WiP: Flush-based Cache Attacks on Modern / Multi-Socket x86 Systems

Guillaume Didier
Universität des Saarlandes, Germany;
Formerly: DGA; Univ. Rennes, Inria, IRISA

Thomas Rokicki
CentraleSupélec, Inria, CNRS,
IRISA, Université de Rennes; France

Augustin Lucas
Département d'Informatique, ENS de Lyon;
Univ. Rennes, Inria, IRISA; France

Cache-based side and covert channels are among the oldest and most widely exploited microarchitectural leakage primitives [3, 4]. They remain essential to both offensive and defensive research: most transient execution attacks rely on such channels to transfer data from the transient to the architectural domains [5]. They are also used to reverse engineer undocumented aspects of modern microarchitectures [1].

On x86, the `clflush` instruction provides an unprivileged way to evict cache lines from all levels of the cache hierarchy, enabling fine-grained observation of memory access behavior within (read-only) memory shared by the attacker and victim. It forms the basis of the Flush+Reload attack [6], which remains a fundamental primitive in cache-based side-channel attacks. Gruss et al. [2] also observed that `clflush` latency depends on the cache line's coherence state on Intel CPUs, leading to the Flush+Flush attack.

Modern CPUs are increasingly complex, with several levels of caches, with sometimes per-core slices in the last-level cache, complex on-chip and off-chip interconnect, and, in multi-socket systems, multiple Numa nodes. The latency of memory accesses thus depends strongly on the relative placement of cores, cache slices, and Numa nodes within the system topology, as illustrated in Figure 1. For example, reading data from memory (cache miss) located in a remote Numa node incurs a higher latency (600 cycles on 2× Sapphire Rapids system) than reading from the local node's memory (only 400). While this behavior is intuitive, its impact on the accuracy, reliability, and timing characteristics of cache-based side-channel attacks has yet to be quantified. This gap motivates our study and leads to our main research question:

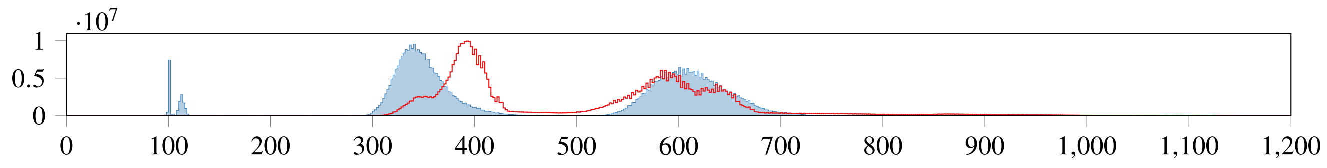
To what extent does the growing complexity of x86 systems affect the effectiveness of flush-based cache attacks?

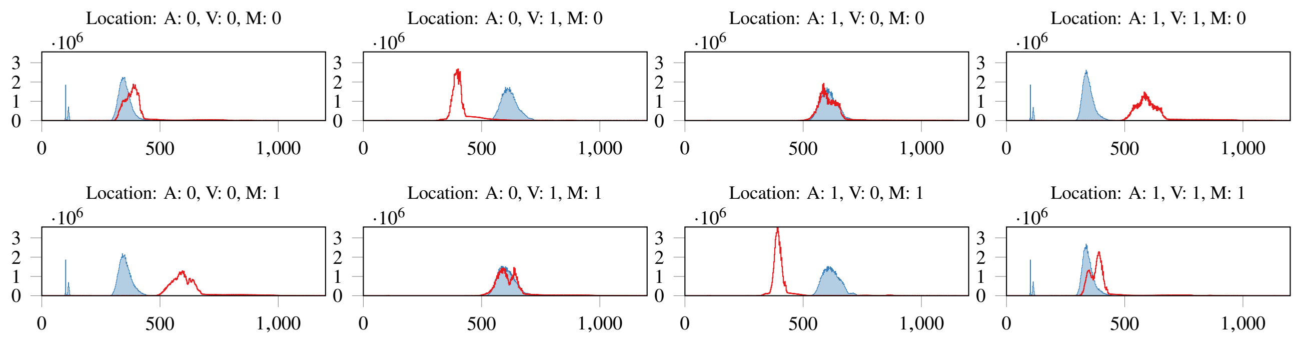
Our study seeks to determine how topology affects the effectiveness of Flush+Flush and Flush+Reload attacks, and how attacker awareness of these factors improves accuracy and bandwidth. We perform a detailed analysis of memory latency across all relevant topological dimensions, including attacker and victim core placement, cache slice, and Numa node, to quantify their respective impacts on timing variability and error rates. To further quantify the gains of topology awareness, we evaluate the bandwidth and error rates of different topological configurations in a covert-channel setting, assessing the speed and robustness of these primitives in practice. To ensure that our observations generalize across platforms, we conduct a large-scale study across 36 Intel and AMD systems, including client, server, and multi-socket architectures.

