Tutorial: Uncovering Side-Channels in Intel SGX Enclaves

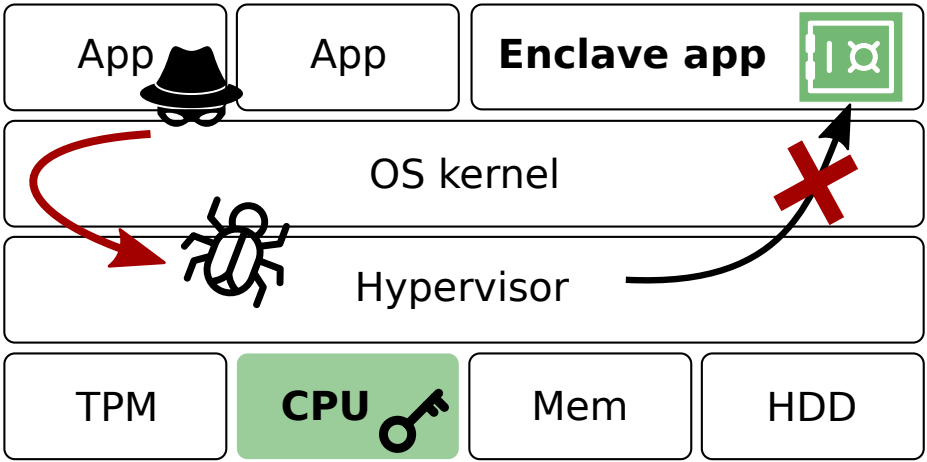
Part 2: Stealing enclave secrets with transient execution

Jo Van Bulck

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

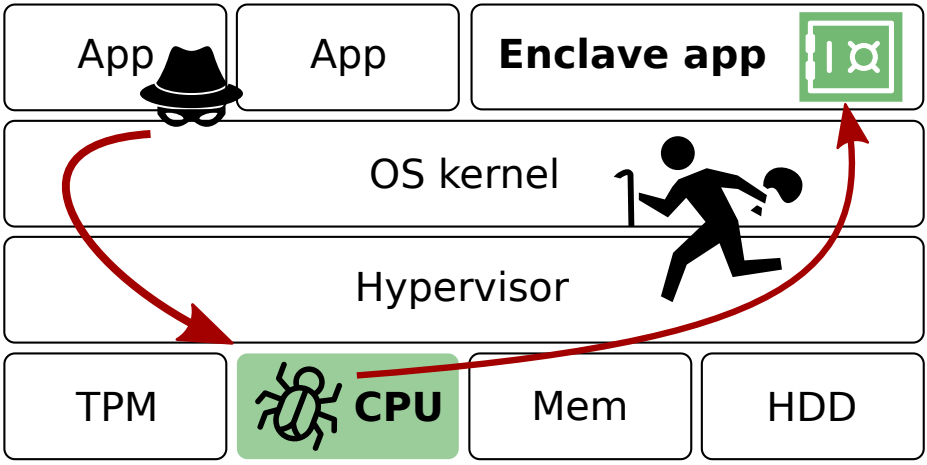
SPACE 2018, December 15, 2018

Enclaved execution attack surface (revisited)



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

Enclaved execution attack surface (revisited)



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]

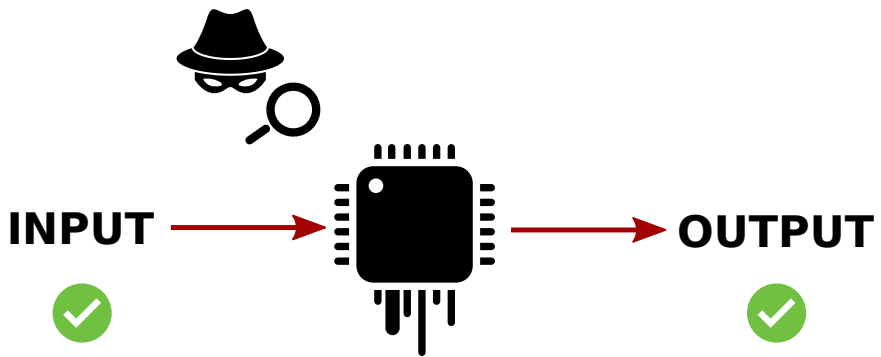
Reflections on trusting trust

"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)

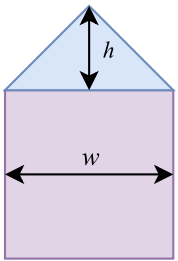
A primer on software security (revisited)

