

Claudio Canella¹, Jo Van Bulck², Daniel Gruss¹

September 23, 2019

¹ Graz University of Technology, ² imec-DistriNet, KU Leuven

Cards Against Confusion

Transient Execution
Tree v1.1.0b
(RDCT/LPFBSDFT)



side channel
= obtaining meta-data and
deriving secrets from it

CHANGE MY MIND

Intel Analysis of Speculative Execution Side Channels

[Download PDF](#)



1 of 12



100%



Intel Analysis of Speculative Execution Side Channels

White Paper



Spectre v1,



Spectre v1, Spectre v2,



Spectre v1, Spectre v2, Meltdown,



Spectre v1, Spectre v2, Meltdown, Spectre v3,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre, RIDL,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre, RIDL, MDS,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre, RIDL, MDS, Fallout,



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre, RIDL, MDS, Fallout, ZombieLoad



Spectre v1, Spectre v2, Meltdown, Spectre v3, LazyFP, Spectre v3.1, Foreshadow, Foreshadow-NG, L1TF, Spectre v1.1, Spectre v4, SpectreRSB, ret2spec, Spectre v5, SmotherSpectre, NetSpectre, RIDL, MDS, Fallout, ZombieLoad
I guess I missed a few...?





- Spectre is about misspeculation



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next





- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
- In all these Meltdown-type attacks:



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
- In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
- In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
- In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately
 - it even does things that are architecturally never allowed



- **Spectre** is about **misspeculation**
 - CPU doesn't know what should happen next
 - makes a good guess
- In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately
 - it even does things that are architecturally never allowed
 - because you can't observe the microarchitectural state anyway!



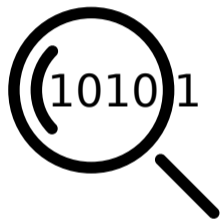
- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
 - In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately
 - it even does things that are architecturally never allowed
 - because you can't observe the microarchitectural state anyway!
- oops ;)



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
 - In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately
 - it even does things that are architecturally never allowed
 - because you can't observe the microarchitectural state anyway!
- oops ;)



- **Spectre** is about misspeculation
 - CPU doesn't know what should happen next
 - makes a good guess
 - In all these Meltdown-type attacks:
 - CPU knows it's doing something wrong
 - still does it deliberately
 - it even does things that are architecturally never allowed
 - because you can't observe the microarchitectural state anyway!
→ oops ;)
- clear difference in behavior







- Step 1: Choose a kernel address and put it in rcx



- Step 1: Choose a kernel address and put it in `rcx`
- Step 2: `mov al, byte [rcx]`



- Step 1: Choose a kernel address and put it in rcx
- Step 2: `mov al, byte [rcx]`
- Step 3: You now got the secret in al



- Step 1: Choose a kernel address and put it in rcx
- Step 2: `mov al, byte [rcx]`
- Step 3: You now got the secret in al



- Step 1: Choose a kernel address and put it in rcx
 - Step 2: `mov al, byte [rcx]`
 - Step 3: You now got the secret in al
- You directly read the value.



- Step 1: Choose a kernel address and put it in `rcx`
- Step 2: `mov al, byte [rcx]`
- Step 3: You now got the secret in `al`

→ You directly read the value. This is not side channels.



- Clear up **naming confusion**



- Clear up **naming confusion**
- Systematic analysis may show **new variants**



- Clear up **naming confusion**
- Systematic analysis may show **new variants**



SPECTRE

- **Spectre**: first class of transient execution attack



- Spectre: first class of transient execution attack
- Exploit control (or data) flow predictions



- 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)



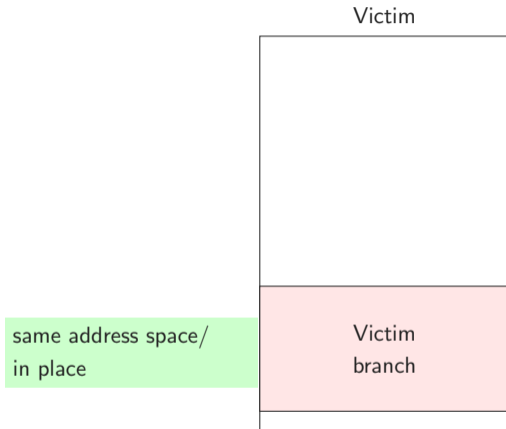
- **Many predictors** in modern CPUs
 - **Branch** taken/not taken (PHT)
 - **Call/Jump** destination (BTB)
 - Function **return** destination (RSB)

