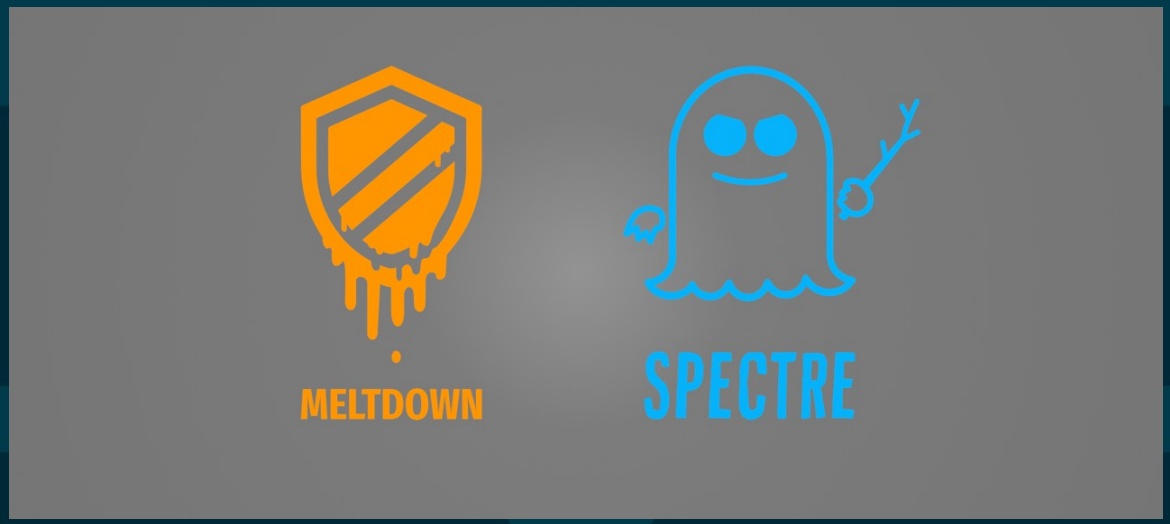
A whole new class of security vulnerabilities:



Greg Scott – gscott@redhat.com

Agenda

- Give credit where credit is due
- Bug hunting – why is this important?
- Ingredients
- The exploits
- The bad news – there are no simple fixes
- What the industry is doing about it
- What we can do about it

Give credit where credit is due

- Collaboration at its best - see https://meltdownattack.com/

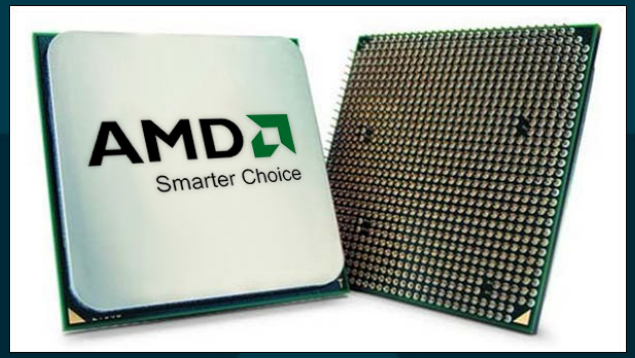
- Three teams independently discovered and reported Meltdown:
- Jann Horn (Google Project Zero),
- Werner Haas, Thomas Prescher (Cyberus Technology),
- Daniel Gruss, Moritz Lipp, Stefan Mangard, Michael Schwarz (Graz University of Technology)

