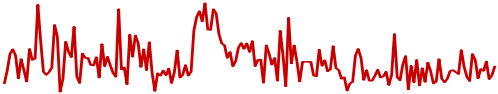


Microarchitectural Side-Channel Attacks for Privileged Software Adversaries

Jo Van Bulck

Public PhD defense, September 14, 2020

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

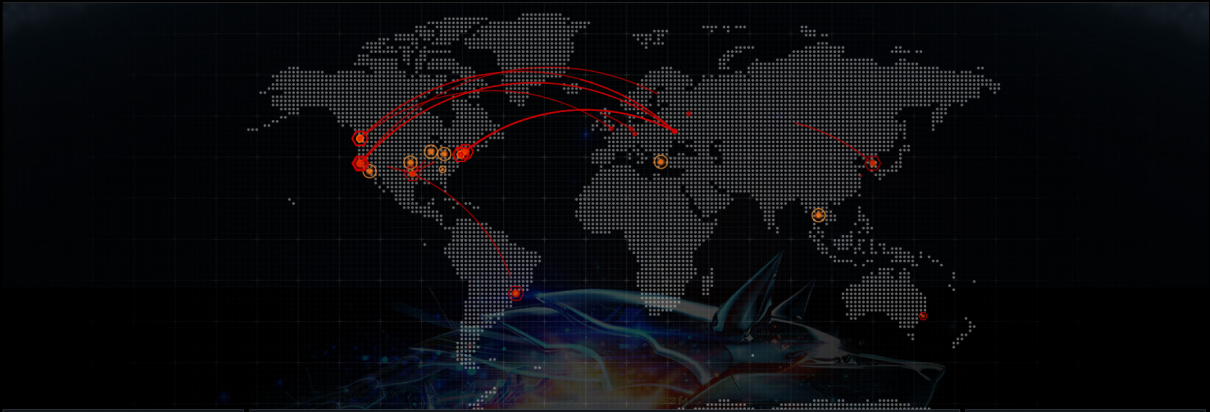
DistriNet

KU LEUVEN


!!
!!
!!
www.freepik.com


SOCIAL DISTANCING
STOP COVID-19
MINIMUM 2M (6FT)
MINIMUM 2M (6FT)
www.freepik.com

Bitdefender CYBERTHREAT REAL-TIME MAP



LEGEND

- ATTACKS
- INFECTIONS
- SPAM

LIVE ATTACKS

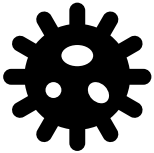
TIME	ATTACK	ATTACK TYPE	ATTACK COUNTRY	TARGET COUNTRY
MON 31 AUG 5:34:44 PM	N/A	ATTACK	UKRAINE	UNITED STATES
MON 31 AUG 5:34:44 PM	N/A	ATTACK	UKRAINE	UNITED STATES
MON 31 AUG 6:00:54 PM	JS:ADWARE.LNKR.A	INFECTION	BRAZIL	N/A
MON 31 AUG 5:34:43 PM	N/A	ATTACK	UKRAINE	UNITED STATES
MON 31 AUG 5:59:52 PM	ADWARE.DEALPLY.1.GEN	INFECTION	BRAZIL	N/A
MON 31 AUG 5:59:49 PM	N/A	SPAM	UNITED STATES	N/A
MON 31 AUG 6:00:01 PM	#CBBULK_LOAD.EXE:0000E001…	INFECTION	UNITED KINGDOM	N/A

LOCATIONS

- UKRAINE
- UNITED STATES
- BRAZIL
- GERMANY
- FRANCE
- CANADA
- ITALY
- UNITED KINGDOM

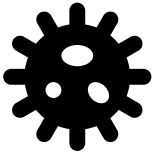
https://threatmap.bitdefender.com/

What do corona and computer viruses have in common?



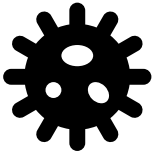
1. No vaccine: adapt to a new reality...

What do corona and computer viruses have in common?

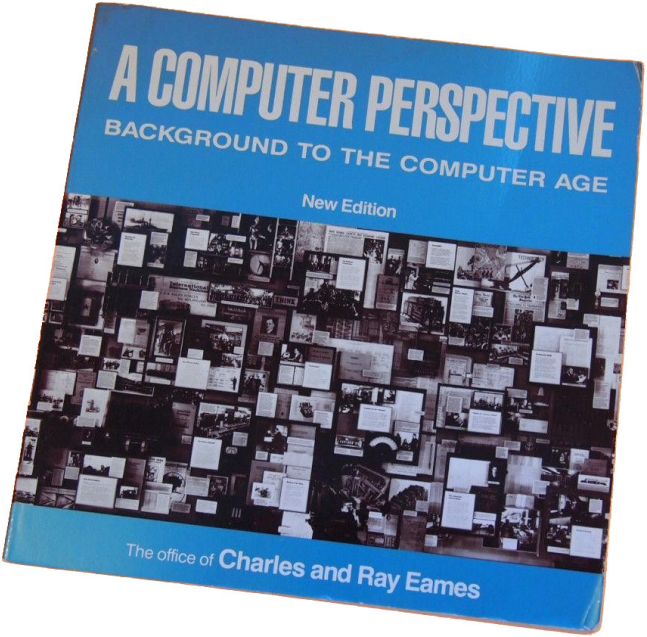


1. No vaccine: adapt to a new reality...
2. Need for physical distancing → **software isolation**

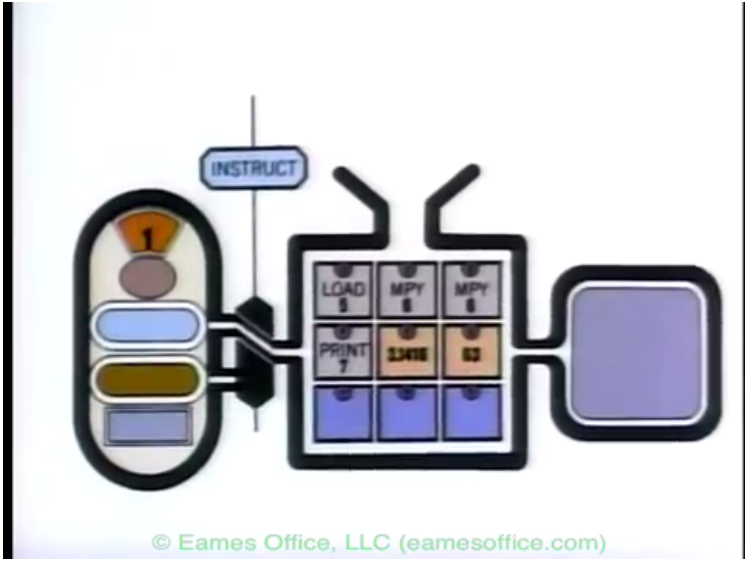
What do corona and computer viruses have in common?



1. No vaccine: adapt to a new reality...
2. Need for physical distancing → **software isolation**
3. Need for testing → **software attestation**

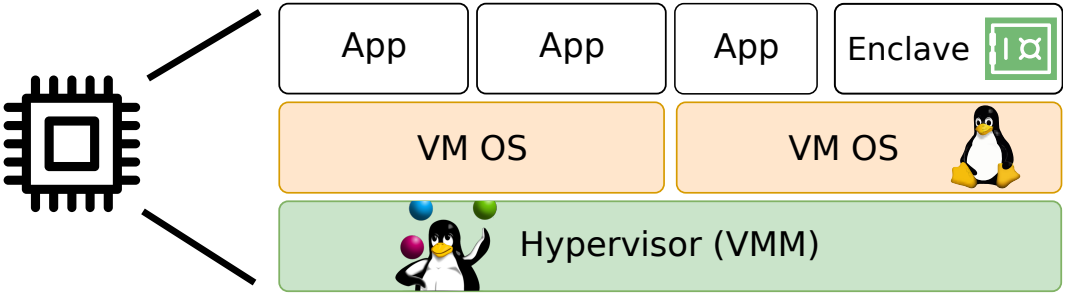

A COMPUTER PERSPECTIVE
BACKGROUND TO THE COMPUTER AGE
New Edition
The office of Charles and Ray Eames

A crash course on computer architecture



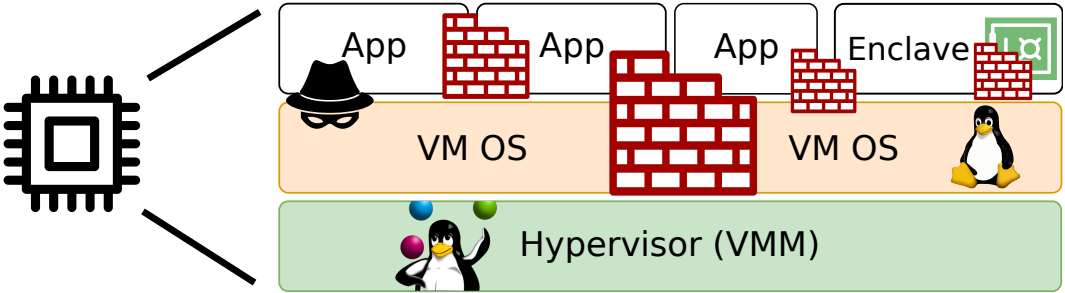

Frank Piessens
Computerwetenschapper KULeuven
Fout computerchips Intel

Processor security: Hardware isolation mechanisms



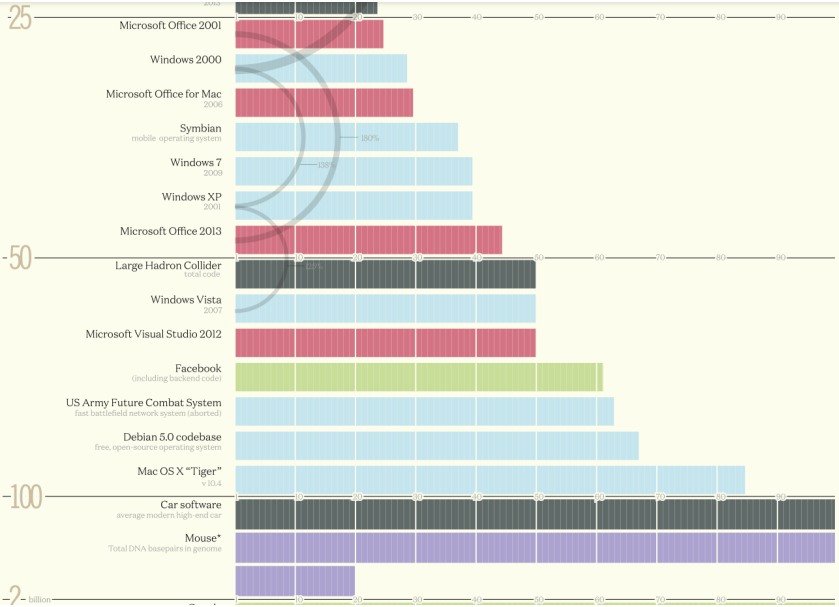
- Different software **protection domains:** applications, virtual machines, enclaves

Processor security: Hardware isolation mechanisms



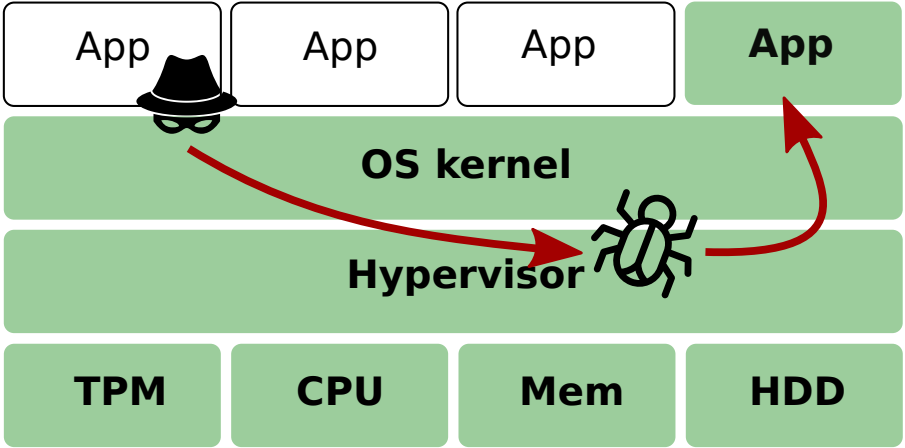
- Different software **protection domains:** applications, virtual machines, enclaves
- CPU builds "walls" for **memory isolation** between applications and privilege levels

