Meltdown and Spectre

New exploits of inherent weaknesses in modern CPUs

presented by Raphael Finkel

Keeping Current, January 10, 2018

History

- Discovered by two teams in 2017: Google Project Zero (Jann Horn) and a group in Maryland and Graz (Paul Kocher, Moritz Lipp, others)
- Announced January 4, 2018.
- The original papers are at `https://meltdownattack.com/`.
- One fundamental architectural flaw, with different exploit paths.
- US CERT alert TA18-004A provides "guidance", but no solution.
- The National Security Agency (NSA) most likely already knew about the underlying flaw, although White House cybersecurity coordinator Rob Joyce says otherwise.

Page tables in Linux

- The top part of the address space of each process contains the kernel.
  - i386: the top 1GB out of 4GB total
  - x86_64: top 128TB out of 256TB total
  - That part is marked as readable only in kernel mode.
  - That part typically also maps all of physical memory (especially on the Intel x86_64, where the address space is 2^{48} bytes)
- Reason: context switch to/from the kernel can avoid switching page tables.
  - Therefore the cached entries from the page table (in the Translation Lookaside Buffer, TLB) need not be flushed.
    - Flushing is slow and repopulating the TLB is expensive.
  - So context switch in and out of the kernel is fast.
- Access to memory that is not readable (but is mapped by the page tables) causes a trap.

Speculative execution of transient instructions

- Modern CPUs execute instructions out of order.
  - They can therefore keep more CPU components busy.
  - They predict the likely direction a conditional branch will go.
- If an instruction causes a trap or if the branch prediction is wrong
  - Any instruction following the failure that has been speculatively evaluated is not committed; its effect, if any, is reverted.
  - Although the uncommitted results are available in the CPU, they are not visible at the architectural level (registers, memory)

Avoiding a trap on invalid memory reference

- The attacks generate traps. They must avoid the trap causing termination.
- Meltdown: in Linux and Windows, processes can arrange for memory-access traps to be handled within the process instead of terminating the process.
  - This technique still requires expensive context switch in and out of the kernel, reducing the bandwidth of the attack.
- Meltdown: Some architectures have a concept of atomic transaction with the TSX instruction
  - Failure within a transaction is rolled back without a trap.
  - This technique makes for a very fast attack.
- Spectre: Place the invalid access instruction on the branch that is expected (and therefore speculatively executed) but not taken.
  - Many CPUs use recent history to predict branches.
  - The attacker can arrange for the history to predict wrongly by poisoning the Branch Target Buffer (BTB), which depends only on the low 20 bits of the branch target address.

Cache lines

- Why
  - Memory is slow.
  - Caches store the contents of recently accessed memory in fast internal memory.
- What
  - Data, instructions, and even page tables are cached.
  - Caches are built in multiple levels, with the last-level cache (LLC) shared by all CPUs in the computer.
- How
  - The address of the memory determines which cache cell (“line”) is used.
  - Addresses separated by the length of a cache line use different cache lines.
  - The CPU looks in caches simultaneously with starting a memory reference.
  - If the cache access succeeds (a *cache hit*), the memory reference is abandoned and the access is very fast.

Cache side-channel attacks

- A cache hit makes access faster.
- By measuring access times, a program can determine whether a cache hit occurred.
- There are many cache side-channel attacks that expose cache hits.
- Meltdown and Spectre use the *Flush+Reload* attack, exposing the LLC.
  - Yarom, Y., and Falkner, K. “Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack.” In *USENIX Security Symposium* (2014).
  - Flush a targeted memory location from the cache with `clflush`.
    - If `clflush` is not available, the attacker uses a variant: *Evict+Reload*.
  - Measure the time needed to reload the data.
  - If the time is unusually short, some other process has reloaded the data meanwhile.

Hardware bug

Abandoned transient instructions modify the cache

- A memory-read instruction might fail because of permission violation.
  - The virtual address is mapped in the page table.
  - The page table entry is marked as “kernel access only”.
- The permission check is performed after the access has completed, and the result is already in the cache.
- A few other instructions following the invalid access might also be executed speculatively.
- All these results are discarded, but the cache update remains.

