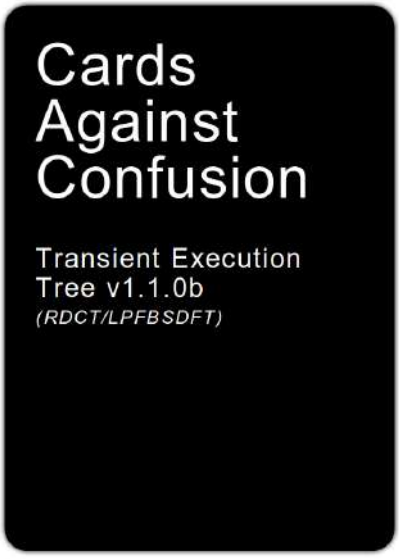
Claudio Canella[1], **Jo Van Bulck**[2], **Daniel Gruss**[1]

September 23, 2019

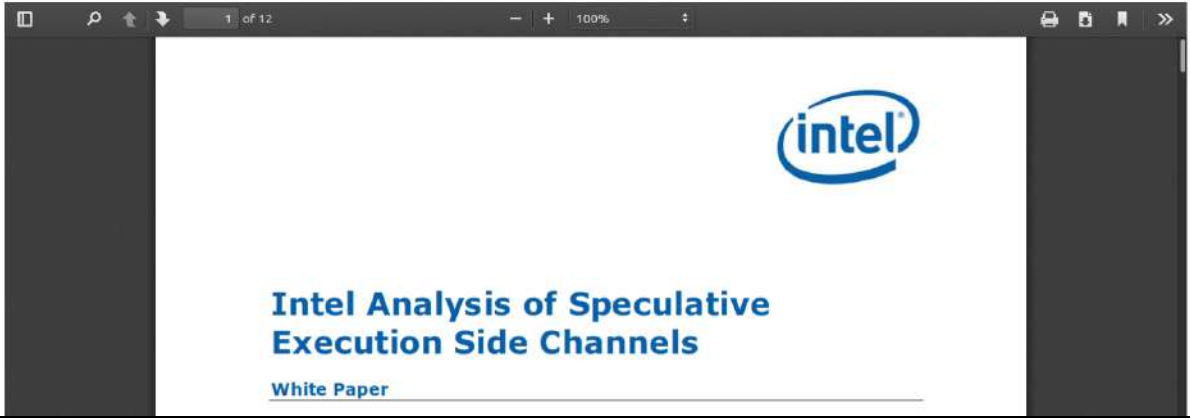
[1] Graz University of Technology, [2] imec-DistriNet, KU Leuven




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

Intel Analysis of Speculative Execution Side Channels

Download PDF



Examples?

Spectre v1,

Examples?

Spectre v1, Spectre v2,

Examples?

Spectre v1, Spectre v2, Meltdown,

Examples?

Spectre v1, Spectre v2, Meltdown, Spectre v3,

Examples?

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

Examples?

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

Examples?

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

Examples?

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

Examples?

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

Examples?

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,

Examples?

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

Examples?

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

Examples?

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

Examples?

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,

Examples?

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,

Examples?

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,

Examples?

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,

Examples?

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,

Examples?

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

Examples?

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...?

Isn't this just all Spectre?

Isn't this just all Spectre?

Isn't this just all Spectre?

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - CPU doesn't know what should happen next

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - CPU doesn't know what should happen next
  - makes a good guess

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - CPU doesn't know what should happen next
  - makes a good guess
- In all these Meltdown-type attacks:

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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!

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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!

  $\rightarrow$ oops ;)

Isn't this just all Spectre?

- **Spec**tre is about mis**spec**ulation
  - 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!

  $\rightarrow$ oops ;)

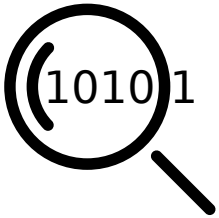
Isn't this just all Spectre?