@SIOUXSIEW @XTOTL thespinoff.co.nz



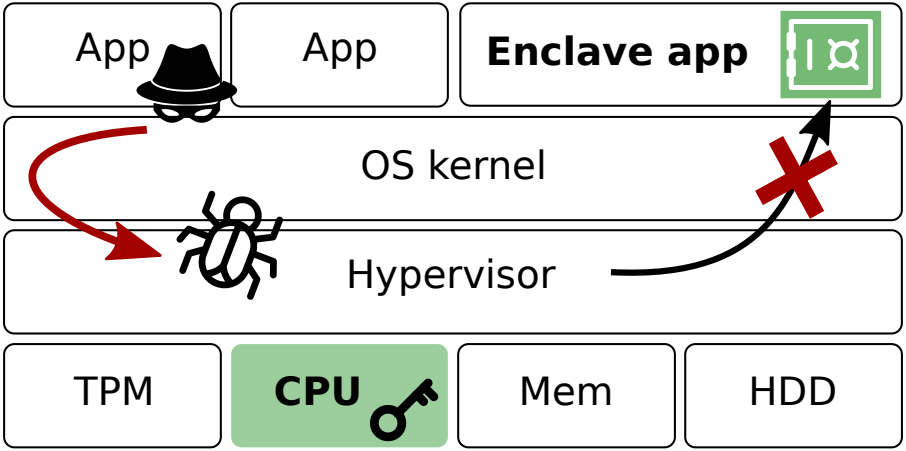
https://informationisbeautiful.net/visualizations/million-lines-of-code/

Enclaved execution: Reducing the bubble



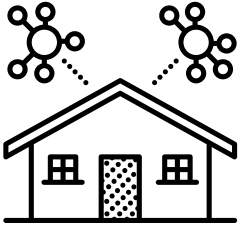
Traditional **layered designs:** large trusted computing base

Enclaved execution: Reducing the bubble



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

Overview: Processor enclaves for self-quarantining

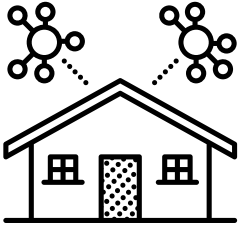


≈ Vault for sensitive code and data

→ Trusted "bubble" in untrusted world

- 2008-2014: Research prototypes (e.g., Sancus)

Overview: Processor enclaves for self-quarantining



≈ Vault for sensitive code and data

→ Trusted "bubble" in untrusted world

- 2008-2014: Research prototypes (e.g., Sancus)
- 2015: Intel Software Guard Extensions (SGX)

KITGURU

Intel alters design of 'Skylake' processors to enhance security

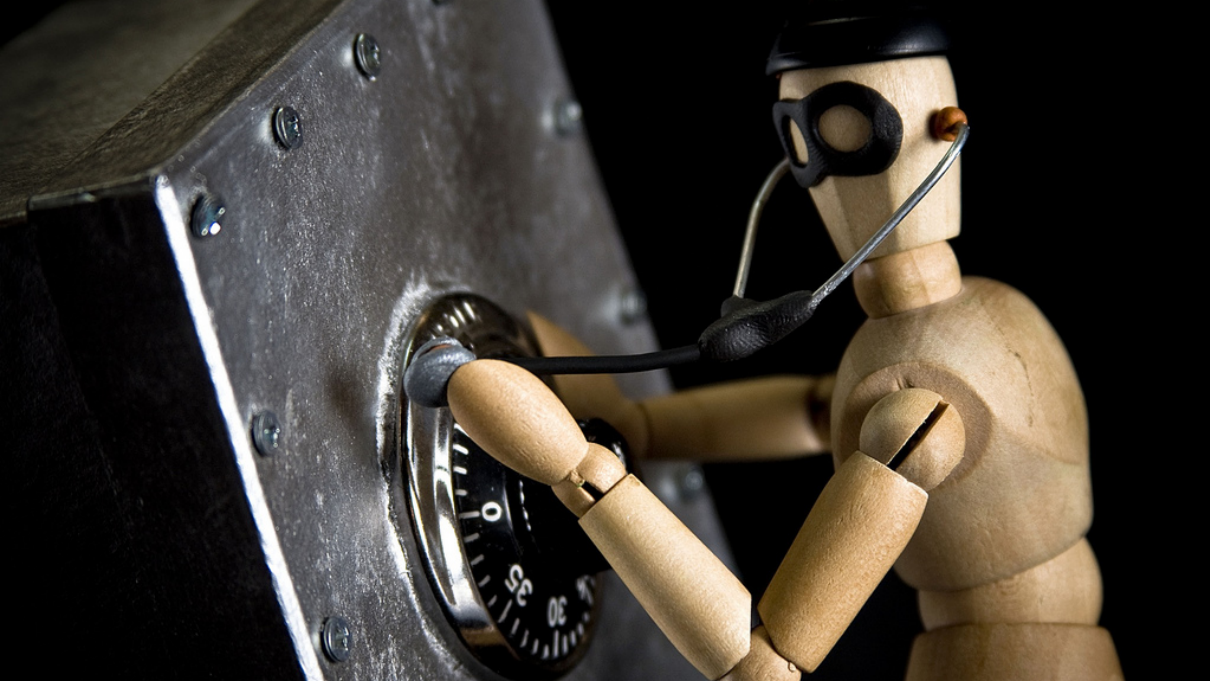
Anton Shilov October 3, 2015 APU, CPU

THE TECH REPORT

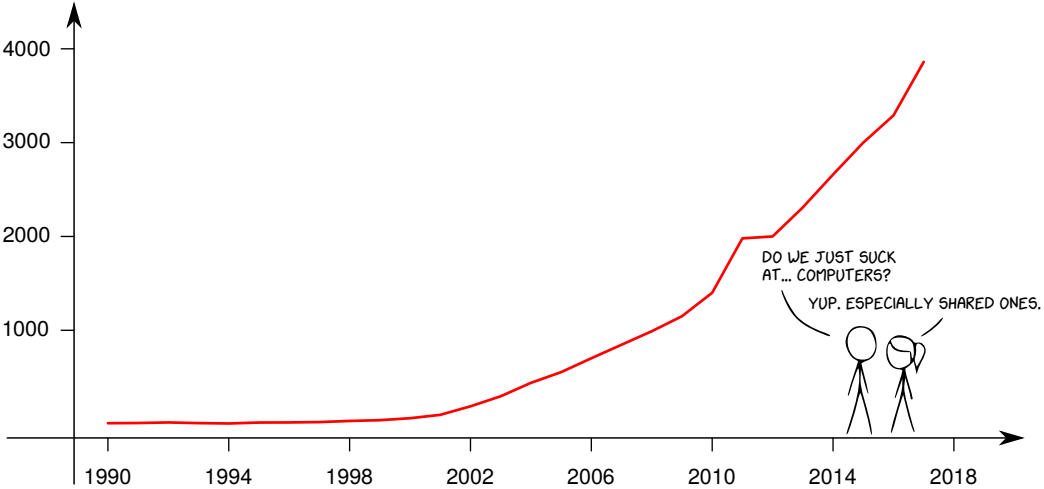
Intel to begin shipping Skylake CPUs with SGX enabled

BY BEN FUNK / 8:08 AM, OCTOBER 5, 2015 / 0 COMMENTS


VAULT DOOR
WEIGHT: 22 1/2 Tons
THICKNESS: 22 Inches
STEEL: 11 Layers of Special Cutting and Drill Resistant
LOCKS: 4 Hamilton Watch Movements for Time Locks

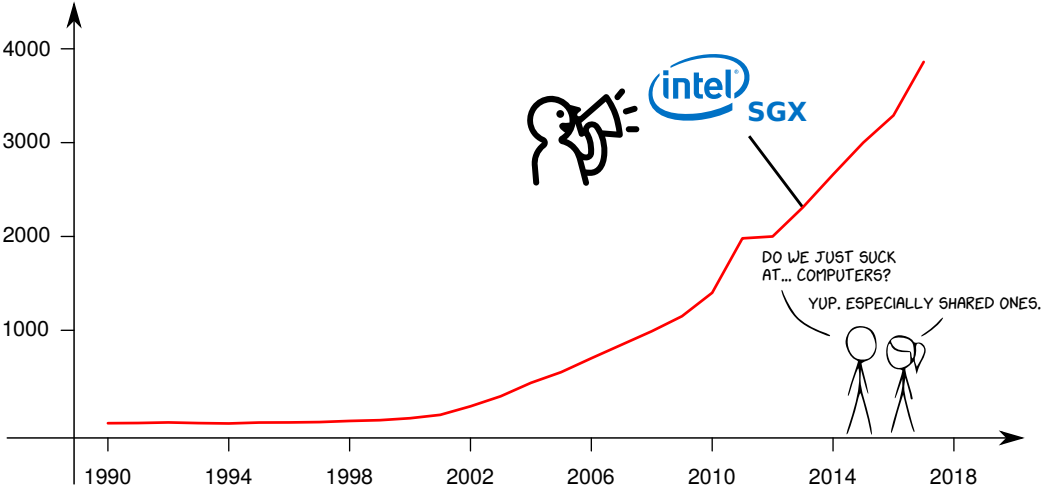


Evolution of “side-channel attack” research



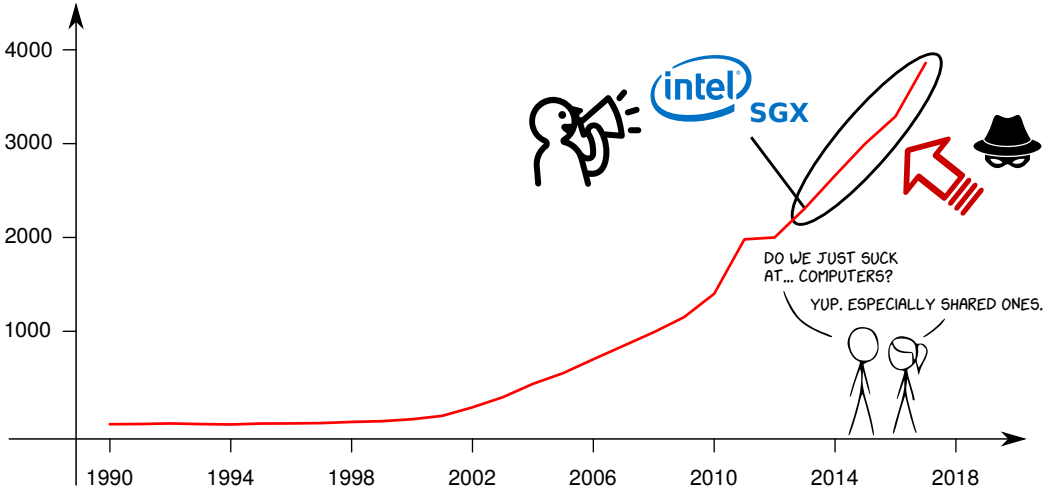
Based on github.com/Pold87/academic-keyword-occurrence and xkcd.com/1938/