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




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

Out-of-order and speculative execution

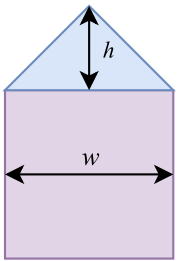



```
int area(int h, int w)
{
  int triangle = (w*h)/2;
  int square   = (w*w);
  return triangle + square;
}
```

Key **discrepancy:**

- Programmers write sequential instructions

Out-of-order and speculative execution



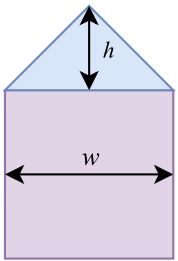

```
int area(int h, int w)
{
    int triangle = (w*h)/2;
    int square   = (w*w);
    return triangle + square;
}
```

Key **discrepancy:**

- Programmers write sequential instructions
- Modern CPUs are inherently parallel

⇒ *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!

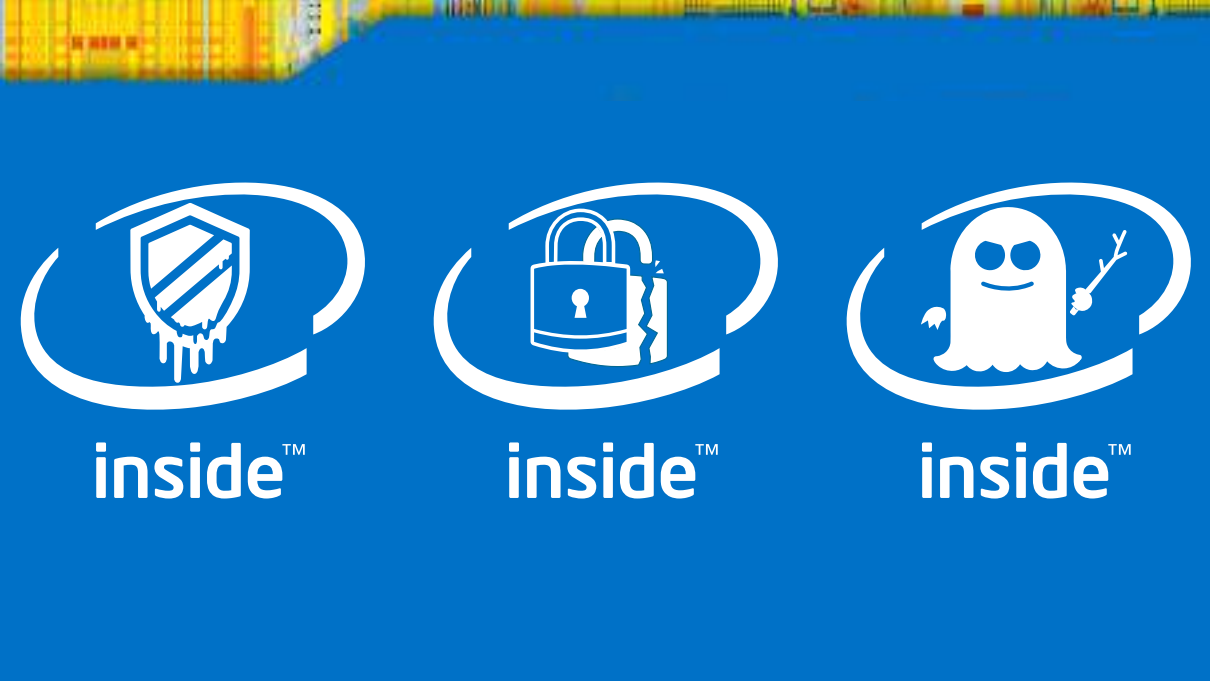
Key finding of 2018

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

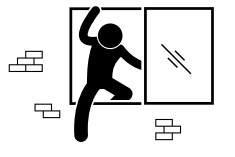
Transient world (microarchitecture) may temp bypass architectural software intentions:

