Leaky Processors and the RISE of Hardware-Based Trusted Computing

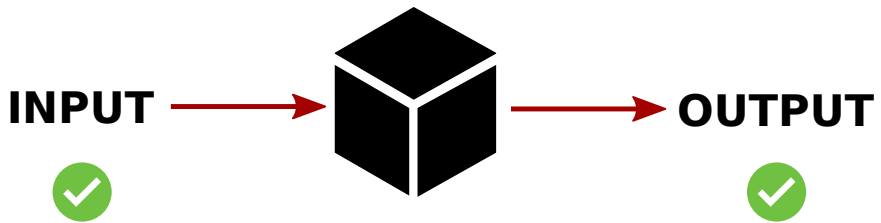
Jo Van Bulck

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

1st RISE Annual Conference, November 14, 2018

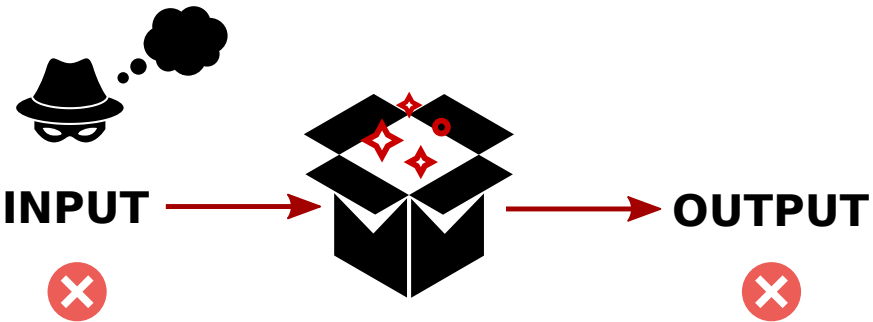
A primer on software security

Secure program: convert all input to *expected output*



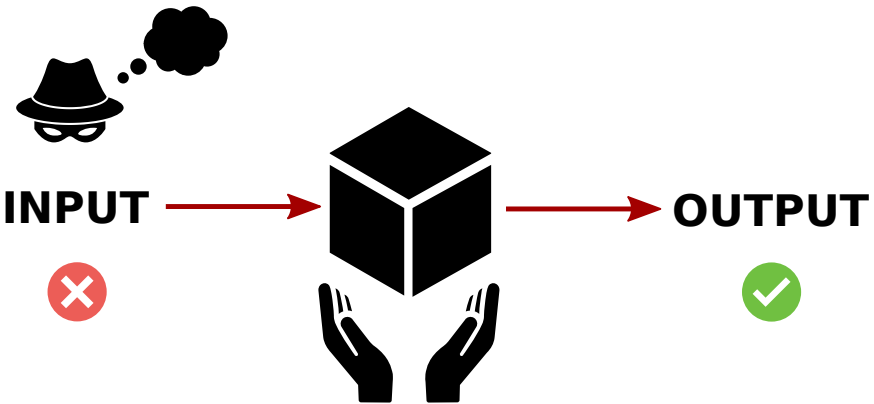
A primer on software security

Buffer overflow vulnerabilities: trigger *unexpected behavior*

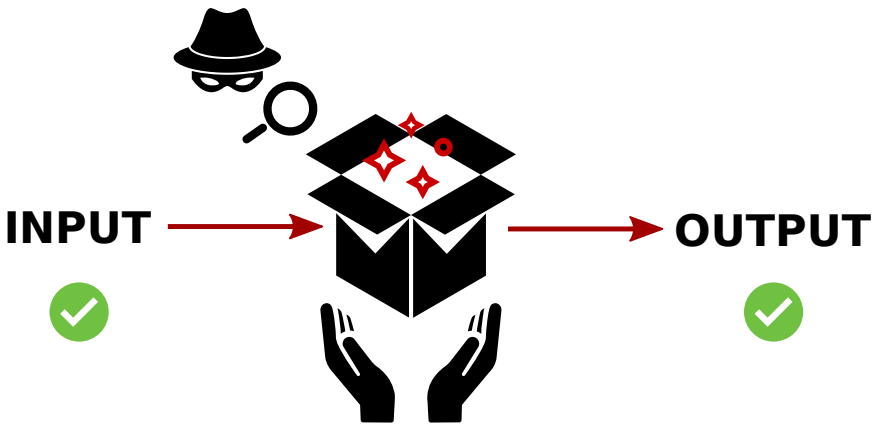


A primer on software security

Safe languages & formal verification: preserve *expected behavior*

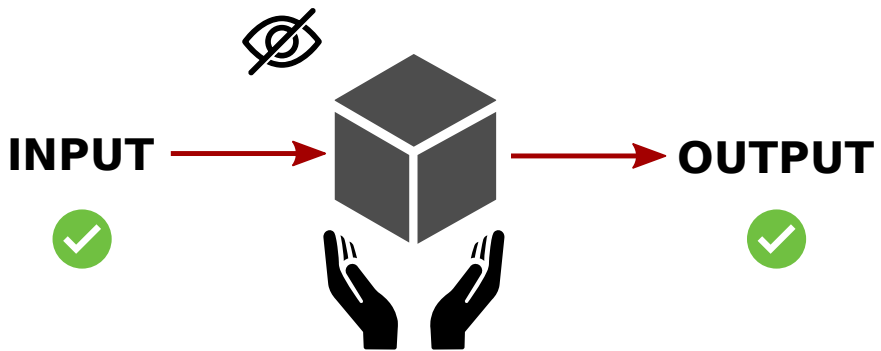


Side-channels: observe *side-effects* of the computation



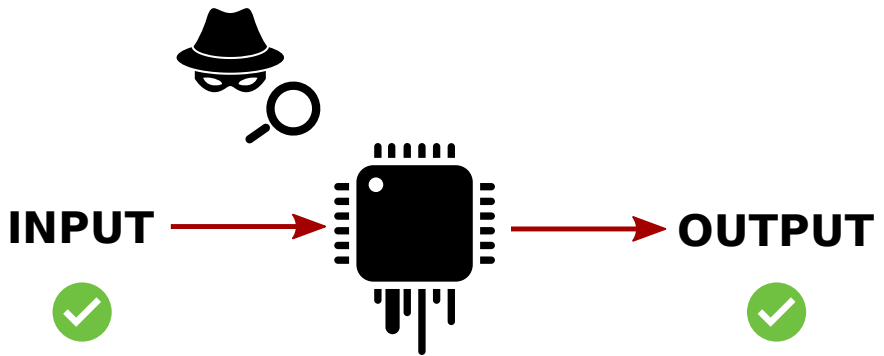
A primer on software security

Constant-time code: eliminate *secret-dependent* side-effects

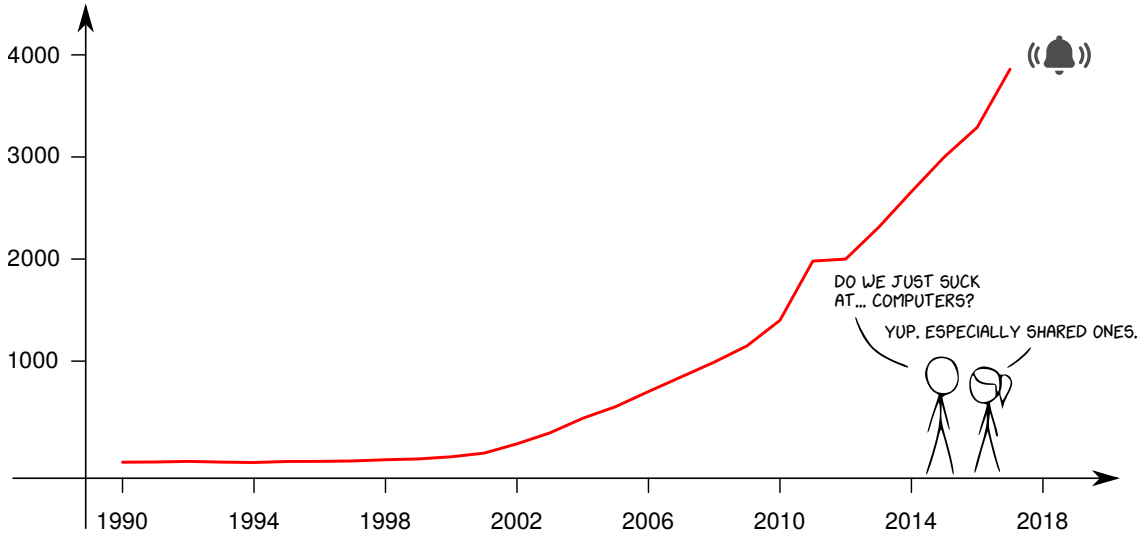


A primer on software security

Transient execution: *HW optimizations* do not respect SW abstractions (!)

