Poster: Systematic Analysis of Hardware-Based Spectre Countermeasures

Marton Bognar
DistriNet, KU Leuven
Leuven, Belgium
marton.bognar@kuleuven.be

Jens Sprengers
DistriNet, KU Leuven
Leuven, Belgium
jens.sprengers@kuleuven.be

Frank Piessens
DistriNet, KU Leuven
Leuven, Belgium
frank.piessens@kuleuven.be

***Abstract*—Spectre attacks exploit side effects of speculative execution in modern processors. Unlike other transient execution attacks, such as Meltdown, which could be permanently patched, Spectre variants continue to affect most CPUs manufactured today. In response, the research community and processor vendors proposed a plethora of countermeasures and detection tools, some of which have been integrated into commercial hardware or in critical pieces of software.**

There are multiple surveys that attempt to study Spectre countermeasures with varying levels of rigor and focusing on different aspects. Our study will focus on hardware-based Spectre countermeasures, which have been shown to generally offer much better performance than software-based approaches. Despite their numbers, no framework exists that allows for a meaningful comparison between these countermeasures. In this poster, we propose building a framework that uses a graph-based visualization strategy to compare the enforcement mechanisms and security guarantees of these countermeasures. Together with other evaluation data, such as hardware cost, our goal is to offer an objective and comprehensive comparison method for hardware-based Spectre countermeasures.

1. Introduction

The discovery of Spectre attacks [12] sparked a new research direction and forced processor manufacturers to deploy mitigations. These attacks exploit the speculative behavior of modern processors, such as branch prediction and memory disambiguation. When these mechanisms incur a misprediction, causing the execution of an incorrect control or data flow, secret data might be accessed incorrectly. If the incorrectly executed (transient) instructions encode the secret data in the microarchitectural state (e.g., through a secret-dependent memory address affecting the cache state), this can be recovered later, even after the architectural state has been rolled back upon detecting the misprediction. As the vulnerabilities result from the intended behavior of the optimizations, mitigation is not straightforward. Software mitigations often incur high overheads and offer incomplete security, directing interest toward hardware-based defenses. On commercial CPUs, features such as `IBPB`, `(e)IBRS`, and `SSBD` were introduced to flush microarchitectural state or disable optimizations that cause vulnerabilities. In academia, many proposals aim to eliminate secret leakage by changing the hardware or using a hardware-software co-design, where the software still determines the protection scope.

To keep track of all relevant research and assess trade-offs, survey and SoK papers are invaluable. While many studies summarize or classify transient execution attacks and defenses, they all suffer from shortcomings or are insufficient for comparing hardware-based countermeasures. Most studies are outdated [2], [13] and do not cover more recent attacks, such as those targeting predictors in modern Apple processors [4], [10], [11]. Another aspect, often overlooked by defenses, is the origin of the secret data (whether the target application can access it architecturally), which Guarnieri et al. [7] refer to as the *sandboxing* or *constant-time* model. While one study analyzes this distinction in detail [3], this work considers only software-based mitigations. The only study that focuses on hardware-based countermeasures [9] does not offer a detailed security analysis. The lack of established frameworks for the comparison of security properties leads to claims that are overly broad or have the potential to become outdated ("defends against all Spectre variants") or are used ambiguously in different works ("all Spectre-v1 attacks"). Moreover, to the best of our knowledge, no current systematization includes techniques used for automatic validation of transient execution security.

In our work, we aim to address this gap with a framework centered around a (visual) graph-based model of all speculation-based attacks. Our framework will consist of a well-defined graph that models different stages of Spectre attacks. We can then map countermeasures to the graph, identifying how different strategies aim to stop different stages of the attack, similar to the representation of Szekeres et al. [14] for memory safety. To complete our comparison, we will collect additional metrics, such as the extent of required software changes, performance impact, and hardware cost.

