Side-Channel Attacks on BTree

Dana Dachman-Soled Stuart Nevans Locke Shir Maimon
Robert Metzger Aria Shahverdi
Laura B. Sullivan-Russett

December 6, 2017

Abstract

An increasing need for parallelism have pushed the CPU designers to let different processes running at the same time on a shared hardware resource. This can be both beneficial in terms of doing more computation in fixed amount of time but can potentially introduce new challenges which has not been considered before. Another increasing service offered today is cloud computing. Similar to the idea of parallelism, a hardware is shared with different users allowing them to run their process independently. There has been efforts to secure each process in software space, however the possible leakages in micro-architecture has been ignored. Since processes basically share same resource, one process can act a spy process and track the usage of the shared resource and gain some information about the other process which we call it victim process. One example of this shared resource is a cache. Two processes run on a same core share L1/L2/L3 cache while two processes running on different core share L3 cache.

Caches were identified as a possible source of leakage as early as 2005 in a work by Percival [1]. This is one of the preliminary works in the area which mentioned that the caches are possible source of side-channel attacks in a processor. Zhang et al. [**?**] showed an attack which could retrieve cryptographic key by an attacker which is running on separate VM from victim. In another work Yarom and Falkner [3] showed Flush and Reload attack on L3 cache. In this work the attacker and victim are using shared library which gives an ability to attacker to flush a line which is being used by victim. Acıiçmez [**?**] showed for the first time that instruction cache can also be a source of leakage and attacked OpenSSL's RSA implementation.

There is a timing difference between cache hit and cache miss.

As it can be seen in Figure 1 the blue distribution is for the case when the accessed data is loaded from memory, hence slow access, and the red distribution is for the case where data is loaded from cache, hence fast access. In other word, based on the location of data the amount of time it takes to load the data varies. The attack scenario works as following. Two process will share a cache. Attacker monitors one line of the shared cache. Based on the time it takes for the attacker

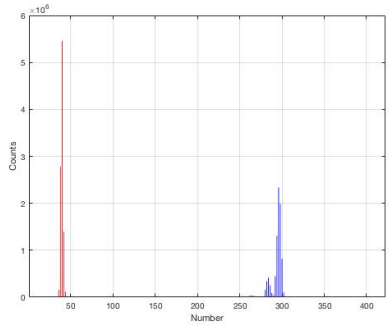

Figure 1: Time ot takes to access a data when it is loaded from Cache vs. when it is loaded from Memory

to access that line of the cache, it can decide whether the victim has accessed that line or not.

Most of the work in this area has been focused on the applications related to cryptography [2, 3, **?**]. In this work, we extend this line of work to non-cryptographic applications, namely database.

1 BTree

We attacked BTree which is a self-balancing tree data structure that keeps data sorted and allows searches. The specific implementation we targeted has tree structure and also all the node in leaves are connected via a pointer. In this work we are interested in the range queries which returns all the value between a and b. The search for correct entries is done by first traversing from the root down to the first value which is equal to a. One at the leaf level, using the linked list, the traverse will continue to find the first element which is greater than b. The intuition for our attack is that by finding the number of traversal happening at the leaf level we know the volume of response. Using prior work [4], a database can be reconstructed purely by knowing the distribution of the volume of range queries and the probability of any particular range query being called and it requires at least $O(N^4 \log N)$ range queries. It means that if we can retrieve the size of enough random range queries, i.e. $O(N^4 \log N)$, we can reconstruct the database. That is, we can know exactly which keys are in the database which is not good if even membership of an entry should be private.

2 Prime and Probe

In order to find the number of accessed entries at the leave level we performed prime and probe attack which introduced by Osvik et al. [2]. The attack works as following,

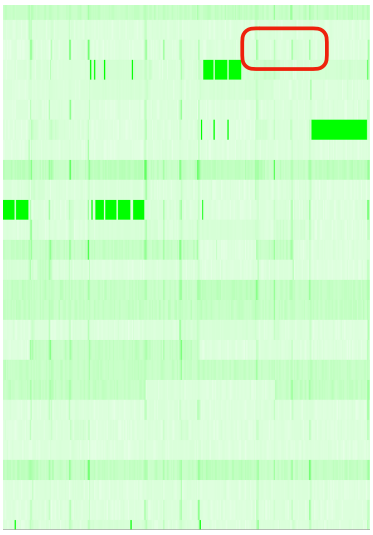

Figure 2: Heatmap of Instruction cache when Specific function is called

1. Attacker **primes** cache by filling it with own data
2. Victim process runs
3. Attacker **probes** cache and records timings
4. If victim process accessed cache sets, the time to probe sets will increase

The attack was done on intel i7-5600U (should be double checked) processor running Ubuntu 16.04. We scheduled attacker and victim on the same core so it means that they share all the level of the cache. We targeted L1 Instruction cache to count number of times a function executed by tracking the cache set it accessed. The function calls tell us exactly how many records are returned in a range query.

Figure 2 shows a snapshot of Heatmap for the instruction cache activities when our target function is being called. Each row represents one cache set (64 for L1 instruction cache) and the column represents time. As it can be seen we could detect 4 function calls in couple of sets. By aggregating over all the sets we were able to reduce the noise. The result is presented in Figure 3, where each group of 3 peaks show one range queries.

3 Future Work

As for the future work we will target SQLite as a real example of database and we are going to extend our attack to that case.

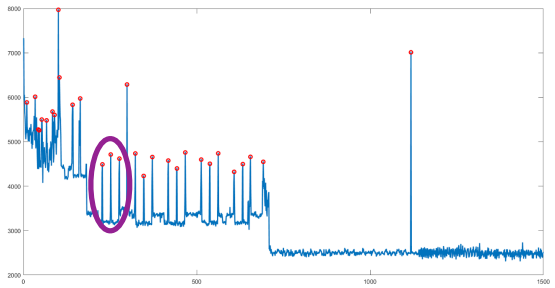

Figure 3: Reduced Noise Measurement

References

[1] Colin Percival. Cache missing for fun and profit.

[2] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of aes. In *Cryptographers Track at the RSA Conference*, pages 1–20. Springer, 2006.

[3] Yuval Yarom and Katrina Falkner. Flush+reload: A high resolution, low noise, l3 cache side-channel attack. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 719–732, San Diego, CA, 2014. USENIX Association.

[4] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1329–1340, New York, NY, USA, 2016. ACM.