50	●	○	○	○	○	○	○	✗
'17	Stealthy PTE	~ Page	○	●	○	●	○	○	●	✓
'17	DarkROP	~ Page	○	○	●	○	○	○	○	✓
'17	SGX-Step	✓ 0 - 1	●	○	●	●	○	○	○	✓
'18	Off-limits	✓ 0 - 1	●	○	●	○	○	●	○	✓
'18	Single-trace RSA	~ Page	○	○	●	○	○	○	○	✓
'18	Foreshadow	✓ 0 - 1	●	○	●	○	●	○	○	✓
'18	SgxPectre	~ Page	○	○	●	○	○	○	○	✓
'18	CacheQuote	✗ > 1	●	○	○	○	○	○	○	✓
'18	SGXlinger	✗ > 1	●	○	○	○	○	○	○	✗
'18	Nemesis	✓ 1	●	○	●	●	○	○	●	✓

Yr	Attack	Temporal resolution	APIC		PTE			Desc		Drv
			IRQ	IPI	#PF	A/D	PPN	GDT	IDT	
'19	Spoiler	✓ 1	●	○	○	●	○	○	●	✓
'19	ZombieLoad	✓ 0 - 1	●	○	●	●	○	○	●	✓
'19	Tale of 2 worlds	✓ 1	●	○	●	●	○	○	●	✓
'19	MicroScope	~ 0 - Page	○	○	●	○	○	○	○	✗
'20	Bluethunder	✓ 1	●	○	○	○	○	○	●	✓
'20	Big troubles	~ Page	○	○	●	○	○	○	○	✓
'20	Viral primitive	✓ 1	●	○	●	●	○	○	●	✓
'20	CopyCat	✓ 1	●	○	●	●	○	○	●	✓
'20	LVI	✓ 1	●	○	●	●	●	○	●	✓
'20	A to Z	~ Page	○	○	●	○	○	○	○	✓
'20	Frontal	✓ 1	●	○	●	●	○	○	●	✓
'20	CrossTalk	✓ 1	●	○	●	○	○	○	●	✓
'20	Online template	~ Page	○	○	●	○	○	○	○	✓
'20	Déjà Vu NSS	~ Page	○	○	●	○	○	○	○	✓

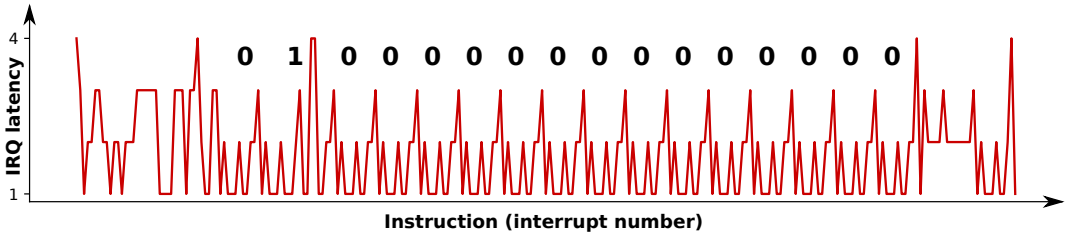
Idea 2: Privileged interrupts for microarchitectural leakage

Nemesis attack: Inferring key strokes from Sancus enclaves



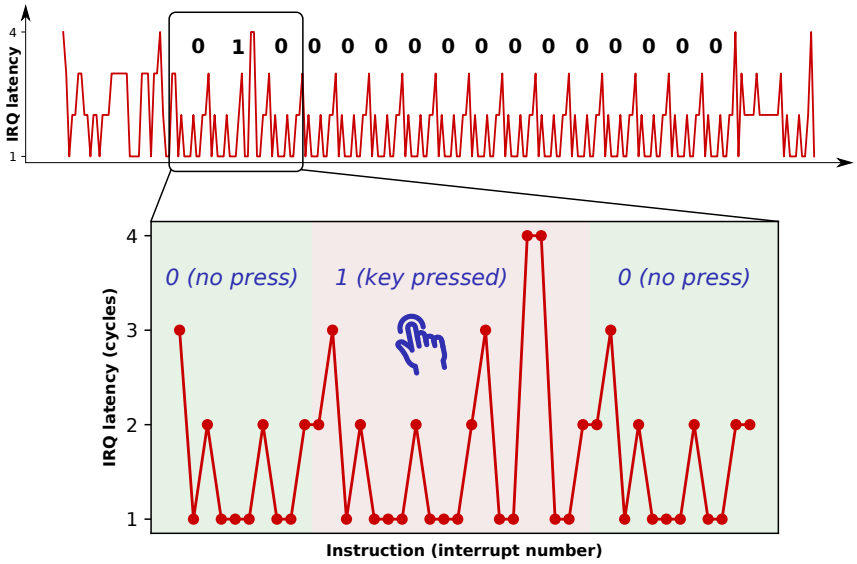
Enclave x-ray: Start-to-end trace enclaved execution