2. The Spectre attack graph

Our systematization framework is centered on a graph representation of speculative execution attacks, including each possible step of the attack flow, e.g., preparation of the microarchitectural state, triggering transient execution, and exfiltrating secret data. In this graph, each path from the start of the execution to the leakage of a secret constitutes a

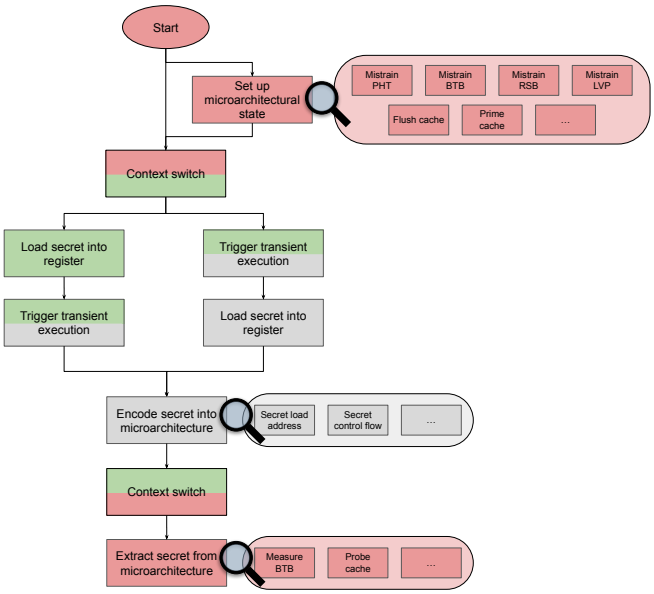


Figure 1. Simplified structure of the systematization framework. Magnifying indicates possible instances of a given state.

Spectre attack variant. This is similar to the representation used for memory safety attacks by Szekeres et al. [14], and also Spectre attacks in earlier works [2], [8]. However, our model differs from prior work in some significant aspects. First, we cover new types of speculative execution attacks that were not known at the time of prior analyses, such as load address and load value prediction [10], [11] and data-dependent prefetching [4]. Second, we make a distinction between the sandboxing and the constant-time leakage model [7], differentiating based on the source of the leaked data, which many proposed defenses in the past did not take into account. Once our graph is constructed (a draft of which is shown in Figure 1), we will continue by mapping the defenses onto the attack stages. Specifically, we note which attack stage is stopped by each proposed hardware-based defense. Using the graph-based model, this mapping offers a powerful overview of the protection scope, revealing possible attack flows that are not covered by the given defense. We believe that our framework can effectively pinpoint *blind spots* of current defenses and enable a meaningful comparison between the security guarantees of different approaches. Combined with the analysis in Section 3, our framework can guide the development of future countermeasures, extending the protection scope of current proposals.

3. Additional analysis

Other metrics. While the graph serves as a tool to qualitatively compare the security guarantees of Spectre countermeasures, we also plan to collect other metrics that can determine the feasibility of an extension. Concretely, we will quantify the necessary software changes to support the hardware-based countermeasure, its performance impact, and its implementation details, such as complexity and incurred hardware cost. We anticipate a wide variety of these factors, as countermeasures are implemented on several platforms, such as the gem5 simulator [5], the BOOM [15], NaxRiscv [1], and Proteus [6] processor designs.

Tools. Similar to defense strategies, there are research tools to automatically detect the presence or absence of Spectre attacks on hardware designs. By also mapping these tools onto the graph, we would be able to determine which tools can be used to validate the security guarantees of different defenses, by checking whether a tool covers all the attacks that the mapping shows to be stopped by the defense.

References

[1] H. Andrianatrehina, R. Lashermes, J. Paturel, S. Rokicki, and T. Rubiano, "Exploring speculation barriers for risc-v selective speculation," in *International Conference on Availability, Reliability and Security (ARES)*, 2025.

[2] C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss, "A systematic evaluation of transient execution attacks and defenses," in *USENIX Security Symposium*, 2019.