CPU access control bypass

Speculative buffer overflow/ROP


inside™
inside™
inside™

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

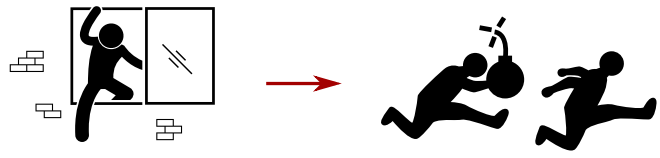
Listing 1: x86 assembly

```
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

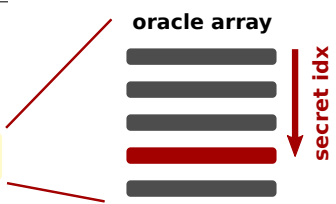
Transient out-of-order window

Listing 1: x86 assembly.

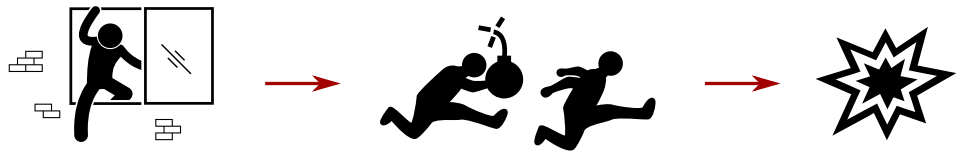
```
1 meltdown:
2     // %rdi: oracle
3     // %rsi: secret_ptr
4
5     movb (%rsi), %al
6     shl $0xc, %rax
7     movq (%rdi, %rax), %rdi
8     retq
```

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)

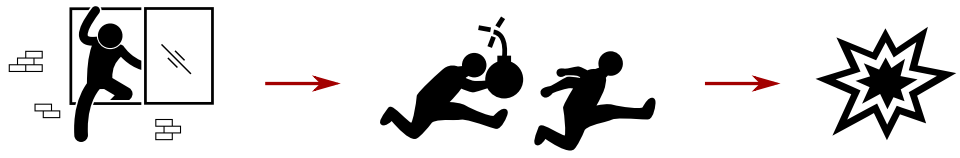
Listing 1: x86 assembly.

```
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



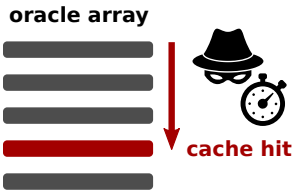
Unauthorized access | Transient out-of-order window | **Exception handler**

Listing 1: x86 assembly.

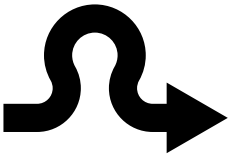
```
1 meltdown:
2     // %rdi: oracle
3     // %rsi: secret_ptr
4
5     movb (%rsi), %al
6     shl $0xc, %rax
7     movq (%rdi, %rax), %rdi
8     retq
```

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

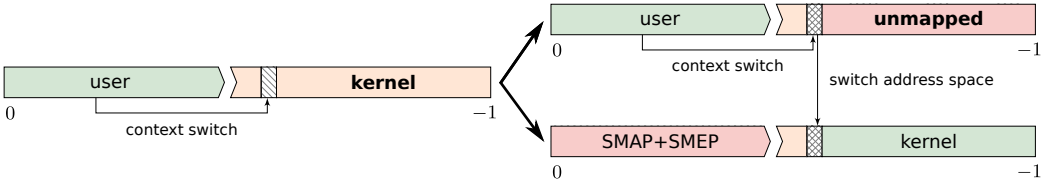


- OS software fix for faulty hardware (↔ future CPUs)

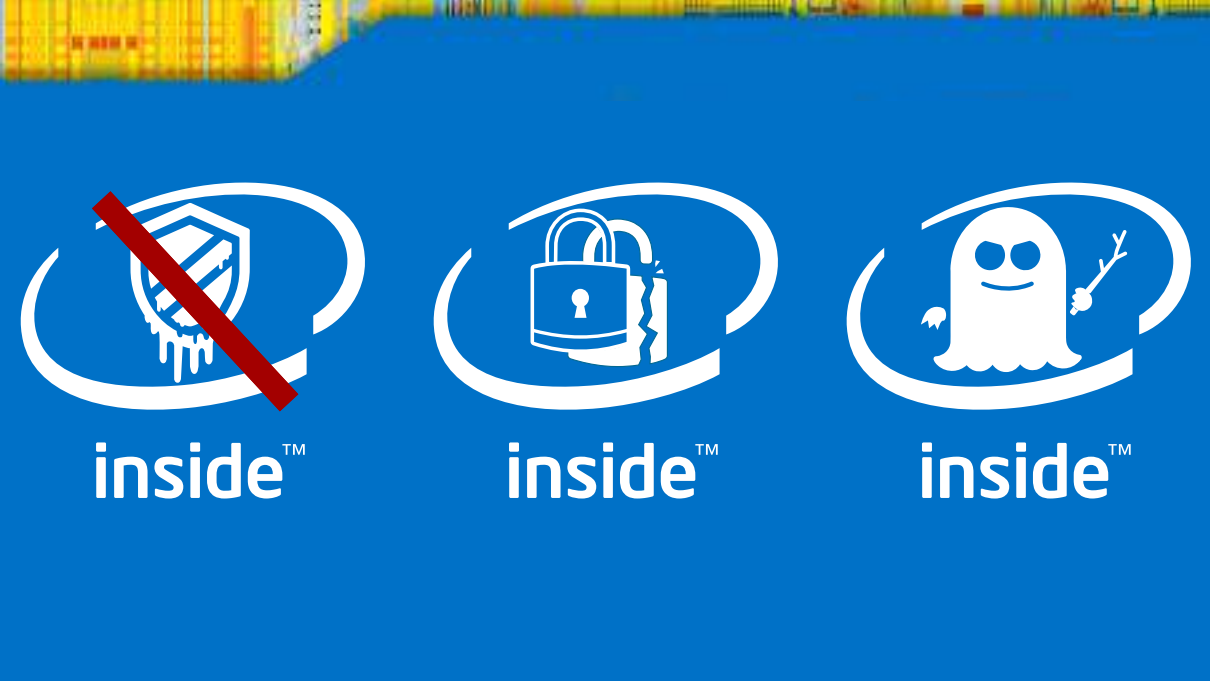
Mitigating Meltdown: Unmap kernel addresses from user space