Evolution of “side-channel attack” research

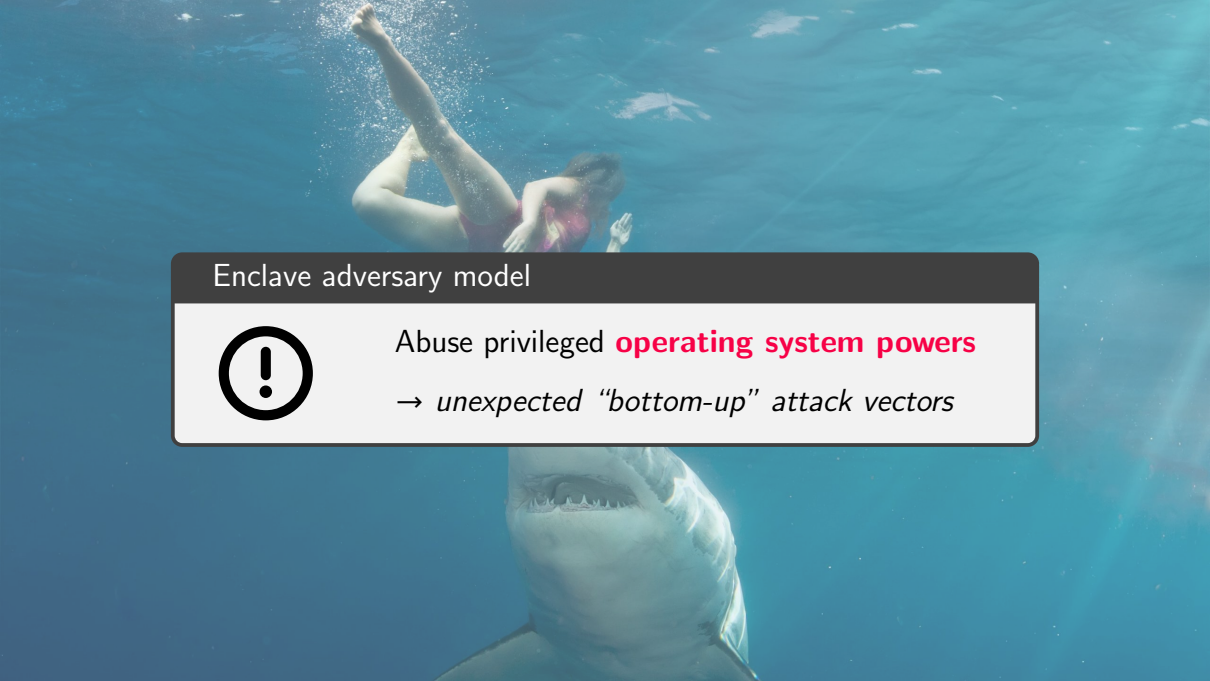


Based on github.com/Pold87/academic-keyword-occurrence and xkcd.com/1938/

Side-channel attacks and trusted computing (focus of this PhD)



Based on github.com/Pold87/academic-keyword-occurrence and xkcd.com/1938/

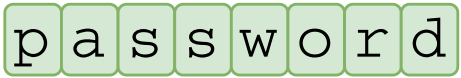


Enclave adversary model

Abuse privileged **operating system powers**

→ *unexpected "bottom-up" attack vectors*

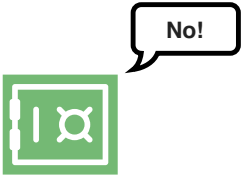
Case study: Comparing a secret password

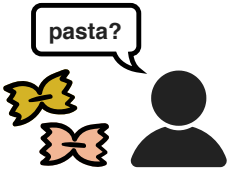


Case study: Comparing a secret password

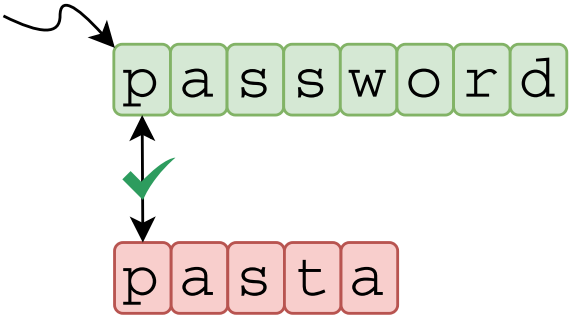
password

pasta

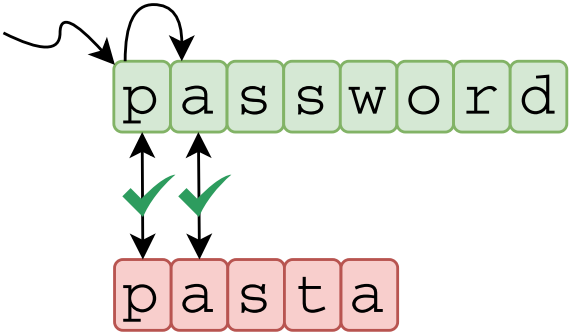




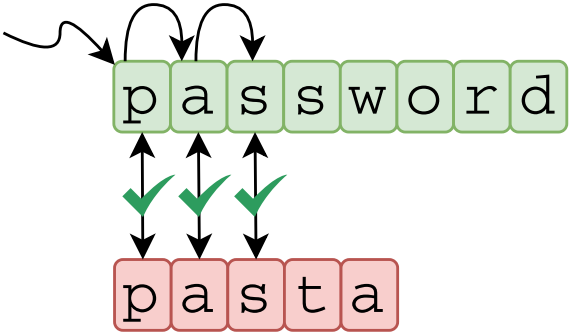
Case study: Comparing a secret password



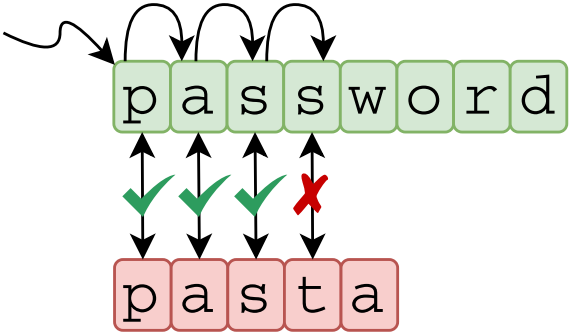
Case study: Comparing a secret password



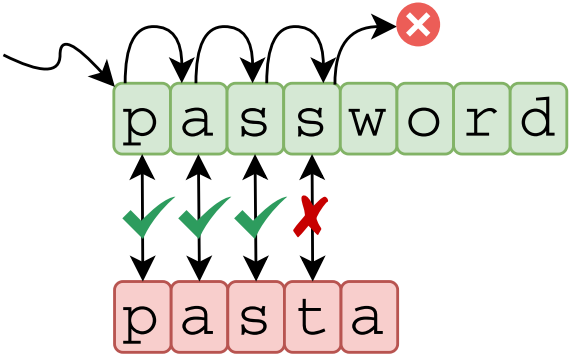
Case study: Comparing a secret password



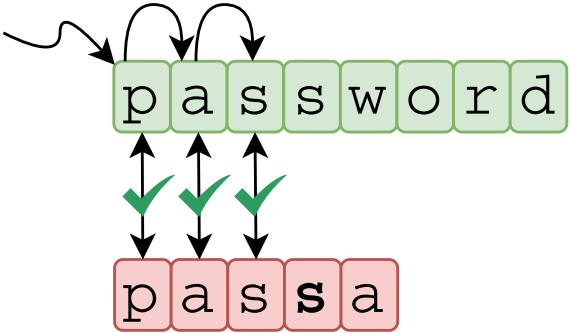
Case study: Comparing a secret password



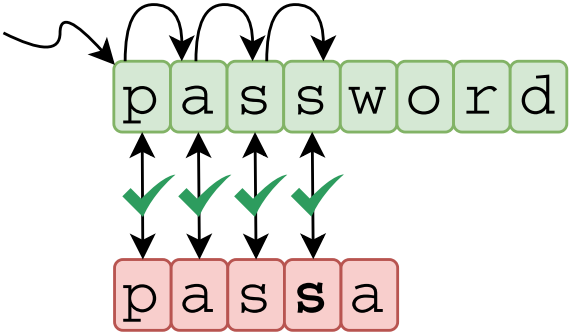
Case study: Comparing a secret password



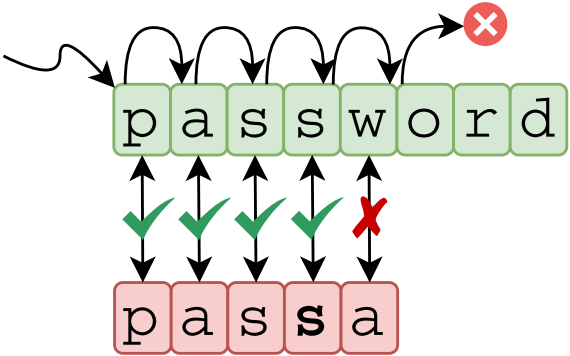
Case study: Comparing a secret password



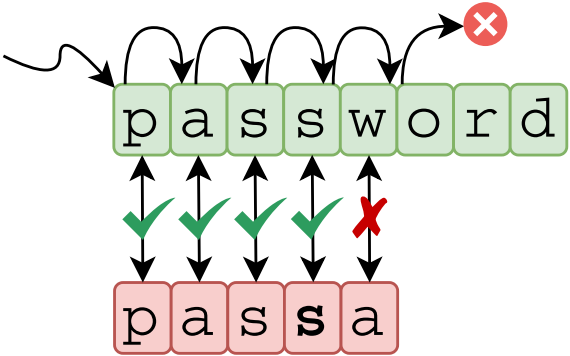
Case study: Comparing a secret password

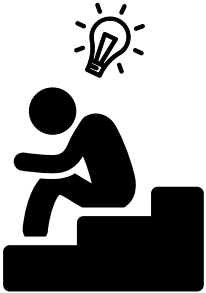


Case study: Comparing a secret password



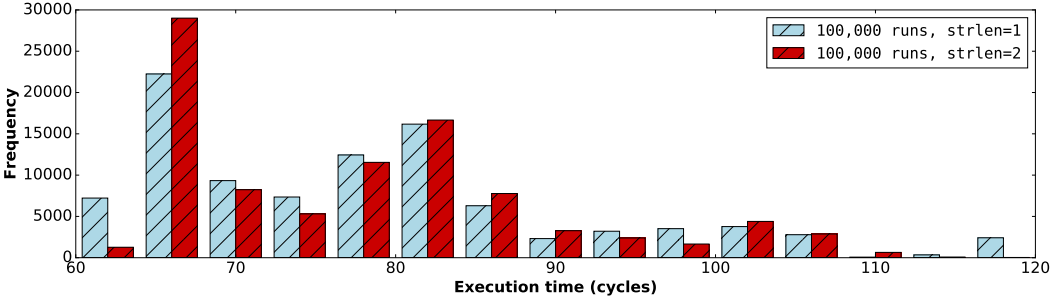
Case study: Comparing a secret password





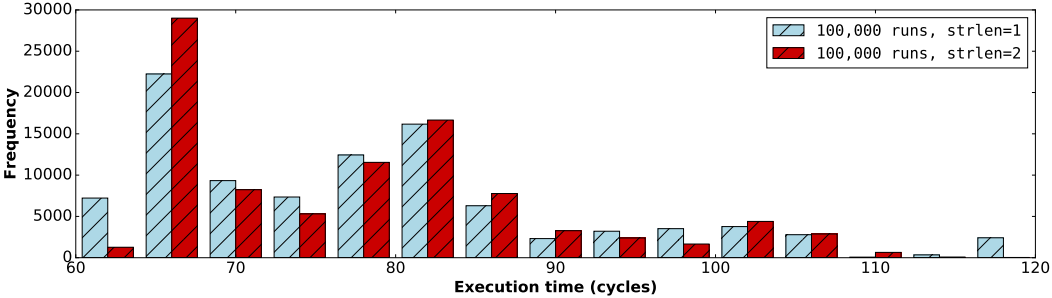
Overall execution time reveals correctness of individual password bytes!

Building the side-channel oracle with execution timing?