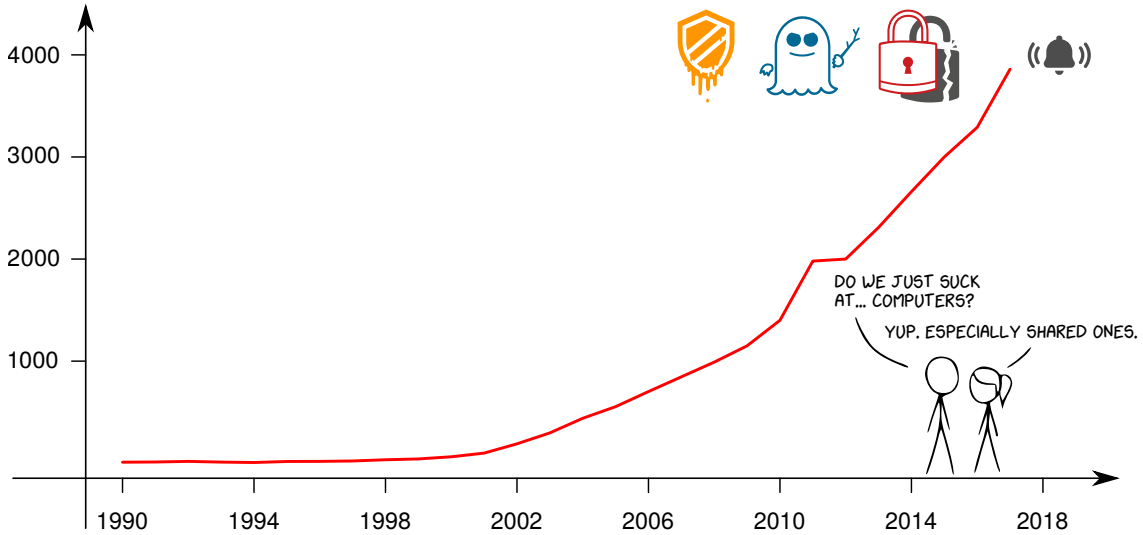


Evolution of “side-channel attack” occurrences in Google Scholar



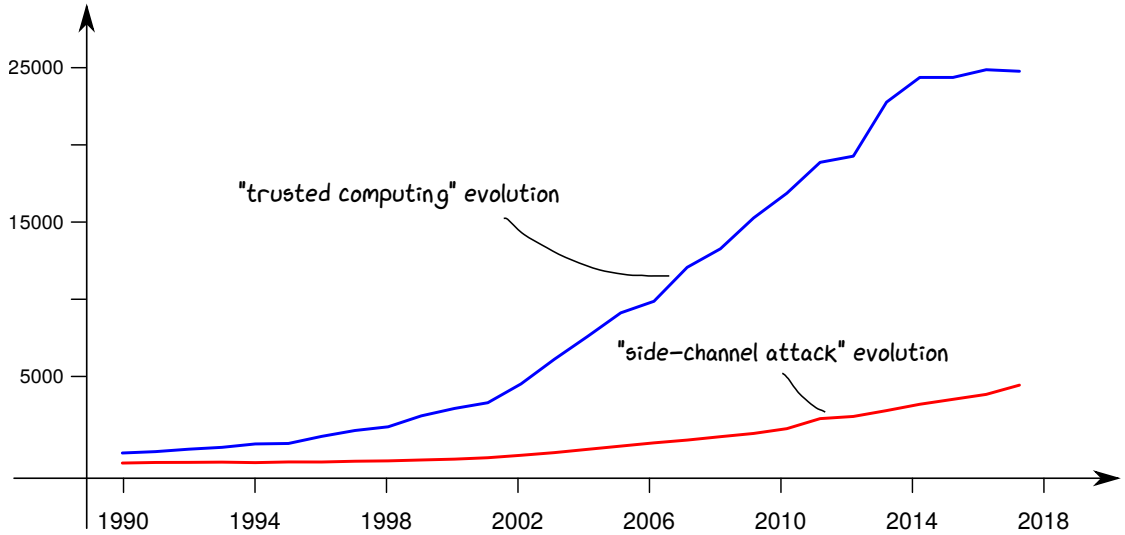
Based on github.com/Pold87/academic-keyword-occurrence and xkcd.com/1938/

Evolution of “side-channel attack” occurrences in Google Scholar



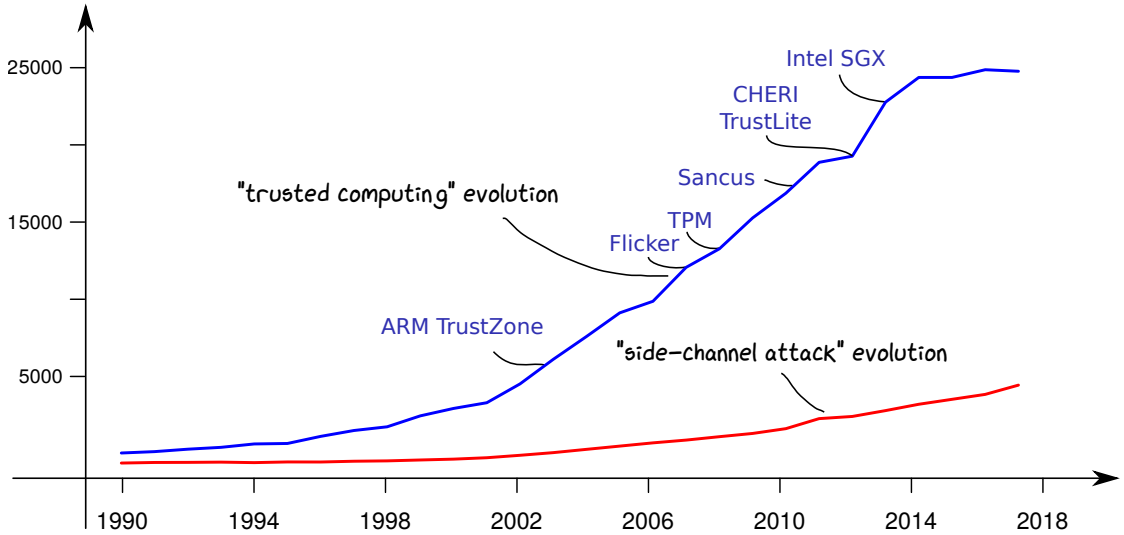
Based on github.com/Pold87/academic-keyword-occurrence and xkcd.com/1938/

The bigger picture: The RISE of hardware-based trusted computing



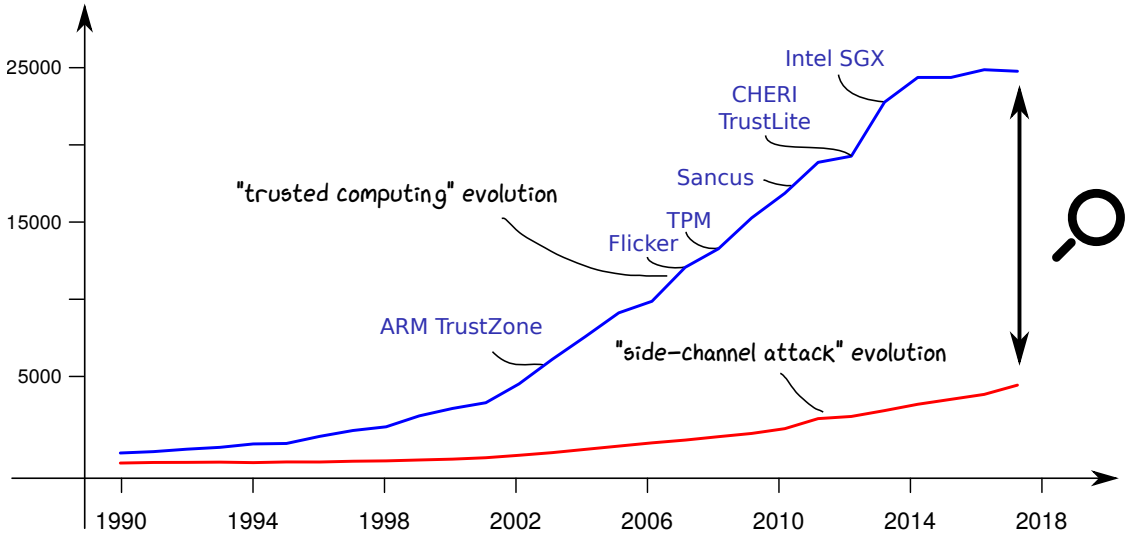
Based on github.com/Pold87/academic-keyword-occurrence