Nemesis attack: Inferring key strokes from Sancus enclaves



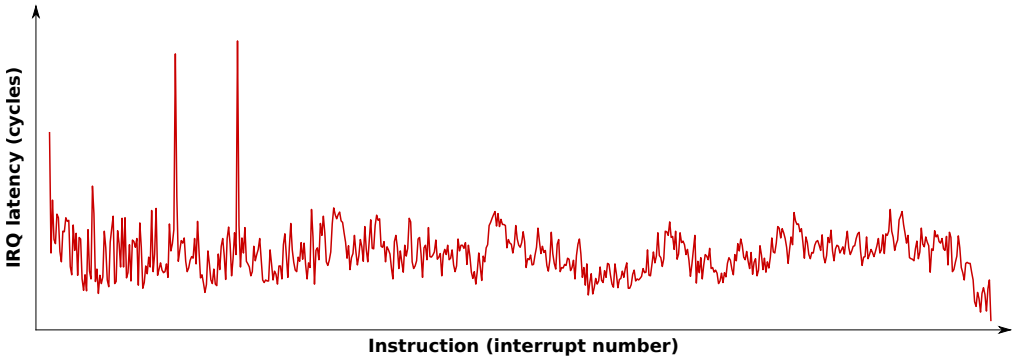
Enclave x-ray: Keymap bit traversal (ground truth)

Nemesis attack: Inferring key strokes from Sancus enclaves



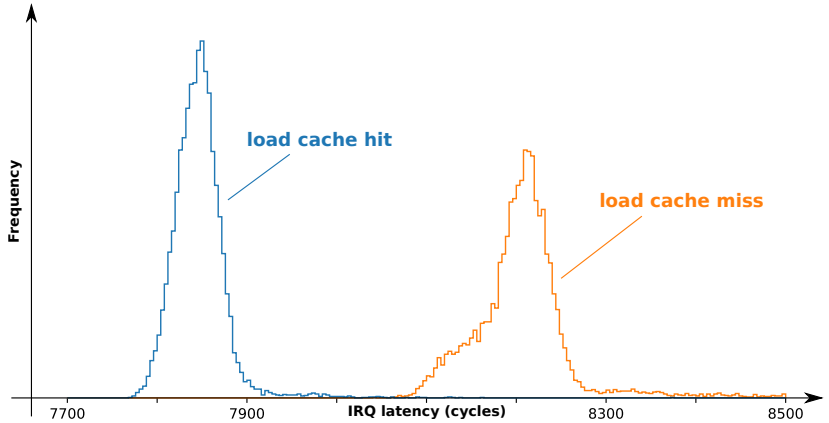
Nemesis attack: Single-stepping Intel SGX enclaves in practice

Enclave x-ray: Start-to-end trace enclaved execution



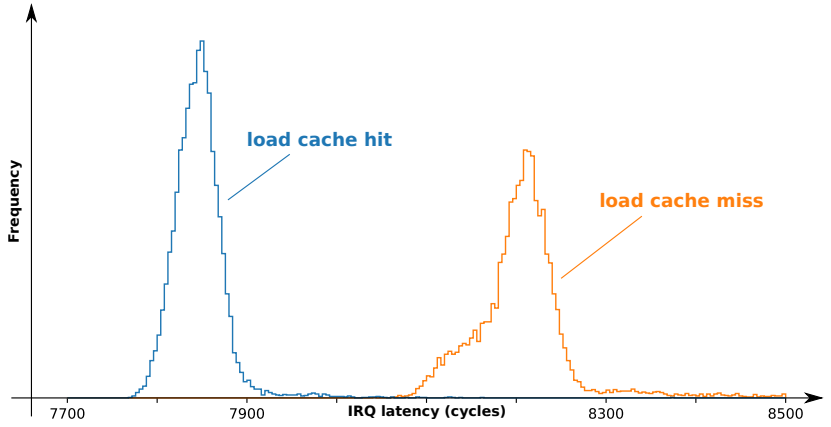
Intel SGX microbenchmarks: Measuring x86 cache misses

Timing leak: Reconstruct *microarchitectural state*



Intel SGX microbenchmarks: Measuring x86 cache misses

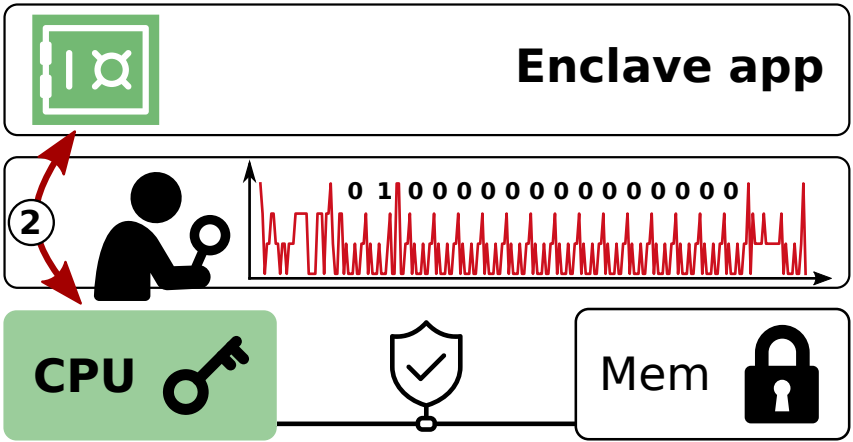
Timing leak: Many more → see paper!