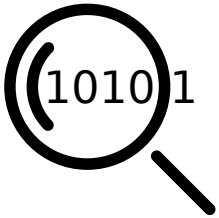
- **Spec**tre is about mis**spec**ulation
  - CPU doesn't know what should happen next
  - makes a good guess
- In all these Meltdown-type attacks:
  - CPU knows it's doing something wrong
  - still does it deliberately
  - it even does things that are architecturally never allowed
  - because you can't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

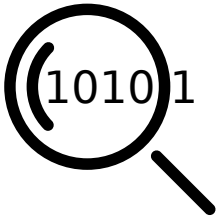
  $\rightarrow$ oops ;)

$\rightarrow$ clear difference in behavior

Isn't this side channels?

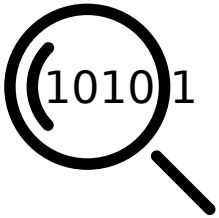

10101


10101

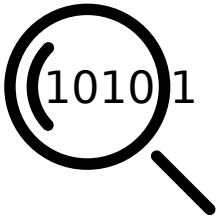


- Step 1: Choose a kernel address and put it in `rcx`

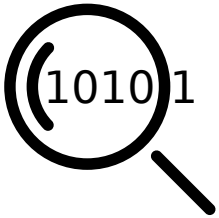
Isn't this side channels?



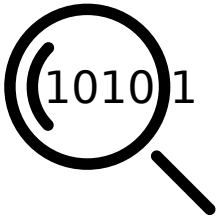
- Step 1: Choose a kernel address and put it in `rcx`
- Step 2: `mov al, byte [rcx]`



- 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`



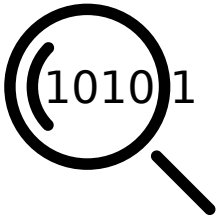
- Step 1: Choose a kernel address and put it in `rcx`
- Step 2: `mov al, byte [rcx]`
- Step 3: You now got the secret in `al`



- 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`

$\rightarrow$ You directly read the value.

Isn't this side channels?



- 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`

$\rightarrow$ You directly read the value. This is not side channels.

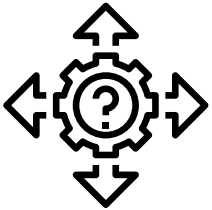
- Clear up naming confusion

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

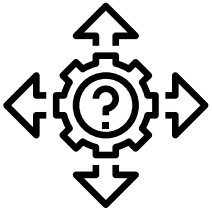
- Clear up naming confusion
- Systematic analysis may show new variants

- Spectre: first class of transient execution attack

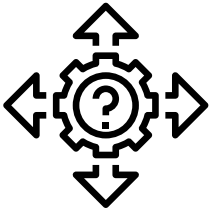
- Spectre: first class of transient execution attack
- Exploit control (or data) flow predictions



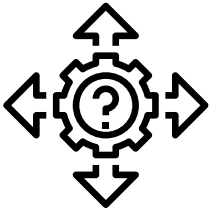
- Many predictors in modern CPUs



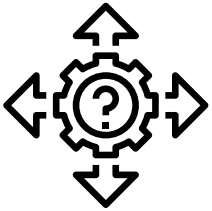
- Many predictors in modern CPUs
  - Branch taken/not taken (PHT)



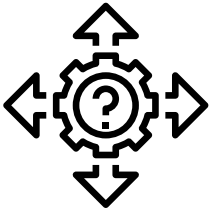
- Many predictors in modern CPUs
  - Branch taken/not taken (PHT)
  - Call/Jump destination (BTB)



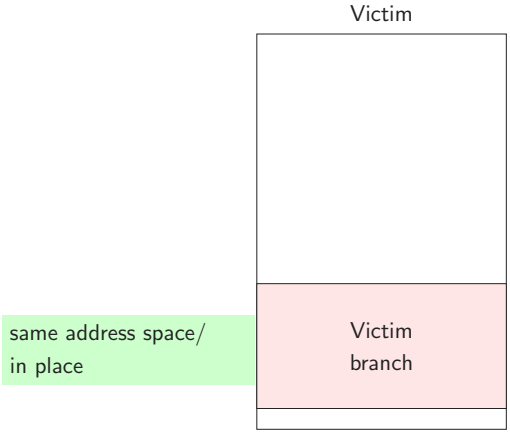
- Many predictors in modern CPUs
  - Branch taken/not taken (PHT)
  - Call/Jump destination (BTB)
  - Function return destination (RSB)