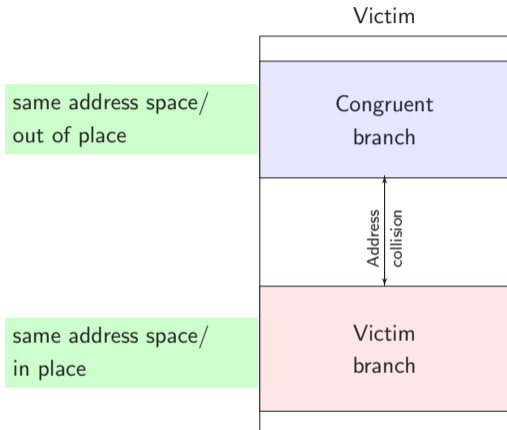


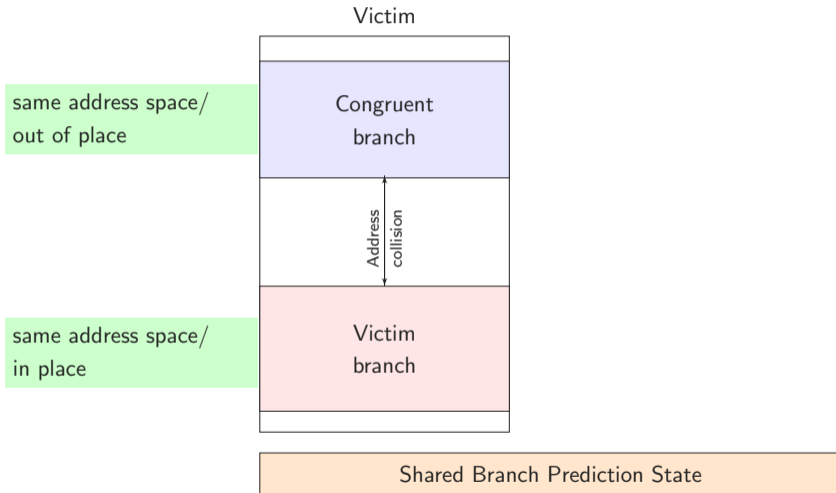
- **Many predictors** in modern CPUs
 - **Branch** taken/not taken (PHT)
 - **Call/Jump** destination (BTB)
 - Function **return** destination (RSB)
 - **Load** matches previous store (STL)

