ZombieLoad

Cross-Privilege-Boundary Data Sampling

Michael Schwarz\textsuperscript{1}, Moritz Lipp\textsuperscript{1}, Daniel Moghimi\textsuperscript{2}, Jo Van Bulck\textsuperscript{3}, Julian Stecklina\textsuperscript{4}, Thomas Prescher\textsuperscript{4}, Daniel Gruss\textsuperscript{1}

November 11, 2019

\textsuperscript{1} Graz University of Technology, \textsuperscript{2} Worcester Polytechnic Institute, \textsuperscript{3} imec-DistriNet, KU Leuven, \textsuperscript{4} Cyberus Technology
A new Meltdown-type transient-execution attack
• A new Meltdown-type transient-execution attack
• Leaks data on Intel CPUs
ZombieLoad

- A new Meltdown-type transient-execution attack
- Leaks data on Intel CPUs
- Really new? Published in May 2019...
Intel Zombieload bug fix to slow data centre computers

ZombieLoad attack lets hackers steal data from Intel chips

'Zombieload' Flaw Lets Hackers Crack Almost Every Intel Chip Back to 2011. Why's It Being Downplayed?

Only New CPUs Can Truly Fix ZombieLoad and Spectre
ZombieLoad

- CVE-2018-12130
- CVE-2019-11091
Microarchitectural Data Sampling (MDS)

ZombieLoad
- CVE-2018-12130
- CVE-2019-11091

RIDL
- CVE-2018-12127
- CVE-2018-12130
- CVE-2019-11091
Microarchitectural Data Sampling (MDS)

- **ZombieLoad**
  - CVE-2018-12130
  - CVE-2019-11091
  - CVE-2019-11135

- **RIDL**
  - CVE-2018-12127
  - CVE-2018-12130
  - CVE-2019-11091

- **Fallout**
  - CVE-2018-12126
Variant 1
Kernel Mapping

Variant 3
Microcode-Assisted Page-Table Walk
Variant 2 embargoed until November 12, 2019
Variants 2 embargoed until November 12, 2019

Only variant without hardware mitigations
Variant 2 embargoed until November 12, 2019

Only variant without hardware mitigations

→ Works on MDS-resistant Cascade Lake CPUs
March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)
March 28, 2018 We report Meltdown on fill buffer (CVE-2019-11091)

September 12, 2018 VUSec reports fill-buffer leakage (RIDL)
Timeline

March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)
September 12, 2018  VUSec reports fill-buffer leakage (RIDL)
April 12, 2019  We report ZombieLoad Variant 1 (CVE-2018-12130)
March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)

September 12, 2018  VUSec reports fill-buffer leakage (RIDL)

April 12, 2019  We report ZombieLoad Variant 1 (CVE-2018-12130)
  →  All embargoed until May 14, 2019 (MDS)

April 24, 2019  We report ZombieLoad Variant 2 (CVE-2019-11135)
March 28, 2018 We report Meltdown on fill buffer (CVE-2019-11091)

September 12, 2018 VUSec reports fill-buffer leakage (RIDL)

April 12, 2019 We report ZombieLoad Variant 1 (CVE-2018-12130)
   → All embargoed until May 14, 2019 (MDS)

April 24, 2019 We report ZombieLoad Variant 2 (CVE-2019-11135)

May 10, 2019 We report Variant 2 on Cascade Lake
March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)

September 12, 2018  VUSec reports fill-buffer leakage (RIDL)

April 12, 2019  We report ZombieLoad Variant 1 (CVE-2018-12130)

→ All embargoed until May 14, 2019 (MDS)

April 24, 2019  We report ZombieLoad Variant 2 (CVE-2019-11135)

May 10, 2019  We report Variant 2 on Cascade Lake

May 11, 2019  Call with Intel
March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)

September 12, 2018 VUSec reports fill-buffer leakage (RIDL)

April 12, 2019  We report ZombieLoad Variant 1 (CVE-2018-12130)
→ All embargoed until May 14, 2019 (MDS)