The bigger picture: The RISE of hardware-based trusted computing



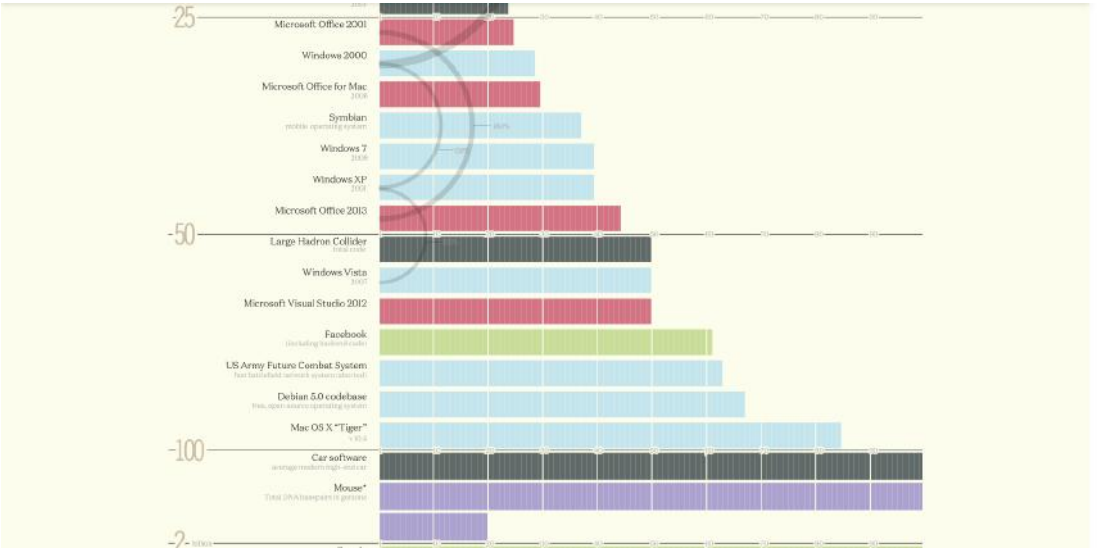
Based on github.com/Pold87/academic-keyword-occurrence

The bigger picture: The RISE of hardware-based trusted computing



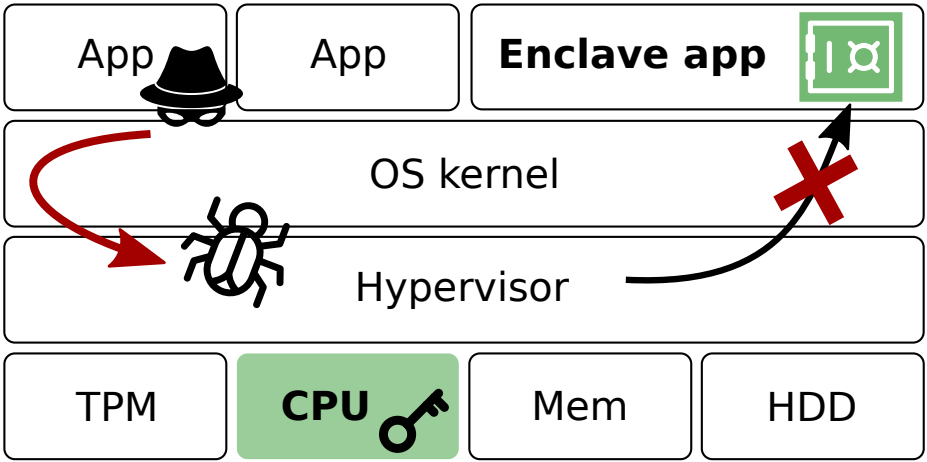
Based on github.com/Pold87/academic-keyword-occurrence

Enclaved execution attack surface: TCB reduction



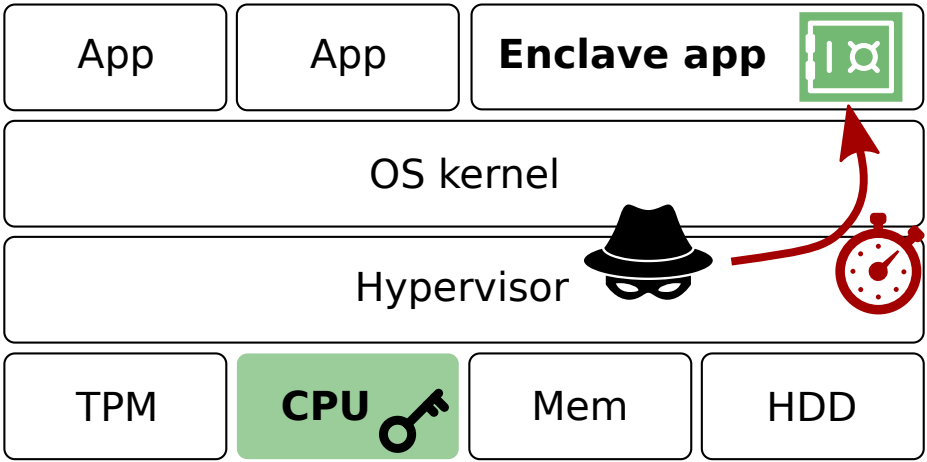
https://informationisbeautiful.net/visualizations/million-lines-of-code/

Enclaved execution attack surface: TCB reduction



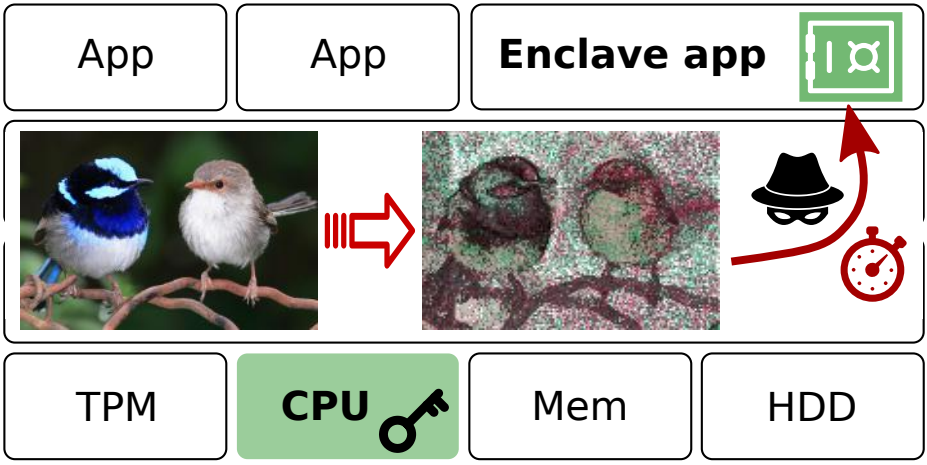
Intel SGX promise: hardware-level **isolation and attestation**

Enclaved execution attack surface: Privileged side-channel attacks



Untrusted OS → new class of powerful **side-channels**

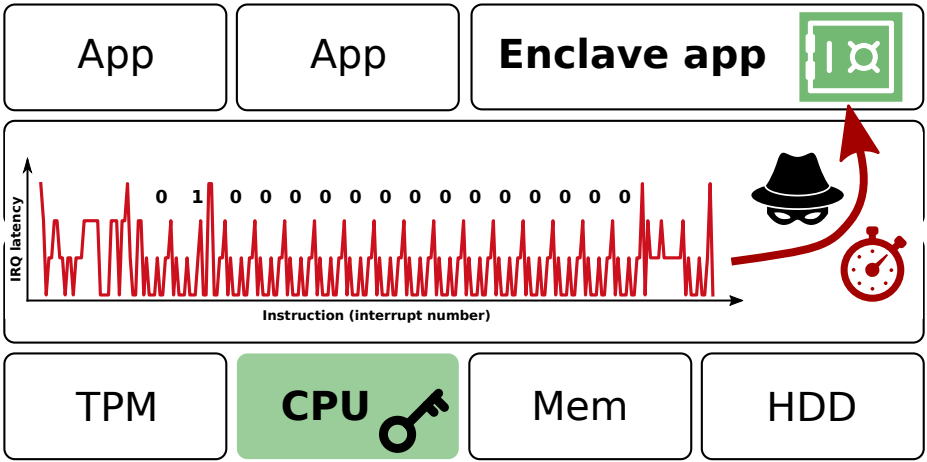
Enclaved execution attack surface: Privileged side-channel attacks