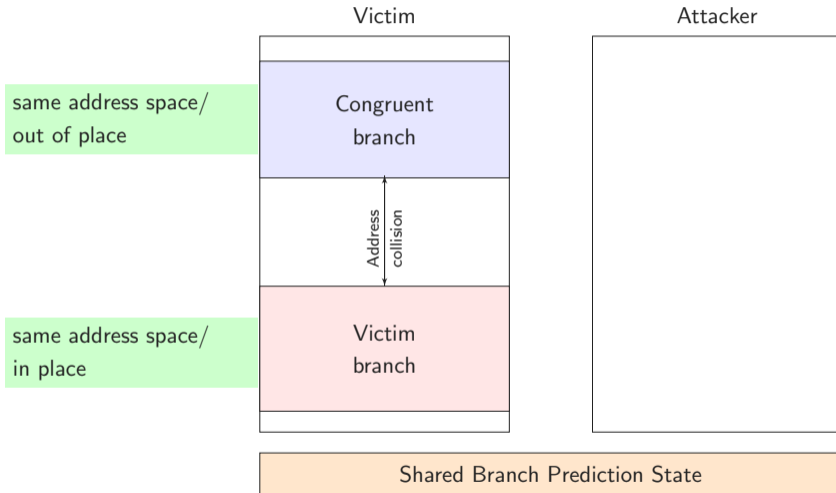


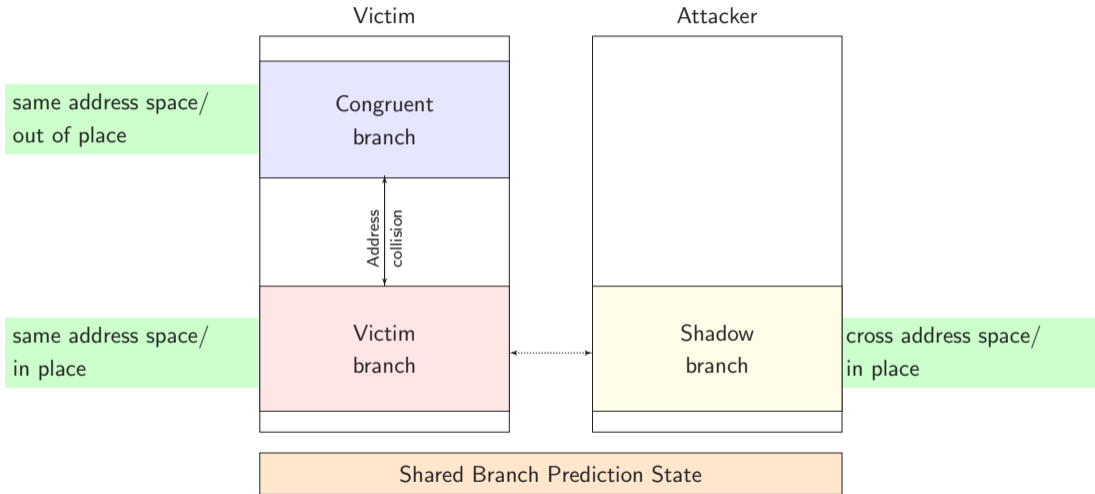
- **Many predictors** in modern CPUs
 - **Branch** taken/not taken (PHT)
 - **Call/Jump** destination (BTB)
 - Function **return** destination (RSB)
 - **Load** matches previous store (STL)
- Most are even **shared** among processes