Van Bulck et al. "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic", CCS 2018.

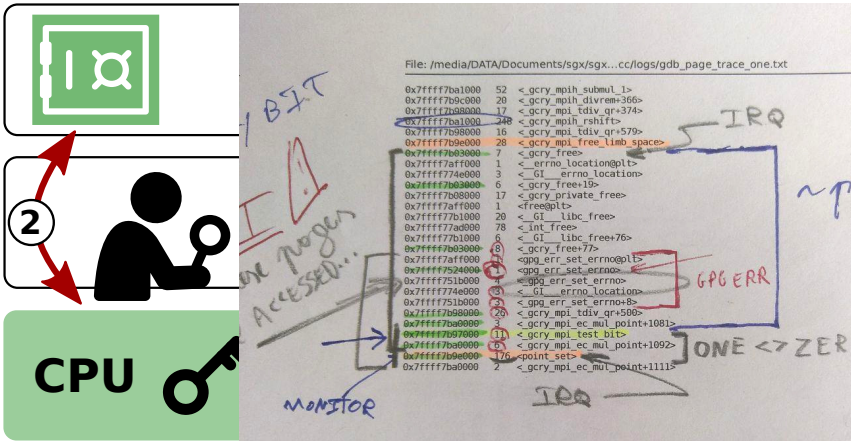
Idea 3: Privileged page tables for transient data leakage

Thesis outline: Privileged side channels (interrupt latency)



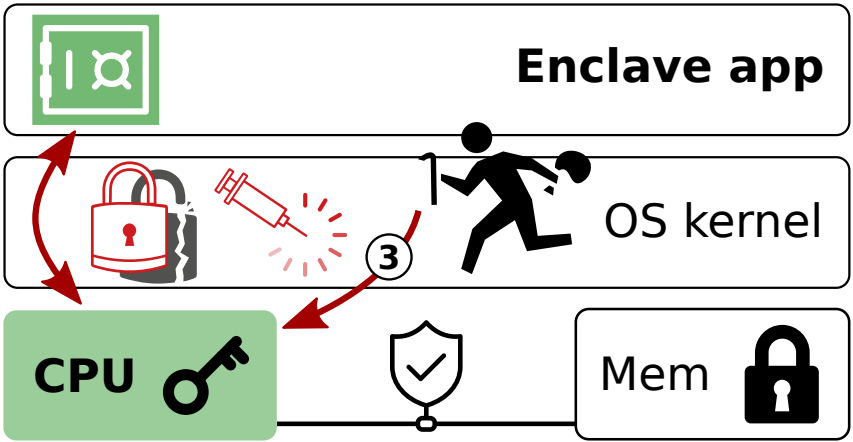
Van Bulck et al. "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic", CCS 2018.

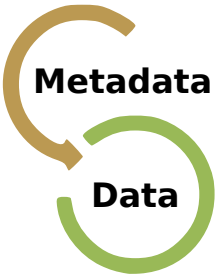
Thesis outline: Privileged side channels (page table accesses)



Van Bulck et al. "Telling Your Secrets Without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution", USENIX Security 2017.

Thesis outline: Transient-execution attacks (Foreshadow, LVI)



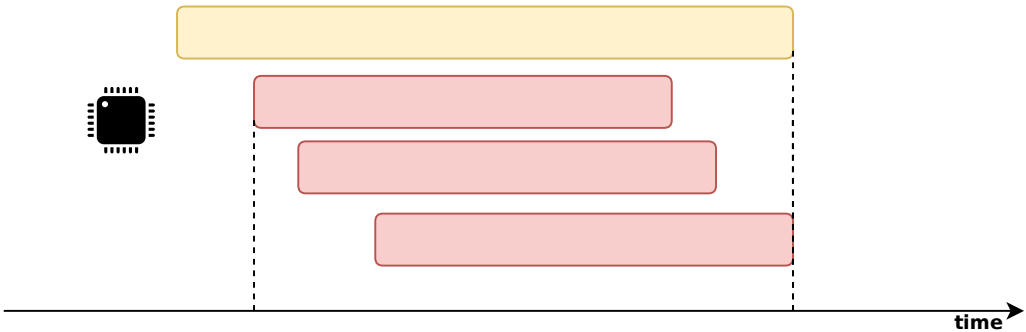


Van Bulck et al. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution", USENIX Security 2018.

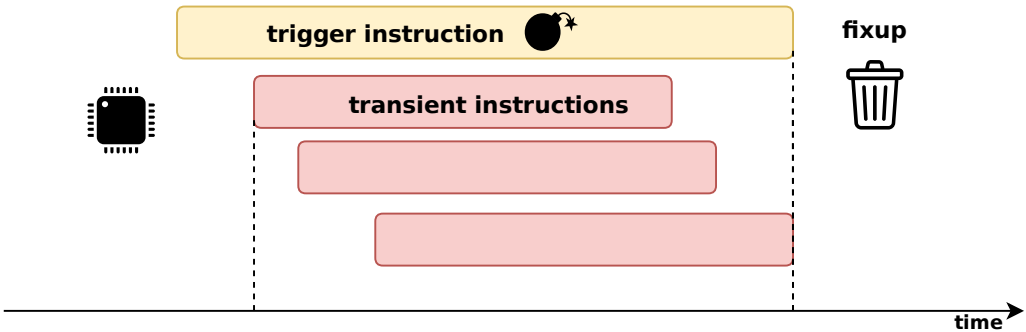
Van Bulck et al. "LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection", S&P 2020.