Untrusted OS → new class of powerful **side-channels**

Xu et al. "Controlled-channel attacks: Deterministic side-channels for untrusted operating systems", IEEE S&P 2015 [XCP15]

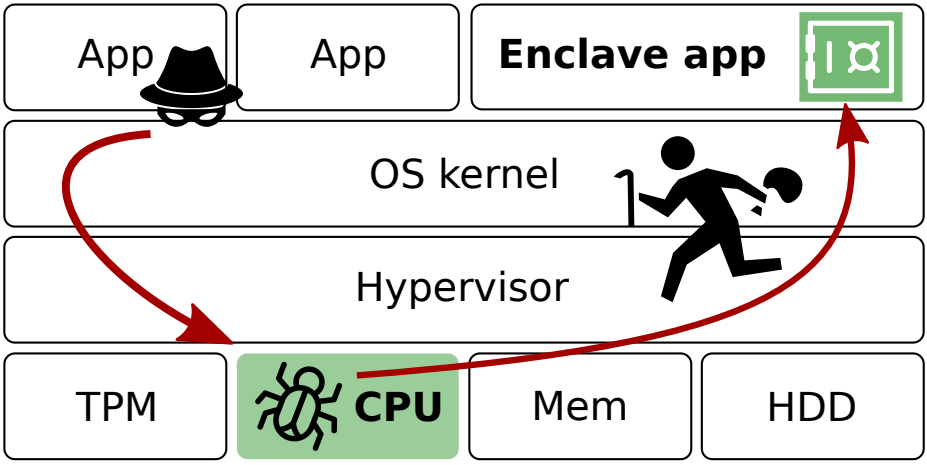
Enclaved execution attack surface: Privileged side-channel attacks



Untrusted OS → new class of powerful **side-channels**

Van Bulck et al. "Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic", CCS 2018 [VBPS18]

Enclaved execution attack surface: Transient execution attacks

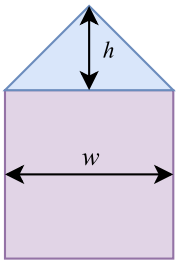


Trusted CPU → exploit **microarchitectural bugs/design flaws**

Van Bulck et al. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution", USENIX 2018 [VBMW+18]


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

Out-of-order and speculative execution

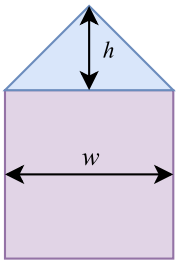



```
int area(int h, int w)
{
  int triangle = (w*h)/2;
  int square   = (w*w);
  return triangle + square;
}
```

Key **discrepancy:**

- Programmers write sequential instructions

Out-of-order and speculative execution



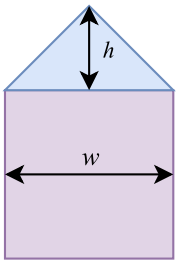

```
int area(int h, int w)
{
    int triangle = (w*h)/2;
    int square   = (w*w);
    return triangle + square;
}
```

Key **discrepancy:**

- Programmers write sequential instructions
- Modern CPUs are inherently parallel

$\Rightarrow$ *Speculatively 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

$\Rightarrow$ *Speculatively execute instructions ahead of time*

Best-effort: What if triangle fails?

$\rightarrow$ Commit in-order, roll-back square

$\ldots$ But side-channels may leave traces (!)


STRANGER THINGS
EXPLORING THE
UPSIDE DOWN

Transient execution attacks: Welcome to the world of fun!

CPU executes ahead of time in **transient world**

- Success → *commit* results to normal world ☺
- Fail → *discard* results, compute again in normal world ☹

Transient execution attacks: Welcome to the world of fun!

CPU executes ahead of time in **transient world**

- Success → *commit* results to normal world ☺
- Fail → *discard* results, compute again in normal world ☹

Transient world (microarchitecture) may temp bypass architectural software intentions:

Delayed exception handling

Control flow prediction

Transient execution attacks: Welcome to the world of fun!

Key finding of 2018

$\Rightarrow$ *Transmit secrets from transient to normal world*

Transient world (microarchitecture) may temp bypass architectural software intentions:

Delayed exception handling

Control flow prediction

Transient execution attacks: Welcome to the world of fun!

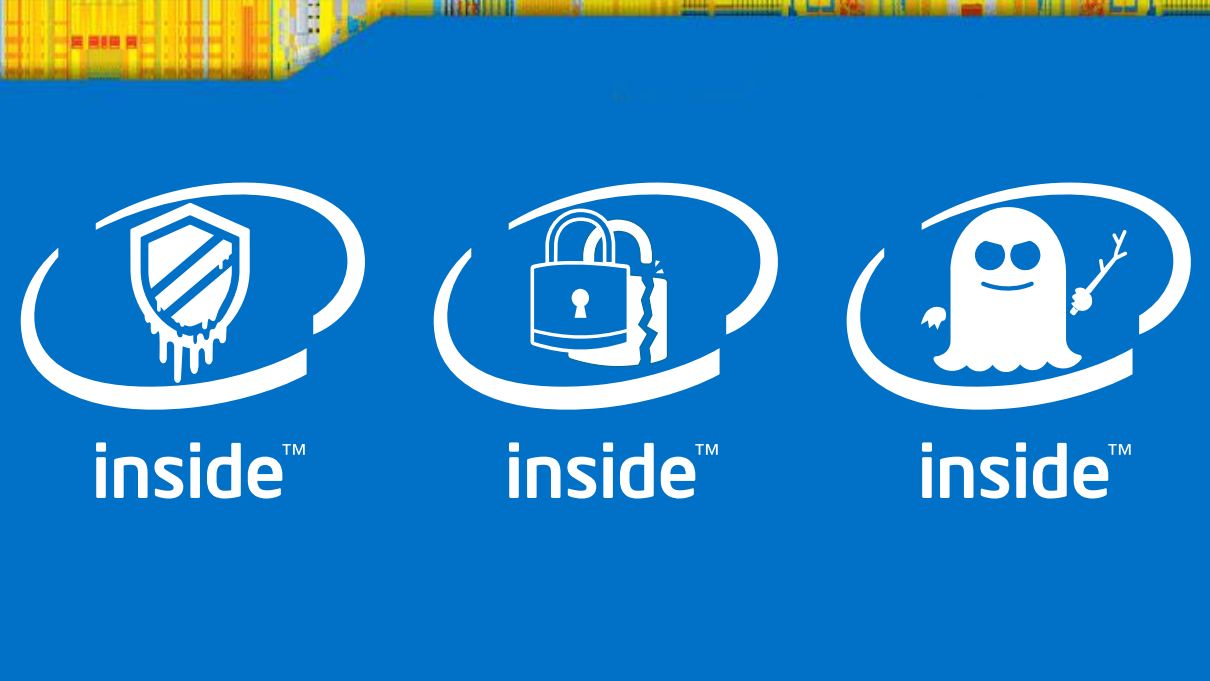
Key finding of 2018

$\Rightarrow$ *Transmit secrets from transient to normal world*

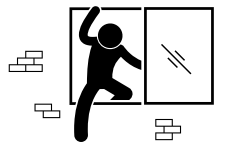
Transient world (microarchitecture) may temp bypass architectural software intentions:

CPU access control bypass

Speculative buffer overflow/ROP


inside™
inside™
inside™

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

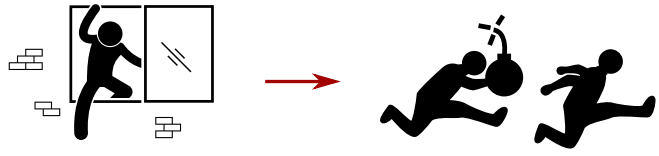
Listing 1: x86 assembly

```
1 meltdown:
2   // %rdi: oracle
3   // %rsi: secret_ptr
4
5   movb (%rsi), %al
6   shl $0xc, %rax
7   movq (%rdi, %rax), %rdi
8   retq
```

Listing 2: C code.

```
1 void meltdown(
2       uint8_t *oracle,
3       uint8_t *secret_ptr)
4 {
5   uint8_t v = *secret_ptr;
6   v = v * 0x1000;
7   uint64_t o = oracle[v];
8 }
```

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

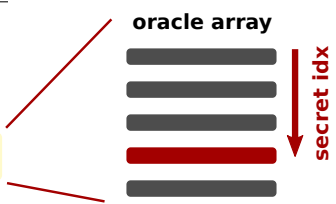
Transient out-of-order window

Listing 1: x86 assembly.