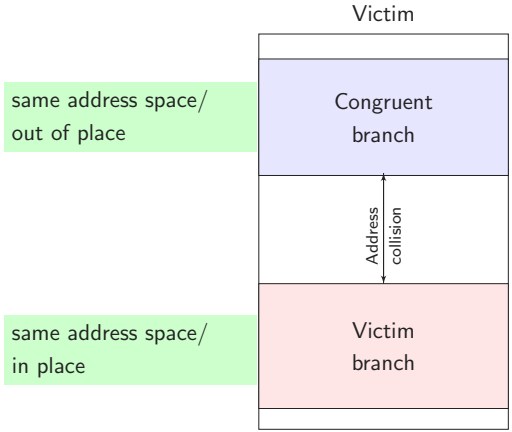


- Many predictors in modern CPUs
  - Branch taken/not taken (PHT)
  - Call/Jump destination (BTB)
  - Function return destination (RSB)
  - Load matches previous store (STL)

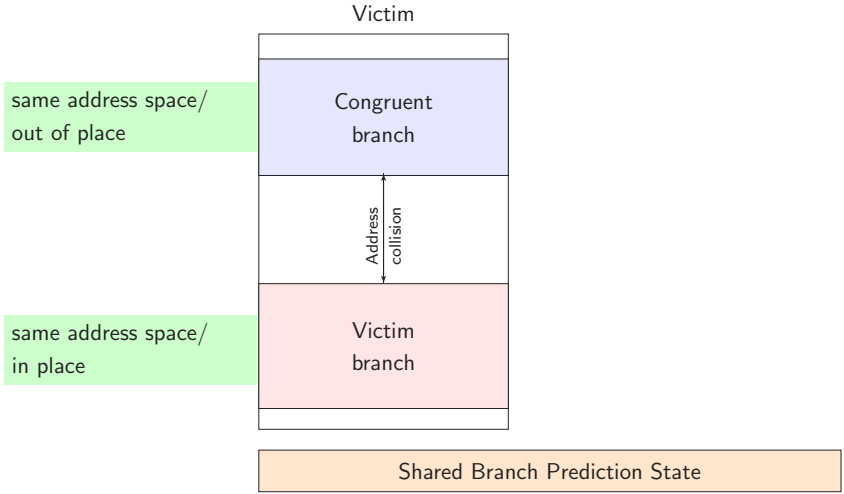


- Many predictors in modern CPUs
  - Branch taken/not taken (PHT)
  - Call/Jump destination (BTB)
  - Function return destination (RSB)
  - Load matches previous store (STL)
- Most are even shared among processes

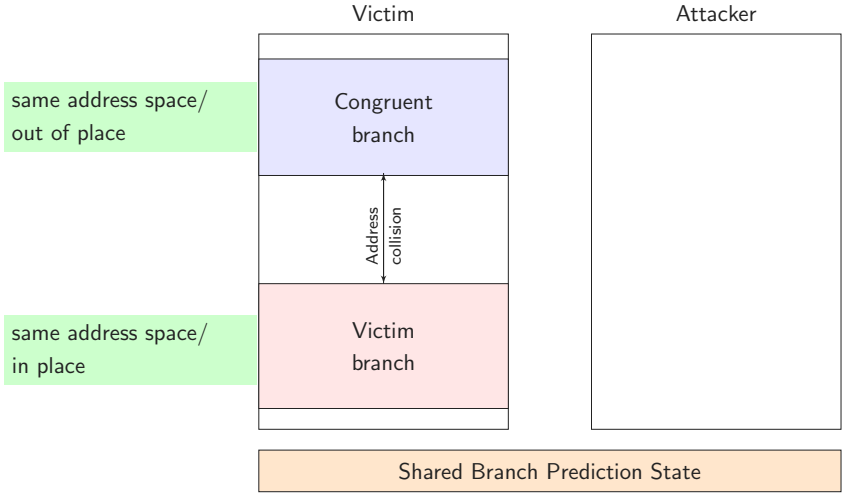

Victim
same address space/
in place
Victim
branch


Victim
same address space/
out of place
Congruent
branch
Address
collision
same address space/
in place
Victim
branch

