Establishing a Standard Security Benchmark Suite

Transient Execution Attacks

Leo Ge Dr. Tamara Lehman[2]

University of Colorado Boulder, Department of Electrical and Computer Engineering
[2]Supervisor

October 7, 2023

Motivation

```
1          ...
2          if (i < 10){
3              j = array[i];
4          }
5          ...
6
```

What if this could be a security vulnerability?

→ Goal: *Write software to test for Spectre vulnerabilities.*

Background: Speculative / Transient Execution

- Out-of-order / speculative execution
- Branch prediction
- Consequences of speculative execution

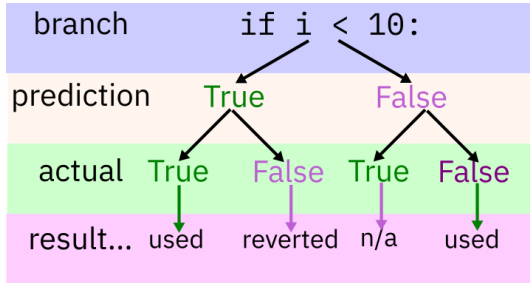

Figure: Diagram of Speculative Outcomes. 'Actual' represents the actual outcome of the branch. 'Result' represents whether the speculative execution result is used.

Speculative Execution Attacks

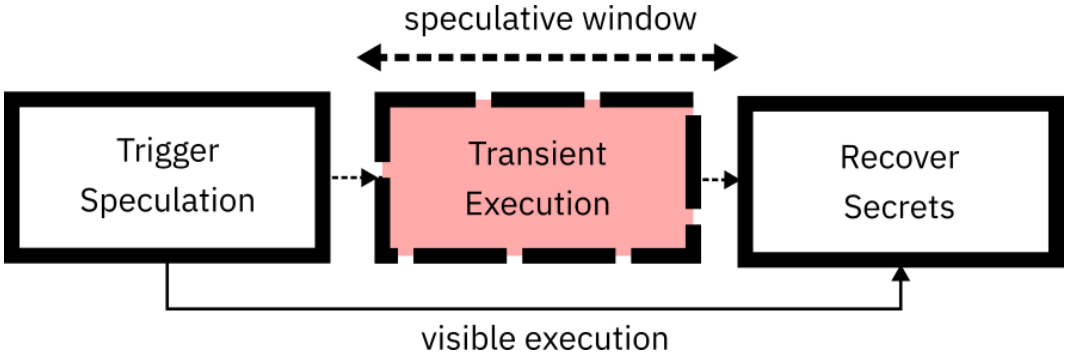

Figure: Basic structure of Speculative Execution Attack (inspired by Canella et. al). While the transient execution does actually happen, the data is reverted, making it seem invisible. However, the *state* is changed.

- Concept of speculative execution attacks, Spectre by Canella et. al [2]

Goals

Goal: *Write software to test for Spectre vulnerabilities.*

- Software should...
  - Target gem5 and native environments
  - Be oriented towards research use – mitigations, architecture testing

Methods

- The gem5 simulator [1]
- Software refinement
  - Traditional debugging
  - gem5 debug traces
  - Computer architecture theory, gem5 source examination
- Two-part division of speculative triggers

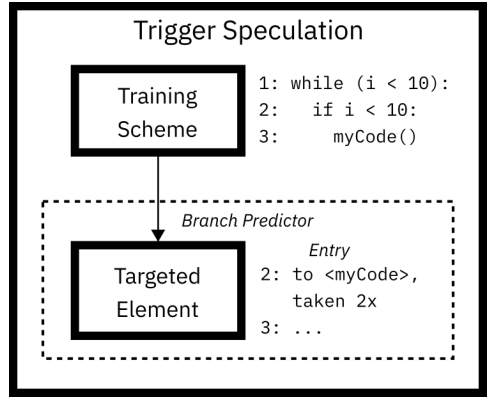

Figure: Canella et. al's paper further divides the speculation trigger into a *training scheme* and a *target element*. The example shown is for a PHT attack.

Speculation Triggers

- Branch predictors rely on history, "training" is the creation of that history
- Parts of the branch predictor have separate histories, and can thus be separately targeted
  - Pattern History Table (PHT): *whether to enter*: `if (i < 10) foo()`
    Train by taking branches
  - Branch Target Buffer (BTB): *where to enter*: `i = foo_ambiguous()`
    Train by calling an ambiguous function
  - Return Stack Buffer (RSB): *where to exit to*: `return 0`
    Train by manipulating stack, calling functions

Results, Contributions

- Operational programs testing for Spectre attacks on three architectural elements
- Successful adaptation of an RSB attack to gem5
- Increased standardisation and modularity of source code

```
attempting to extract secret 'foobar'

pht test...
reading character... time: 38721, time: 38036, success: 0x66='f' score=2
reading character... time: 38221, time: 38442, success: 0x6F='o' score=2
reading character... time: 38262, time: 38127, success: 0x6F='o' score=2
reading character... time: 38010, time: 37834, success: 0x62='b' score=2
reading character... time: 37893, time: 37813, success: 0x61='a' score=2
reading character... time: 38559, time: 38241, success: 0x72='r' score=2

btb test...
reading character... time: 28200, time: 27259, success: 0x66='f' score=2
reading character... time: 27247, time: 27186, success: 0x6F='o' score=2
reading character... time: 27149, time: 27147, success: 0x6F='o' score=2
reading character... time: 27155, time: 27142, success: 0x62='b' score=2
reading character... time: 27141, time: 27144, success: 0x61='a' score=2
reading character... time: 27127, time: 27128, success: 0x72='r' score=2

rsb test...
reading character... time: 16653, time: 16639, success: 0x66='f' score=2
reading character... time: 16626, time: 16626, success: 0x6F='o' score=2
reading character... time: 16626, time: 16626, success: 0x6F='o' score=2
reading character... time: 16626, time: 16626, success: 0x62='b' score=2
reading character... time: 16626, time: 16626, success: 0x61='a' score=2
reading character... time: 16626, time: 16626, success: 0x72='r' score=2
```

Figure: Example execution output.

Conclusions, Limitations

- Learning about computer architecture, lower-level programming
- Incomplete coverage of Canella et al's classification due to gem5's architecture
- Limited testing instrumentation to evaluate mitigations
  - Only measures cycles per extraction, extraction accuracy
  - Expand to specific parts of attack: allows identification of what part is mitigated / by how much

Acknowledgements

Thanks to Dr. Lehman, Ange-Thierry Ishimwe, and the Engineering Excellence Fund.

Citations

[1] Nathan Binkert et al. "The Gem5 Simulator". In: *SIGARCH Comput. Archit. News* 39.2 (Aug. 2011), pp. 1–7. ISSN: 0163-5964. DOI: 10.1145/2024716.2024718. URL: https://doi.org/10.1145/2024716.2024718.

[2] Claudio Canella et al. "A Systematic Evaluation of Transient Execution Attacks and Defenses". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 249–266. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/canella.