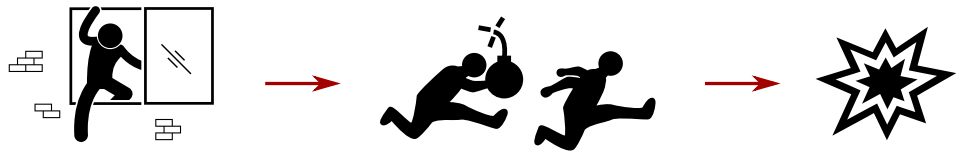
```
1 meltdown:
2     // %rdi: oracle
3     // %rsi: secret_ptr
4
5     movb (%rsi), %al
6     shl $0xc, %rax
7     movq (%rdi, %rax), %rdi
8     retq
```

Listing 2: C code.

```
1 void meltdown(
2         uint8_t *oracle,
3         uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```



Meltdown: Transiently encoding unauthorized memory



Unauthorized access

Transient out-of-order window

Exception
(discard architectural state)

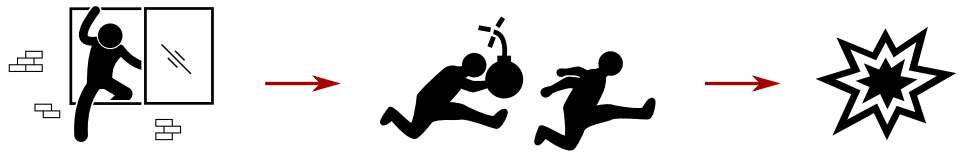
Listing 1: x86 assembly.

```
1 meltdown:
2     // %rdi: oracle
3     // %rsi: secret_ptr
4
5     movb (%rsi), %al
6     shl $0xc, %rax
7     movq (%rdi, %rax), %rdi
8     retq
```

Listing 2: C code.

```
1 void meltdown(
2         uint8_t *oracle,
3         uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```

Meltdown: Transiently encoding unauthorized memory



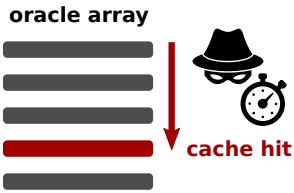
Unauthorized access | Transient out-of-order window | **Exception handler**

Listing 1: x86 assembly.

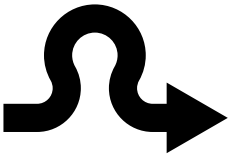
```
1 meltdown:
2     // %rdi: oracle
3     // %rsi: secret_ptr
4
5     movb (%rsi), %al
6     shl $0xc, %rax
7     movq (%rdi, %rax), %rdi
8     retq
```