- 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?~~

SPECTRE-LIKE FLAW UNDERMINES INTEL PROCESSORS' MOST SECURE ELEMENT

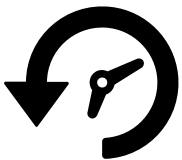
I'M SURE THIS WON'T BE THE LAST SUCH PROBLEM —

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

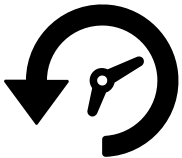


1. Cache secrets in L1

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

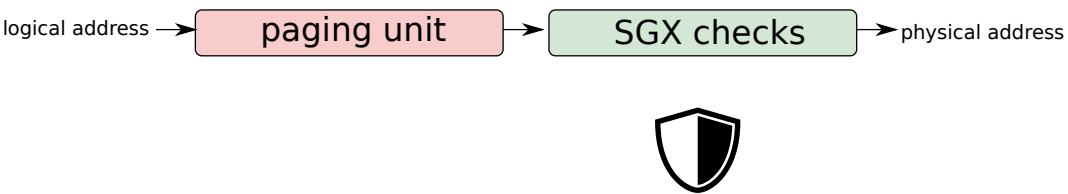
- Enclaved memory reads `0xFF`
- Meltdown "bounces back" (~ mirror)

Intra-enclave view

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

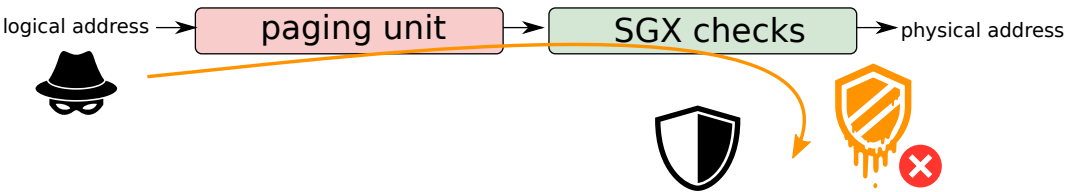
Building Foreshadow: Evade SGX abort page semantics

Note: SGX MMU sanitizes *untrusted* address translation



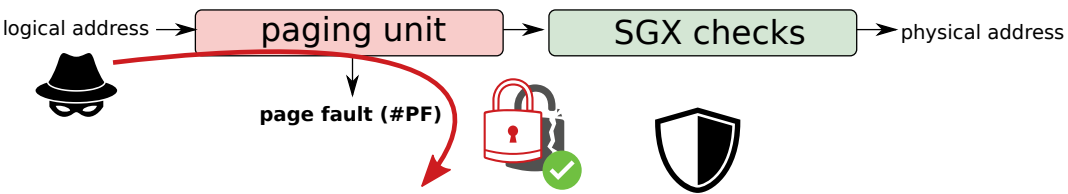
Building Foreshadow: Evade SGX abort page semantics

