Advanced Topics in Computer Architecture

Secure Processors 2: Speculative Execution Attacks

Dr. Jonathan Woodruff

UNIVERSITY OF CAMBRIDGE
Computer Science & Technology

Story of Transient Execution Vulnerabilities

- Big surprise in 2017/18!
  - Discovered concurrently by Jann Horn from Google's Project Zero, Werner Haas and Thomas Prescher from Cyberus Technology, and Daniel Gruss, Moritz Lipp, Stefan Mangard and Michael Schwarz from Graz University of Technology
  - Disclosed responsibly
- Large overheads were incurred to mitigate
  - Up to 17% recorded in Amazon for Meltdown mitigation
  - Core i7 8086K: mean 17% overhead for microcode Spectre mitigation
- This is much more overhead than anyone had previously endured for a side-channel attack against general computation

  What is different?

Introduction

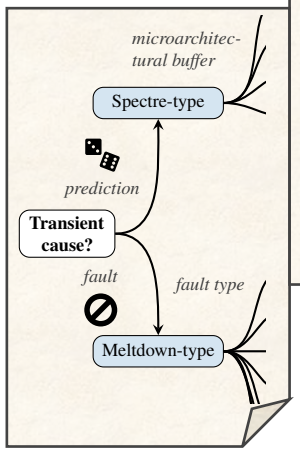
- Introduction to transient execution attacks (Meltdown & Spectre)
- Introduction to defences
- Introduction to testing for transient execution vulnerabilities and verification of transient execution vulnerability defences

Transient Execution Attacks

- Transient execution definition:
  Speculative execution which has failed and is "squashed" in the pipeline.
- Attacks can leak the result of illegal behaviour during transient execution

- Two stages of transient execution attacks:
  - Trigger illegal behaviour that produces secret value
  - Exfiltrate via side-channel
    - Encode in micro-architectural state
    - Decode in architectural state

- Classes of transient execution attacks:
  - Meltdown – leverage transient execution due to exception/fault
  - Spectre – leverage transient execution due to failed prediction



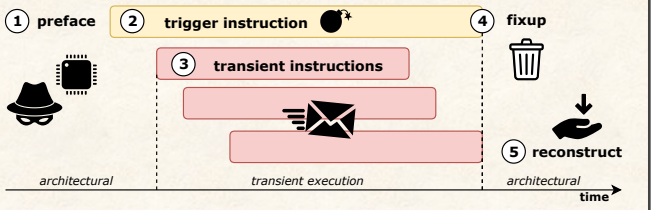
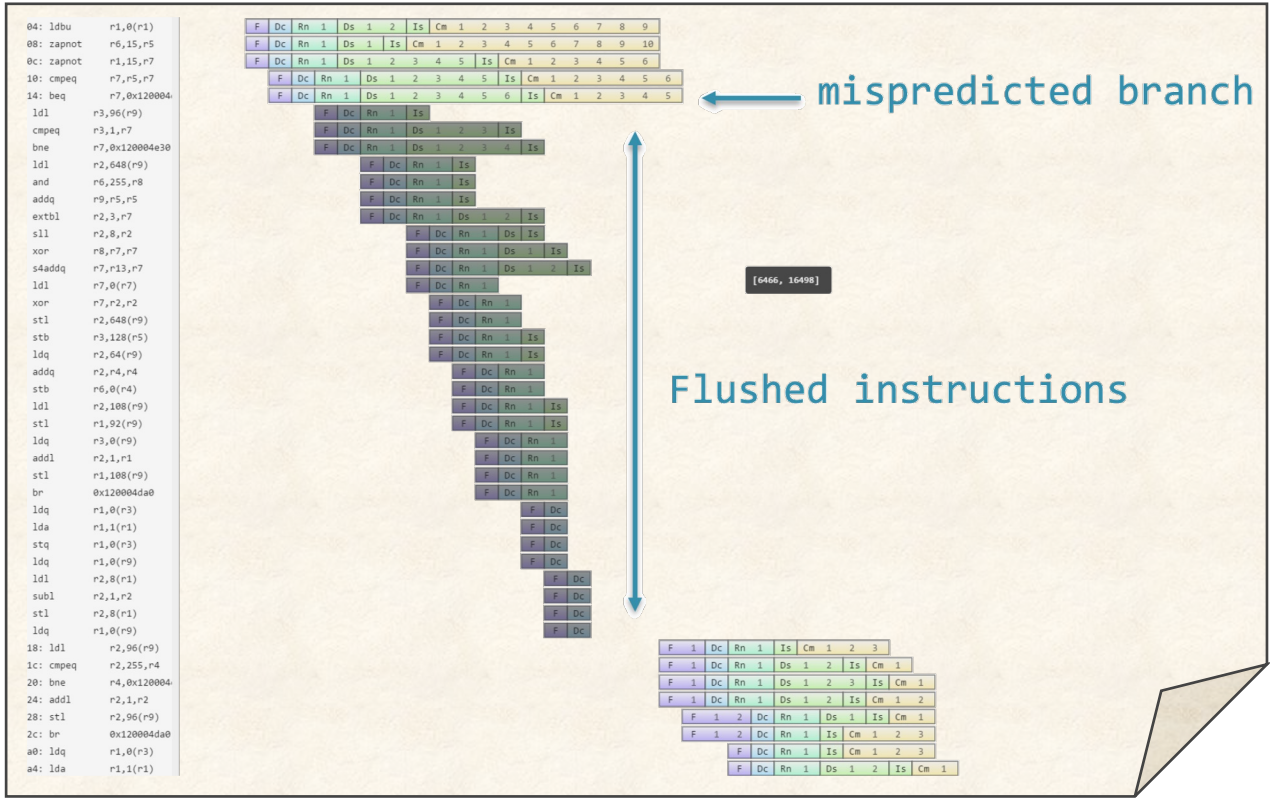