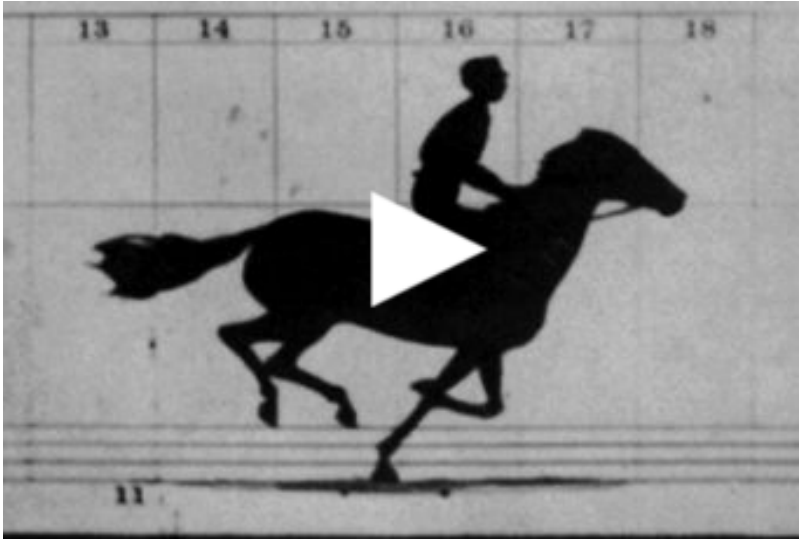


Building the side-channel oracle with execution timing?

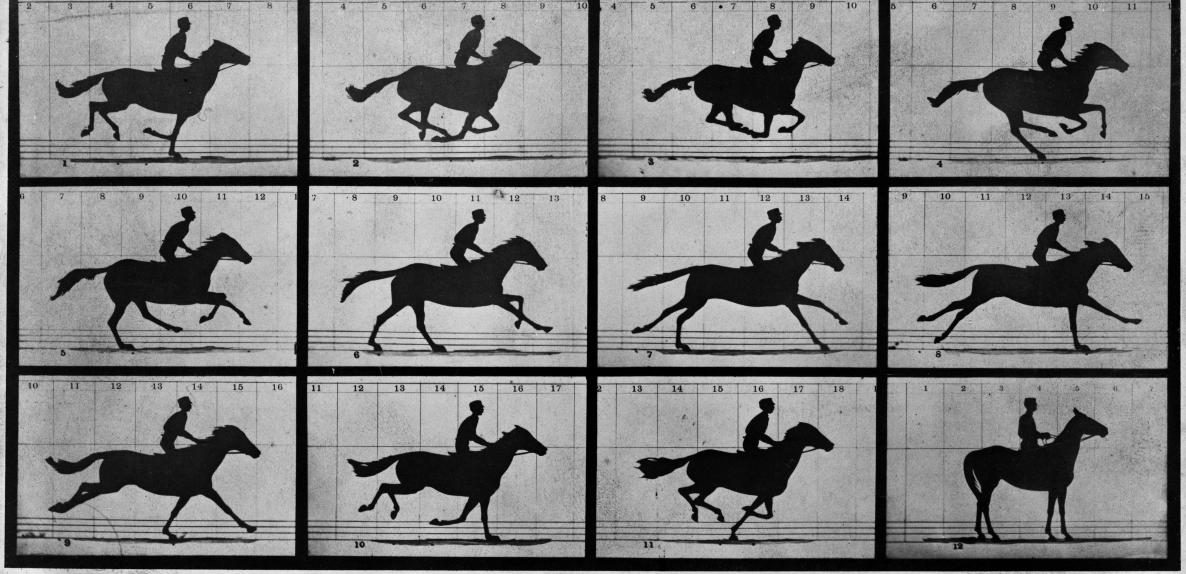
Too noisy: modern x86 processors are lightning fast. . .



Analogy: Studying galloping horse dynamics



https://en.wikipedia.org/wiki/Sallie_Gardner_at_a_Gallop



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

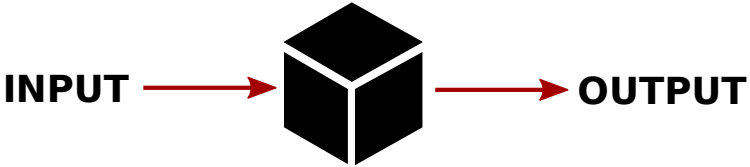
THE HORSE IN MOTION.

Illustrated by
MUYBRIDGE.

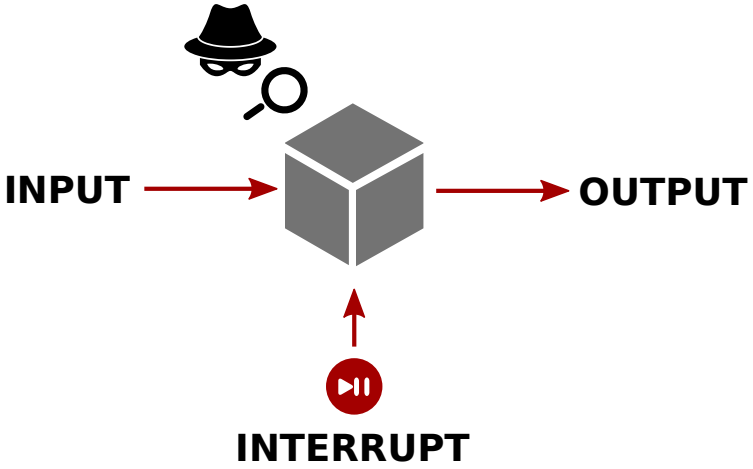
AUTOMATIC ELECTRO-PHOTOGRAPH.

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

SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time



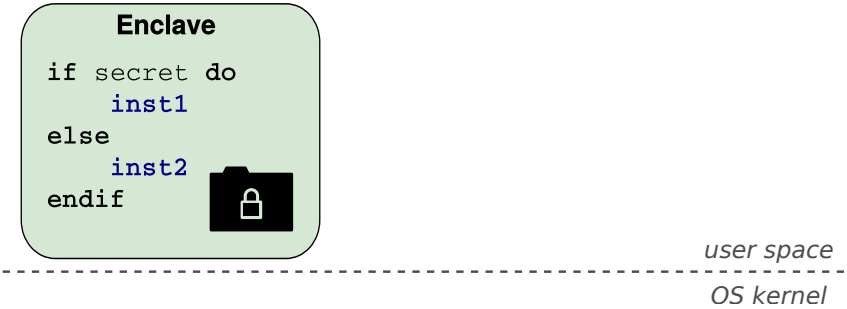
SGX-Step: Executing enclaves one instruction at a time

SGX-Step

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

Watch 22 | Star 245 | Fork 52

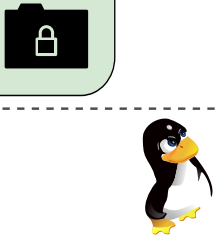
SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time

Enclave

```
if secret do
    inst1
else
    inst2
endif
```

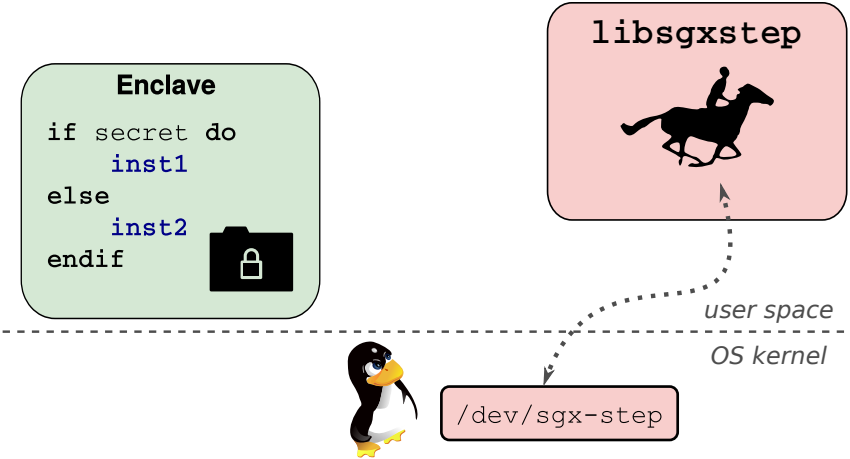


user space

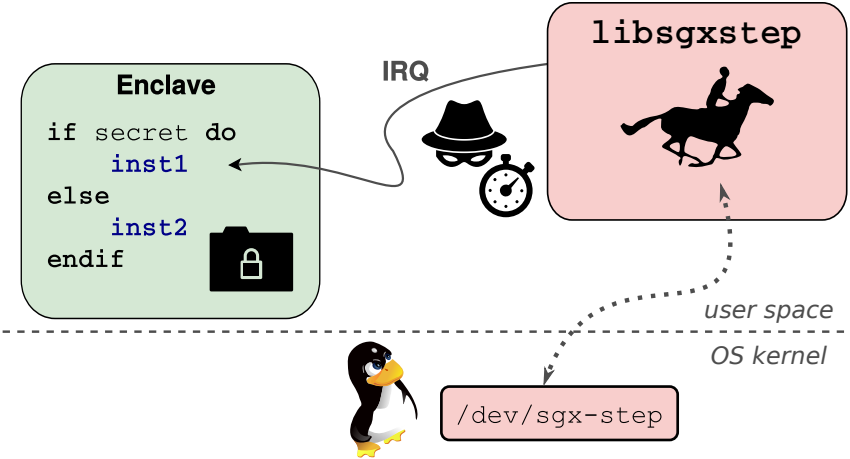
OS kernel

`/dev/sgx-step`

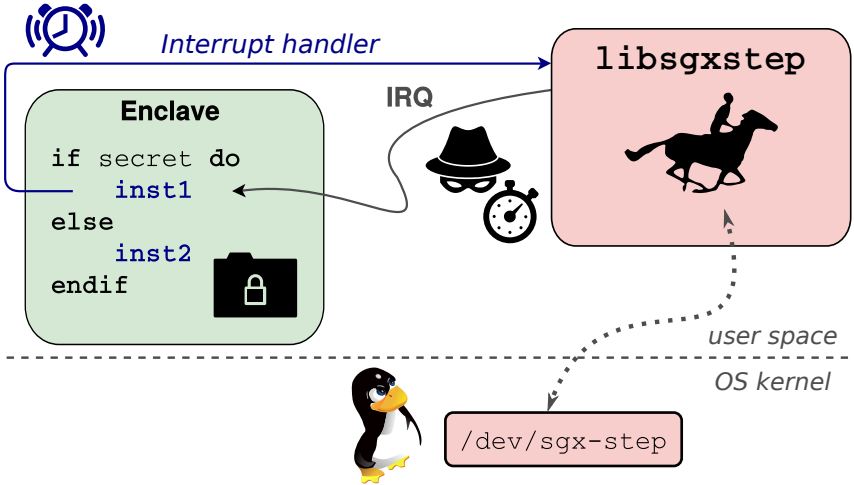
SGX-Step: Executing enclaves one instruction at a time



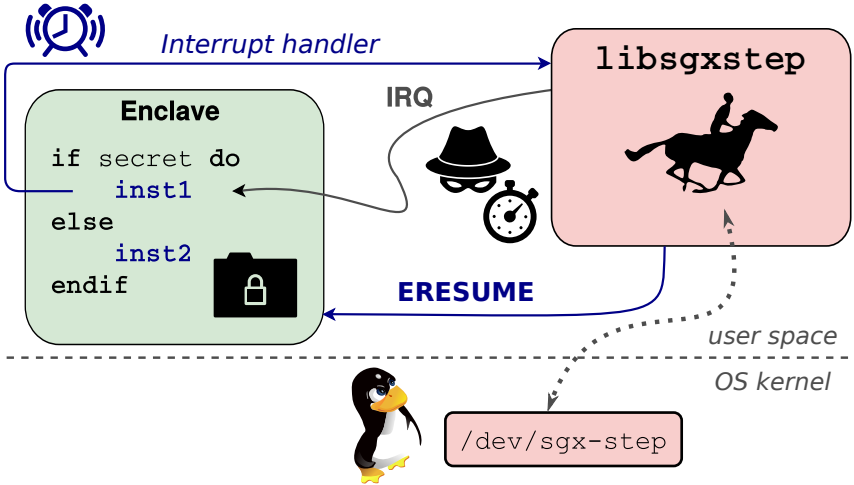
SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time