[3] S. Cauligi, C. Disselkoen, D. Moghimi, G. Barthe, and D. Stefan, "Sok: Practical foundations for software spectre defenses," in *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[4] B. Chen, Y. Wang, P. Shome, C. W. Fletcher, D. Kohlbrenner, R. Paccagnella, and D. Genkin, "Gofetch: Breaking constant-time cryptographic implementations using data memory-dependent prefetchers," in *USENIX Security Symposium*, 2024.

[5] R. Choudhary, J. Yu, C. Fletcher, and A. Morrison, "Speculative privacy tracking (spt): Leaking information from speculative execution without compromising privacy," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021.

[6] L. Daniel, M. Bognar, J. Noorman, S. Bardin, T. Rezk, and F. Piessens, "Prospect: Provably secure speculation for the constant-time policy," in *USENIX Security Symposium*, 2023.

[7] M. Guarnieri, B. Köpf, J. Reineke, and P. Vila, "Hardware-software contracts for secure speculation," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[8] Z. He, G. Hu, and R. B. Lee, "New models for understanding and reasoning about speculative execution attacks," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021.

[9] G. Hu, Z. He, and R. B. Lee, "Sok: Hardware defenses against speculative execution attacks," in *International Symposium on Secure and Private Execution Environment Design (SEED)*, 2021.

[10] J. Kim, J. Chuang, D. Genkin, and Y. Yarom, "Flop: Breaking the apple m3 cpu via false load output predictions," in *USENIX Security Symposium*, 2025.

[11] J. Kim, D. Genkin, and Y. Yarom, "Slap: Data speculation attacks via load address prediction on apple silicon," in *IEEE Symposium on Security and Privacy (S&P)*, 2025.

[12] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[13] A. Randal, "This is how you lose the transient execution war," 2023. [Online]. Available: https://arxiv.org/abs/2309.03376

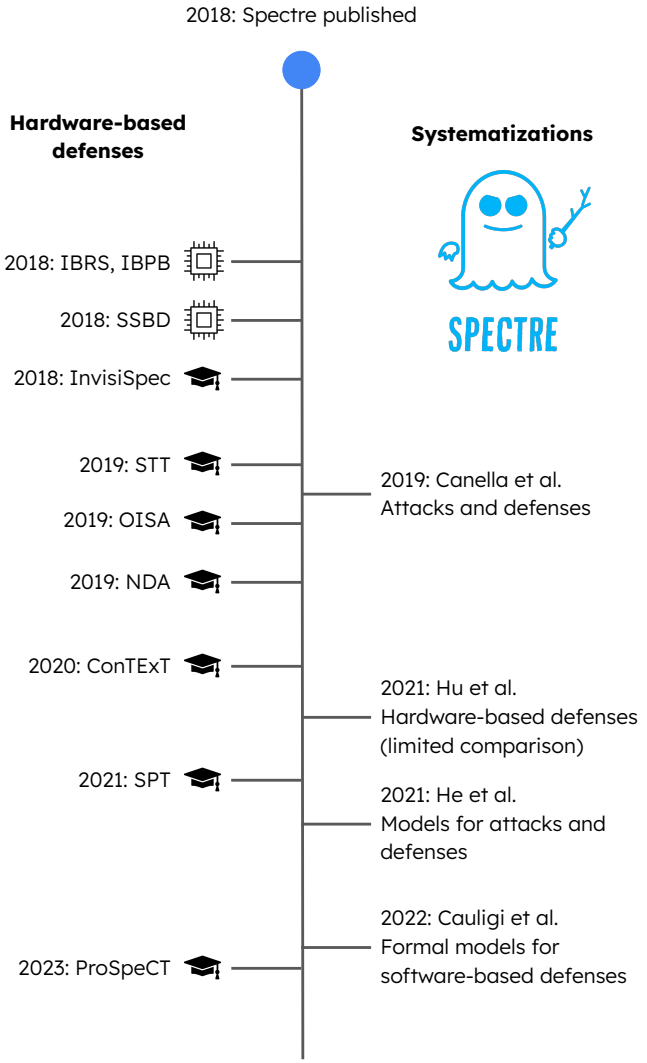
[14] L. Szekeres, M. Payer, T. Wei, and D. Song, "Sok: Eternal war in memory," in *IEEE Symposium on Security and Privacy (S&P)*, 2013.