April 24, 2019  We report ZombieLoad Variant 2 (CVE-2019-11135)

May 10, 2019  We report Variant 2 on Cascade Lake

May 11, 2019  Call with Intel

May 12, 2019  Not allowed to publish Variant 2
March 28, 2018  We report Meltdown on fill buffer (CVE-2019-11091)
September 12, 2018  VUSec reports fill-buffer leakage (RIDL)
April 12, 2019  We report ZombieLoad Variant 1 (CVE-2018-12130)
    →  All embargoed until May 14, 2019 (MDS)
April 24, 2019  We report ZombieLoad Variant 2 (CVE-2019-11135)
May 10, 2019  We report Variant 2 on Cascade Lake
May 11, 2019  Call with Intel
May 12, 2019  Not allowed to publish Variant 2
    →  Additional embargo until November 12, 2019 (TAA)
<table>
<thead>
<tr>
<th>Fill Buffer, Load Ports, ?</th>
<th>Fill Buffer, Load Ports</th>
<th>Store Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>All loads &amp; stores</td>
<td>Uncached loads &amp; stores</td>
<td>Stores</td>
</tr>
<tr>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✅</td>
<td>✗ (before Cascade Lake)</td>
<td>✗ (before Cascade Lake)</td>
</tr>
</tbody>
</table>

ZombieLoad works **despite** software **mitigations** and even on **MDS-resistant** CPUs (e.g., Cascade Lake)
ZombieLoad Cache-line Conflicts

Mapping $v_1$

- cache line
- Page
ZombieLoad Cache-line Conflicts
ZombieLoad Cache-line Conflicts

Mapping $v_2$

cache line

flush

Mapping $v_1$

Page
### User Memory

|    | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  | M  | N  | O  | P  | Q  | R  | S  | T  | U  | V  | W  | X  | Y  | Z  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| char value = faulting[0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
char value = faulting[0]

mem[value]

Out of order
char value = faulting[0]

User Memory

K

Out of order
There is no noise.

Noise is just someone else’s data.
Complex Load Situations

- Execution Engine
  - Scheduler
    - Execution Units
      - ALU, AES, ...
      - ALU, FMA, ...
      - ALU, Vect, ...
      - ALU, Branch
    - Load data
    - Store data
    - AGU
  - Load Buffer
  - Store Buffer
    - L1 Data Cache
    - DTLB
    - LFB

- Core Memory
- CDB

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Complex Load Situations

Execution Engine

- Reorder buffer
- Scheduler
- Execution Units
  - ALU, AES, ...
  - ALU, FMA, ...
  - ALU, Vect, ...
  - ALU, Branch

Core Memory

- Load Buffer
- Store Buffer
- L1 Data Cache
- DTLB
- LFB

Load data
Load data
Store data
AGU

mov al, byte [rcx]

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Complex Load Situations

... mov al, byte [rcx] ...
Complex Load Situations

...mov al, byte [rcx]...

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Complex Load Situations

... 

mov al, byte [rcx] 
...
Complex Load Situations

Execution Engine

Reorder buffer

Scheduler

Execution Units

Execution Units

ALU, AES, . . .

ALU, FMA, . . .

ALU, Vect, . . .

ALU, Branch

Load data

Store data

AGU

CDB

Core Memory

Load Buffer

Store Buffer

L1 Data Cache

DTLB

LFB

... mov al, byte [rcx] ...

Core Memory

Load data

Store data

AGU

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Complex Load Situations

The diagram illustrates the execution engine of a processor with various execution units such as ALUs, AES, FMA, and Vect. The load data and store data operations are handled by the Load Buffer and Store Buffer, respectively. The L1 Data Cache, DTLB, and LFB are components of the core memory.

The execution units (μOP) pass through the reorder buffer and scheduler before being executed by the execution engine.