We find that topology awareness, in particular Numa awareness, can reduce error rates of attacks by up to an order of magnitude compared to topology-unaware methods. For instance, on `esterel41`, a two-socket Intel Sapphire Rapids machine, topology-unaware Flush+Flush has an average error rate of 27.55%, while the topology-aware average is 1.53%. If the attacker is not merely aware, but is allowed to select the best configuration, then the error rate drops below 0.01%.

We also present Load/Flush+Reload, a more accurate variant of Flush+Reload, which provides a higher bandwidth, however limited to a covert-channel scenario. On a single-socket Zen 5 machine, it achieves a true capacity of 4.98 Mbit/s, a 2× improvement over Flush+Flush and Flush+Reload, which only achieve, respectively, 1.95 Mbit/s and 2.27 Mbit/s.

— We demonstrate that topology is a major contributor to load and `clflush` timing, with Numa being the most significant factor in multi-socket systems.
— We show that, on modern systems, accurate Flush+Reload and Flush+Flush attacks are enhanced by topology awareness. As illustrated in Table 1, the hit/miss classification error rate of a Flush+Reload is reduced by 2 order of magnitudes by taking into account NUMA-topology on a 2-socket Sapphire Rapids processor.
— We present an evaluation of the impact of various parameters on cache attack performance, on 36 x86 systems.
— We show that Flush+Flush is widely applicable on x86 machines, including AMD and multi-socket systems, and may be more accurate than Flush+Reload in many cases.
— We present a new covert-channel primitive, Load/Flush+Reload, whose bandwidth can reach 2× that of Flush+Reload and Flush+Flush, and far lower error rates.


(a) Topology-unaware histogram, weighing equally all possible configurations



(b) NUMA-aware histograms, under the NUMA-AVM model.

Figure 1: Flush+Reload timing histograms on `esterel41`, (2× Sapphire Rapids). In red, outlined, loads of an invalid line (I) *i.e.*, cache misses. In blue, filled, loads of a victim-exclusive line (E), *i.e.*, cache hits.

Table 1: Summary for `esterel41`, (2× Sapphire Rapids).

Model	F+R	F+F	LF+R
Topology-Unaware	27.55	34.15	< 0.01
Topology-Aware	1.53	6.57	< 0.01
Best Topology-Aware	< 0.01	< 0.01	< 0.01

Acknowledgments

Multi-socket experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr/).

This project started at IRISA, with support from the French DGA, and was then pursued at Saarland University. This work has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101020415).

We would also like to thank Stefan GAST for the measurements ont the `lab32` (Zen4c) machine, and Jan REINEKE for his advice and thorough feedback.

References

[1] Guillaume Didier, Clémentine Maurice, Antoine Geimer, and Walid J. Ghandour. Characterizing Prefetchers using CacheObserver. In *SBAC-PAD*, 2022.

[2] Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+Flush: A Fast and Stealthy Cache Attack. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2016.

[3] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, 2006.

[4] Colin Percival. Cache missing for fun and profit. In *BSDCan*, 2005.

[5] Wenjie Xiong and Jakub Szefer. Survey of transient execution attacks and their mitigations. *ACM Computing Surveys*, 2021.

[6] Yuval Yarom and Katrina Falkner. Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *23rd USENIX Security Symposium*, 2014.

A Artifacts

Our current experimental code can be found on GitHub[1]. We also provide an online supplement on Zenodo, available at `https://doi.org/10.5281/zenodo.17987842`, which contains our current results, for all 36 machines. We also intend to make available the experimental results files (> 10 GiB) and the rust library needed to deserialize them, in a later version of the Zenodo upload.

[1] `https://github.com/GuillaumeDIDIER/flush-based-cache-attacks-modern-x86`