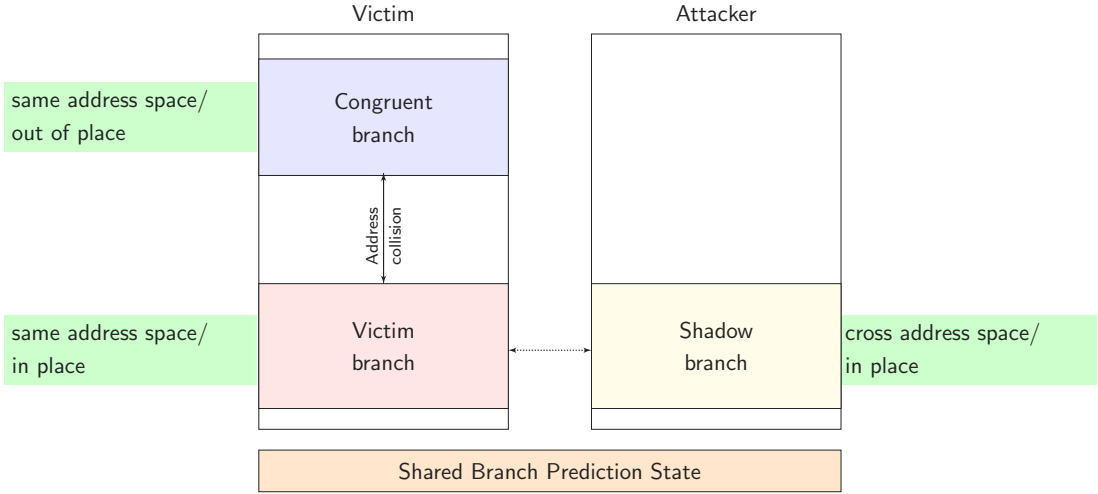
Spectre Mistraining



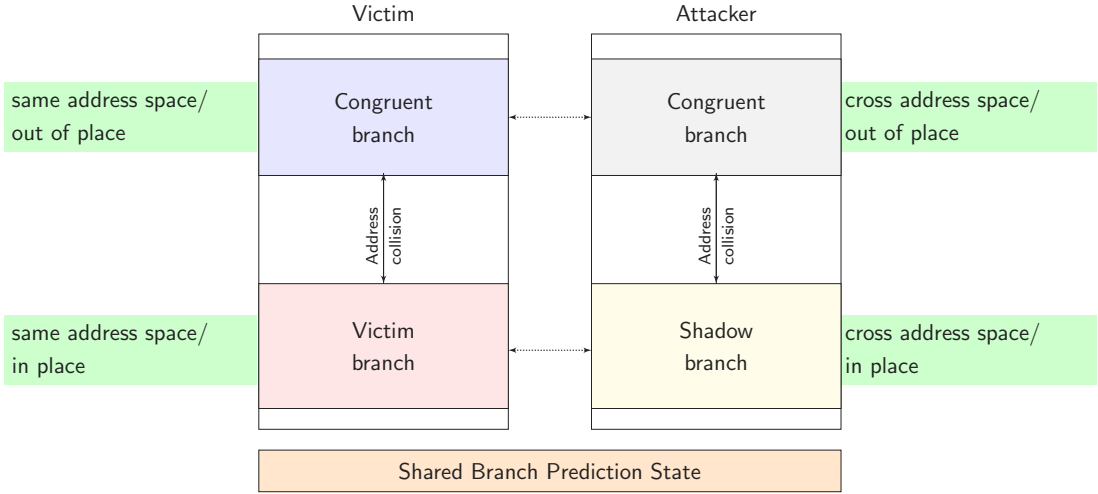
Spectre Mistraining



Spectre Mistraining

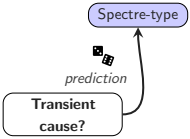


Spectre Mistraining

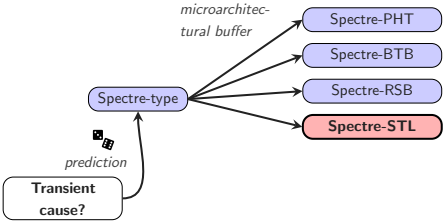


Transient cause?