WHAT IF I TOLD YOU
YOU CAN CHANGE RULES MID-GAME
imgflip.com

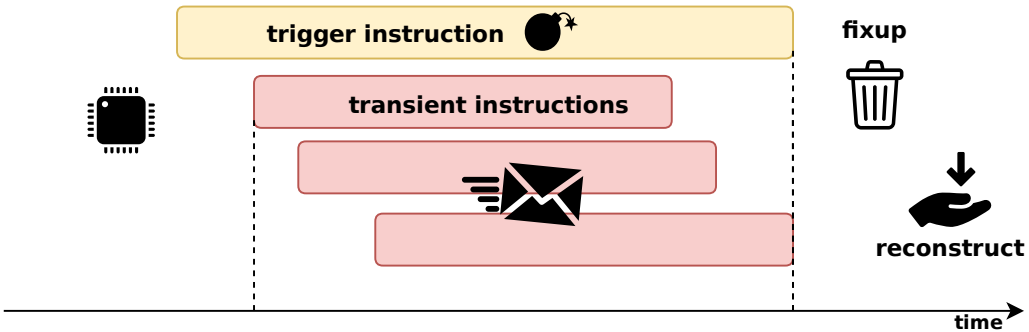
Abusing out-of-order and speculative execution



Abusing out-of-order and speculative execution

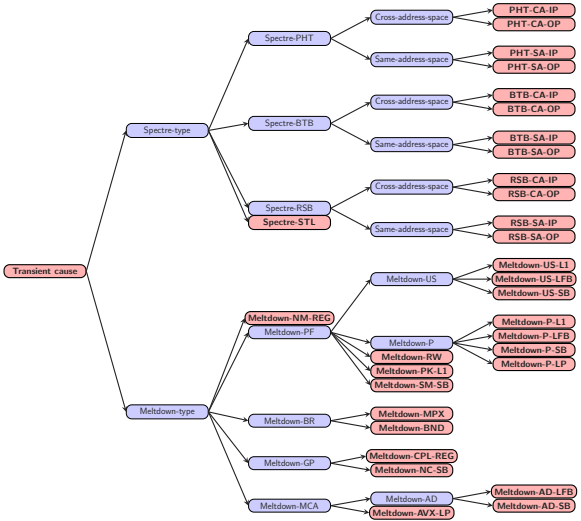


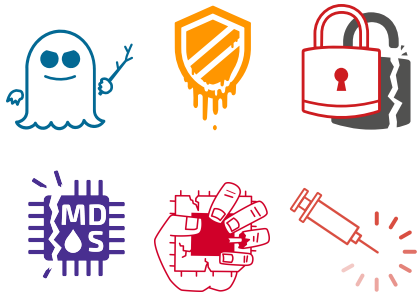
Abusing out-of-order and speculative execution



The transient-execution zoo

https://transient.fail

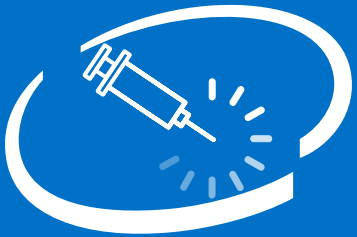




Canella et al. "A systematic evaluation of transient execution attacks and defenses", USENIX Security 2019

inside™

inside™



inside™

Rumors: Meltdown immunity for SGX enclaves?

Meltdown melted down everything, except for one thing

"[enclaves] remain protected and completely secure"

— *International Business Times, February 2018*

ANJUNA'S SECURE-RUNTIME CAN PROTECT CRITICAL APPLICATIONS AGAINST THE MELTDOWN ATTACK USING ENCLAVES

"[enclave memory accesses] redirected to an abort page, which has no value"