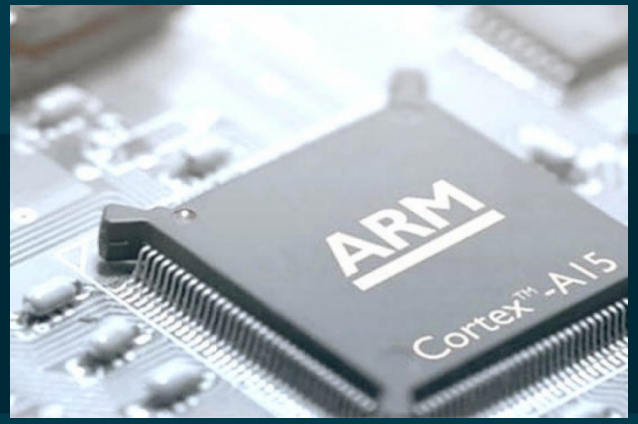
- Two people independently discovered and reported Spectre:
- Jann Horn (Google Project Zero) and Paul Kocher in collaboration with, in alphabetical order, Daniel Genkin (University of Pennsylvania and University of Maryland), Mike Hamburg (Rambus), Moritz Lipp (Graz University of Technology), and Yuval Yarom (University of Adelaide and Data61)

More credit

- First reported to Intel and other chip makers June 1, 2017
- That led to a mad scramble behind the scenes to address it.
- Went public Jan. 3, 2018, one week earlier than planned, after an article appeared in The Register.
- And that led to another mad scramble to get the updates out.
- See this article from Wired Magazine (Andy Greenburg, Jan. 7, 2018) for a great writeup on how researchers pieced it together: https://www.wired.com/story/meltdown-spectre-bug-collision-intel-chip-flaw-discovery/

Why is this important and why should we care?





Because every modern computer chip has the problem.

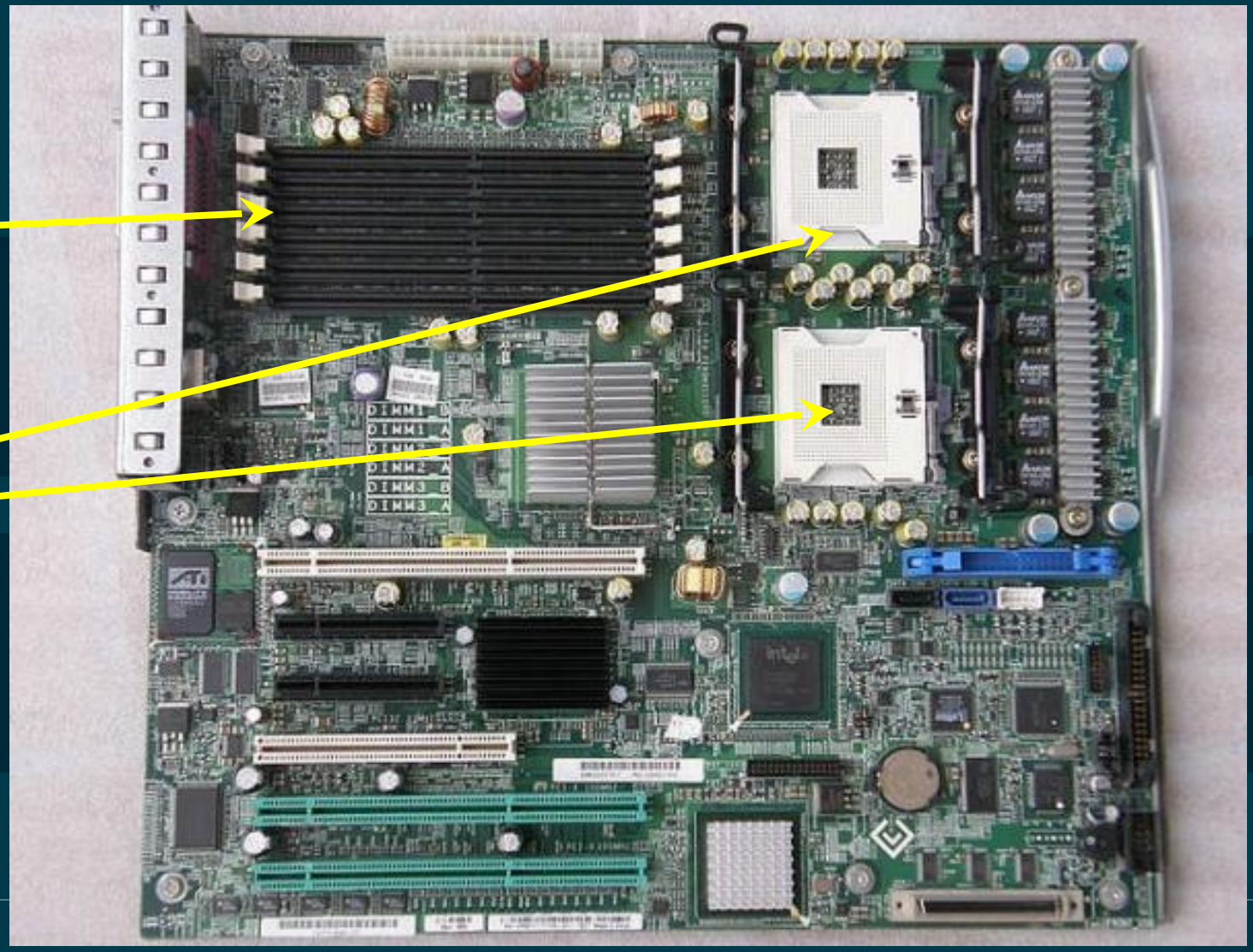
Which means every modern computer device is vulnerable