Figure 2: High-level overview of a transient execution attack in 5 phases: (1) prepare microarchitecture, (2) execute a *trigger instruction*, (3) *transient instructions* encode unauthorized data through a microarchitectural covert channel, (4) CPU retires trigger instruction and flushes transient instructions, (5) reconstruct secret from microarchitectural state.

From: Canella, Claudio, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. "A systematic evaluation of transient execution attacks and defenses." In 28th USENIX Security Symposium (USENIX Security 19), pp. 249-266. 2019.

Transient Execution: Definition

- Pipelines speculate to achieve parallelism
  - That instructions will not trap
  - Branch prediction
  - Store/load independence
  - Many others…
- Failed speculation must be detected, and "all" effects squashed to avoid corrupting architectural state
- Transient execution = failed speculation



From: Ryota Shioya. "Visualizing the out-of-order CPU model." In 2018 ASPLOS Learning Gem5 Tutorial. IEEE, 2018.

Transient execution affects timing !

Stages of Transient Execution Attacks

- Access secret in transient execution
  - Example: Meltdown-US loads kernel memory from user space
  - Example: Spectre PHT can perform out-of-bounds load
- Exfiltrate secret through side channel
  - Example: cache
    - Encode: Perform secret-dependent cache-line load of array
    - Decode: Measure time to load array elements
  - Example: variable-delay arithmetic
    - Encode: Perform variable-delay arithmetic using the secret
    - Decode: Measure time spent in transient execution

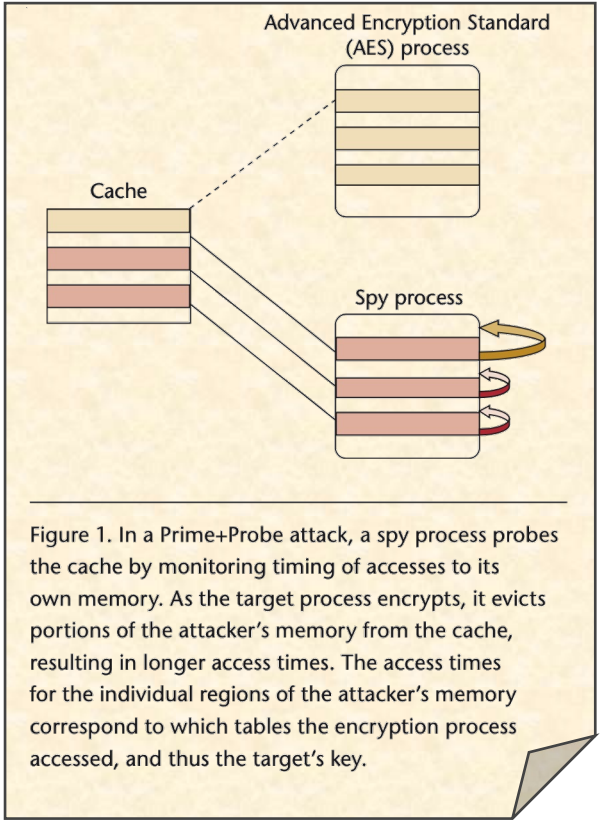


Figure 1. In a Prime+Probe attack, a spy process probes the cache by monitoring timing of accesses to its own memory. As the target process encrypts, it evicts portions of the attacker's memory from the cache, resulting in longer access times. The access times for the individual regions of the attacker's memory correspond to which tables the encryption process accessed, and thus the target's key.

