datto

The Meltdown attack

A clever attack against an obscure flaw.

Fred Mora - System Engineering, Datto

Agenda

- What is Meltdown?
- How does it work?
- Has it been exploited yet?
- Mitigation

Ooooh, look, a bug with a logo, so we know it's for real!

Disclaimer

- That stuff is complicated
- This is not a bloody academic class
- I tried to simplify things and make them accessible
- I hope I didn't completely blow it with oversimplification
- But keep in mind that there is more to it, so refer to primary sources if you ~~feel frisky~~ want to understand the whole problem.

What is Meltdown?

All right, what did you geeks break now?

- Not much.
- It's just a flaw affecting all Intel processors
- And maybe some other architectures as well
- And you cannot fix it because it depends on the processor hardware

- Actually, to this day, Intel does not seem to have a plan for a fix.
- AMD is immune, though.

What is Meltdown? (cont'd)

OMG. Another Intel bug?

- Yeah, getting old.
- Some insiders blame Intel management for cutting the Architecture Validation dept
- These guys were doing architecture review
- When you cut QA, you get more bugs...
- The reason is that the validation was slowing things down.
- And was costly, too.
- To their credit, this is not an obvious flaw.

How does Meltdown work?

What did they do this time?

- First, we need to know about *out-of-order execution* and *speculative execution*.
- The CPU has several internal computation units.
- If a program keeps them all busy, it will run much faster.
- Some program statements depend on each other, other are independent.

- Example:
  R1, R2, etc. are registers (a.k.a internal memory)
  R1 = R2 * 5
  R3 = R4 + 2
  R5 = R1 +1
- The CPU can simultaneously calculate the values of R1 and R3, in any order.
- But it has to wait until R1 is computed to calculate R5

How does Meltdown work? (cont'd)

OK, what else?

- We also need to know about *speculative execution*.
- When the program hits a test (e.g., an IF statement), it cannot decide which path to follow until the tested value can be calculated
- This can leave internal computing units idle.
- So why not take some educated guesses and try to optimize this?

- We look at the most likely branch of the test and we execute it on a copy of the registers.
- We keep the changes in this auxiliary register banks until we can decide
- Then either:
  - we commit: we validate the calculations and transfer the aux register values to the actual registers
  - Or we retire: we scrape these calculations.
- By keeping executing instructions in parallel, the CPU increases throughput.

How does Meltdown work? (cont'd)

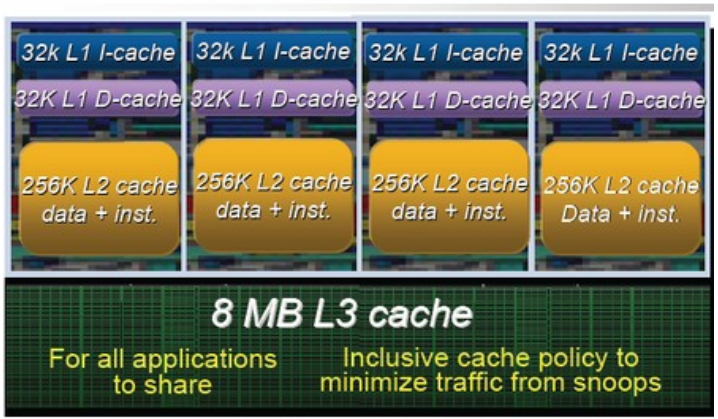
Go on...

- Now let's talk about *memory protection*.
- The CPU keeps track of memory by pages (e.g. 4 Kbytes on Linux).
- Pages belong to the user space or the kernel space
- User processes cannot access kernel pages or pages belonging to different processes.

How does Meltdown work? (cont'd)

Are we done with the boring exposition? What is this, Charles Dickens?