Listing 2: C code.

```
1 void meltdown(
2         uint8_t *oracle,
3         uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```



Mitigating Meltdown: Unmap kernel addresses from user space

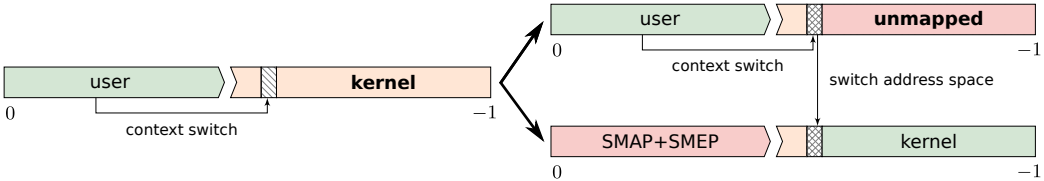


- OS software fix for faulty hardware ($\leftrightarrow$ future CPUs)

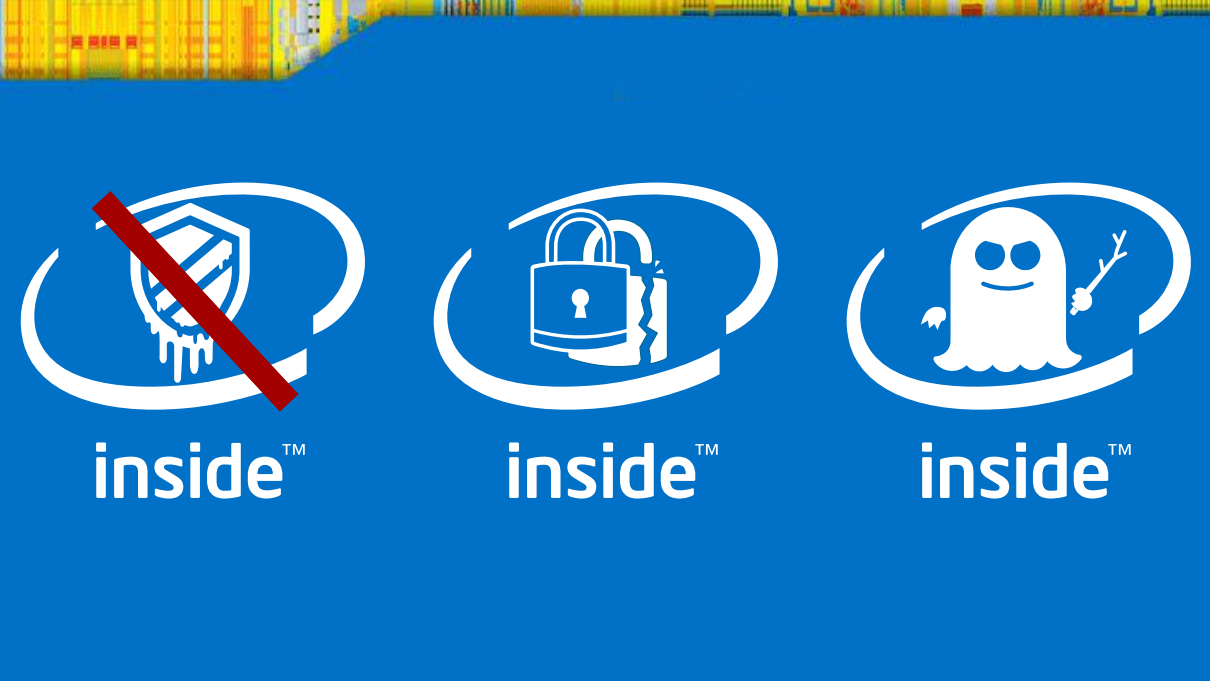
Mitigating Meltdown: Unmap kernel addresses from user space

- OS software fix for faulty hardware (↔ future CPUs)
- Unmap kernel from user *virtual address space*

→ Unauthorized physical addresses out-of-reach (˜cookie jar)



Gruss et al. "KASLR is dead: Long live KASLR", ESSoS 2017 [GLS+17]


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

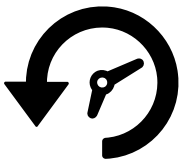
I'M SURE THIS WON'T BE THE LAST SUCH PROBLEM —

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

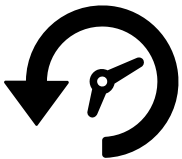


1. Cache secrets in L1

2. Unmap page table entry

3. Execute Meltdown

Building Foreshadow



1. Cache secrets in L1

2. Unmap page table entry

3. Execute Meltdown

L1 terminal fault challenges

Foreshadow can read unmapped physical addresses from the cache (!)

Challenge: Reading unmapped secrets with Foreshadow

Untrusted world view

- Enclaved memory reads `0xFF`

Intra-enclave view

- Access enclaved + unprotected memory

Challenge: Reading unmapped secrets with Foreshadow

Untrusted world view

- Enclaved memory reads `0xFF`

Intra-enclave view

- Access enclaved + unprotected memory
- SGXpectre in-enclave code abuse

Challenge: Reading unmapped secrets with Foreshadow

Untrusted world view

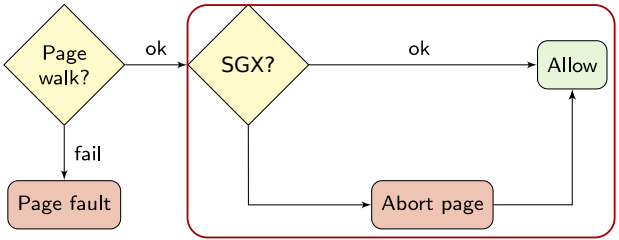
- Enclaved memory reads `0xFF`
- Meltdown "bounces back" (~ mirror)

Intra-enclave view

- Access enclaved + unprotected memory
- SGXpectre in-enclave code abuse

Building Foreshadow: Evade SGX abort page semantics

Note: SGX MMU sanitizes *untrusted* address translation

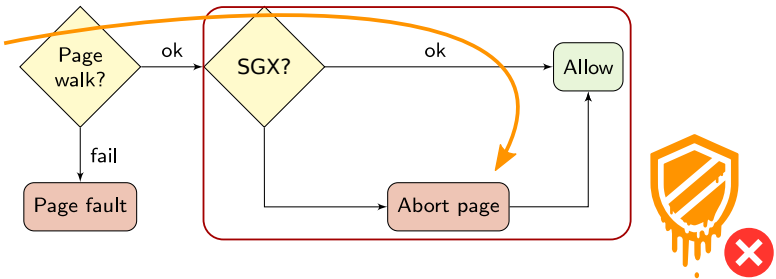


Abort page semantics:
An attempt to read from a non-existent or disallowed resource returns all ones for data (abort page). An attempt to write to a non-existent or disallowed physical resource is dropped. This behavior is unrelated to exception type abort (the others being Fault and Trap).

`https://software.intel.com/en-us/sgx-sdk-dev-reference-enclave-development-basics`

Building Foreshadow: Evade SGX abort page semantics

Straw man: (Transient) accesses in non-enclave mode are dropped

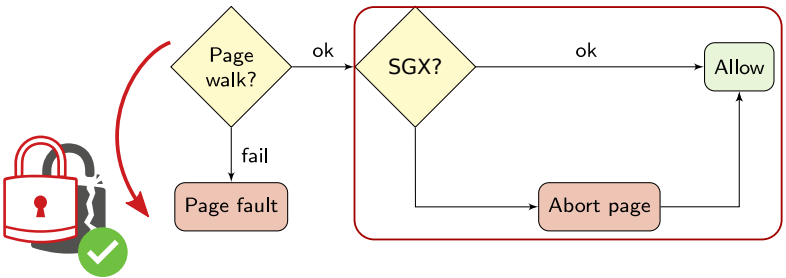


Abort page semantics:
An attempt to read from a non-existent or disallowed resource returns all ones for data (abort page). An attempt to write to a non-existent or disallowed physical resource is dropped. This behavior is unrelated to exception type abort (the others being Fault and Trap).

https://software.intel.com/en-us/sgx-sdk-dev-reference-enclave-development-basics

Building Foreshadow: Evade SGX abort page semantics

Stone man: Bypass abort page via *untrusted* page table

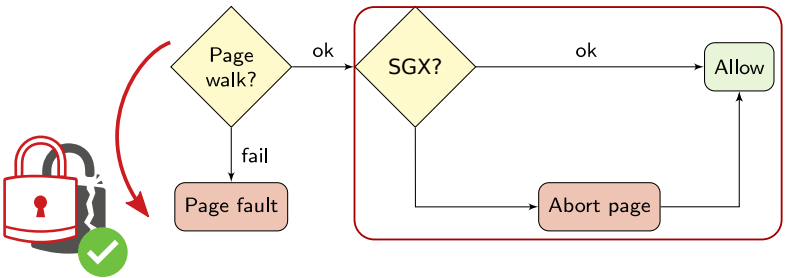


Xu et al. "Controlled-channel attacks: Deterministic side-channels for untrusted operating systems", IEEE S&P 2015 [XCP15]

Van Bulck et al. "Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution", USENIX 2017 [VBWK+17]

Building Foreshadow: Evade SGX abort page semantics

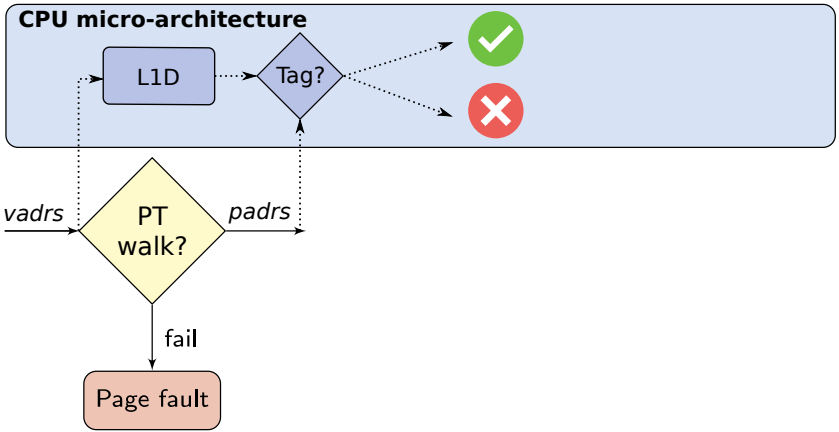
Stone man: Bypass abort page via *untrusted* page table



Unprivileged system call

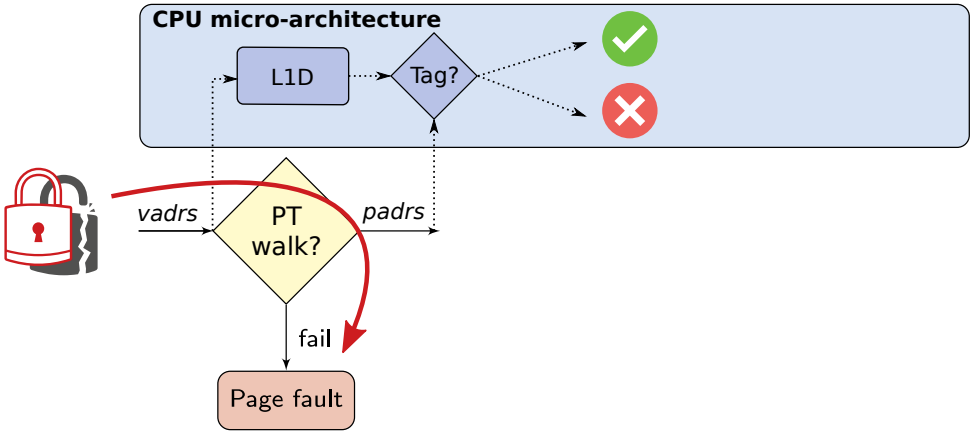
```
mprotect( secret_ptr & 0xFFF, 0x1000, PROT_NONE );
```

Foreshadow-NG: Breaking the virtual memory abstraction



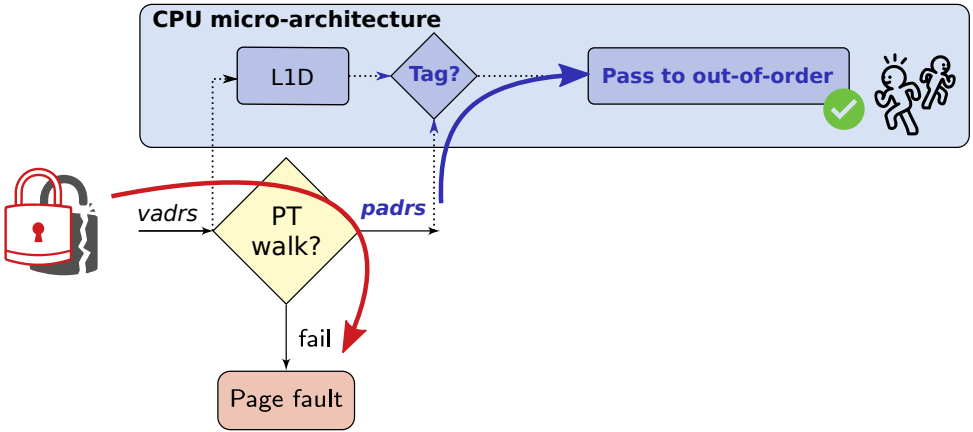
L1 cache design: Virtually-indexed, physically-tagged

Foreshadow-NG: Breaking the virtual memory abstraction



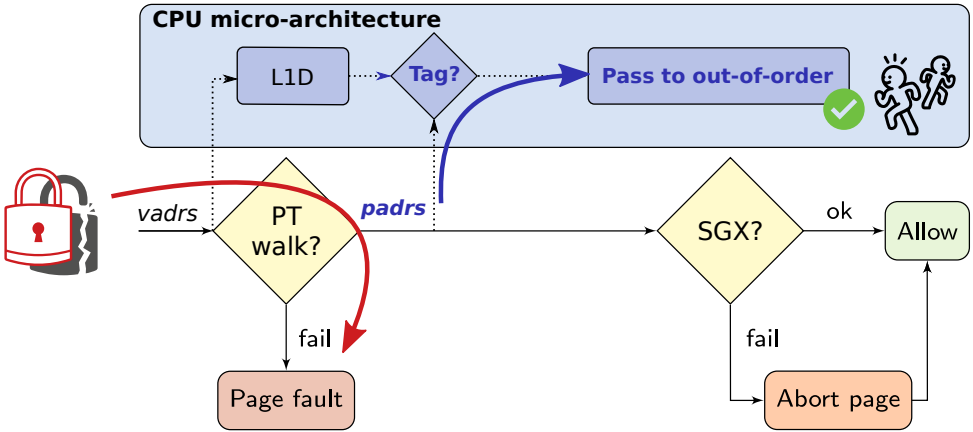
Page fault: Early-out address translation

Foreshadow-NG: Breaking the virtual memory abstraction



