CSE 291: Securing Processors Assignment 1

Brandon Saldanha (A59011109)

I. Cache Configuration

Cache Level	Line Size	Total Size	Associativity	Number of sets	Raw latency
L1	64B	32KB * 2	8	64	4
L2	64B	256KB	8	512	12
L2	64B	8MB	16	8192	30

II. Lines in Cache

Cache Level	Lines
L2	512*8 = 4096
L2	8192*16 = 131072

III. Cache access tihe threshold

Cache Level	Lines
L2_L3	50
L3_DRAM	200

IV. Communication Protocol

The attack is carried out by a `receiver` process that snoops the cache waiting for the `sender` process to make a cache access. This cache access can be discerned by the `receiver` using a side channel maneuver.

The first step in the attack is to make sure that the processor frequency is constant as this help get us consistent measurements for our timing attack. Once way to do this is to set a constant frequency for the process by disabling DVFS. As this is not a valid solution, we instead run a set of instructions that have high compute intensity and make sure that the processor is running in the performance mode [1].

The main idea used for the attack is the Prime+Probe attack described in [2]. Based on experiments performed on the cache access patterns that revealed the time required to access data in different cache levels, we conclude that differentiating between L1 and L2 cache access times and an L2 cache access is difficult as there is quite some overlap in the timing of the two. To mount a successful cache attack in this scenario, we use the L2 cache as the target for our Prime+Probe. We run two processes on the same processor core via Simultaneous Multithreading [3]. This ensures that both the processes utilize the same L2 cache which creates contention between the processes for the cache.

The Prime+Probe attack on the entire cache causes quite a bit of noise which makes the communication between processes unreliable. To combat this, we identify addresses that map to specific sets in both the `sender` and the `receiver` processes. This is done by allocating a buffer that has the same size as the L2 cache and then identifying addresses that map to specific sets. We know that each line in all levels of the cache are 64 bytes wide which implies the 6 LSB bits in the cache are used to indicate block offset. The L2 cache has a total of 512 sets which are indexed by the 9 LSB bits obtained after ignoring the

6 bits used for block offset. We used this to create our eviction set stored in a local buffer in both the `receiver` and `sender` processes. From our experiments we find that the L2 cache is virtually indexed because of which we can ignore the virtual to physical address translation.

To mount a successful attack, we prime 8 pre-selected sets in the L2 cache with addresses decided in the previous step in the `receiver` process. We then use the `sender` to access required sets based on the input in the L2 cache. Post this, we probe the L2 cache in the `receiver` to find the timing difference in specific sets which then reveal to us the encoded pattern sent by the `sender`. The `sender` uses binary encoding viz. using a set to represent a bit in the secret to be transmitted. If the bit corresponding to the set is ”1” the set is accessed by the data in the sender which leads to a miss by the probe in the `receiver`. In the case where the bit is ”0”, we do not access the set which shows up as a hit on the `receiver's` end.

Using the timing difference between a hit and a miss, we can decode the secret and then display it on the terminal.

References

[1] Intel Corp., Rafael J. Wysocki, https://www.kernel.org/doc/html/v4.19/admin-guide/pm/intel_pstate.html, 2015

[2] D. A. Osvik, A. Shamir, and E. Tromer, “Cache attacks and countermeasures: the case of AES,” http://www.cs.tau.ac.il/ tromer/papers/cache.pdf, Nov 2005.

[3] Converting Thread-Level Parallelism Into Instruction-Level Parallelism via Simultaneous Multithreading, Jack L. Lo, Joel S. Emer, Henry M. Levy, Rebecca L. Stamm, Dean M. Tullsen, S. J. Eggers, ACM Transactions on Computer Systems, August 1997, pp. 322-354.