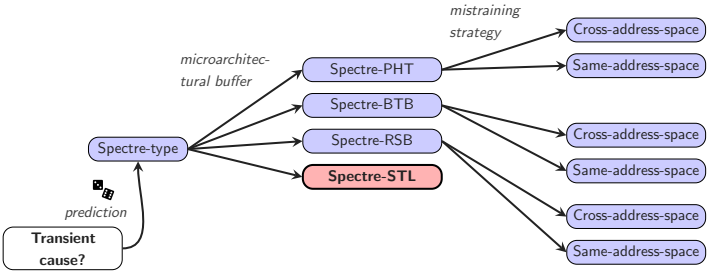
Spectre Variants



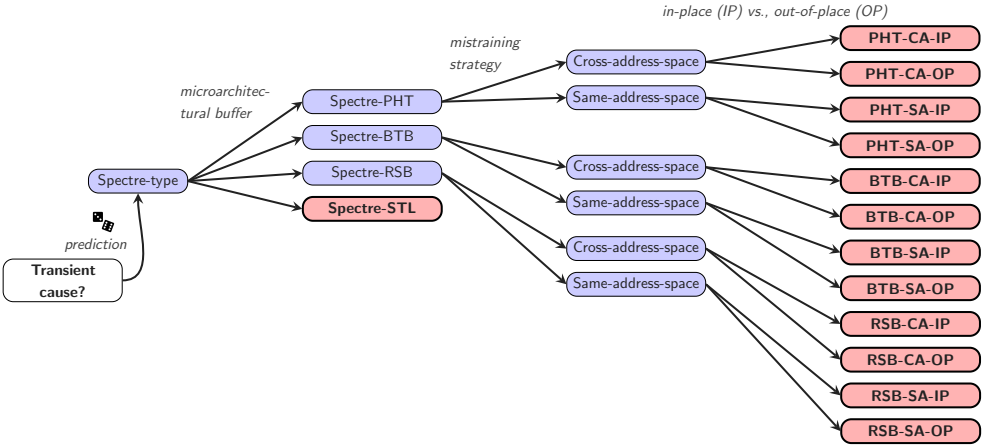
Spectre Variants



Spectre Variants



Spectre Variants



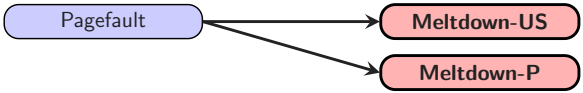
- Meltdown is a separate class of transient execution attack

