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
Secure program: convert all input to *expected output*
Buffer overflow vulnerabilities: trigger unexpected behavior
Safe languages & formal verification: preserve expected behavior
Side-channels: observe side-effects of the computation
Constant-time code: eliminate secret-dependent side-effects
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

DO WE JUST SUCK AT... COMPUTERS?
YUP. ESPECIALLY SHARED ONES.

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

Enclaved execution attack surface: Privileged side-channel attacks

Untrusted OS $\rightarrow$ new class of powerful side-channels

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

Key discrepancy:
- Programmers write sequential instructions

```c
int area(int h, int w)
{
    int triangle = (w*h)/2;
    int square = (w*w);
    return triangle + square;
}
```
Out-of-order and speculative execution

Key discrepancy:
- Programmers write sequential instructions
- Modern CPUs are inherently parallel

$\Rightarrow$ Speculatively execute instructions ahead of time

```c
int area(int h, int w)
{
    int triangle = (w*h)/2;
    int square = (w*w);
    return triangle + square;
}
```
Out-of-order and speculative execution

Key **discrepancy:**
- Programmers write *sequential* instructions
- Modern CPUs are inherently *parallel*

⇒ *Speculatively execute instructions ahead of time*

**Best-effort:** What if triangle fails?
- → Commit in-order, **roll-back** square
  - But **side-channels** may leave traces (!)
Transient execution attacks: Welcome to the world of fun!

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

- Success $\rightarrow$ *commit* results to normal world 😊
- Fail $\rightarrow$ *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 $\rightarrow$ commit results to normal world 😊
- Fail $\rightarrow$ 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

⇒ 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

⇒ *Transmit secrets from transient to normal world*

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

- **CPU access control bypass**
- **Speculative buffer overflow/ROP**
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

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.

```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.

```c
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 (\leftrightarrow\ future CPUs)
- Unmap kernel from user virtual address space
  - Unauthorized physical addresses out-of-reach (\sim\ cookie jar)

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?

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

L1 terminal fault challenges

1. Cache secrets in L1  
2. Unmap page table entry  
3. Execute Meltdown

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
- SGXspectre 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
- SGXspectre in-enclave code abuse
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).

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


Building Foreshadow: Evade SGX abort page semantics

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

Unprivileged system call

```c
mprotect( secret_ptr & 0xFFF, 0x1000, PROT_NONE );
```
CPU micro-architecture

L1D → Tag?

Tag?

PT walk?

vadrs → padrs

fail

Page fault

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

**CPU micro-architecture**

L1D → Tag? → Pass to out-of-order

PT walk?

vadrs → padrs

fail → Page fault

**L1-Terminal Fault:** match *unmapped physical address* (!)
Foreshadow-NG: Breaking the virtual memory abstraction

- CPU micro-architecture
  - L1D
  - Tag?
  - Pass to out-of-order

- PT walk?
  - vadrs
  - padrs

- SGX?
  - ok
  - Allow

- Page fault
- Abort page

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)

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

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

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.


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

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

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

Remote attestation and secret provisioning

Challenge-response to prove **enclave identity**

- App enclave

Can you keep a secret?
Foreshadow fallout: Dismantling the SGX ecosystem

CPU-level key derivation

Intel == trusted 3rd 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 . . . 😞
EPID key extraction with Foreshadow

Active **man-in-the-middle**: read + modify all local and remote secrets (!)
Research challenges: Universal classification and evaluation

- Transient cause?
- Spectre-type: Spectre-PHT, Spectre-BTB, Spectre-RSB, Spectre-STL
  - Microarchitectural buffer
  - Cross-address-space vs. Same-address-space
  - Mistraining strategy

  - Cross-address-space vs. Same-address-space

- Prediction fault

Canella et al. “A Systematic Evaluation of Transient Execution Attacks and Defenses”, arXiv preprint [CVBS\textsuperscript{+}18]
Reflections on Post-Meltdown Trusted Computing
A Case for Open Security Processors

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
Nemesis: Studying rudimentary CPU interrupt logic

Overview

⇒ Interrupts leak instruction execution times
⇒ 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
A systematic evaluation of transient execution attacks and defenses.

KASLR is dead: Long live KASLR.

Spectre attacks: Exploiting speculative execution.

Meltdown: Reading kernel memory from user space.

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.

Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution.

J. Van Bulck, F. Piessens, and R. Strackx.
Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic.
Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution.

Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution.

Y. Xu, W. Cui, and M. Peinado.
Controlled-channel attacks: Deterministic side channels for untrusted operating systems.