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

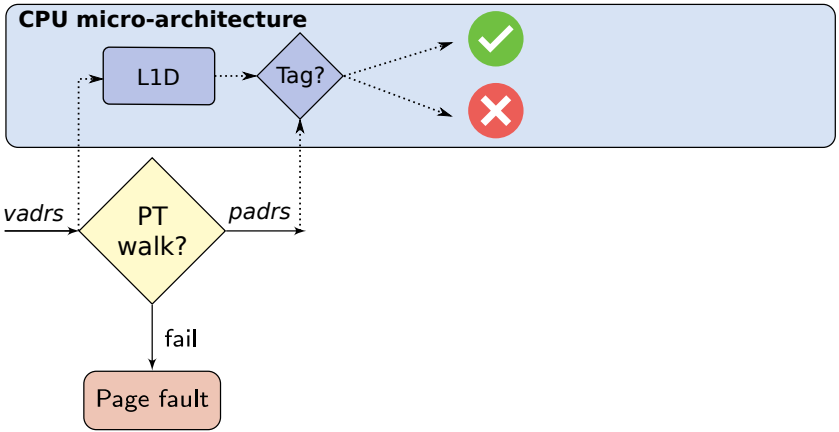


Building Foreshadow: Evade SGX abort page semantics

Foreshadow: Bypass abort page via *untrusted* page table

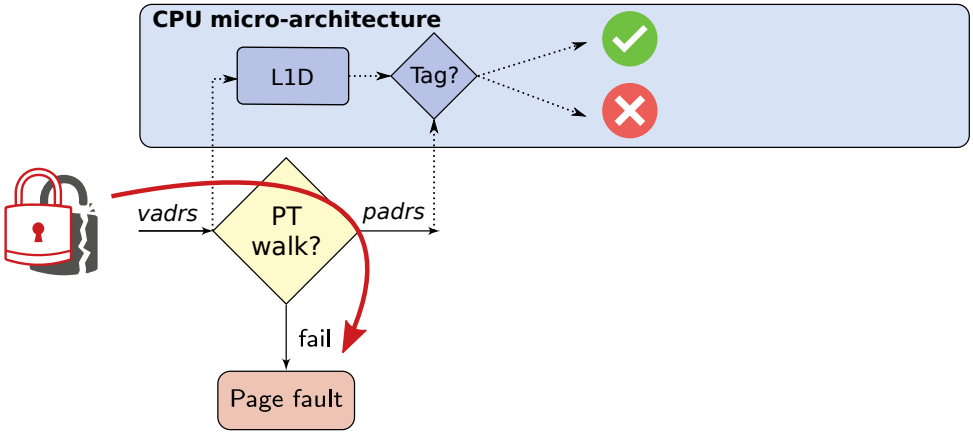


Foreshadow-NG: Breaking the virtual memory abstraction



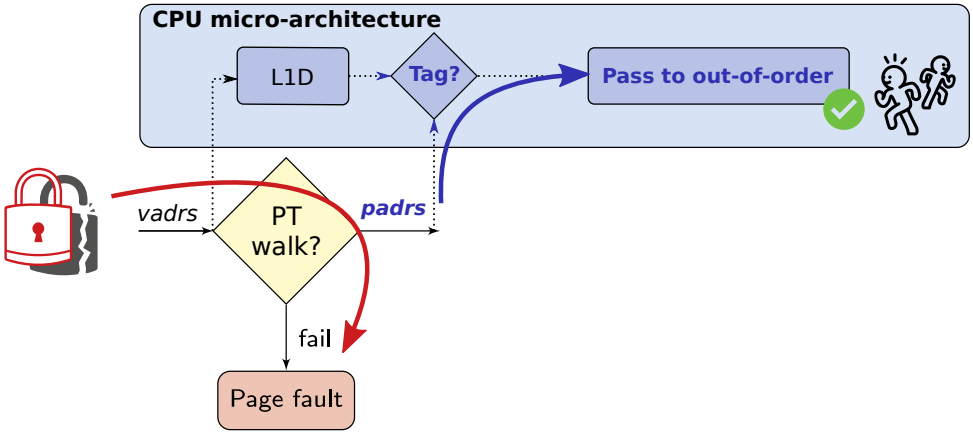
L1 cache design: Virtually-indexed, physically-tagged

Foreshadow-NG: Breaking the virtual memory abstraction



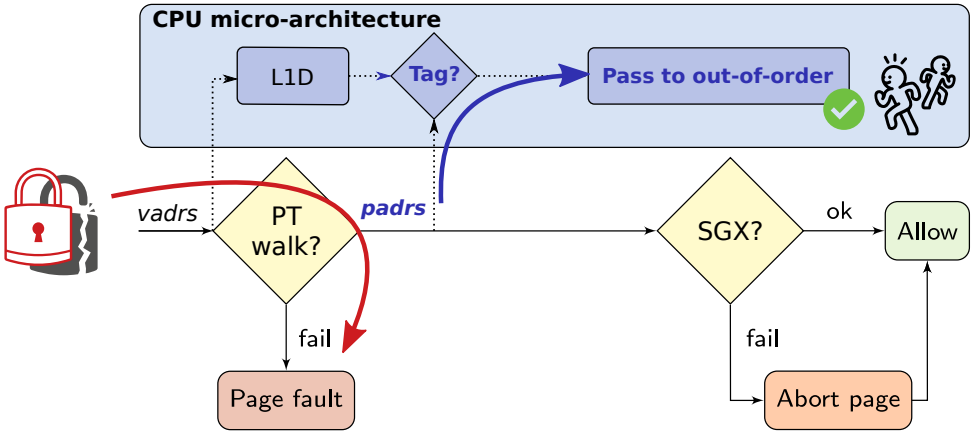
Page fault: Early-out address translation