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

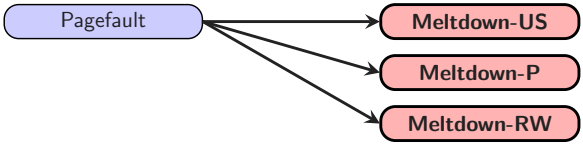
Pagefault

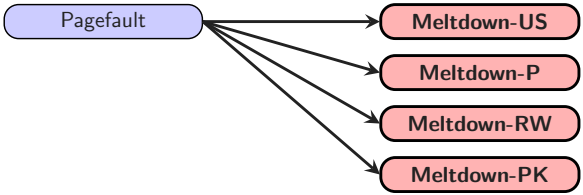
Meltdown Variants

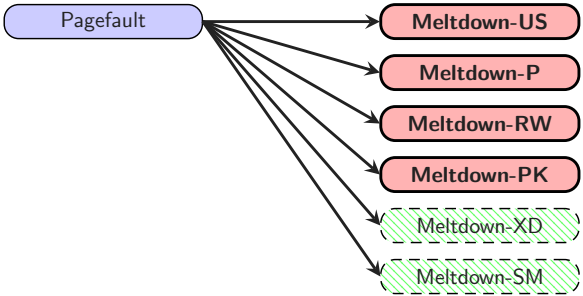
Meltdown Variants



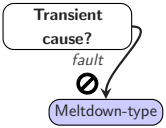
Meltdown Variants



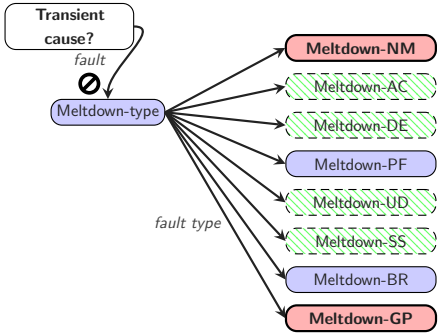

Pagefault
Meltdown-US
Meltdown-P
Meltdown-RW
Meltdown-PK


Pagefault
Meltdown-US
Meltdown-P
Meltdown-RW
Meltdown-PK
Meltdown-XD
Meltdown-SM

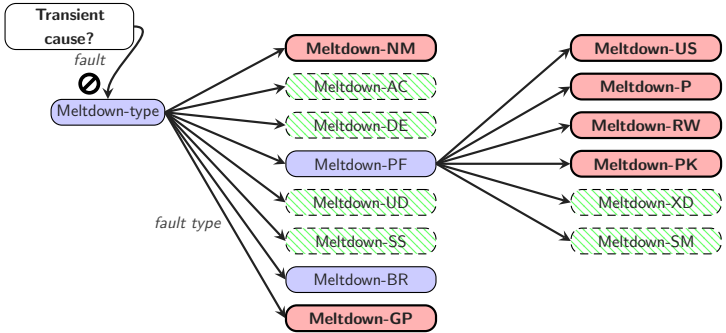
Transient cause?


Transient cause?
fault
Meltdown-type

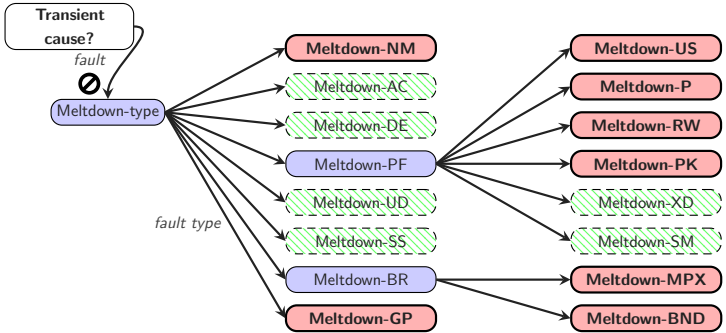
Meltdown Tree



Meltdown Tree



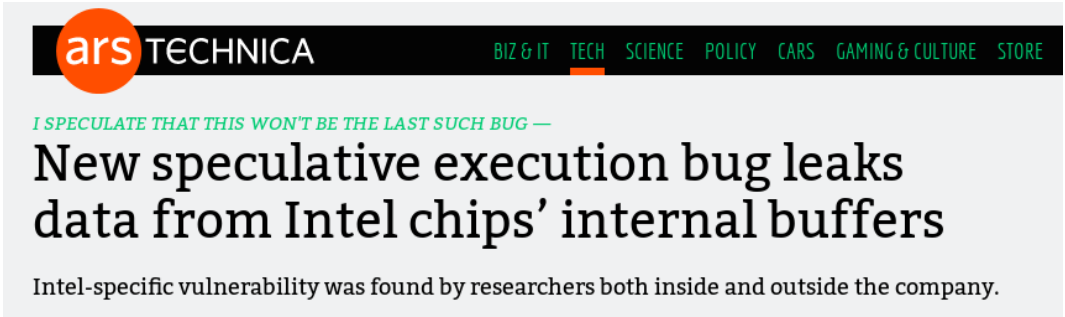
Meltdown Tree



WIRED BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY TRANSPORTATION

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.