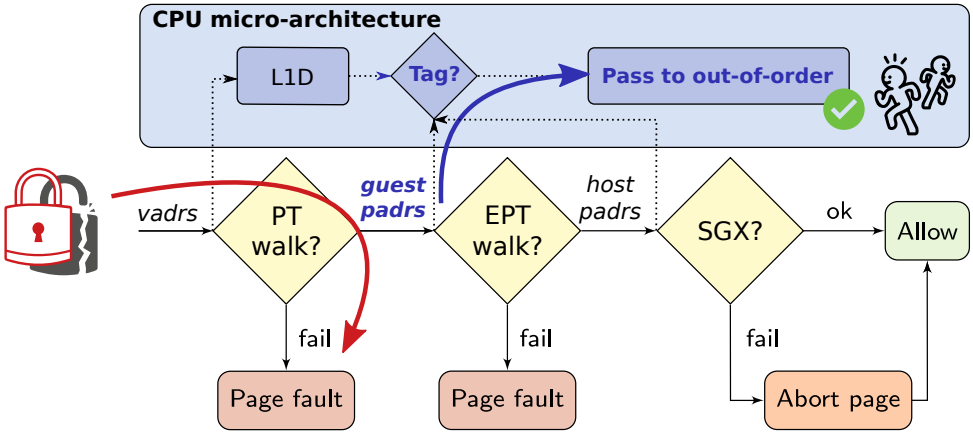
L1-Terminal Fault: match *unmapped physical address* (!)

Foreshadow-NG: Breaking the virtual memory abstraction



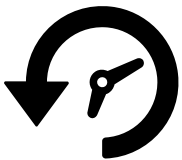
Foreshadow-SGX: bypass enclave isolation

Foreshadow-NG: Breaking the virtual memory abstraction



Foreshadow-VMM: bypass virtual machine isolation

Mitigating Foreshadow

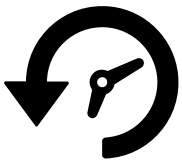


1. Cache secrets in L1

2. Unmap page table entry

3. Execute Meltdown

Mitigating Foreshadow



1. Cache secrets in L1

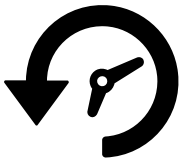
2. Unmap page table entry

3. Execute Meltdown

Future CPUs
(silicon-based changes)

https://newsroom.intel.com/editorials/advancing-security-silicon-level/

Mitigating Foreshadow



1. Cache secrets in L1

2. Unmap page table entry

3. Execute Meltdown

OS kernel updates
(sanitize page frame bits)

https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/L1TF

Mitigating Foreshadow

1. Cache secrets in L1

Intel microcode updates

2. Unmap page table entry

3. Execute Meltdown

⇒ **Flush L1** cache on enclave/VMM exit + **disable HyperThreading**

https://software.intel.com/security-software-guidance/software-guidance/l1-terminal-fault

Some good news?

A lingering risk: Because Foreshadow, Spectre, and Meltdown are all hardware-based flaws, there's no guaranteed fix short of swapping out the chips. But security experts say the weaknesses are incredibly hard to exploit and that there's no evidence so far to suggest this year's chipocalypse has led to a hacking spree. Still, if your computer offers you an urgent software upgrade, be sure to take it immediately.

https://www.technologyreview.com/the-download/611879/intels-foreshadow-flaws-are-the-latest-sign-of-the-chipocalypse/

For the latest Intel security news, please visit security newsroom.

For all others, visit the Intel Security Center for the latest security information.

L1TF is a highly sophisticated attack method, and today, Intel is not aware of any reported real-world exploits.

https://www.intel.com/content/www/us/en/architecture-and-technology/l1tf.html

Some good news?

Azure confidential computing: Microsoft boosts security for cloud data

Microsoft is rolling out new secure enclave technology for protecting data in use.

By Liam Tung | September 18, 2017 -- 13:17 GMT (14:17 BST) | Topic: Cloud

https://www.zdnet.com/article/azure-confidential-computing-microsoft-boosts-security-for-cloud-data/

Some good news?

Azure confidential computing: Microsoft boosts security for cloud data

Microsoft is rolling out new secure enclave technology for protecting data in use.

By Liam Tung | September 18, 2017 -- 13:17 GMT (14:17 BST) | Topic: Cloud

https://www.zdnet.com/article/azure-confidential-computing-microsoft-boosts-security-for-cloud-data/

Foreshadow fallout: Dismantling the SGX ecosystem

Remote attestation and secret provisioning

Challenge-response to prove **enclave identity**

Foreshadow fallout: Dismantling the SGX ecosystem

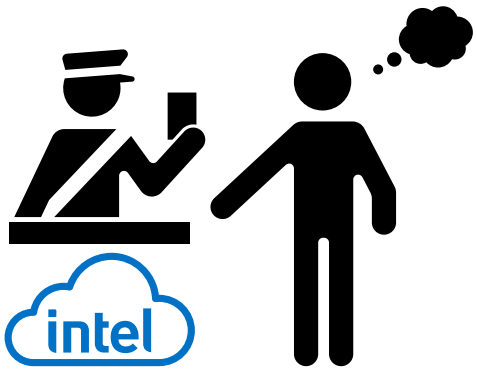
CPU-level key derivation

Intel == trusted 3th party (shared **CPU master secret**)

Foreshadow fallout: Dismantling the SGX ecosystem

CPU-level key derivation

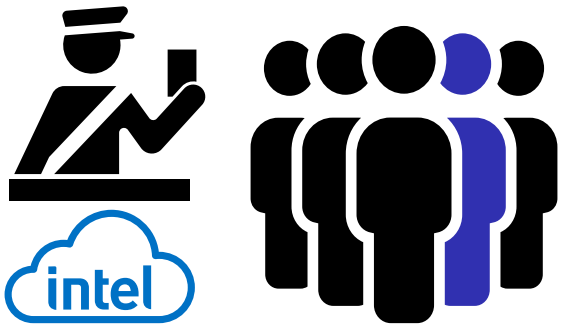
Intel == trusted 3th party (shared **CPU master secret**)



Foreshadow fallout: Dismantling the SGX ecosystem

Fully anonymous attestation

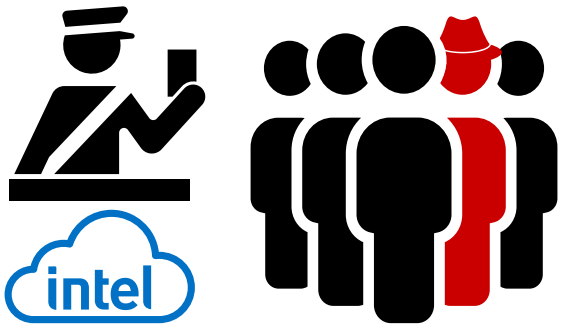
Intel Enhanced Privacy ID (EPID) **group signatures** ☺



Foreshadow fallout: Dismantling the SGX ecosystem

The dark side of anonymous attestation

Single **compromised EPID key** affects millions of devices . . . ☹

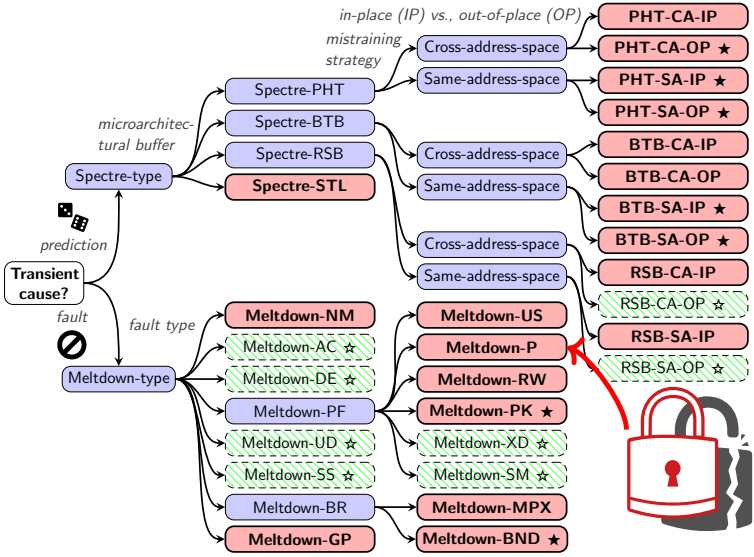


Foreshadow fallout: Dismantling the SGX ecosystem

EPID key extraction with Foreshadow

Active **man-in-the-middle:** read + modify all local and remote secrets (!)