Foreshadow-NG: Breaking the virtual memory abstraction



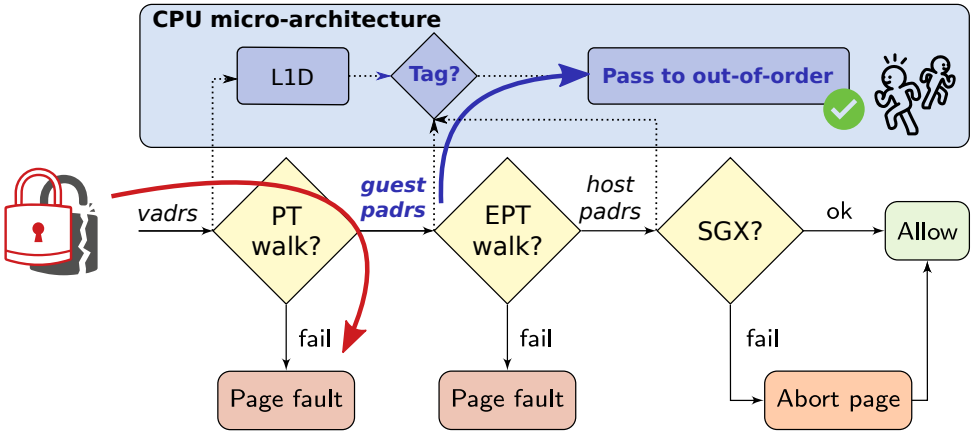
L1-Terminal Fault: match *unmapped physical address* (!)

Foreshadow-NG: Breaking the virtual memory abstraction



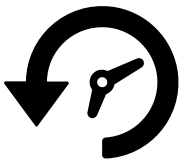
Foreshadow-SGX: bypass enclave isolation

Foreshadow-NG: Breaking the virtual memory abstraction



Foreshadow-VMM: bypass virtual machine isolation

Mitigating Foreshadow

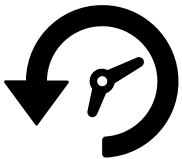


1. Cache secrets in L1

2. Unmap page table entry

3. Execute Meltdown

Mitigating Foreshadow



1. Cache secrets in L1

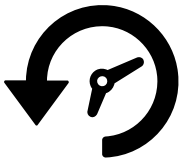
2. Unmap page table entry

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

2. Unmap page table entry

3. Execute Meltdown

Intel microcode updates

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

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

Mitigating Foreshadow/L1TF: Hardware-software cooperation

```
jo@gropius:~$ uname -svp
Linux #41~16.04.1-Ubuntu SMP Wed Oct 10 20:16:04 UTC 2018 x86_64

jo@gropius:~$ cat /proc/cpuinfo | grep "model name" -m1
model name      : Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz

jo@gropius:~$ cat /proc/cpuinfo | egrep "meltdown|l1tf" -m1
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf

jo@gropius:~$ cat /sys/devices/system/cpu/vulnerabilities/meltdown | grep "Mitigation"
Mitigation: PTI

jo@gropius:~$ cat /sys/devices/system/cpu/vulnerabilities/l1tf | grep "Mitigation"
Mitigation: PTE Inversion; VMX: conditional cache flushes, SMT vulnerable

jo@gropius:~$
```

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 enclaves technology for protecting data in use.

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 enclaves technology for protecting data in use.