Meltdown

The attack speculatively executes the following code. It starts with an invalid access, so the results are discarded.

```
    ; rcx holds a kernel address
    ; rbx points to a probe array (256*4KB)
    ;   no part of the probe array is cached at first
retry:
    mov al, byte [rcx]  ; generates a permissions trap
                       ; rax gets the byte at that address
    shl rax, 12          ; rax *= 4K (page size)
    jz retry             ; a zero could be failure
    mov rbx, qword [rbx + rax] ; read the probe array
                ; the access modifies the cache
```

Meanwhile, the attack iterates over all 256 pages of the probe array and measures the access time for the first cache line on each page. If no cache hit, attack assumes the value 0.

Effectiveness of Meltdown

- The Flush+Reload attack is the bottleneck.
- It is more efficient to transmit only 1 bit instead of 8 bits.
- Recent versions of Linux randomize where they map physical memory into kernel virtual space (KASLR: kernel address space layout randomization), but only 128 tests are needed to find the place.
- Meltdown can read kernel memory in Windows 10, which also maps a large amount of physical memory (but not linearly).
- Meltdown works from inside containers such as Docker.
- About 500KB/s, error rate 0.02%, using TSX on an Intel Core i7-6700K.
- About 120KB/s, error rate 0.03%, using exception handling (Linux and Windows)
- Meltdown was not successful on ARM or AMD CPUs, even though a toy example does work.

Software fixes to Meltdown

- Operating system level: Use separate page tables for user and kernel modes.
  - The KAISER patch (Kernel Address Isolation to have Side-channels Efficiently Removed) for the x86_64 does this; it is not yet standard in Linux, but Ubuntu as well as RedHat and others have released new kernels with KAISER in place.
    - Still, several privileged memory locations must be mapped into user space. They do not contain privileged information.
  - Effect: up to 30% slowdown (but typically 5%), because context switch is slower
- Browsers: To prevent JavaScript implementation of Meltdown, reduce the precision of the clock
  - Firefox 57.0.4: reduced the precision of `performance.now()` from 5μs to 20μs and disabled `SharedArrayBuffer`.
  - Similar fixes are also in IE, Edge, Chrome 64.

Hardware fixes to Meltdown

- Disable out-of-order execution.
  - Not a viable solution; the performance effect is severe.
- Serialize the permission check and the register fetch.
  - Most likely not a viable solution; the performance effect is large.
- “Hard split” of user and kernel space, with a new CPU control register.
  - If hard split is enabled, the kernel must reside in the upper half of the address space, the user in the lower half.
  - A memory fetch can immediately determine a violation (user mode trying to access upper half) without accessing the data.
  - Almost no performance impact.
  - Backwards compatible (old operating systems would not enable hard-split mode).

Return-oriented programming (ROP): an attack method

We discuss this general attack in CS270.

- Identify byte sequences within an existing program that consist of one or more instructions followed by the `ret` instruction.
- Such a segment is called a *gadget*.
- If the attacker can put arbitrary data on the stack (due to a buffer-overflow vulnerability), it can cause the victim to execute a sequence of gadgets.
- Address-space layout randomization (ASLR) makes ROP harder.

Spectre

(speculative execution side-channel attack)

- Setup
  - Mis-train the CPU, poisoning the BTB, so CPU will make an erroneous prediction.
  - Prepare the side channel by flushing the relevant parts of the cache.
- Induce the victim to speculatively perform operations it would not normally perform.
  - Read from an attacker-chosen address
  - Perform a memory operation that modifies the cache state to expose the value.
- Recover the exposed data by timing read operations.
  - Method 1: if the memory operation is to a place the attacker can read
  - Method 2: call the code again with an in-bounds value
- This attack is very difficult to accomplish
  - It requires intimate knowledge of the victim's code to find usable gadgets.
  - It must exploit a buffer overflow vulnerability to invoke those gadgets.
  - Poisoning the BTB works differently on different processors.

Software fixes to Spectre

Spectre only divulges data within the user process.

- Browser: Display each website with a separate process (Chrome strict site isolation).

What an ordinary user should do

- Don’t panic.
- Keep software up to date (ameliorating the risk of Spectre, which depends on an existing software vulnerability)
- Avoid putting secrets (like financial passwords) in software on shared computers.