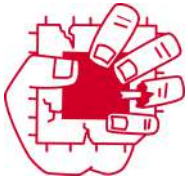


ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

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

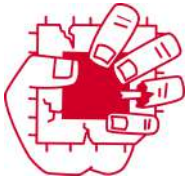
New speculative execution bug leaks data from Intel chips' internal buffers

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 micrcode 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 micrcode assists ($\supset$ exceptions)

$\Rightarrow$ **How to classify in our tree + lessons learned?**

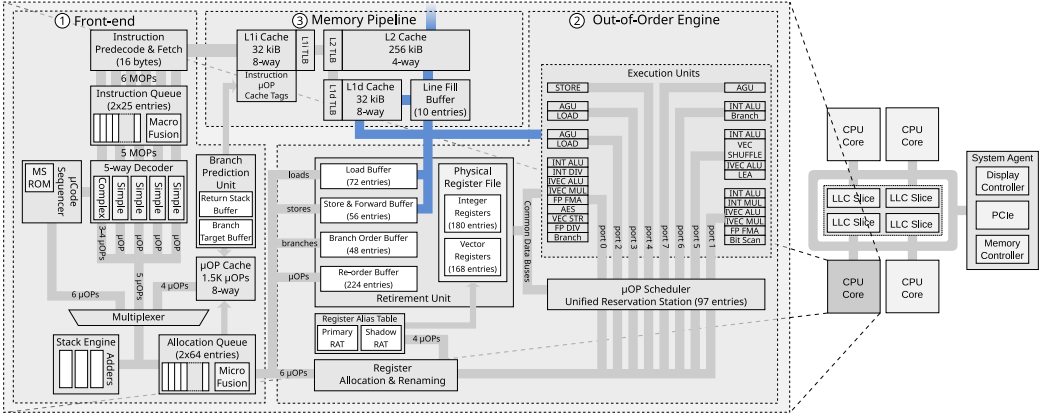


Figure 1: by Stephan van Schaik (`https://mdsattacks.com/`).

- 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!

$\Rightarrow$ **MD-faulttype-BUF** naming scheme

⇒ **MD-faulttype-BUF** naming scheme

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

$\Rightarrow$ **MD-faulttype-BUF** naming scheme

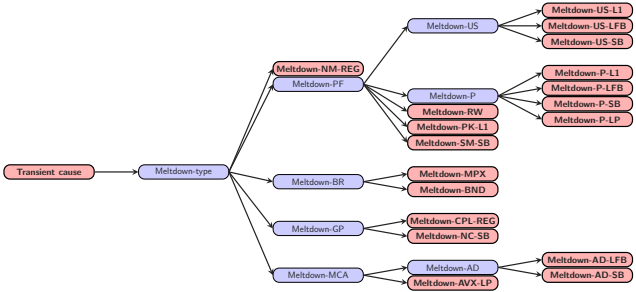
Update leaves – leakage source: REG, L1, LFB, SB, LP

