RETROSPECTIVE: New Attacks and Defense for Encrypted-Address Cache

Moinuddin Qureshi (Georgia Tech)

This ISCA-2019 paper was a follow-up to our prior MICRO-2018 paper (CEASER), which proposed a low-cost randomized cache design using encrypted-address. This paper proposed faster eviction-set algorithms, which broke prior secure caches, and a robust randomized cache (CEASER-S). This paper represented a significant jump in both attacks and defense for randomized caches. Personally, this was truly a fun paper, where for several months, I had posed the problem of eviction-set generation as a ”spy” problem to dozens of people, to see if they could come up with an even better solution. The second attack that exploits the replacement policy was discovered accidentally while making the conference slides for MICRO-2018. Finally, the solution was a simple combination of CEASER with Seznec's brilliant idea of skewing. We hope to share the backstory, the context, and the impact of the paper.

I. The Backdrop

While my group had previously done work on improving the performance of secure memory designs (e.g. SYNERGY at HPCA-2018), coming up with new secure architectures was not really a focus of our group. In Jan 2018, we became aware of the Spectre/Meltdown attacks, which greatly increased our interest in security. One of the initial secure processor designs that we were exploring (with a student in my graduate class) was to invalidate lines that were brought on the wrong path. My PhD student, Gururaj Saileshwar (who was also then the TA for my class) had correctly determined that such a design would not be secure as an adversary can simply do Prime+Probe to learn which set was invalidated. We realized that Prime+Probe could be defeated with a randomized cache. Unfortunately, the randomized cache designs at that time (based on the indirection table) were designed for L1 cache, and scaling them to LLC would cause unacceptable storage overheads. Furthermore, they relied on OS support to protect the indirection table. Ideally, we wanted a randomized cache design without relying on any indirection table or OS support.

II. CEASER: Randomization via Encryption

About a decade earlier, while at IBM, I had worked on a similar problem of avoiding indirection tables in the context of wear-leveling. At MICRO-2009 we had presented *Randomized Start-Gap*, an algorithmic wear leveling design (with block cipher for randomization) that eliminated indirection tables for wear leveling. The insights for our randomized cache were inspired by Start-Gap. Unfortunately, this insight came only a month before the MICRO submission deadline, and at that time all my graduate students were busy as each was leading a submission to MICRO-2018, so they were unavailable to work on secure caches. Rather than postponing this work to HPCA, I decided to use the week of the upcoming spring break to work on the paper, and the result was CEASER.

CEASER had two main ideas. First, using encrypted line-address (using a block cipher) to break the line-to-set mapping. Second, remapping, whereby the line-to-set mapping changes dynamically (by changing the key of the block-cipher). CEASER required negligible storage and performance overheads. To evaluate the security properties of CEASER, we had use the state-of-the-art eviction set algorithm [1] at that time, which required $O(L^2)$ accesses, which meant even with a remapping rate of 1% (one line move per 100 accesses the cache) CEASER could provide years of security. Thus, CEASER was a practical design for securing large caches.

III. Towards Faster Eviction-Set Generation

The security of CEASER relies on the ability of the attacker to form an *eviction sets* (a group of lines that map to the same set so as to cause an eviction from that set). The best-known algorithm [1] at that time had three steps: (1) Launch L random lines at the cache and access them again, redo L until you get at-least one miss, (2) Converge on the eviction set by removing one line and testing the remaining lines to see if there is still a conflict miss, if so discard the removed line, else retain the removed line. Repeat until you are left with only conflicting lines (3) Use the eviction set to perform an attack. This algorithm has quadratic complexity.

The insight to form faster eviction came on the day after submitting the camera-ready version of CEASER, and it happened with the simple thought experiment: what would happen if we removed 2 lines (a group) from ”L” instead of a single line? We would get almost 2x faster attack, as very few lines tend to map to the same set. For this new attack (called Group Elimination Method or GEM), the main question was what should be the size of the group. After some experimentation and analysis, we had concluded that it is best to have W+1 equal-sized groups for a W way cache (so each group would have L/(W+1) lines).

It was unclear at that time if this group size was the best or if there are even faster search methods to converge on the conflicting lines. To help with finding the best algorithm, I had created a *spy-problem* and shared it with a dozen friends and students, and posted it on a popular puzzle website [3]. While we received a lot of creative solutions, none of them was faster than the solution that we already had. Soon, we discovered that the search algorithm was unnecessary.

IV. From Fast Search to No Search

The classical search-based identification of conflicting lines relies on the assumption that you do not know the replacement policy of the cache. Turns out that replacement policy plays a critical role. This became clear while I was making the conference presentation for CEASER (MICRO-2018). I had an animation to explain the existing eviction set algorithm. It had an access pattern with 5 lines (ABCDE) out of which three lines (ACE) conflicted on the same set. Once a LRU-managed cache is accessed with ABCDE and then again with ABCDE, you get misses for all the conflicting lines (ACE), due to the thrashing property of LRU. Then, there is no need for a search algorithm. Thus, accidentally, we discovered a second, much faster attack, that did not require any search algorithm and reduced the complexity from L^2 of prior work and $L.W$ for GEM to 2L. We generalized this line of attack to other replacement policies (such as RRIP and Random) as well. Our attack, that exploits the property of the replacement policy, is currently the fastest known eviction set algorithm.