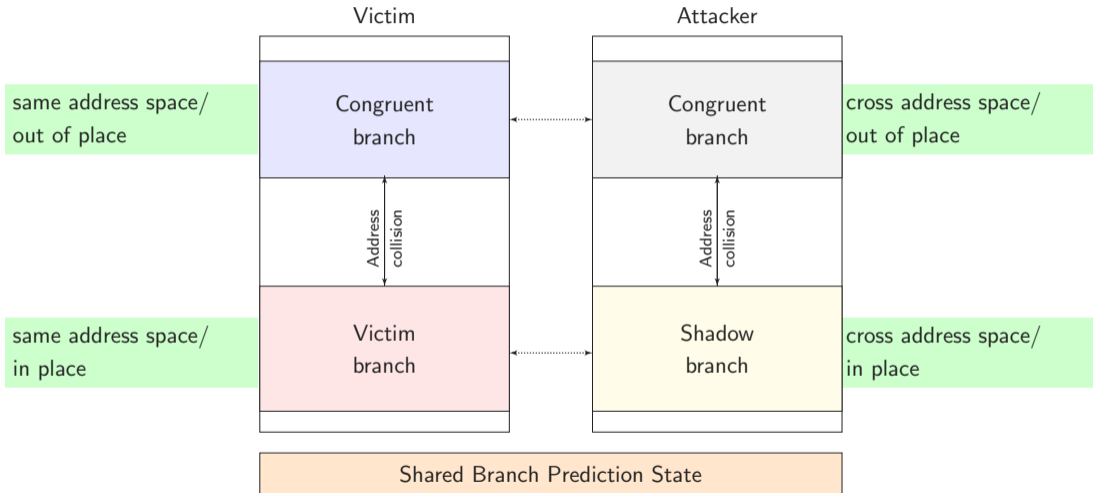




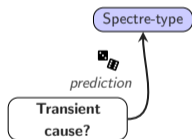


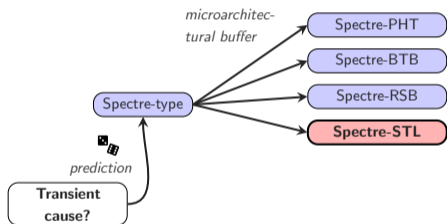


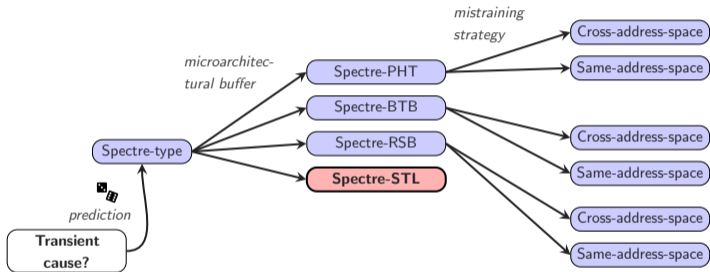


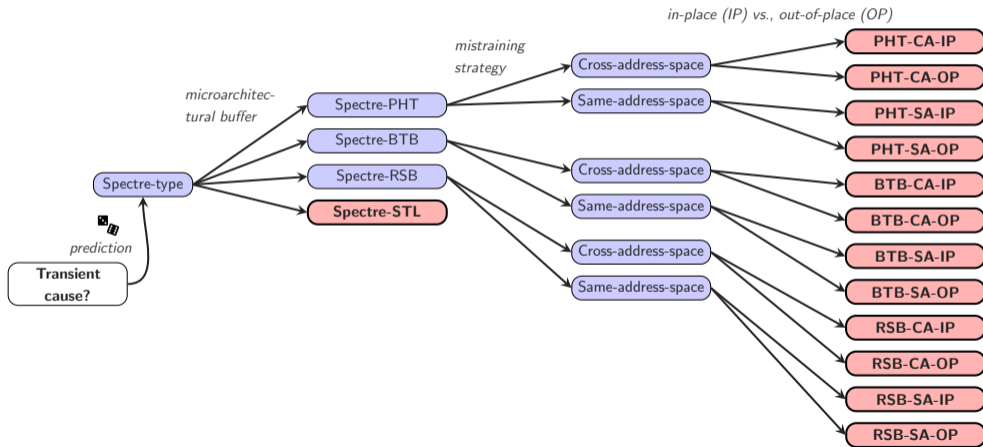


Transient
cause?











MELTDOWN

- **Meltdown** is a separate class of transient execution attack



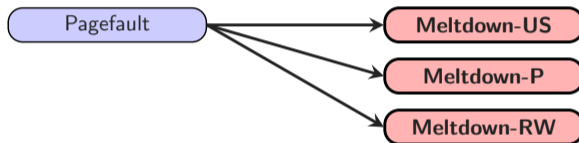
MELTDOWN

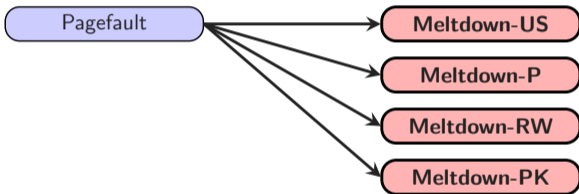
- **Meltdown** is a separate class of transient execution attack
- Exploit **lazy fault handling**