redhat.

Ingredients – cache

RAM sockets

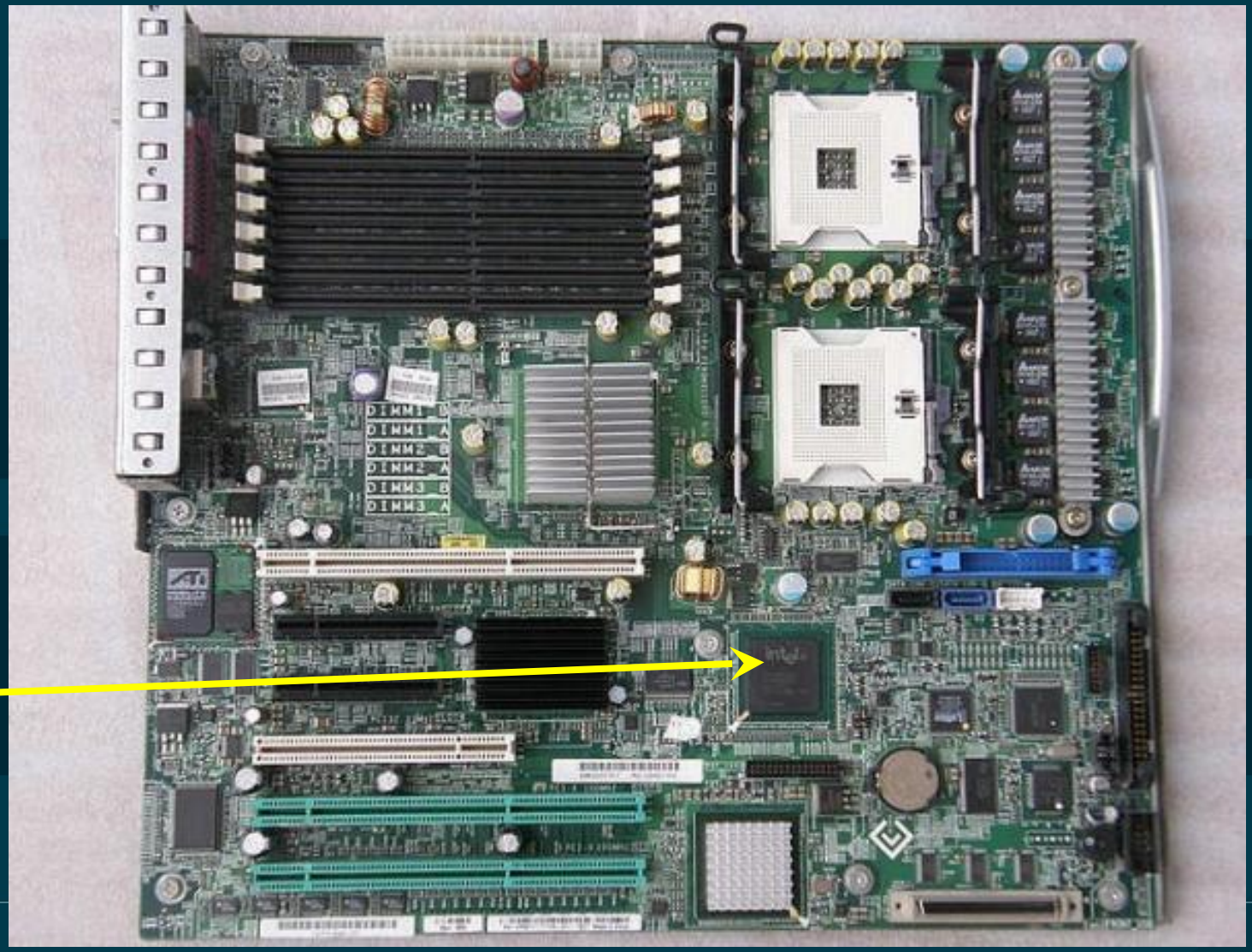
CPU sockets



Why is cache a big deal?

- 1 sec = 1000 ms
- 1 ms = 1000 usec (10^6 usec in a second.)
- 1 usec = 1000 ns (10^9 ns in a second.)
- A L1 cache reference takes around 0.5 ns. An L2 reference is about 7 ns. Let's just average it out to, say, 3 ns.
- A main memory reference takes around 100 ns.
- Pretend my 3 ns cache round trip is one minute; this means my main memory round trip takes about ½ hour.

Ingredients – firmware and microcode



Ingredients- concurrency

Ingredients – prefetching and pipelining

Ingredients – speculative execution

Putting it together

- ***Isolation*** is a bedrock computer security concept. It means no process should be able to look inside another process or the kernel without following strict interface rules.
- But speculative execution doesn't follow the rules.
- Speculatively execute a sequence of machine instructions to access memory you're not supposed to touch.
- Once the speculation proves to be wrong, the microcode is supposed to restore state back the way it was.
- And it does… except for the cache.
- A little bit of clever, non-privileged code breaks isolation and destroys civilization.
- And this bug has been in nearly all computer chips since around 1995.

The exploits we know about as of early 2018

- Spectre variant 1 known as Bounds Check Bypass, CVE-2017-5753
- Spectre variant 2, known as Branch Target Injection, CVE-2017-5715
- Meltdown – variant 3, Rogue Data Cache Load, CVE-2017-5754