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

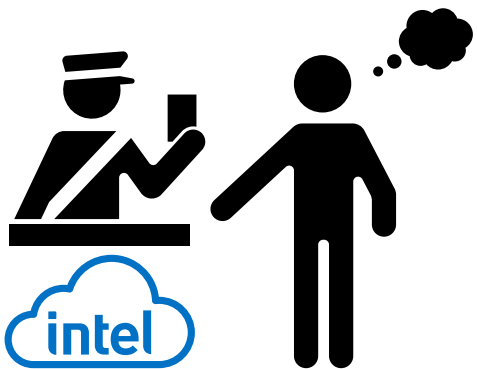
CPU-level key derivation

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

Foreshadow fallout: Dismantling the SGX ecosystem

CPU-level key derivation

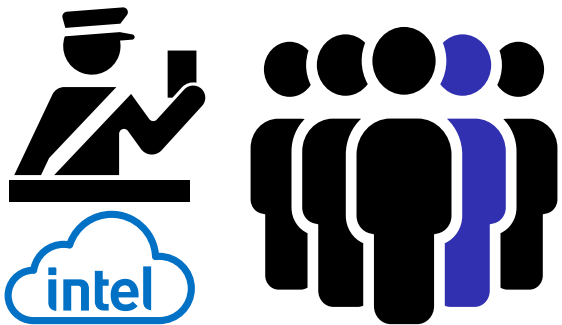
Intel == trusted 3th party (shared **CPU master secret**)



Foreshadow fallout: Dismantling the SGX ecosystem

Fully anonymous attestation

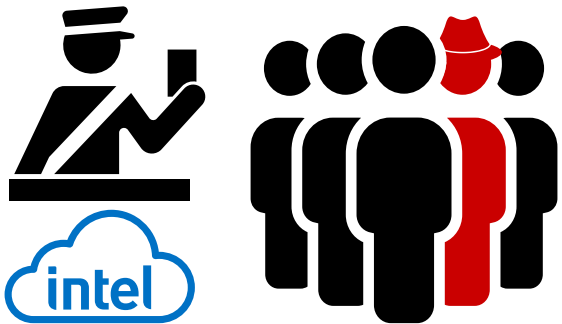
Intel Enhanced Privacy ID (EPID) **group signatures** ☺



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

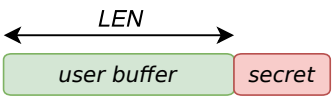
Active **man-in-the-middle:** read + modify all local and remote secrets (!)


inside™


inside™


inside™

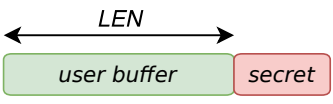
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory

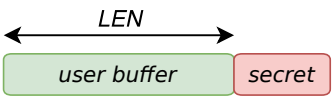
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory
- Branch can be mistrained to speculatively (i.e., ahead of time) execute with $idx \geq LEN$ in the **transient world**

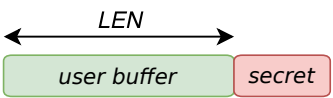
Spectre v1: Speculative buffer over-read




```
if (idx < LEN)
{
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory
- Branch can be mistrained to speculatively (i.e., ahead of time) execute with $idx \geq LEN$ in the **transient world**
- Side-channels leak out-of-bounds secrets to the **real world**

Mitigating Spectre v1: Inserting speculation barriers

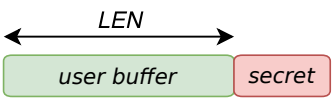



```
if (idx < LEN)
{

  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory

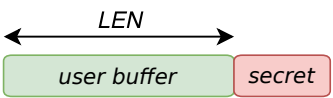
Mitigating Spectre v1: Inserting speculation barriers




```
if (idx < LEN)
{
  asm("lfence\n\t");
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory
- Insert **speculation barrier** to tell the CPU to halt the transient world until *idx* got evaluated ↔ performance ☹

Mitigating Spectre v1: Inserting speculation barriers



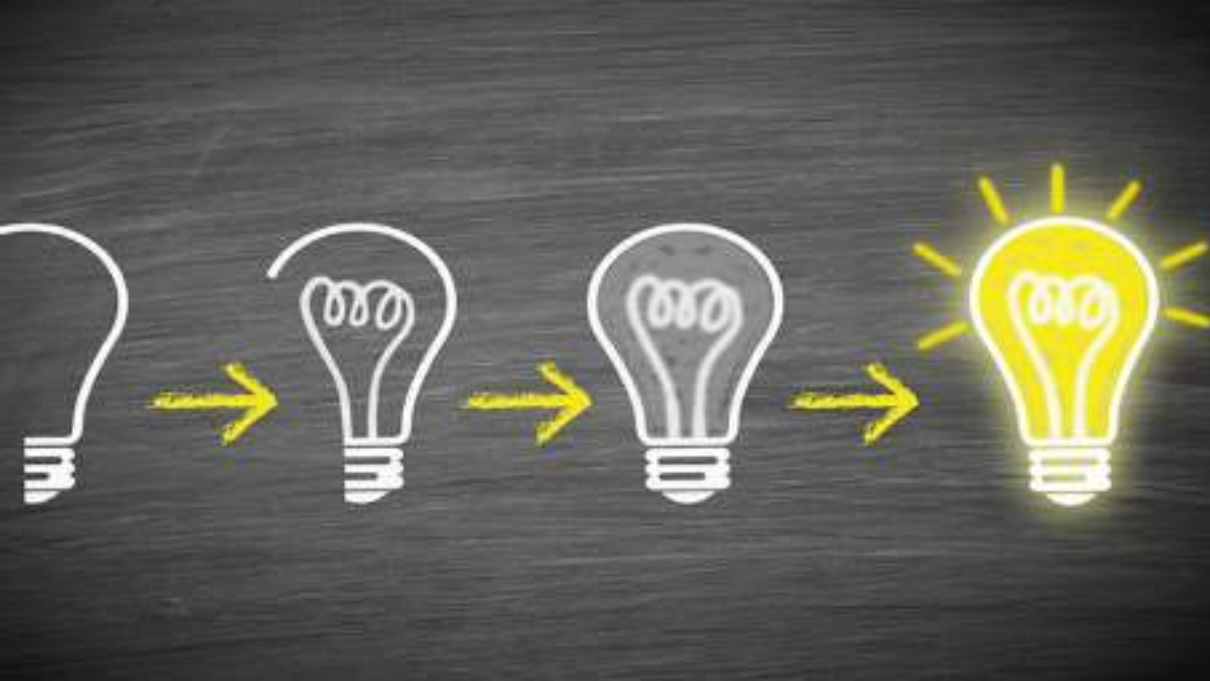