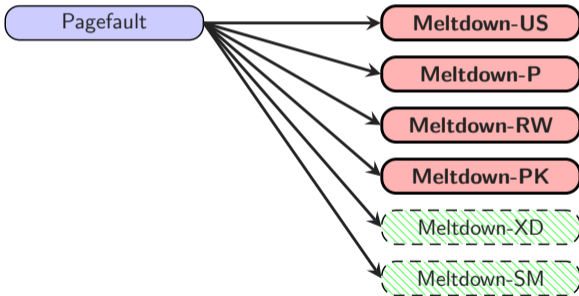
Pagefault



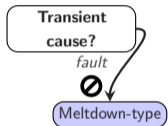


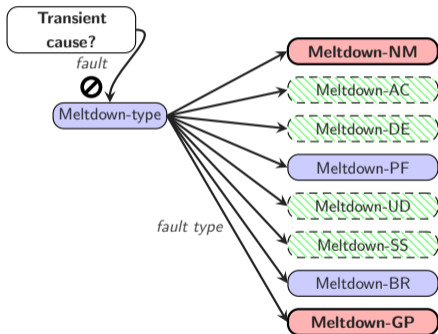


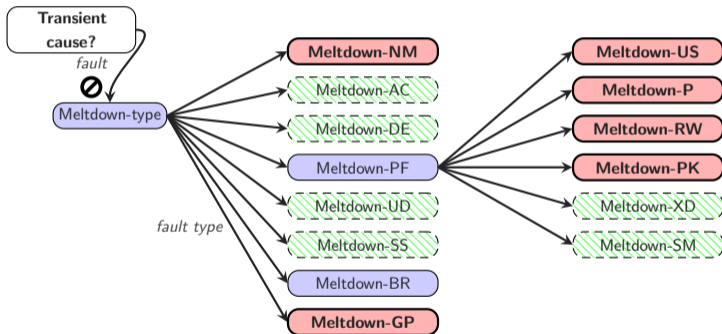


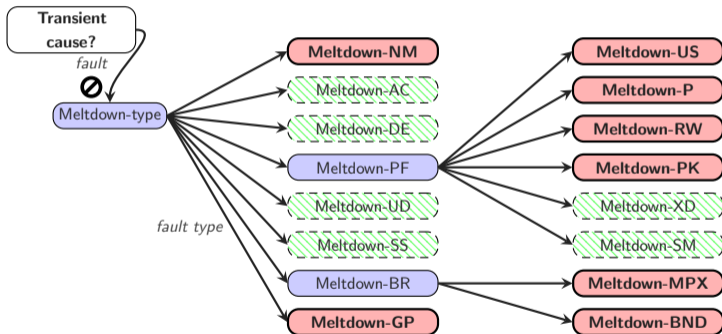


Transient
cause?









Meltdown Redux: Intel Flaw Lets Hackers Siphon Secrets from Millions of PCs

Two different groups of researchers found another speculative execution attack that can steal all the data a CPU touches.



I SPECULATE THAT THIS WON'T BE THE LAST SUCH BUG —

New speculative execution bug leaks data from Intel chips' internal buffers

Intel-specific vulnerability was found by researchers both inside and outside the company.



- May 2019: 3 new **Meltdown-type** attacks
- Leakage from: line-fill buffer, store buffer, load ports





- May 2019: 3 new **Meltdown-type** attacks
- Leakage from: line-fill buffer, store buffer, load ports
- Key take-aways:
 1. Leakage from **intermediate buffers** (\supset L1D)
 2. Transient execution through **microcode assists** (\supset exceptions)



- May 2019: 3 new **Meltdown-type** attacks
- Leakage from: line-fill buffer, store buffer, load ports
- Key take-aways:
 1. Leakage from **intermediate buffers** (\supset L1D)
 2. Transient execution through **microcode assists** (\supset exceptions)

⇒ **How to classify in our tree + lessons learned?**

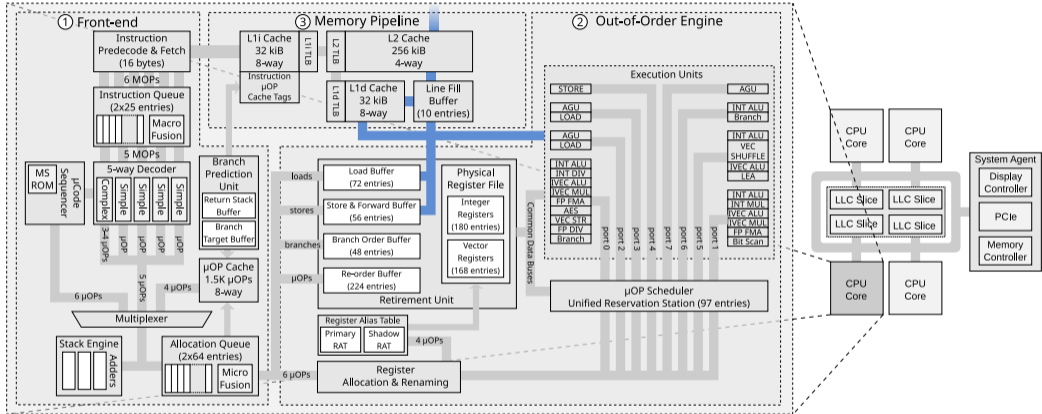


Figure 1: by Stephan van Schaik (<https://mdsattacks.com/>).