[15] J. Yu, L. Hsiung, M. E. Hajj, and C. W. Fletcher, "Data oblivious ISA extensions for side channel-resistant and high performance computing," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

Systematic Analysis of Hardware-Based Spectre Countermeasures

Marton Bognar, Jens Sprengers, Frank Piessens
KU Leuven, Belgium
marton.bognar@kuleuven.be

An incomplete historical overview



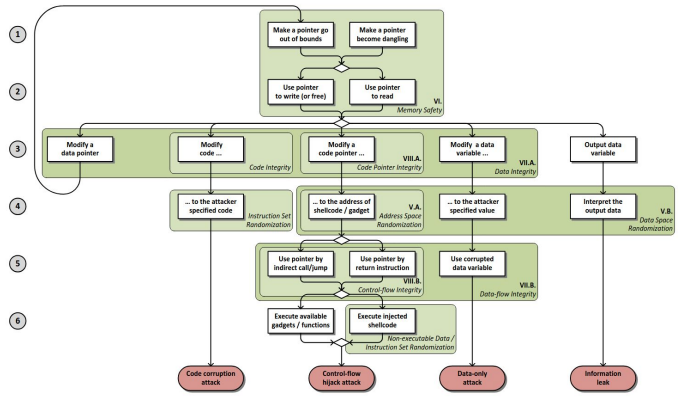
Takeaway: no current systematization of hardware-based defenses for security comparison

Our goal:

- Enable precise reasoning about security
- Compare coverage of existing defenses
- Identify gaps in current mechanisms

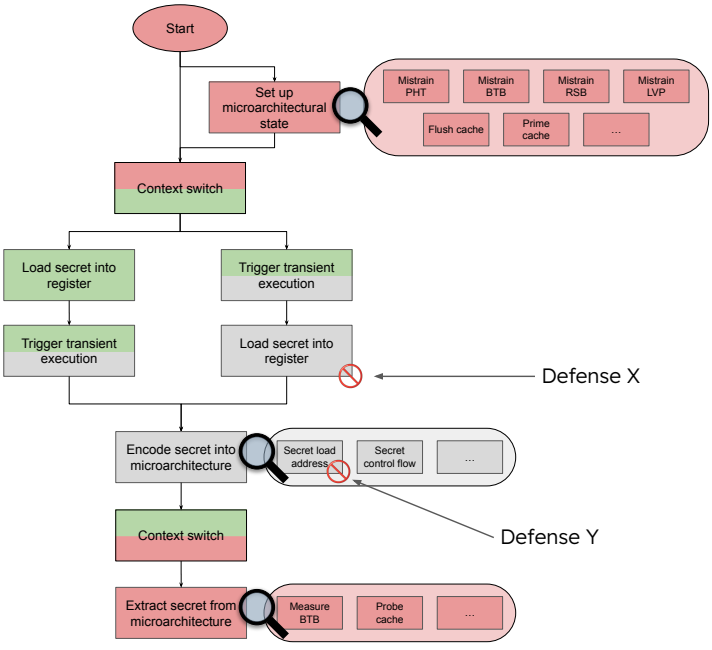
Graph-based analysis

Classification of memory safety issues and mitigations (Szekeres et al., S&P'13):



Similar approach:

- **Stages** of a Spectre attack
  - Secret access – sandboxing/constant-time (Guarnieri et al., S&P'21)
  - Recent primitives: novel Apple predictors
- Mapping of **defenses**
  - Impacted (stopped) attack stage



Additional comparative metrics

- Performance impact
- Required software changes
- Implementation platform
  - Academic - commercial
  - Simulator - hardware design
  - Hardware cost (if available)

Future directions

- Integrating **Spectre detection tools** into the classification
  - Hardware fuzzing
  - Verification-based approaches
- Designing a unified defense
  - Cutting all possible paths in the graph

Cyber Security Flanders DistriNet KU LEUVEN