```
if (idx < LEN)
{
  asm("lfence\n\t");
  s = buffer[idx];
  t = lookup[s];
  ...
}
```

- Programmer *intention:* never access out-of-bounds memory
- Insert **speculation barrier** to tell the CPU to halt the transient world until *idx* got evaluated ↔ performance ☹
- Huge error-prone **manual effort,** no reliable automated compiler approaches yet...

index : kernel/git/torvalds/linux.git

Linux kernel source tree

Age	Commit message (Expand)	Author	Files	Lines
3 days	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net	Linus Torvalds	56	-274/+795
4 days	vhost: Fix Spectre V1 vulnerability	Jason Wang	1	-0/+2
2018-10-19	Merge tag 'usb-4.19-final' of git://git.kernel.org/pub/scm/linux/kernel/git/g...	Greg Kroah-Hartman	7	-27/+65
2018-10-19	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net	Greg Kroah-Hartman	57	-187/+253
2018-10-19	Merge tag 'for-gkh' of git://git.kernel.org/pub/scm/linux/kernel/git/rdma/rdma	Greg Kroah-Hartman	2	-0/+6
2018-10-17	ptp: fix Spectre v1 vulnerability	Gustavo A. R. Silva	1	-0/+4
2018-10-17	usb: gadget: storage: Fix Spectre v1 vulnerability	Gustavo A. R. Silva	1	-0/+3
2018-10-16	RDMA/ucma: Fix Spectre v1 vulnerability	Gustavo A. R. Silva	1	-0/+3
2018-10-16	IB/ucm: Fix Spectre v1 vulnerability	Gustavo A. R. Silva	1	-0/+3
2018-09-25	Merge tag 'tty-4.19-rc6' of git://git.kernel.org/pub/scm/linux/kernel/git/gre...	Greg Kroah-Hartman	6	-7/+30
2018-09-18	tty: vt_ioctl: fix potential Spectre v1	Gustavo A. R. Silva	1	-0/+4
2018-09-14	Merge tag 'char-misc-4.19-rc4' of git://git.kernel.org/pub/scm/linux/kernel/g...	Linus Torvalds	10	-34/+73
2018-09-12	Merge tag 'pci-v4.19-fixes-1' of git://git.kernel.org/pub/scm/linux/kernel/gi...	Linus Torvalds	8	-25/+41
2018-09-12	misc: hmc6352: fix potential Spectre v1	Gustavo A. R. Silva	1	-0/+2
2018-09-11	switchtec: Fix Spectre v1 vulnerability	Gustavo A. R. Silva	1	-0/+4
2018-08-29	Merge tag 'hwmon-for-linus-v4.19-rc2' of git://git.kernel.org/pub/scm/linux/k...	Linus Torvalds	5	-12/+32
2018-08-26	hwmon: (nct6775) Fix potential Spectre v1	Gustavo A. R. Silva	1	-0/+2
2018-08-17	Merge tag 'drm-next-2018-08-17' of git://anongit.freedesktop.org/drm/drm	Linus Torvalds	44	-156/+346



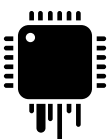
Conclusions and take-away

https://foreshadowattack.eu/

⇒ 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. 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.

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.