— *Anjuna Security, Inc., March 2018*

~~Rumors: Meltdown immunity for SGX enclaves?~~

LILY HAY NEWMAN SECURITY 08.14.18 01:00 PM

SPECTRE-LIKE FLAW UNDERMINES INTEL PROCESSORS' MOST SECURE ELEMENT

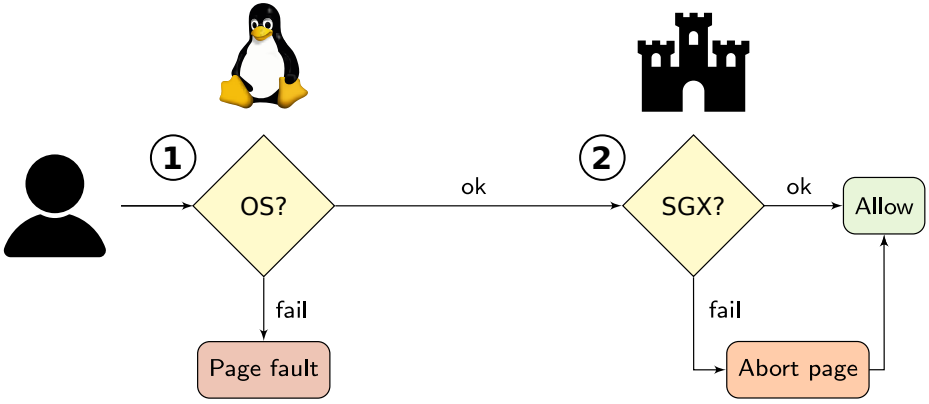
I'M SURE THIS WON'T BE THE LAST SUCH PROBLEM —

Intel's SGX blown wide open by, you guessed it, a speculative execution attack

Speculative execution attacks truly are the gift that keeps on giving.

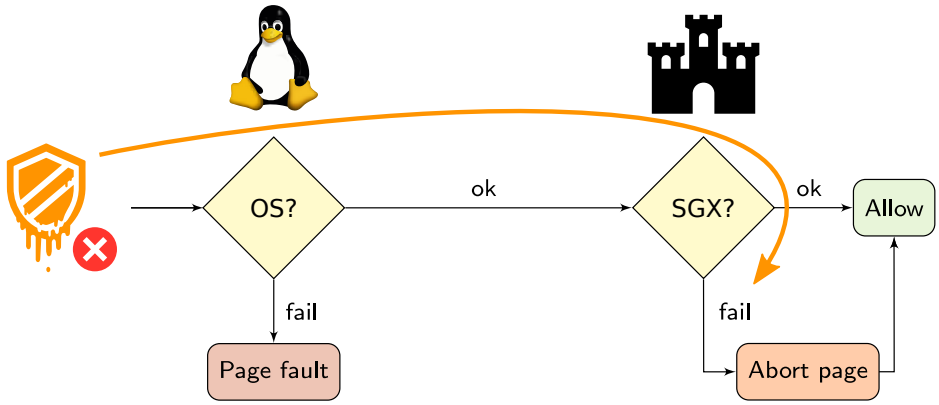
https://wired.com and https://arstechnica.com