- Optimization: only implement fast-path in **silicon**
- More complex edge cases (slow-path) in **microcode**



- Optimization: only implement fast-path in **silicon**
- More complex edge cases (slow-path) in **microcode**
- Need help? Re-issue the load with a **microcode assist**
 - assist == “microarchitectural fault”



- Optimization: only implement fast-path in **silicon**
- More complex edge cases (slow-path) in **microcode**
- Need help? Re-issue the load with a **microcode assist**
 - assist == “microarchitectural fault”
- Example: setting A/D bits in the page table walk
 - Likely many more!



⇒ **MD-faulttype-BUF** naming scheme



⇒ **MD-faulttype-BUF** naming scheme

Update leaves – leakage source: REG, L1, LFB, SB, LP



⇒ **MD-faulttype-BUF** naming scheme

Update leaves – leakage source: REG, L1, LFB, SB, LP

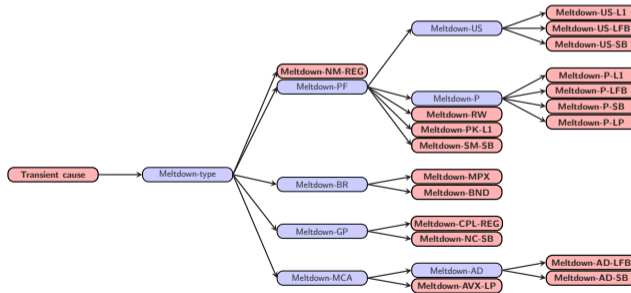
Add sub-branch – trigger Meltdown via μ -code assists