SGX-Step: Executing enclaves one instruction at a time

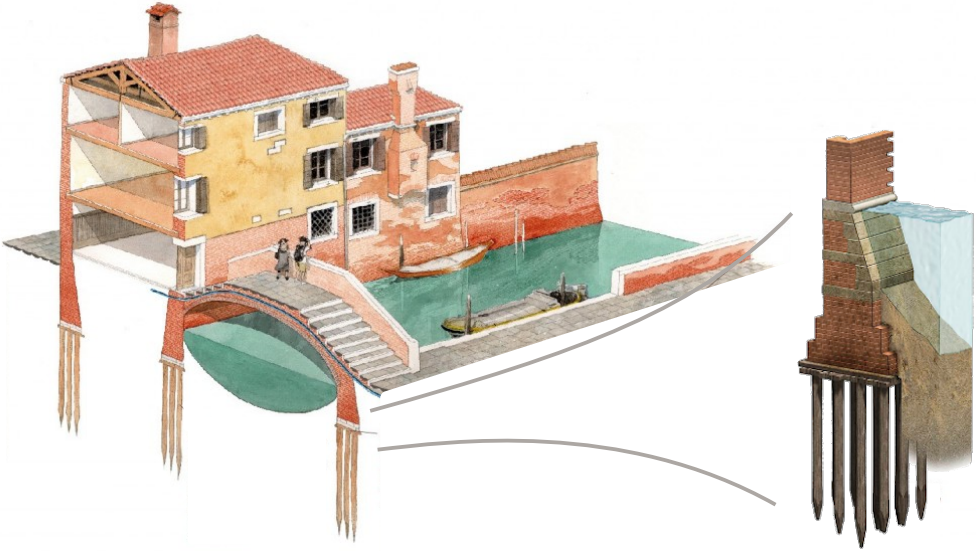


Building a deterministic password oracle with SGX-Step

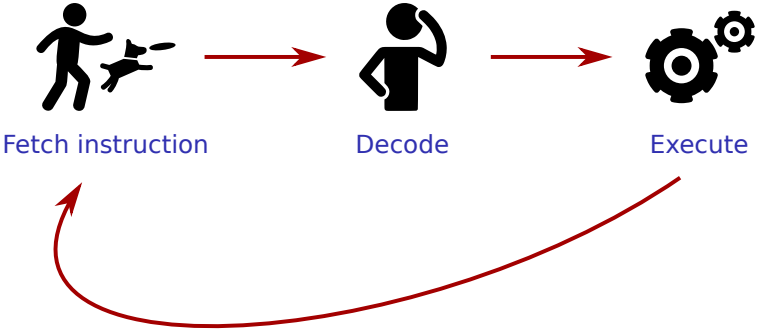
```
[idt.c] DTR.base=0xfffffe0000000000/size=4095 (256 entries)
[idt.c] established user space IDT mapping at 0x7f7ff8e9a000
[idt.c] installed asm IRQ handler at 10:0x56312d19b000
[idt.c] IDT[ 45] @0x7f7ff8e9a2d0 = 0x56312d19b000 (seg sel 0x10); p=1; dpl=3; type=14; ist=0
[file.c] reading buffer from '/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$
```

From architecture. . .

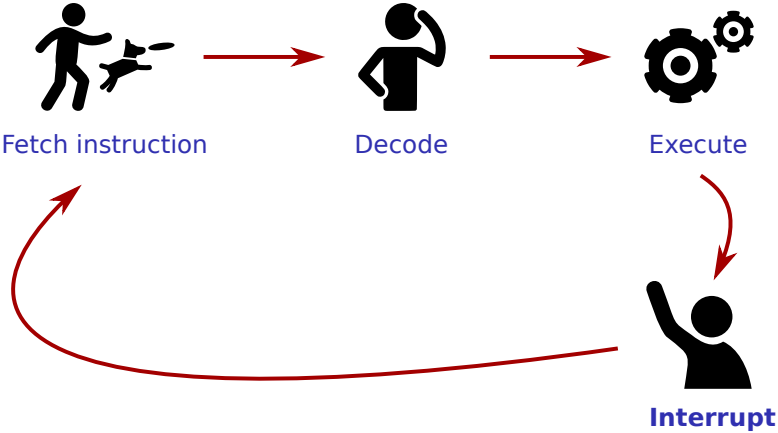
From architecture. . . to microarchitecture



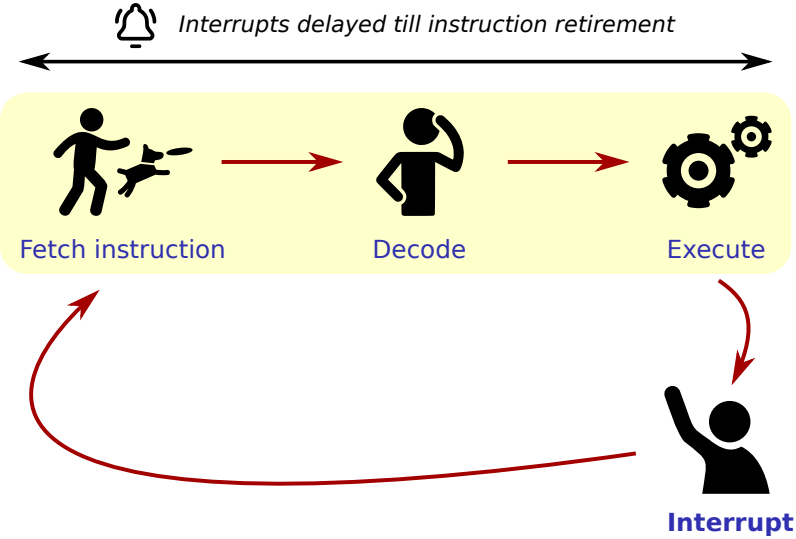
Back to basics: Fetch decode execute CPU operation



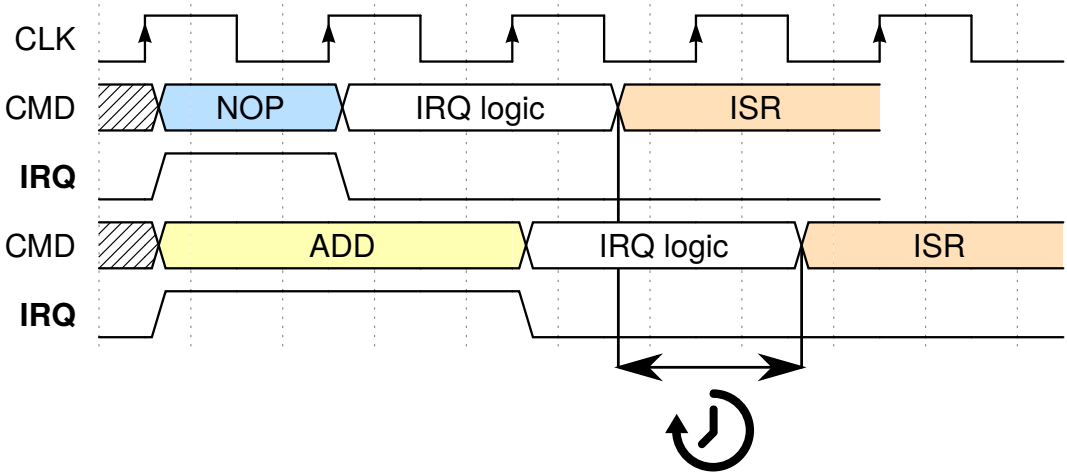
Back to basics: Fetch decode execute CPU operation



Back to basics: Fetch decode execute CPU operation

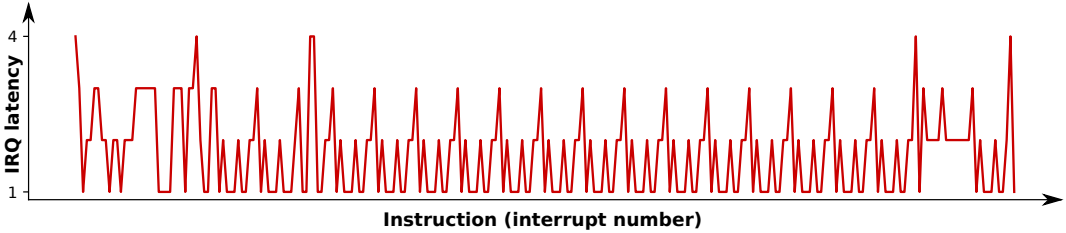


Wait a cycle: Interrupt latency as a side channel



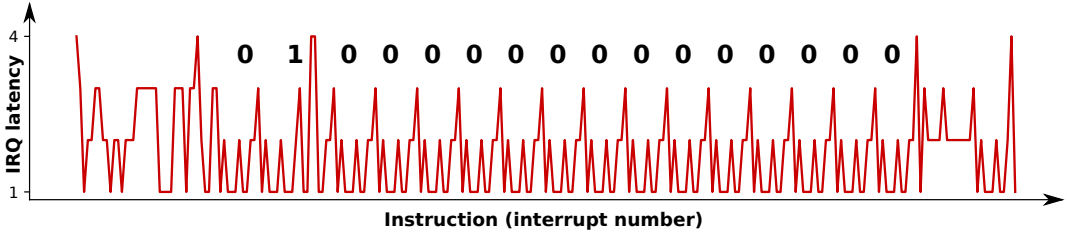

TIMING LEAKS
EVERYWHERE
imgflip.com

Nemesis attack: Inferring key strokes from Sancus enclaves



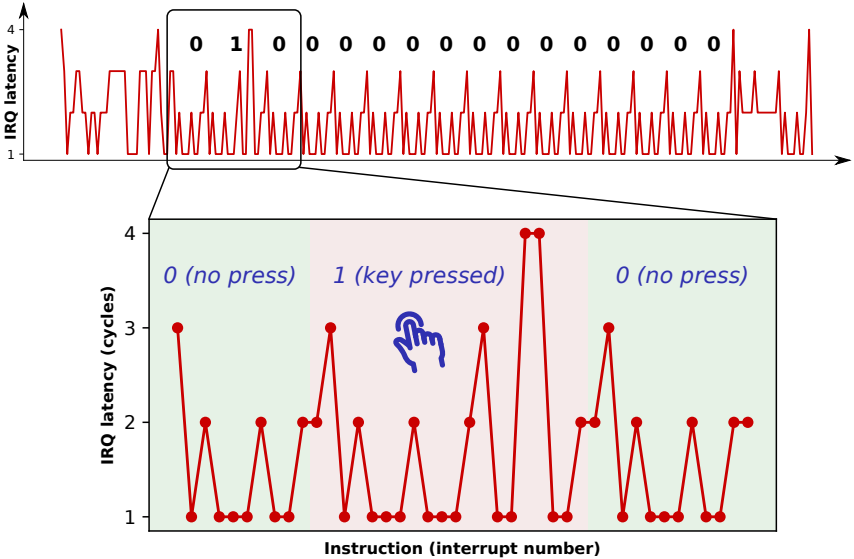
Enclave x-ray: Start-to-end trace enclaved execution

Nemesis attack: Inferring key strokes from Sancus enclaves



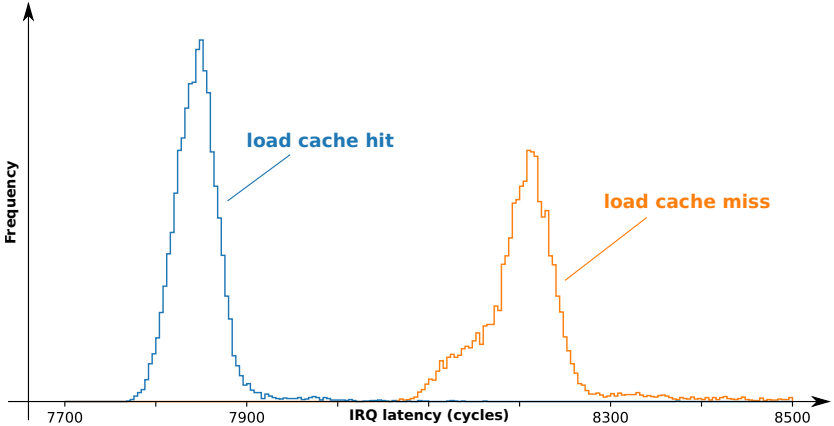
Enclave x-ray: Keymap bit traversal (ground truth)