- Almost! Let's also discuss *caching*.
- The CPU clock runs much faster than RAM access.
- Also, it's faster to access consecutive addresses in RAM.
- So the CPU keeps recently accessed RAM in its local cache
- There is actually 3 levels of cache on a modern CPU

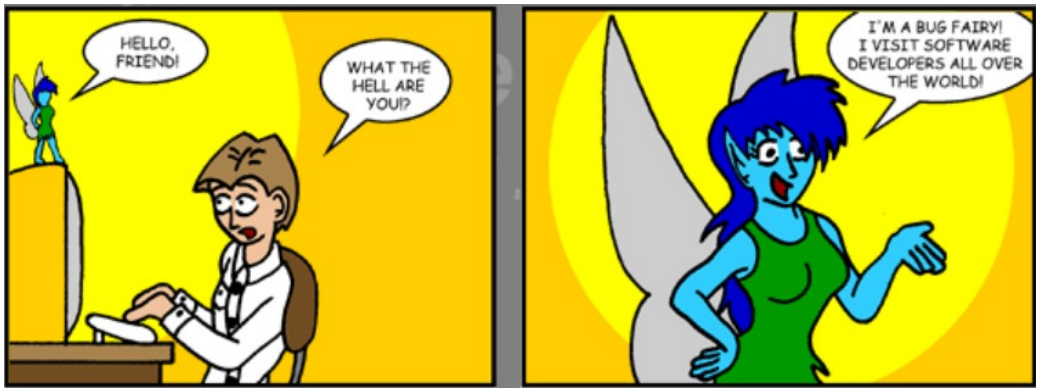


Intel i7 core cache organization
Credit: Tom's Hardware

How does Meltdown work? (cont'd)

Sounds reasonable so far. What's the catch?

- The catch is that the Good Idea Fairy visited the Intel CPU designers.
- Note that the Good Idea Fairy has an evil twin sister called the Bug Fairy, and it's hard to tell which one visits you.



From Andy Weir's old online comic *Casey and Andy*

How does Meltdown work? (cont'd)

Er, what was the good idea?

- Memory protection is a royal pain to implement, see.
- And slow, too.
- So why not disable it during speculative execution?
- After all, if a memory access turns out to be forbidden, the instructions will never be committed, and the result will remain unseen.

How does Meltdown work? (cont'd)

I have a bad feeling about this...

- Let's combine the ingredients we reviewed into a little recipe.
- Say I have a memory location that my program cannot read, called PROTECTED
- And an address that I can read, called READABLE, that I haven't touched yet.
- I write the content of address A as [A].
- My program contains this:

```
if (bit 0 of [PROTECTED] == 1) {
    [READABLE] = 0
}
```

- With a few well-crafted loops, I coerce the CPU into speculatively executing the contents of the if branch.
- In case the lowest bit of PROTECTED is indeed 1, the READABLE address will be accessed
- Which means that it will be loaded in the CPU cache, even though the branch will always be discarded.
- Then, I read that same location again and measure the timing. Is it fast or slow?
- If it's fast, it means READABLE is in the cache…
- Which means I now have one bit of PROTECTED.
- Whooo… I know a secret!

How does Meltdown work? (cont'd)

All of that for one bit? Big deal.

- I can extend this recipe and recover several bits at once
- And then I can iterate through the memory
- It's tedious, but computers are very good at tedious things.
- Ultimately, from a user process, I can read the memory of the whole computer.
- As a demo, one of the original researchers showed how to dump the passwords stored in his browser process!

Has Meltdown been exploited?

Eeep!

- Yeah, big deal indeed.
- But the flaw has not been exploited yet
- Exploit requires executing unknown code on your machine
- Your editor and email client won't start trying to dump your memory suddenly
- But if you are running customer VMs…
- You are at the mercy of the first bad apple.
- A variant of this timing attacks have also been implemented in javascript
- Firefox and Chrome (among others) have incorporated protection against these attacks
- Make sure you use the latest browser versions.

Mitigation (cont'd)

What if I find that the KPTI patches slow down my games?

- The patches come with a kernel boot option called nopti to disable the patches
- If your favorite programs are slowing down, and you aren't running random code all the time, you might disable KPTI at boot.
- If you are running VMs with customer-provided or unknown content, keep KPTI active.
- Patch the kernel and reboot.

Mitigation

So VM hosts are at risk?

- Among others
- If you run unknown programs in VMs, you are a textbook target
- You need to update the kernel ASAP.
- The mitigation patches work by flushing the cache in cases that can be exploited
- There is no more timing difference because all access cases are slower.
- Slow down to be determined.
- The kernel fixes started their life as KAISER
- Linus asked for something more neutral
- Acronyms UASS and FUCKWIT were briefly considered
- The final name for the patches is KPTI (Kernel Page Table Isolation).

Questions?