No dynamic voltage, no smp, set affinity, rdtsc measurement.

rdt

הדברים שמודדים הם מאוד מהירים.

Side channel attack protections:

CAT, TSX.

Cache attacks can leak RSA keys and counter ASLR.

Usually through shared CPU data and instruction cache.\

Spatial granularity

hyperthreads, partitioning

Flush and reload attack: Deduplication, flush instruction

caches of different cpus are connected

Evict and reload attack: same but by refilling cache

No flush in ARM?

Statically allocated memory?

Prime and probe?

CPU socket?

!!!!!

We can access the branches to put them in the cache, then we know that the victim visited these branches!!

Must identify which set the instruction goes to.

RSA – can check access to the key scrambling tables. Trying to find the possible key by checking the addresses – the blocks used, if the cache is not fully associative.

Attack: we can fill specific sets before running the hypercalls!

We can graph power usage and then find which algorithms are being used.

KVM?

Timing can let you find which websites were visited because of the JavaScript used.

You can still AES keys across apps.

Android – shared core libraries.

Shared libraries in general.

Inclusivity

Lockdown

Trusted execution. Can attack LLC. Advantage – no application scheduling problems in TSX. Even the OS can be malicious. The OS can stop the victim.

Idea: if the hypercall is deterministic – we fill up LLC, then run the hypercall. See what location it used, then remember those – by accessing all of our locations.

The end goal is getting branches. If we know what the binary is, we know which memory accesses happen right after branches. We might be able to tell if a branch was taken or not this way.

Algorithm:

1. Fill the LLC with known blocks that won't be used by hypercall.
2. Look for every basic block that begins with a memory access. (This step can be skipped)
3. Run the hypercall. Access LLC in a loop and record which memory blocks are slower. Find which tags/sets are occupied. (Depends on eviction policy!)
4. If the cache uses an LRU or Pseudu-LRU eviction policy, can find how many sets of each tag got filled by the hypercall by evicting them slowly. This will help for when the two branch targets share a tag.

Code that's spatially close is likely in the same tag in different sets – that coul

Countermeasures:

1. Developers discern secret information from nonsecret. Give the secrets a different set. Can identify variables related to the secret through analysis like taint. You measure the cache timing traces. There are no automated ways to do this.
2. Putting both targets of a branch in the same instruction block.
3. We can load memory, in large chunk, before making if-elses. Likely there are redundant reads.
4. Divide LLC into “colors” like how registers are protected.
5. CAT: Hypervisor can mark sets as unevictable. But only 4 cache sections are protected. Usually combined with coloring.
6. Behavior detection: HPCs can track muarch events. Can find LLC attacks. Can find hackers repeatedly flushing the same addresses.

CAT requires a pretty new hardware. Doesn’t work with memory deduplication.

CAT and page coloring won’t work on TEE as it doesn’t trust the OS.

24/4

Do clobber to memory.

RDTSCD

יש משהו בשם ISOLCPU בGRUB כדי שחלק מהליבות לא ירוץ עליהן כלום.

אחרי זה Task set כדי להריץ את הפרוסס על הליבה המתאימה. או ע״י schedsetaffinity תשים את עצמך על הקור שעשית לו בידוד.

IRQaffinity לבדוק שאין פסיקות בליבה.

elixir.bootlin.com/linux/latest/source/arch/x86/include/asm

Try rdtscp for memory. Try memory fencing.