Nemesis attack: Inferring key strokes from Sancus enclaves



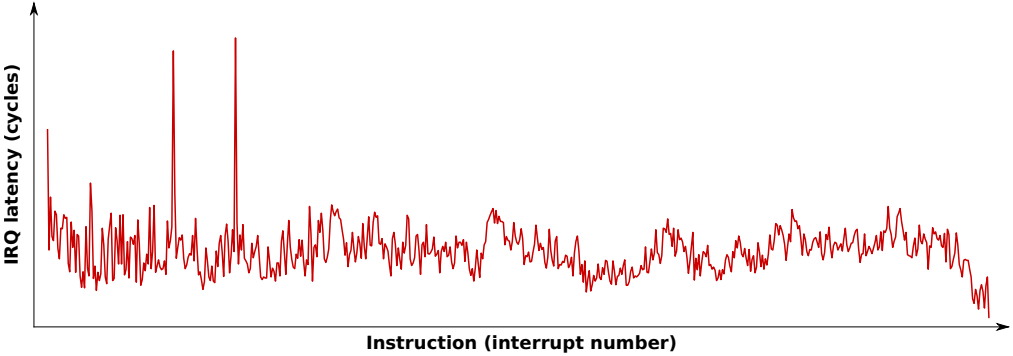
Intel SGX microbenchmarks: Measuring x86 cache misses

Timing leak: reconstruct *microarchitectural state*



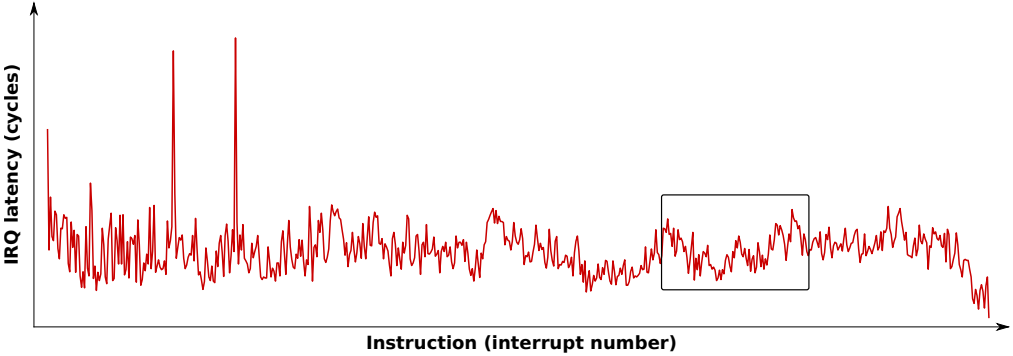
Single-stepping Intel SGX enclaves in practice

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



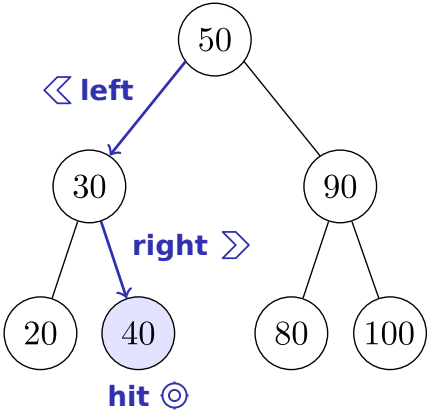
Single-stepping Intel SGX enclaves in practice

Enclave x-ray: Zooming in on `bsearch` function



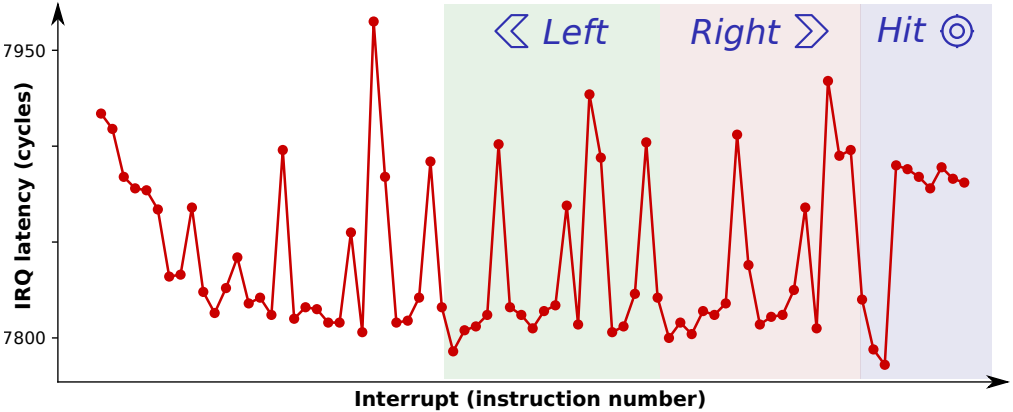
De-anonymizing SGX enclave lookups with interrupt latency

Adversary: Infer secret lookup in known sequence (e.g., DNA)

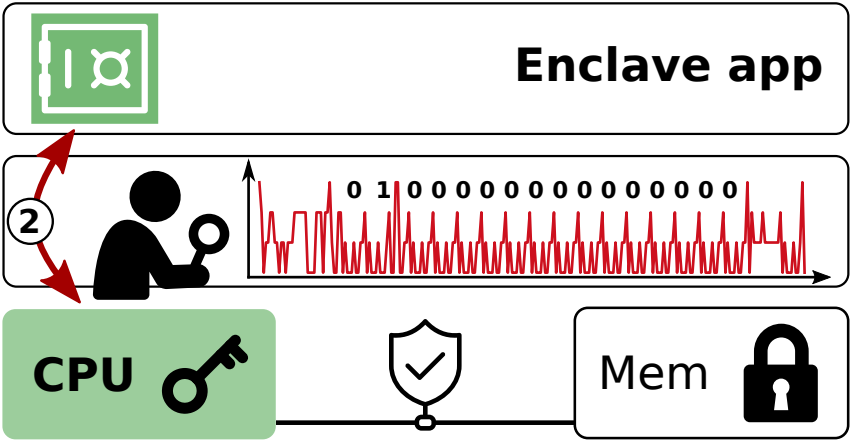


De-anonymizing SGX enclave lookups with interrupt latency

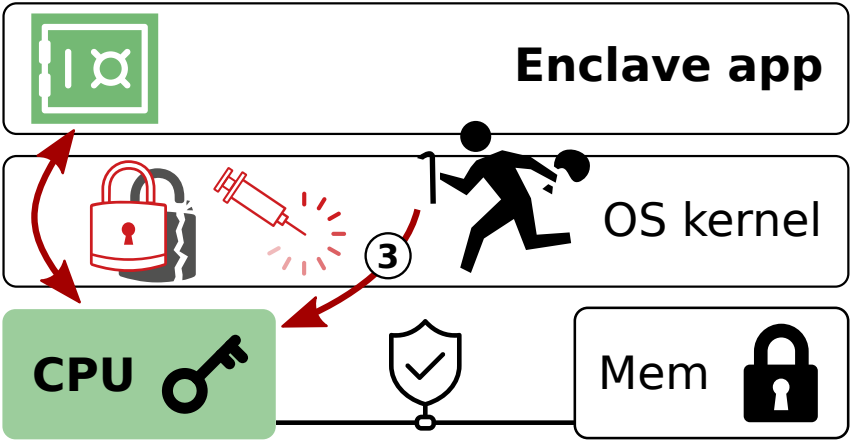
Goal: Infer lookup → reconstruct `bsearch` control flow

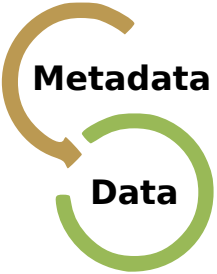


Thesis outline: Privileged side-channel attacks



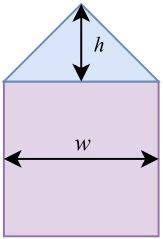
Thesis outline: Transient-execution attacks






WHAT IF I TOLD YOU
YOU CAN CHANGE RULES MID-GAME

Out-of-order and speculative execution

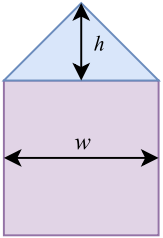



```
int area(int h, int w)
{
  int triangle = (w*h)/2;
  int square   = (w*w);
  return triangle + square;
}
```

Key **discrepancy:**

→ Programmers write sequential instructions

Out-of-order and speculative execution




```
int area(int h, int w)
{
  int triangle = (w*h)/2;
  int square   = (w*w);
  return triangle + square;
}
```

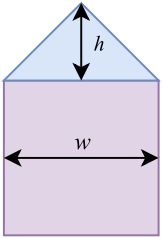
Key **discrepancy:**

→ Programmers write sequential instructions

↔ Modern CPUs are inherently parallel

⟹ *Execute instructions ahead of time*

Out-of-order and speculative execution



Roll-back

Overflow exception

```
int area(int h, int w)
{
  int triangle = (w*h)/2;
  int square   = (w*w);
  return triangle + square;
}
```

Key **discrepancy:**

→ Programmers write sequential instructions

↔ Modern CPUs are inherently parallel

⟹ *Execute instructions ahead of time*

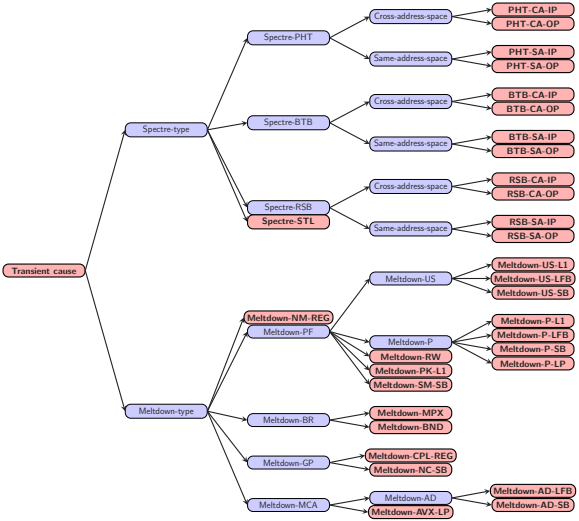
Best effort: What if triangle fails?

→ *Commit in-order, roll-back square*

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

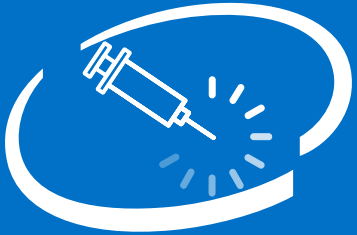
The transient-execution zoo

https://transient.fail

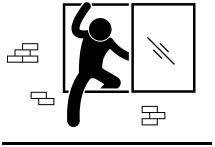



inside™


inside™


inside™

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

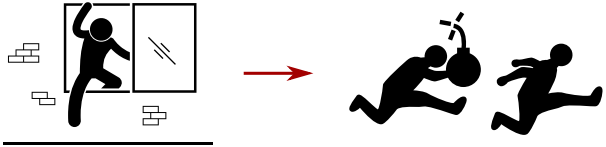
Listing 1: x86 assembly

```
meltdown:
  // %rdi: oracle
  // %rsi: secret_ptr

  movb (%rsi), %al
  shl $0xc, %rax
  movq (%rdi, %rax), %rdi
  retq
```

Listing 2: C code.

```
void meltdown(
        uint8_t *oracle,
        uint8_t *secret_ptr)
{
  uint8_t v = *secret_ptr;
  v = v * 0x1000;
  uint64_t o = oracle[v];
}
```

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

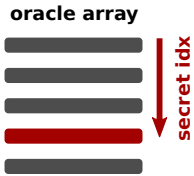
Transient out-of-order window

Listing 1: x86 assembly.

```
meltdown:
  // %rdi: oracle
  // %rsi: secret_ptr

  movb (%rsi), %al
  shl $0xc, %rax
  movq (%rdi, %rax), %rdi
  retq
```

Listing 2: C code.

```
void meltdown(
        uint8_t *oracle,
        uint8_t *secret_ptr)
{
  uint8_t v = *secret_ptr;
  v = v * 0x1000;
  uint64_t o = oracle[v];
}
```



Meltdown: Transiently encoding unauthorized memory

Unauthorized access → Transient out-of-order window → **Exception**

(discard architectural state)