From: Lawson, Nate. "Side-channel attacks on cryptographic software." IEEE Security & Privacy 7, no. 6 (2009): 65-68.

Classes of Transient Execution Attacks

- Spectre: misprediction
  - Example: pattern history table
    - Train branch direction predictor that bounds check will not fail (for example)
    - Call with an enormous offset to anywhere in the address space
    - Ensure that the misprediction is not noticed until the secret is read into the core
  - Example: Store-to-load forwarding
    - Train the store aliasing predictor that a load is independent of an earlier store (the common case)
    - Load from a recently stored location such that the load speculatively overtakes the store
    - Ensure the store address resolution is delayed so that the old value is loaded into the core
- Meltdown: exceptions/faults
  - Example Meltdown-US: user-space/kernel page table violation
    - Attempt load of kernel privilege page (as marked in page table) at user privilege
    - Ensure that the exception pipeline flush is delayed until the value is loaded into the core

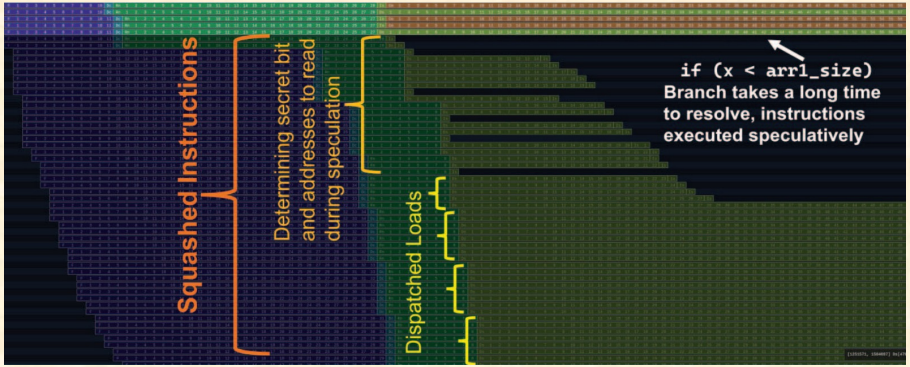


Fig. 12: CPU pipeline trace for Spectre showing loads dispatched during speculation. The bright instructions are executed and committed. The darkened/shaded instructions are flushed due to misspeculation. Loads are still dispatched.

From: Verma, Tarunesh, Achilleas Anastasopoulos, and Todd Austin. "These Aren't The Caches You're Looking For: Stochastic Channels on Randomized Caches." In 2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED), pp. 37-48. IEEE, 2022.

Defenses Against Transient Execution Attacks

- Software
  - Don't use feature that enables speculation

    Discussion Question: How to work around Meltdown-US?

    Hint: *why does translation succeed at all?*
  - Speculation barriers

    Discussion Question: Where might we put barriers to avoid Meltdown-US?
  - Eliminate unsafe speculative paths

    Discussion Question: How might we make a bounds-check conditional branch safe in any case?

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

From: Kocher, Paul, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg et al. "Spectre attacks: Exploiting speculative execution." Communications of the ACM 63, no. 7 (2020): 93-101.

- Hardware
  - Can we limit speculation in "risky" circumstances?
  - Can we define domains for safe speculation?

Testing Transient Execution Attack Defenses

- Can we know if a processor is vulnerable to transient execution attacks?
  - Can we automatically test an existing processor?
  - Can we test an in-development design?
- Approaches
  - Can we automatically discover vulnerabilities with randomized sequences and some sort of expectation?

    What behaviour should be expected? Should this be specified for all implementations?
  - Can we statically check a hardware design to discover where "secret" state is exposed in transient execution?
- A lot of ideas floating round, and lots of things to try!

Papers for this week

- **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

  2019 IEEE Symposium on Security and Privacy

- **Speculative taint tracking: A comprehensive protection for speculatively accessed data**

  Yu, J., Yan, M., Khyzha, A., Morrison, A., Torrellas, J. and Fletcher, C.W.

  2019 International Symposium on Microarchitecture

- **Revizor: Testing black-box CPUs against speculation contracts**

  Oleksenko, O., Fetzer, C., Köpf, B. and Silberstein, M.

  2022 Conference on Architectural Support for Programming Languages and Operating Systems