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

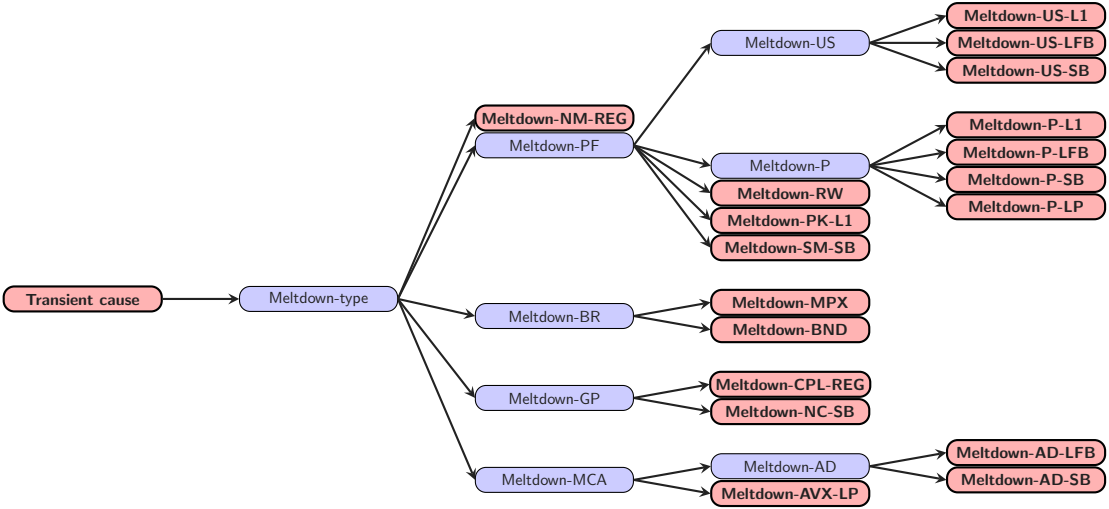
⇒ **MD-faulttype-BUF** naming scheme

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

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



Extended Meltdown tree



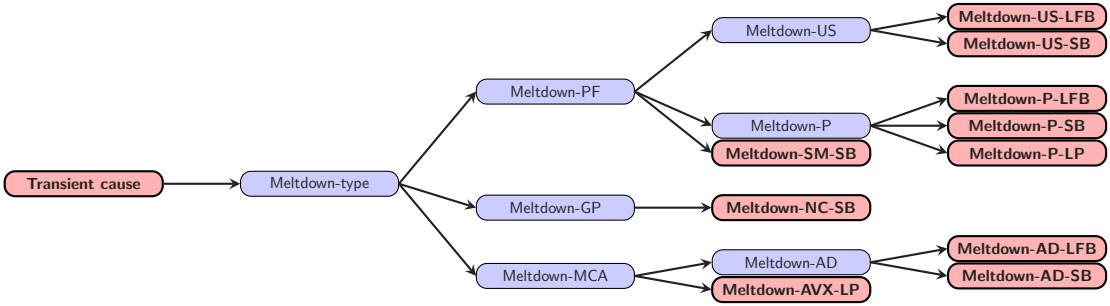
2018 era: Depth-first search (e.g., Foreshadow/L1TF)

2018 era: Depth-first search (e.g., Foreshadow/L1TF)

⇒ Our **systematic analysis** (tree search) revealed several overlooked variants

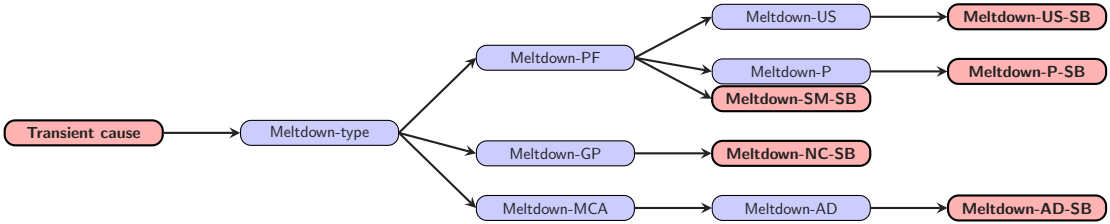
(see Canella et al. "A Systematic Evaluation of Transient Execution Attacks and Defenses", USENIX Security 2019).

2019 era: Breadth-first search (e.g., MDS)



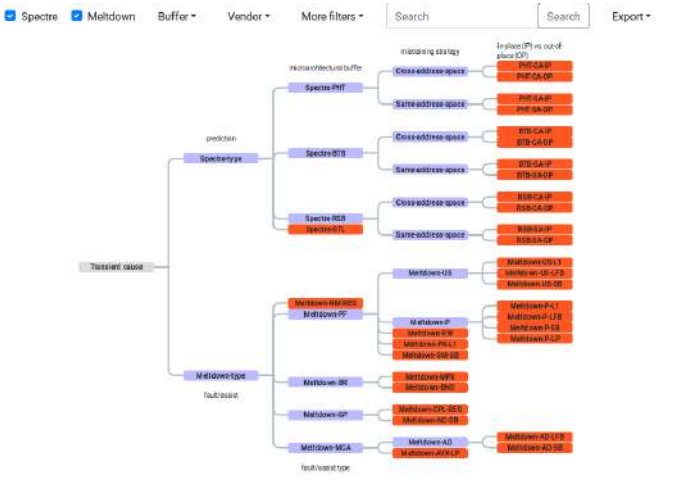
Explore leakage from new **buffers** + microcode **assists**

2019 era: Breadth-first search (e.g., Fallout)



Not "just another buffer", include systematic **fault type analysis**

Interactive JavaScript tree (https://transient.fail)



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 Evtyushkin, 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. $\rightarrow$ understand and build insights
- TikZ/SVG export
- Pull requests welcome! :-)

Claudio Canella[1], **Jo Van Bulck**[2], **Daniel Gruss**[1]

September 23, 2019

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