Last-Level Cache Side-Channel Attacks are Practical

(Kalind Karia)
(`213076001`)

I. Summary

In this paper, the authors present an effective implementation of Prime + Probe side-channel attack on LLC. They have demonstrated the attack on cross-core, cross-VM for GnuPG tool to retrieve the secret key and measured the capacity of the covert channel created. Their technique relies only on cache inclusiveness and large-page mappings used by VMM. The attack can be mounted on sliced-cache based modern architectures.

II. Details

The work presented in the paper adapts the PRIME + PROBE technique for practical LLC attacks by exploiting hardware features such as:

1) inclusive caches (outside control of the cloud provider)
2) large page mappings (controllable and usually enabled in VMM for better performance)

Only other assumption made is that the attacker and victim are co-hosted on the same processor.

Major contributions of the work:

1) asynchronous PRIME + PROBE attack on LLC that does not require sharing of cores or memory between attacker and victim and does not exploit VMM weaknesses.
2) develops two techniques for efficient attack:
    - algorithm for attacker to probe exactly one cache set without the knowledge of virtual-address mapping
    - use temporal access patterns to identify victim's security-critical accesses.
3) achieves the measurable bandwidth of cross-VM covert timing channel as high as 1.2 Mb/s.

Their LLC-based cross-core, cross-VM attack is based on Prime + Probe [**?**] where the attacker learns which cache set is accessed by the victim VM.

Attacker, `A`, runs a spy process to monitor cache usage of victim, `V` as follows:

1) Prime: `A` fills selected cache sets with its own code/data
2) Idle: `A` waits for a pre-configured time while `V` executes and utilizes the cache
3) Probe: `A` again accesses the selected cache sets and measures the time to load each set

If `A` finds that the access latency is more during Probe, then `V` would have evicted some of `A`'s lines during Idle. Conversely, if the access latency is less then `V` has not used or evicted the selected cache sets of `A`.

Challenges for efficient Prime + Probe attack on LLC and how were they overcome are listed below:

1) Visibility of victim's activity at LLC
    - LLC has less visibility into victim's memory activity than L1 since L1 and L2 will satisfy most of the victim's memory accesses.
    - If manipulation to LLC by attacker does not change the state of L1, L2 (private to victim VM), then victim's accesses will never reach LLC and will be hidden from the attacker.
    - It is overcomed by leveraging cache inclusiveness that helps replacing victim's data from complete cache hierarchy without accessing victim's private caches.
2) Infeasibility of priming and probing whole LLC
    - Since LLC is very large in size, it is not feasible to access every set.
    - Hence very few cache sets relevant to victim's accesses are determined and only these sets are monitored during prime and probe.
3) Identify cache sets relevant to victim's accesses without probing the whole LLC
    - Attacker has no idea of the victim's virtual address space and has no control on its mapping.
    - Solution is to scan entire LLC monitoring one cache set at a time, look for temporal access patterns that are relevant to victim's accesses. These patterns are specific to the algorithm used.
    - An eviction set is constructed which is then primed and probed.

The attack is demonstrated by extracting key from (a) secret-dependent execution paths (Square-and-Multiply modular exponentiation) and (b) secret-dependent data access patterns (Sliding-window modular exponentiation) on the GnuPG tool. The core idea of the attack is to monitor the use of squaring operations. It is shown that the exponent can be recovered by observing the time between subsequent squarings.

III. Strengths

- Prime + Probe attack is an efficient asynchronous LLC-based cross-core, cross-VM attack i.e. attacker and victim only have LLC in common that is shared among all the cores of a multi-core processor.

- This attack removes the requirement of shared memory and libraries between attacker and victim process upon which the Flush + Reload attack primarily depends.
- This attack is possible even in modern Intel processors which have sliced cache.
- Much higher bandwidth of up to 1.2 Mb/s is achieved.

IV. Weakness

- Attack depends on the inclusiveness of caches. If the caches are not inclusive, then priming and probing of eviction set will have no effect on victim's local caches and the victim's access will be hidden from the attacker.
- Large page mappings in VMM helps in construction of eviction set without the knowledge of virtual addresses relevant to victim's accesses. Assumption is made that VMM uses large frames to map guest physical memory.
- If the victim executes a fixed-window (or constant-time) exponentiation algorithm, then the notion of monitoring time between subsequent accesses fails and it is impossible to mount the attack.

V. Extensions

- The attack is not tested in a noisy environment or in a real cloud settings. One can experiment the attack in such situations to measure the effects of noise.
- One can experiment the attack's working without the dependence on large pages (but with reduced efficiency).