Building Foreshadow: Evade SGX abort page semantics



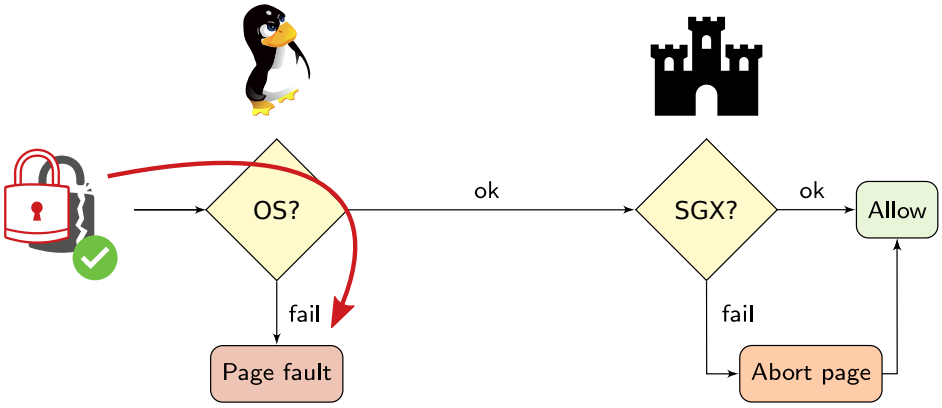
SGX checks prohibit unauthorized access

Building Foreshadow: Evade SGX abort page semantics



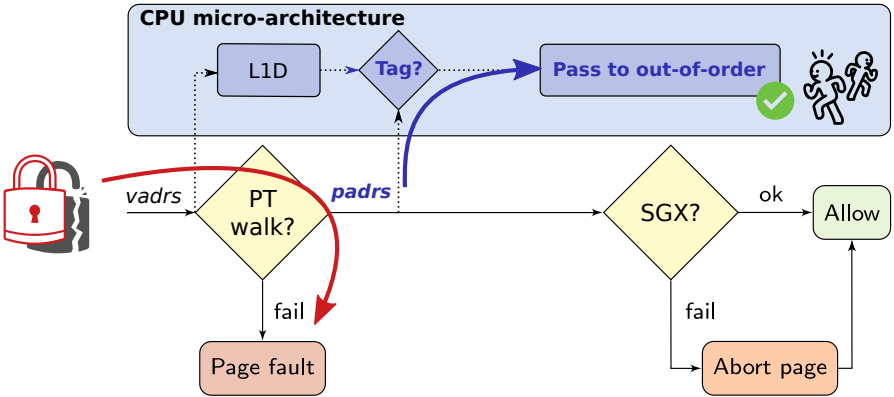
SGX checks prohibit unauthorized access

Building Foreshadow: Evade SGX abort page semantics



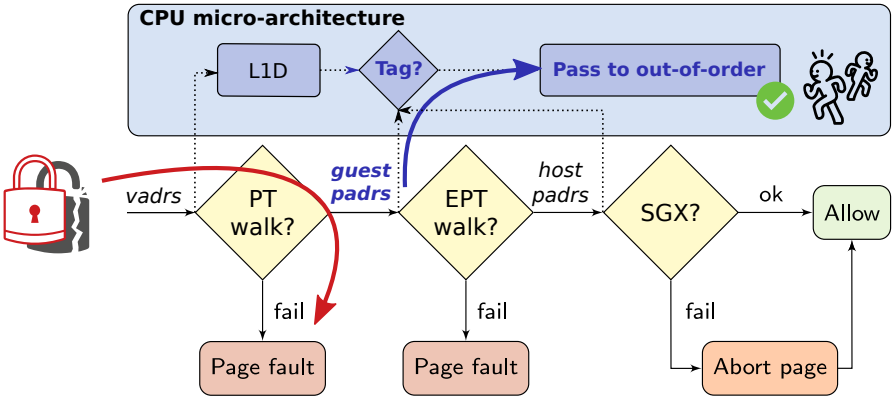
. . . but attackers can **unmap** enclave pages!

The microarchitecture behind Foreshadow



Foreshadow-SGX: Bypass enclave isolation

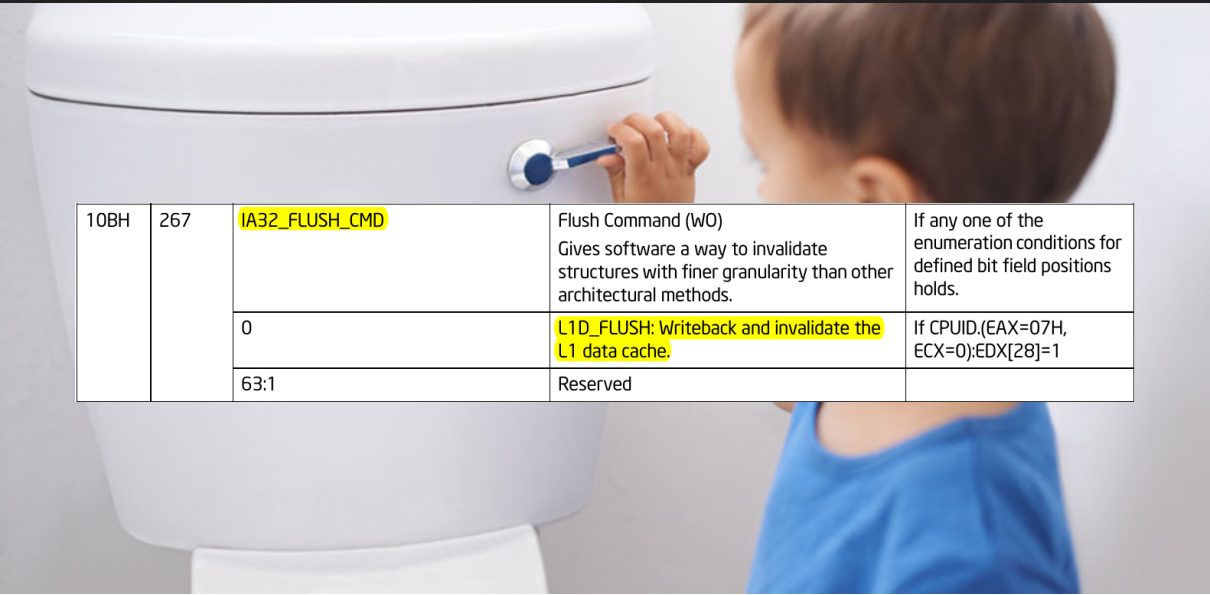
The microarchitecture behind Foreshadow



Foreshadow-VMM: Bypass virtual machine isolation

Mitigating Foreshadow: Flush CPU microarchitecture

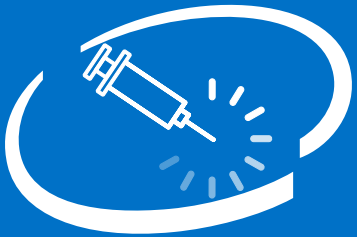
Mitigating Foreshadow: Flush CPU microarchitecture



10BH	267	IA32_FLUSH_CMD	Flush Command (WO) Gives software a way to invalidate structures with finer granularity than other architectural methods.	If any one of the enumeration conditions for defined bit field positions holds.
		0	L1D_FLUSH: Writeback and invalidate the L1 data cache.	If CPUID.(EAX=07H, ECX=0):EDX[28]=1
		63:1	Reserved	