Listing 1: x86 assembly.

```
meltdown:
  // %rdi: oracle
  // %rsi: secret_ptr

  movb (%rsi), %al
  shl $0xc, %rax
  movq (%rdi, %rax), %rdi
  retq
```

Listing 2: C code.

```
void meltdown(
        uint8_t *oracle,
        uint8_t *secret_ptr)
{
  uint8_t v = *secret_ptr;
  v = v * 0x1000;
  uint64_t o = oracle[v];
}
```

Meltdown: Transiently encoding unauthorized memory

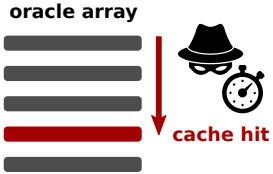
Unauthorized access → Transient out-of-order window → **Exception handler**

Listing 1: x86 assembly.

```
meltdown:
  // %rdi: oracle
  // %rsi: secret_ptr

  movb (%rsi), %al
  shl $0xc, %rax
  movq (%rdi, %rax), %rdi
  retq
```

Listing 2: C code.

```
void meltdown(
        uint8_t *oracle,
        uint8_t *secret_ptr)
{
  uint8_t v = *secret_ptr;
  v = v * 0x1000;
  uint64_t o = oracle[v];
}
```




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

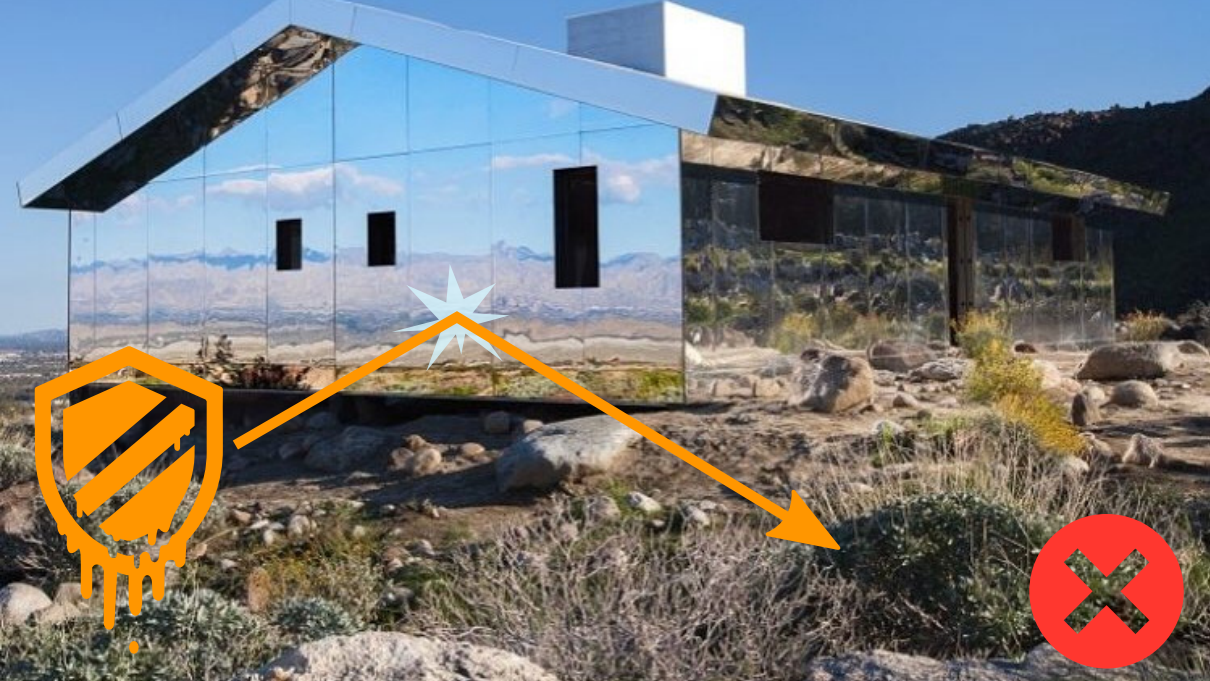
SPECTRE-LIKE FLAW UNDERMINES INTEL PROCESSORS' MOST SECURE ELEMENT

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

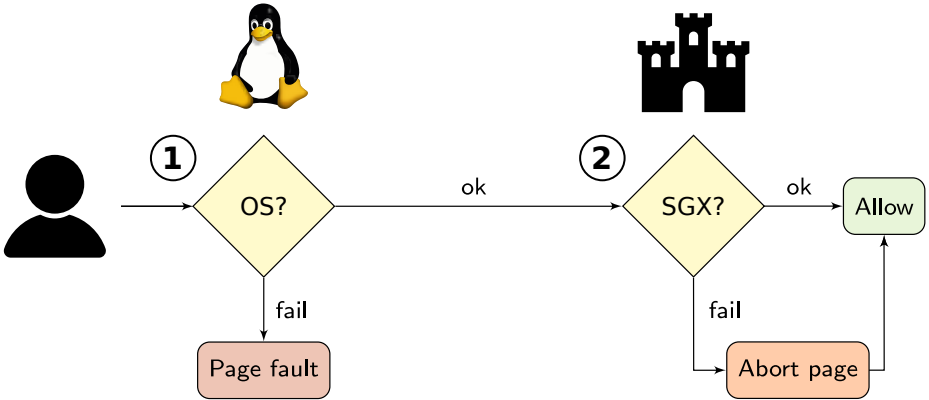
Intel's SGX blown wide open by, you guessed it, a speculative execution attack

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

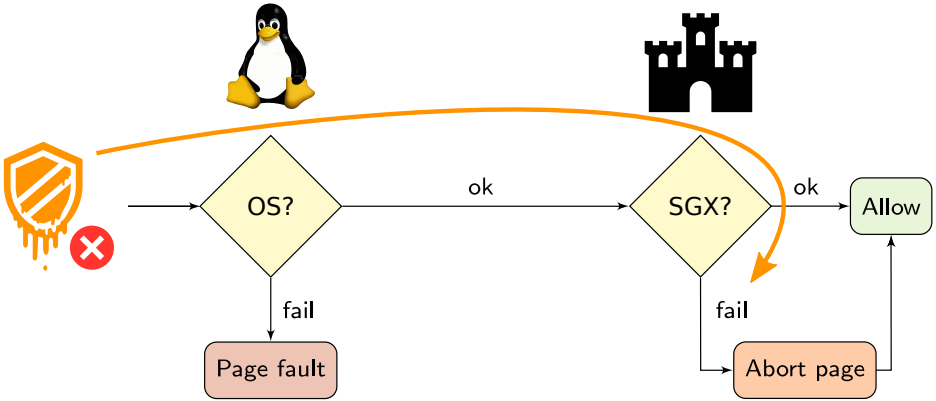
https://wired.com and https://arstechnica.com



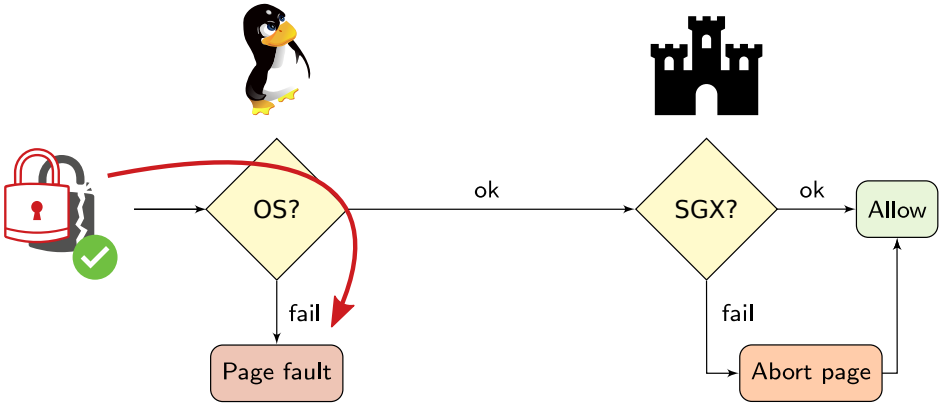
Building Foreshadow: Evade SGX abort page semantics



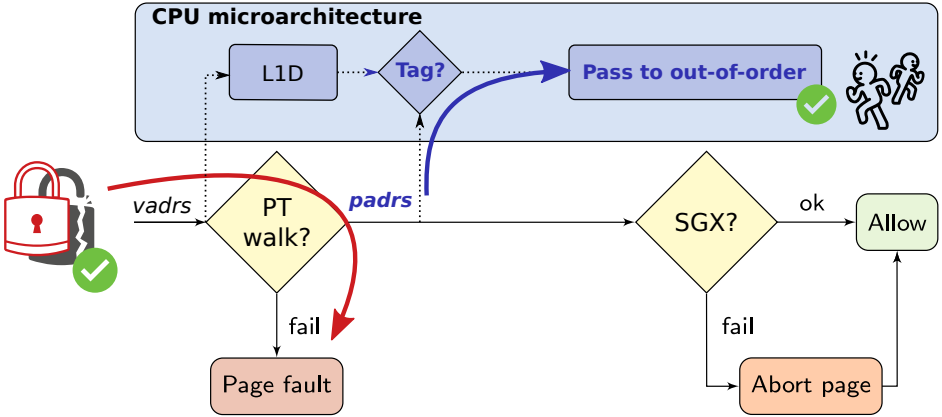
Building Foreshadow: Evade SGX abort page semantics



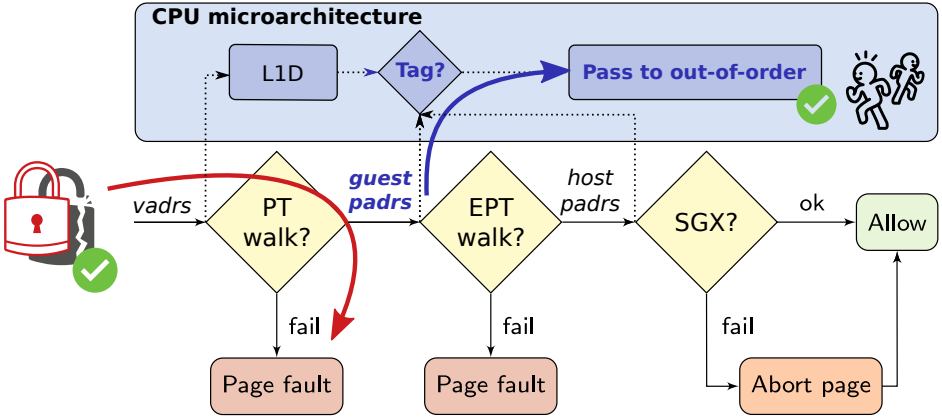
Building Foreshadow: Evade SGX abort page semantics