V. The Impact of Faster Attacks

The security of CEASER (or in general any encrypted address cache) relies on the ability of the attacker to form an eviction set. With an eviction set algorithm that had quadratic complexity, a low rate of remapping (say 1%) was sufficient to ensure security. However, when the complexity reduces to linear, the remapping rate must be increased significantly, to more than 100% (one or more lines moved every access to the cache). This rate of remapping would incur significant performance overheads. Thus, these new attacks render CEASER unusable in practical systems. Other options, such as table-based schemes, while secure, would incur prohibitive storage and performance overheads (while also needing OS support).

VI. The CEASER-S Defense: Combine New with Old

To develop a robust design for a randomized cache, our key insight was to allow flexibility for the given line to be present in multiple sets. This would make it hard for the adversary to form an eviction set for two reasons: First, the eviction set would need to be for all possible places where the line could reside. Second, the lines that form the eviction set also get scattered to multiple possible locations.

Andre Seznec's classic paper from ISCA-93 [5] introduced the skewed-cache design that uses a different hash function for each way, thus allowing a cache line to be mapped to a different set in each way. Our proposed design, CEASER-S (S for Skewed), combined CEASER and Skewed-Cache. CEASER-S splits a 16-way cache into two halves of 8-way, and each half performs CEASER independently (with a different set of keys for the cipher). Thus, CEASER-S not only randomized the line-to-set mapping (using a block cipher) but also the line-to-skew mapping by selecting the skew randomly when the cache line is installed in the cache.

CEASER-S required negligible storage and performance overheads. Given the two dimensions of randomization, CEASER-S could thwart even our new faster attacks on eviction set generations, as it becomes extremely hard for an adversary to converge on a small set of lines that are guaranteed to dislodge the given line from all the skews.

VII. The Initial Reception and Impact

This paper appeared in June 2019, at a time when similar efforts were being pursued in the security community. An Oakland paper [6], which appeared in May 2019, looked at faster eviction set generation using group testing, which was similar to our first attack. However, our second attack (exploiting the replacement policy) continues to be the current fastest method to converge on an eviction set. CEASER-S uses skewed cache for randomizing the location. A USENIX-Security paper, that appeared in Aug 2019, proposed Scatter-Cache [7], which also uses block-ciphers as hash-functions to skewed cache. However, CEASER-S additionally changes the line-to-set mapping continuously thus offering stronger security. Thus, ideas similar to the ones proposed in our ISCA-2019 paper were being argued for, independently, within the security community, providing even greater validation to the insights and applicability of our design. Later studies have also looked at changing the remapping rate of CEASER-S dynamically based on the cache access pattern.

VIII. Ending the Arms Race

Newer attacks motivate strong defenses, and strong defenses motivate better attacks. As guaranteed eviction of a line is hard to do in CEASER-S and ScatterCache, future attacks [2] started exploring probabilistic eviction set generation, whereby a set of lines are identified that can dislodge the given target line with a non-negligible probability. Skewed designs (or in general any design that is not fully associative) would be vulnerable to such probabilistic attacks. For almost two years, there was an arms race between randomized cache designs and intelligent eviction set algorithms. To put an end to the arms race between the cache attacks and randomized caches, we developed a practical fully associative cache, MIRAGE [4], which uses tag-to-data indirection and power-of-two choice load balancing between two skews to virtually eliminate set conflicts, and thus conflict based cache attacks. So far, MIRAGE has remained undefeated for the past two years.

One of the main learning from doing research in randomized cache is that it is important to think about principled security properties first (e.g. fully associative cache) and then implement the design, rather than the other way around.

References

[1] F. Liu *et al.*, "Last-level cache side-channel attacks are practical," in *Security and Privacy (SP),*, 2015.

[2] A. Purnal *et al.*, "Advanced profiling for probabilistic prime+ probe attacks and covert channels in scattercache," *arXiv:1908.03383*, 2019.

[3] M. Qureshi, "The spy problem." [URL]: https://fivethirtyeight.com/features/how-much-will-it-cost-to-sniff-out-the-spies/

[4] G. Saileshwar *et al.*, "MIRAGE: Mitigating Conflict-Based cache attacks with a practical Fully-Associative design," in *USENIX Sec*, 2021.

[5] A. Seznec, "A case for two-way skewed-associative caches."

[6] P. Vila *et al.*, "Theory and practice of finding eviction sets," in *Security and Privacy (SP)*, 2019.

[7] M. Werner *et al.*, "ScatterCache: Thwarting cache attacks via cache set randomization," in *USENIX Security*, 2019.