Code snippet:
```
mov al, byte [rcx]
```

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Complex Load Situations

...
Complex Load Situations

...mov al, byte [rcx]...

data can go to register

Execution Units:
- ALU
- AES
- FMA
- Vect
- Branch
- Load data
- Store data
- AGU

Core Memory:
- Load Buffer
- Store Buffer
- L1 Data Cache
- DTLB
- LFB

Execution Engine:
- Scheduler
- Execution Units
- Reorder buffer
- CDB

Michael Schwarz (@misc0110) et al. — Graz University of Technology
• Complex situations handled in microcode
Microcode Assists

- Complex situations handled in microcode
  - Setting accessed/dirty bit
Complex situations handled in microcode
- Setting accessed/dirty bit
- TSX abort + rollback
• Complex situations handled in microcode
  • Setting accessed/dirty bit
  • TSX abort + rollback
  • ...
Microcode Assists

• Complex situations handled in microcode
  • Setting accessed/dirty bit
  • TSX abort + rollback
  • ...

• Load needs to be re-issued
Microcode Assists

- Complex situations handled in microcode
  - Setting accessed/dirty bit
  - TSX abort + rollback
  - ...

- Load needs to be re-issued
- Meltdown effects due to “microarchitectural fault”
Microcode Assists

- Complex situations handled in microcode
  - Setting accessed/dirty bit
  - TSX abort + rollback
  - ...
- Load needs to be re-issued
- Meltdown effects due to “microarchitectural fault”
- No architectural fault handling required
• Leak data on **same** and **sibling** hyperthread
• Leak data on same and sibling hyperthread

Applications
• Leak data on same and sibling hyperthread

Applications  Operating System
• Leak data on **same** and **sibling** hyperthread

Applications  Operating System  SGX Enclave
• Leak data on *same* and *sibling* hyperthread

Applications

Operating System

SGX Enclave

Virtual Machine
• Leak data on *same* and *sibling* hyperthread

- Applications
- Operating System
- SGX Enclave
- Virtual Machine
- Hypervisor
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Page Number</th>
<th>Physical Page Offset</th>
<th>Virtual Page Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meltdown</td>
<td>51</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Foreshadow</td>
<td>51</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Fallout</td>
<td>51</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>ZombieLoad/</td>
<td>51</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>RIDL</td>
<td>47</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
key\textsubscript{n} (0xD2)

\begin{center}
\begin{tabular}{cccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{tabular}
\end{center}
key\_n (0xD2) \hspace{1cm} (4,4)\text{-domino}_{n,n+1} (0x21)

\begin{array}{cccc}
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{array}
\begin{align*}
\text{key}_n (0xD2) & \quad \text{(4,4)-domino}_{n,n+1} (0x21) \quad \text{key}_{n+1} (0x1C) \\
1 & \quad 1 \quad 0 \quad 1 & \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 & \quad 1 \quad 1 \quad 0 \quad 0
\end{align*}
Results

AES-NI key
Results

AES-NI key

SGX sealing key
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching
Results

- AES-NI key
- SGX sealing key
- Cross-VM covert channel
- Keyword matching
- URL recovery

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Variant 1
Kernel Mapping

5.30 kB/s
### Performance

<table>
<thead>
<tr>
<th>Variant</th>
<th>Description</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant 1</td>
<td>Kernel Mapping</td>
<td>5.30 kB/s</td>
</tr>
<tr>
<td>Variant 2</td>
<td>Transactional Asynchronous Abort</td>
<td>39.66 kB/s</td>
</tr>
</tbody>
</table>

Michael Schwarz (@misc0110) et al. — Graz University of Technology
Performance

Variant 1
Kernel Mapping
5.30 kB/s

Variant 2
Transactional Asynchronous Abort
39.66 kB/s

Variant 3
Microcode-Assisted Page-Table Walk
7.73 kB/s
ZombieLoad Insights

Memory-based Side-Channel Attacks

Instruction Pointer

Address

Michael Schwarz (@misc0110) et al. — Graz University of Technology
ZombieLoad Insights

Instruction Pointer

Memory-based Side-Channel Attacks

Data

Meltdown

Address

Michael Schwarz (@misc0110) et al. — Graz University of Technology
### ZombieLoad Insights

![Diagram](image)