Research challenges: Universal classification and evaluation



Canella et al. "A Systematic Evaluation of Transient Execution Attacks and Defenses", arXiv preprint [CVBS$^+$18]

OPINION

Reflections on Post-Meltdown Trusted Computing

A Case for Open Security Processors

JAN TOBIAS MÜHLBERG AND JO VAN BULCK

;login:
THE USENIX MAGAZINE

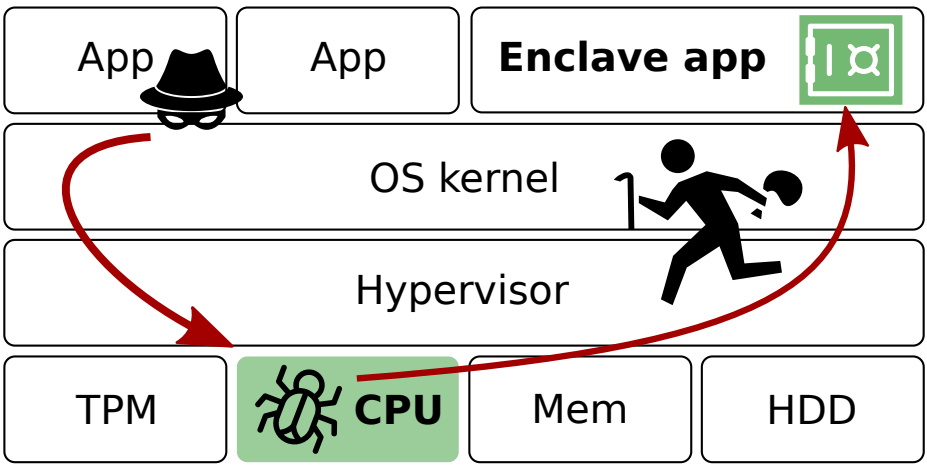
Mühlberg et al. "Reflections on post-Meltdown trusted computing: A case for open security processors", USENIX ;login: magazine, Fall 2018 [MVB18]

Reflections on trusting trust

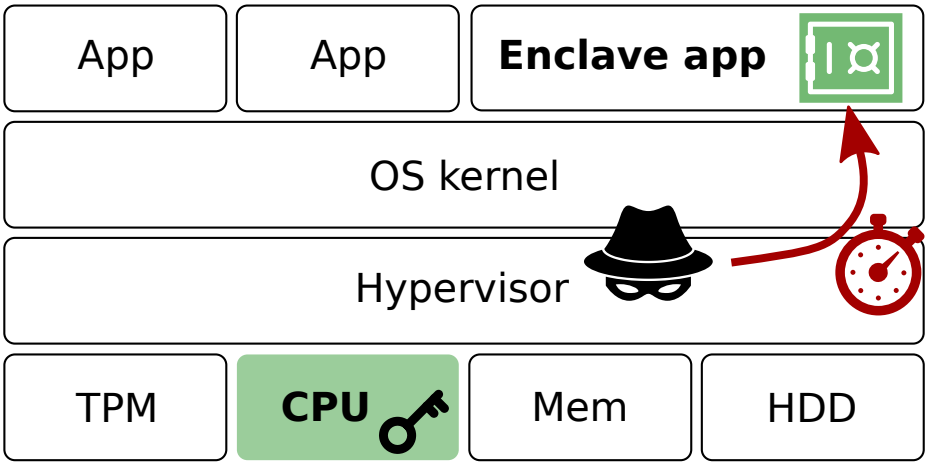
"No amount of source-level verification or scrutiny will protect you from using untrusted code. [...] As the level of program gets lower, these bugs will be harder and harder to detect. A well installed microcode bug will be almost impossible to detect."

— Ken Thompson (ACM Turing award lecture, 1984)

The big picture: Enclaved execution attack surface

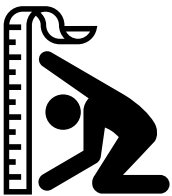


The big picture: Enclaved execution attack surface




SHARING IS NOT CARING
SHARING IS LOSING YOUR STUFF TO OTHERS
imgflip.com

Nemesis: Studying rudimentary CPU interrupt logic

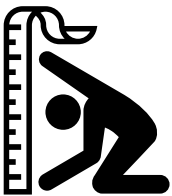


Overview

$\Rightarrow$ Interrupts leak **instruction execution times**

$\Rightarrow$ Determine control flow in **enclave** programs

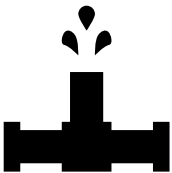
Nemesis: Studying rudimentary CPU interrupt logic



Overview

⇒ Interrupts leak **instruction execution times**

⇒ Determine control flow in **enclave** programs



Research contributions

⇒ (First) remote μ-arch attack on **embedded** CPUs

⇒ Understanding **CPU pipeline** leakage (˜Meltdown)

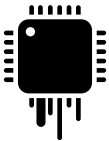

MIND THE GAP

Conclusions and take-away

⇒ New class of **transient execution** attacks

⇒ Importance of fundamental **side-channel research**

⇒ Security **cross-cuts** the system stack: hardware, hypervisor, kernel, compiler, application



References I

C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss.
A systematic evaluation of transient execution attacks and defenses.
arXiv preprint arXiv:1811.05441, 2018.

D. Gruss, M. Lipp, M. Schwarz, R. Fellner, C. Maurice, and S. Mangard.
KASLR is dead: Long live KASLR.
In *International Symposium on Engineering Secure Software and Systems*, pp. 161–176. Springer, 2017.

P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom.
Spectre attacks: Exploiting speculative execution.
In *Proceedings of the 40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg.
Meltdown: Reading kernel memory from user space.
In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, 2018.

J. T. Mühlberg and J. Van Bulck.
Reflections on post-Meltdown trusted computing: A case for open security processors.
;login: the USENIX magazine, Vol. 43(No. 3), Fall 2018.

J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx.
Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution.
In *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, August 2018.

J. Van Bulck, F. Piessens, and R. Strackx.
Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic.
In *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS'18)*. ACM, October 2018.

References II

J. Van Bulck, N. Weichbrodt, R. Kapitza, F. Piessens, and R. Strackx.
Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution.
In *Proceedings of the 26th USENIX Security Symposium*. USENIX Association, August 2017.

O. Weisse, J. Van Bulck, M. Minkin, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, R. Strackx, T. F. Wenisch, and Y. Yarom.
Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution.
Technical Report `https://foreshadowattack.eu/`, 2018.

Y. Xu, W. Cui, and M. Peinado.
Controlled-channel attacks: Deterministic side channels for untrusted operating systems.
In *36th IEEE Symposium on Security and Privacy*. IEEE, May 2015.