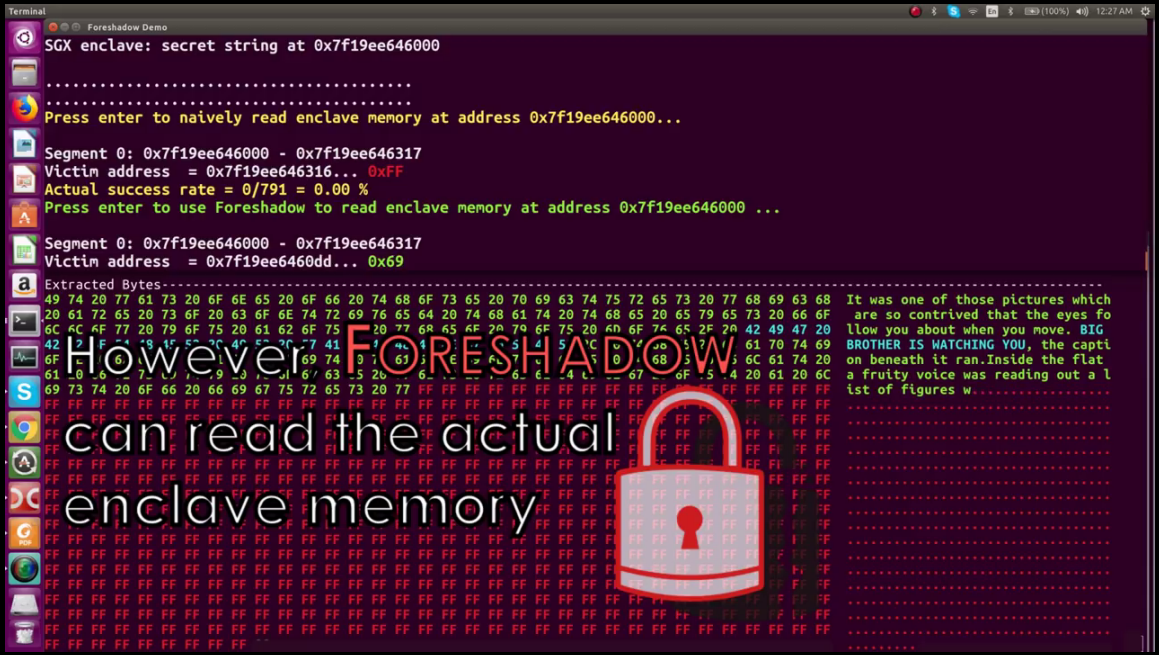


Foreshadow-SGX: Breaking enclave isolation



Foreshadow-NG: Breaking virtual machine isolation




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
It was one of those pictures which are so contrived that the eyes fo llow you about when you move. BIG BROTHER IS WATCHING YOU, the capti on beneath it ran.Inside the flat a fruity voice was reading out a l ist of figures w
However
FORESHADOW
can read the actual
enclave memory

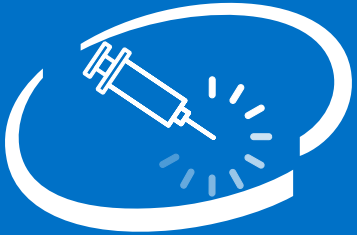
Mitigating Foreshadow: Flush CPU microarchitecture

Mitigating Foreshadow: Flush CPU microarchitecture

10BH	267	IA32_FLUSH_CMD	Flush Command (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™


THE WHITE HOUSE
6:14 PM
WHITE HOUSE
WASHINGTON
BREAKING NEWS
PRES. TRUMP UPDATES PUBLIC ON FEDERAL RESPONSE TO VIRUS
MSNBC

Idea: Can we turn Foreshadow around?

Outside view

- Meltdown: out-of-reach
- Foreshadow: cache emptied

Intra-enclave view

- Access enclave + outside memory

Idea: Can we turn Foreshadow around?

Outside view

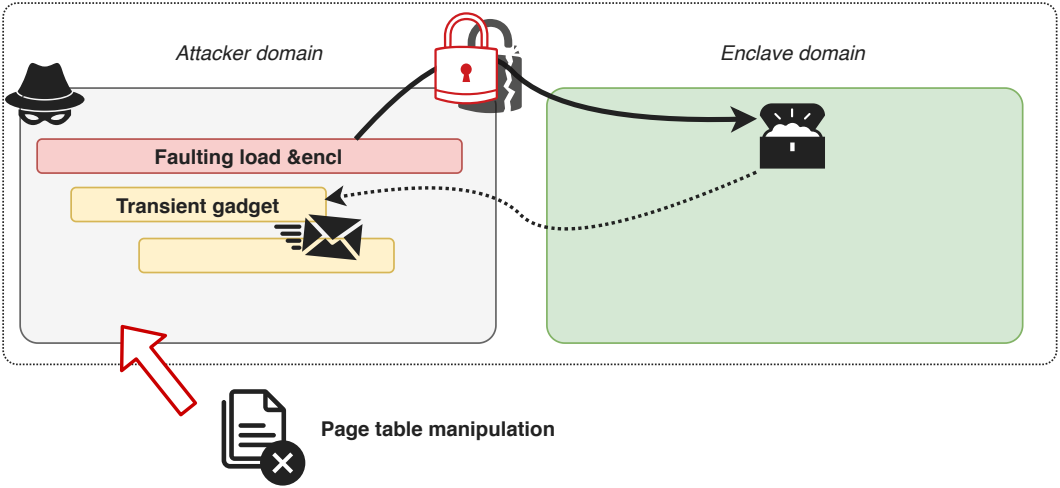
- Meltdown: out-of-reach
- Foreshadow: cache emptied

Intra-enclave view

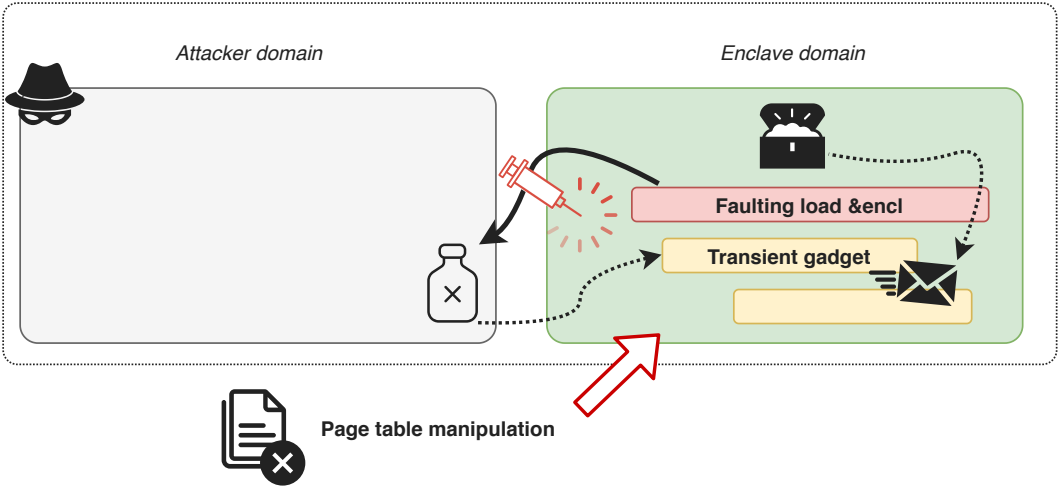
- Access enclave + outside memory

→ Abuse **in-enclave code gadgets!**

Reviving Foreshadow with Load Value Injection (LVI)



Reviving Foreshadow with Load Value Injection (LVI)



FOOD POISONING

Overdue products

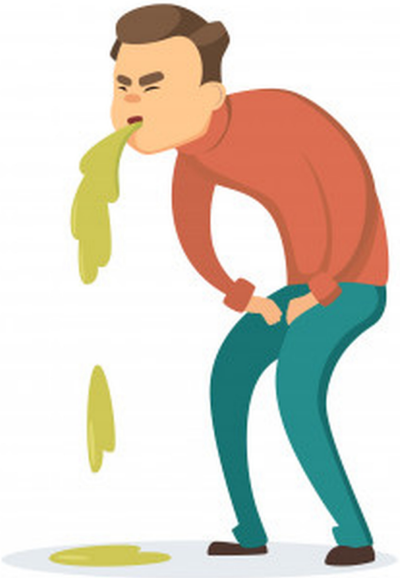
Medicine

Dizziness

Intestinal colic

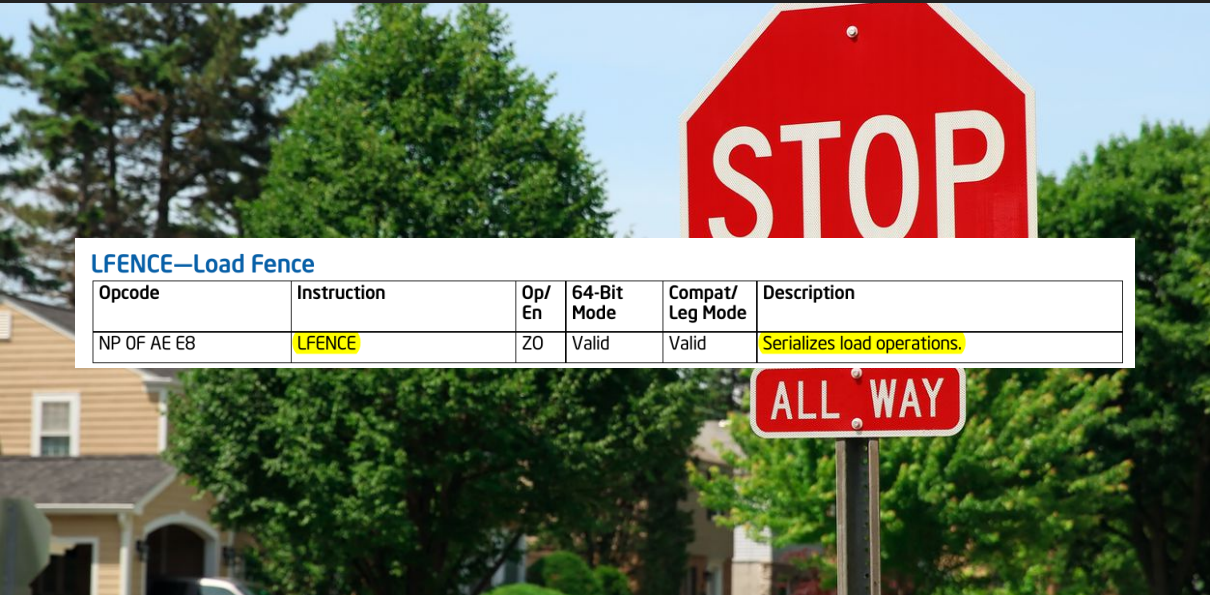
Diarrhea

Headache



Mitigating LVI: Fencing vulnerable load instructions

Mitigating LVI: Fencing vulnerable load instructions

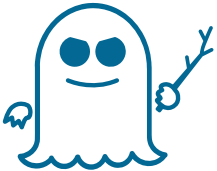


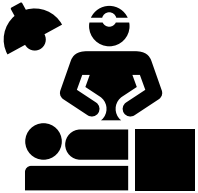
LFENCE—Load Fence

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

Intel architectural enclaves: `lfence` counts

`libsgx_qe.signed.so`



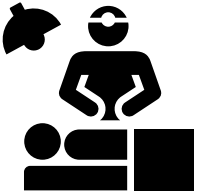


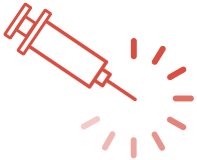
23 fences

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"

LVI performance impact

https://www.phoronix.com

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

The Brutal Performance Impact From Mitigating The LVI Vulnerability

Written by Michael Larabel in Software on 12 March 2020. **Page 1 of 6**. 76 Comments

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

Looking At The LVI Mitigation Impact On Intel Cascade Lake Refresh

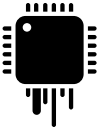
Written by Michael Larabel in Software on 5 April 2020. **Page 1 of 5**. 10 Comments

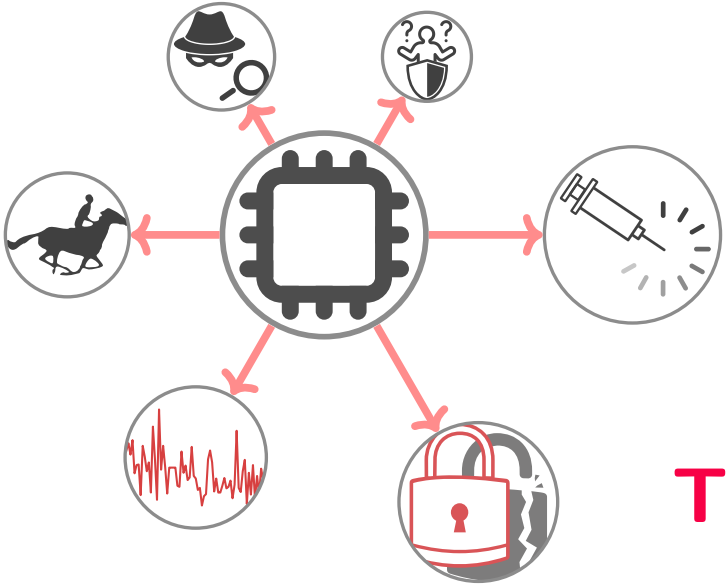
Conclusions and takeaway

⟹ **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!