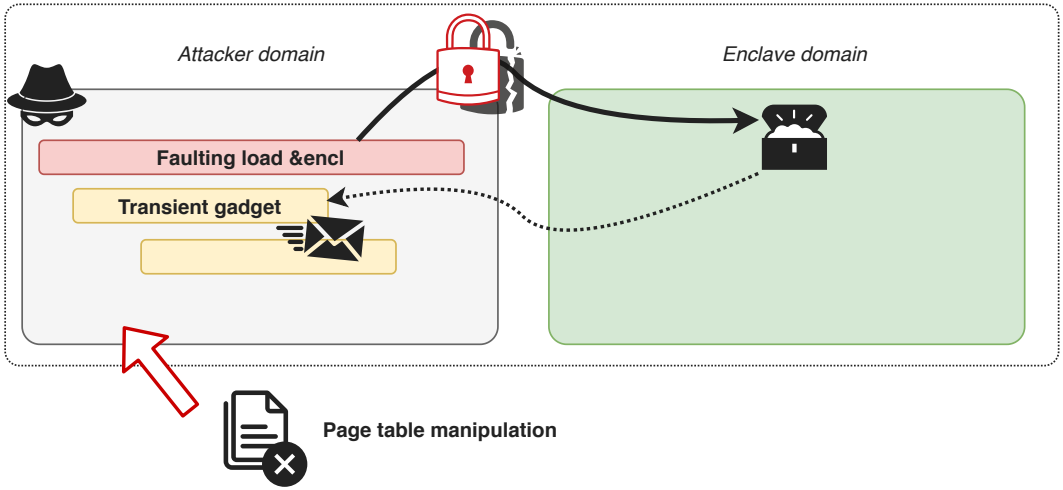
inside™

inside™



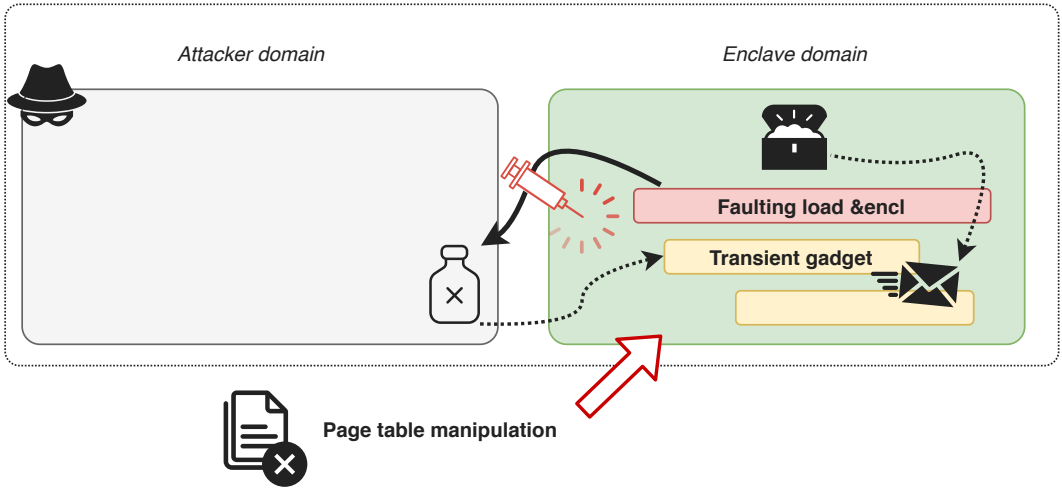
inside™

Idea: Inverting Foreshadow & co. with Load Value Injection (LVI)



Van Bulck et al. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution", USENIX Security 2018.

Idea: Inverting Foreshadow & co. with Load Value Injection (LVI)



Van Bulck et al. "LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection", S&P 2020.