- **Data Sampling**: (this paper)
- **Instruction Pointer**
- **Memory-based Side-Channel Attacks**
- **Address**
- **Meltdown**
- **Data**
• Disable hyperthreading or group scheduling
- Disable hyperthreading or group scheduling
- Overwrite microarchitectural buffers
• Disable hyperthreading or group scheduling
• Overwrite microarchitectural buffers
  • VERW instruction (microcode update)
- Disable hyperthreading or group scheduling
- Overwrite microarchitectural buffers
  - VERW instruction (microcode update)
  - Software sequences
Intel Mitigations

- Disable hyperthreading or group scheduling
- Overwrite microarchitectural buffers
  - VERW instruction (microcode update)
  - Software sequences
- New CPUs which are not affected

<table>
<thead>
<tr>
<th>CPU</th>
<th>Meltdown</th>
<th>Foreshadow</th>
<th>RIDL</th>
<th>Fallout</th>
<th>MLPDS</th>
<th>MDSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>8th/9th gen. Intel Core Coffee Lake</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Intel Xeon Cascade Lake</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
### Intel Mitigations

- Disable hyperthreading or group scheduling
- Overwrite microarchitectural buffers
  - VERW instruction (microcode update)
  - Software sequences
- New CPUs which are not affected

<table>
<thead>
<tr>
<th>CPU</th>
<th>Meltdown</th>
<th>Foreshadow</th>
<th>RIDL</th>
<th>Fallout</th>
<th>MLPDS</th>
<th>MDSUM</th>
<th>ZombieLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td>8th/9th gen. Intel Core Coffee Lake</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>???</td>
</tr>
<tr>
<td>Intel Xeon Cascade Lake</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>???</td>
</tr>
</tbody>
</table>
- Variant 2 works on all CPUs
Circumventing Mitigations

- Variant 2 works on all CPUs
  → Embargoed until November 12, 2019
Circumventing Mitigations

• Variant 2 works on all CPUs
  → Embargoed until November 12, 2019
• Microcode and software sequences do not prevent ZombieLoad
- Variant 2 works on all CPUs
  - Embargoed until November 12, 2019
- Microcode and software sequences do not prevent ZombieLoad
  - Reported on May 16, 2019
Circumventing Mitigations

- Variant 2 works on all CPUs
  → Embargoed until November 12, 2019
- Microcode and software sequences do not prevent ZombieLoad
  → Reported on May 16, 2019
- ZombieLoad might not only leak from fill buffer
• Disable hyperthreading
- Disable hyperthreading
- Flush all buffers on privilege-level change
ZombieLoad Mitigations

- Disable hyperthreading
- Flush all buffers on privilege-level change
  - Fill buffer, store buffer, load ports → VERW

Michael Schwarz (@misc0110) et al. — Graz University of Technology
- Disable hyperthreading
- Flush all buffers on privilege-level change
  - Fill buffer, store buffer, load ports → VERW
  - Flush L1 cache → MSR_IA32_FLUSH_CMD
ZombieLoad Mitigations

- Disable hyperthreading
- Flush all buffers on privilege-level change
  - Fill buffer, store buffer, load ports → VERW
  - Flush L1 cache → MSR_IA32_FLUSH_CMD
- Disable Intel TSX (MSR_TSX_FORCE_ABORT)
Transient Execution Attack Tree

Transient cause
Transient Execution Attack Tree

**Transient cause** — **Meltdown-type**
Transient Execution Attack Tree

Transient cause

Meltdown-type

- Meltdown-PF
- Meltdown-MCA
You can find our proof-of-concept implementation on:

- https://github.com/IAIK/ZombieLoad
• Transient-execution attacks: the gift that keeps on giving
• Transient-execution attacks: the gift that keeps on giving
• Class of Meltdown attacks is larger than expected
Conclusion

- Transient-execution attacks: the gift that keeps on giving
- Class of Meltdown attacks is larger than expected
- CPUs are deterministic - there is no noise
ZombieLoad

Cross-Privilege-Boundary Data Sampling

Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, Daniel Gruss

November 11, 2019

Graz University of Technology