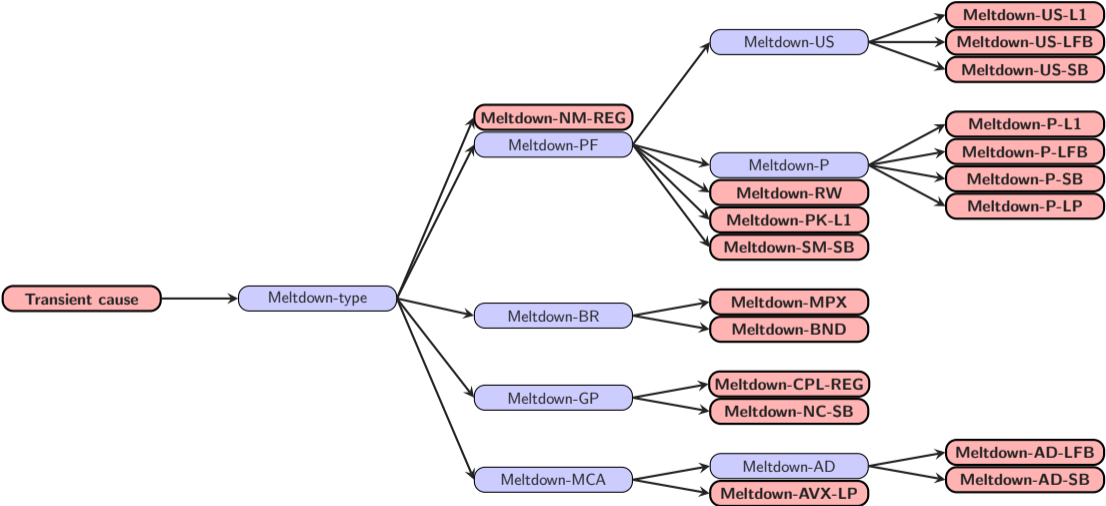


⇒ MD-faulttype-BUF naming scheme

Update leaves – leakage source: REG, L1, LFB, SB, LP

Add sub-branch – trigger Meltdown via μ -code assists

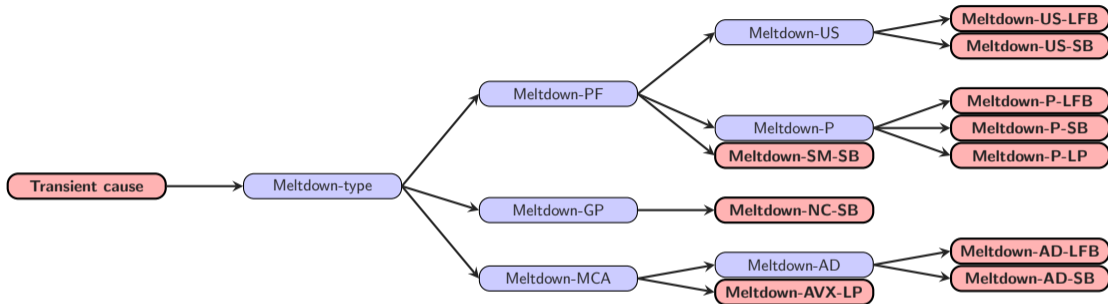




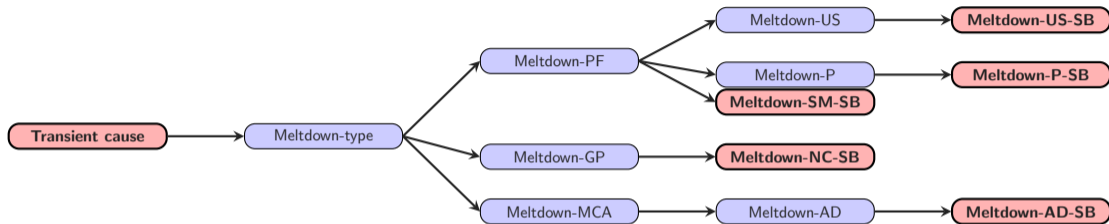




⇒ Our **systematic analysis** (tree search) revealed several overlooked variants (see Canella et al. “A Systematic Evaluation of Transient Execution Attacks and Defenses”, USENIX Security 2019).

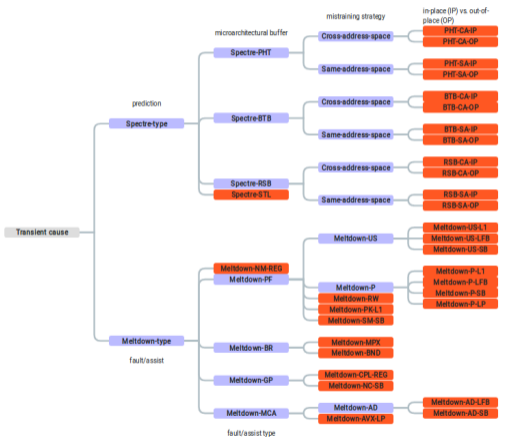


Explore leakage from new **buffers** + microcode **assists**



Not “just another buffer”, include systematic **fault type analysis**

Spectre
 Meltdown
 Buffer ▾
 Vendor ▾
 More filters ▾
 Search Search
 Export ▾



Details

Transient cause



We split the tree based on what the cause for entering transient execution is. If the cause is the handling of a fault or microcode assist upon instruction retirement, we have a Meltdown-type attack. If the cause is a control or data flow prediction, we have a Spectre-type attack.

References

- [A Systematic Evaluation of Transient Execution Attacks and Defenses](#)
 Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtushkin, Daniel Gruss (USENIX Security 2019)
- [Spectre Attacks: Exploiting Speculative Execution](#)
 Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, Yuval Yarom (IEEE S&P 2019)



- Collect information record for each attack:
 - Academic paper references
 - Naming aliases and CVEs
 - Affected vendors (Intel, AMD, ARM)
 - Open-source PoCs
- **Filter** by type, vendor, buffer, etc. → understand and build insights
- TikZ/SVG export
- Pull requests welcome! :-)

Claudio Canella¹, Jo Van Bulck², Daniel Gruss¹

September 23, 2019

¹ Graz University of Technology, ² imec-DistriNet, KU Leuven

Cards Against Confusion

Transient Execution
Tree v1.1.0b
(RDCT/LPFBSDFT)