FOOD POISONING

Overdue products

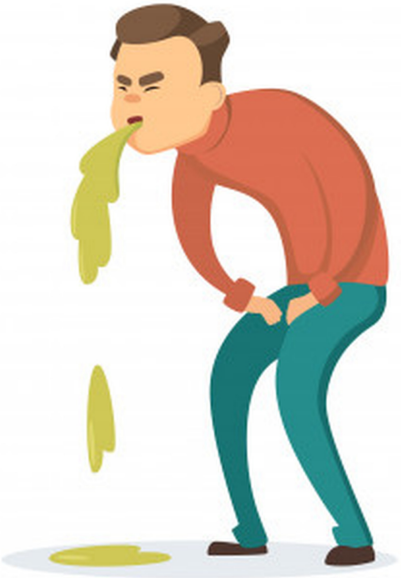
Medicine

Dizziness

Intestinal colic

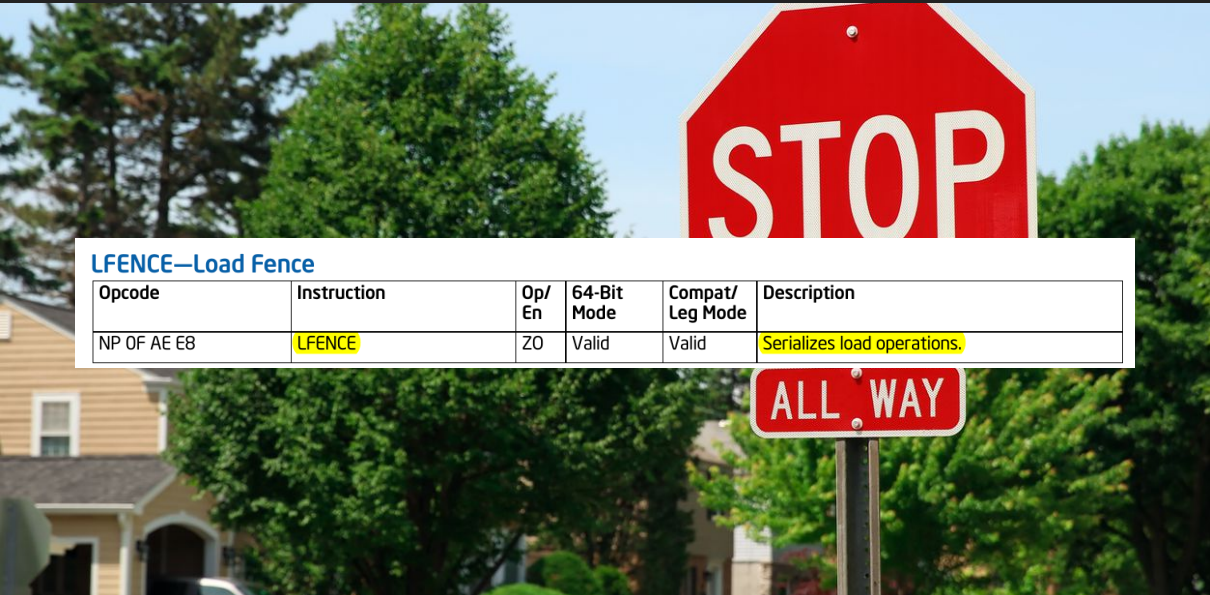
Diarrhea

Headache



Mitigating LVI: Fencing vulnerable load instructions

Mitigating LVI: Fencing vulnerable load instructions



LFENCE—Load Fence

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
NP 0F AE E8	LFENCE	ZO	Valid	Valid	Serializes load operations.

Mitigating LVI: Compiler and assembler support

`-mlfence-after-load`

GNU Assembler Adds New Options For Mitigating Load Value Injection Attack

Written by Michael Larabel in GNU on 11 March 2020 at 02:55 PM EDT. 14 Comments

`-mlvi-hardening`

LLVM Lands Performance-Hitting Mitigation For Intel LVI Vulnerability

Written by Michael Larabel in Software on 3 April 2020. **Page 1 of 3**. 20 Comments

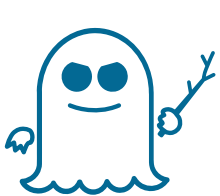
`-Qspectre-load`

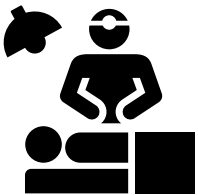
More Spectre Mitigations in MSVC

March 13th, 2020

Intel architectural enclaves: `lfence` counts

`libsgx_qe.signed.so`



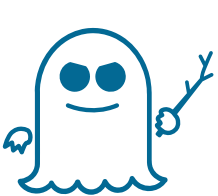


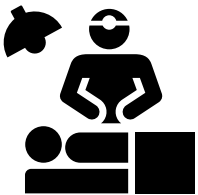
23 fences

October 2019—"surgical precision"

Intel architectural enclaves: `lfence` counts

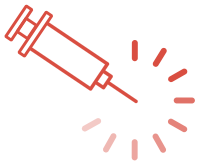
`libsgx_qe.signed.so`





23 fences

October 2019—"surgical precision"



49,315 fences

March 2020—"big hammer"

Outlook: Future and ongoing research directions

1. **Universal attack primitives:** Intel TDX, AMD SEV, ARM?
   → Adversary capabilities, hardware vs. software monitor, automation, etc.

Outlook: Future and ongoing research directions

1. **Universal attack primitives:** Intel TDX, AMD SEV, ARM?
   → Adversary capabilities, hardware vs. software monitor, automation, etc.
2. **Hardware extensions** for next-gen TEEs: MSP430-Sancus, RISC-V
   → Provable security & limitations, availability, SMAP-like restrictions, etc.

Outlook: Future and ongoing research directions

1. **Universal attack primitives:** Intel TDX, AMD SEV, ARM?
   → Adversary capabilities, hardware vs. software monitor, automation, etc.
2. **Hardware extensions** for next-gen TEEs: MSP430-Sancus, RISC-V
   → Provable security & limitations, availability, SMAP-like restrictions, etc.
3. **Transparent shielding:** Enclave runtime, compiler
   → Fuzzing, formal verification of the enclave interface
   → Compile-time hardening for *incremental* side-channel resistance

Outlook: Future and ongoing research directions

1. **Universal attack primitives:** Intel TDX, AMD SEV, ARM?
   → Adversary capabilities, hardware vs. software monitor, automation, etc.
2. **Hardware extensions** for next-gen TEEs: MSP430-Sancus, RISC-V
   → Provable security & limitations, availability, SMAP-like restrictions, etc.
3. **Transparent shielding:** Enclave runtime, compiler
   → Fuzzing, formal verification of the enclave interface
   → Compile-time hardening for *incremental* side-channel resistance
4. Towards **transient safety:** Redefining the hardware-software contract
   → Efficient containment of Spectre (long term) vs. LVI (short term)

Thank you!