Bulck 2018

Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution

Meltdown in the Enclave

- Intel’s secure enclave, SGX promises an isolated memory space
- Can run code that is trusted, prevent indirect access of protected space
- But it uses the processor core and cache hierarchy
- Get it to run a job so it has data in cache
- Beforehand, set up array of 256 slots at 4K intervals

Complex Setup

- Revoke all permissions on enclave
- Step through array to re-establish array slot TLB entries
- Run out-of-order instructions to do an indirect access to the array using a cached value as the pointer
- Provide exception handler for page fault that does a timing scan of cache

Optimizations

- Use page aliasing ro reduce effect of mprotect call
- Hide exception handling in a transaction to suppress faults
- Flush cache before enclave execution to make room
- Repeatedly load the indirect location to keep it in cache
- Pin victim enclave to a core to reduce interference

Preemptive Strike

- Force enclave to exit early to ensure data stays in cache
- Use advanced interrupts to single step the enclave
- SGX stores it registers in a frame of a fixed SSA stack
- Revoke execute permission to force a page fault, which causes a SSA frame to refill, bringing it into the cache
- Read out the register values

Root Adversary Read of SGX

- Evict the a page from the enclave page cache then reload it
- Reload copies the page in L1 and doesn’t evict it
- Enables a root process to read enclave content without executing it

More Attacks

- Can steal Launch Enclave keys
- Can steal keys from Quoting Enclave

Discussion

Yan 17

Secure Hierarchy-Aware Cache Replacement Policy

Spy Process

- Knows addresses of interest
- Flushes them (or evicts through conflict)
- Observes when they are reloaded
- If addresses are dependent on data, then the data values can be recovered
- Note that this is not like Meltdown or Specter

cflush and Inclusion Victims

- Evicts from entire cache hierarchy including copies on other cores
- Used e.g., to get output into DRAM for DMA access
- Can flush pages shared with a victim process(e.g., a shared library)
- Inclusive caches keep copies at all levels below highest residence
- If a lower level has an eviction, then copies above need to be evicted

SHARP replacement

- Prioritize eviction of non-private lines
- If none, look for a line private to only one process
- If none, increment an alarm count and do a random evict
- Interrupt if count exceeds a threshold

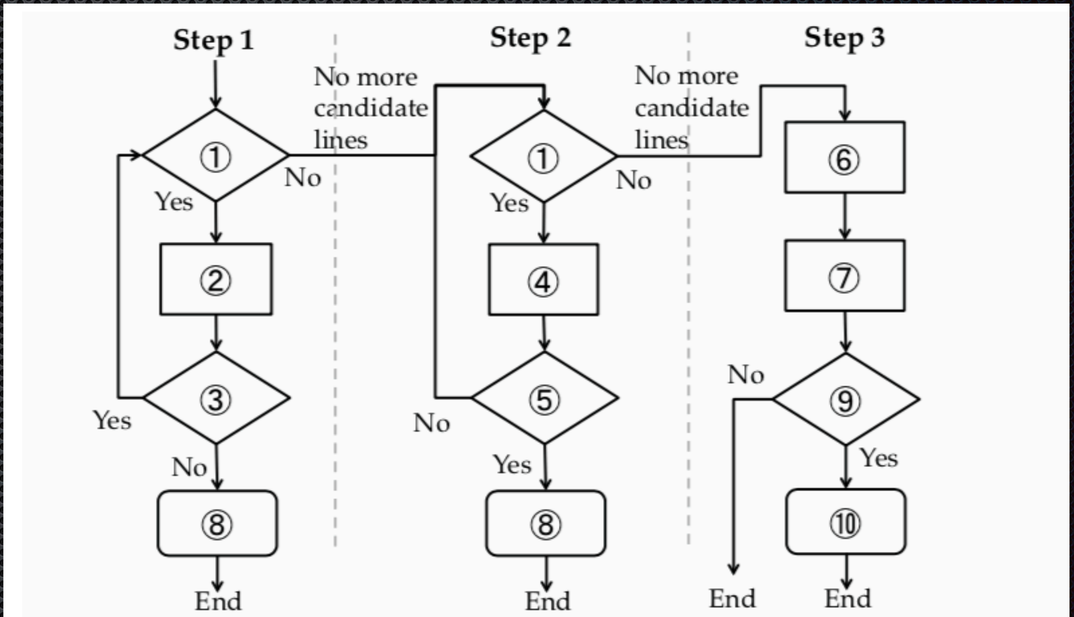

Figure 2: SHARP replacement algorithm.

Core Valid Bits

- Exist for directory based shared memory management
- May not be up to date — can give false positives for private lines
- Can add a query mechanism to ask cores to update the CVB for a line
- But that won’t scale with core count, so just query for the first N lines in a set

Modify cflush

- Shouldn’t have to flush read-only or executable pages
- Change cflush to only work for writeable pages
- Avoids flush on shared libraries, and cause an exception (need OS change)
- If a page is marked copy-on-write, then subsequent spy flushes will be to its own copy

Discussion