See https://meltdownattack.com/

Meltdown – variant 3, Rogue Data Cache Load, CVE-2017-5754

- Every user thinks they own the whole machine.
- The OS depends on hardware to enforce permissions.
- I want to read an address in kernel space I'm not supposed to see.
- The system executes the instruction ahead of time so it's ready when my program gets to it. ***Regardless of whether I have permission or not.***
- If this turns out to be an illegal address, my program takes an exception and the ~~hardware~~ microcode restores its state.
  - All except the cache.
- I flush the cache, and now main memory and the cache agree.
- Easiest to exploit, easiest to fix

Meltdown mitigation

- Don’t depend on hardware to enforce memory page permissions; do it with software in the kernel.
- Separate kernel and user page tables; take a context switch when looking at kernel pages.
- Take a 5 to 30 percent performance hit when accessing kernel pages.
- This was in the first wave of patches from January, 2018.

Spectre variant 1, Bounds Check Bypass, CVE-2017-5753

- Write a program to call a function in the kernel that looks like this:

```
if (x < array1_size)
    y = array2[array1[x] * 256];
```

- Pick an out-of-bounds value for x, call the function, and it will return without running the second line.
- But the microcode ***will*** speculatively execute that second line and leave a legacy of it in the cache.
- Hard to exploit because I need to find a value for x that points to the secret I want.

Spectre variant 2, Branch Target Injection, CVE-2017-5715

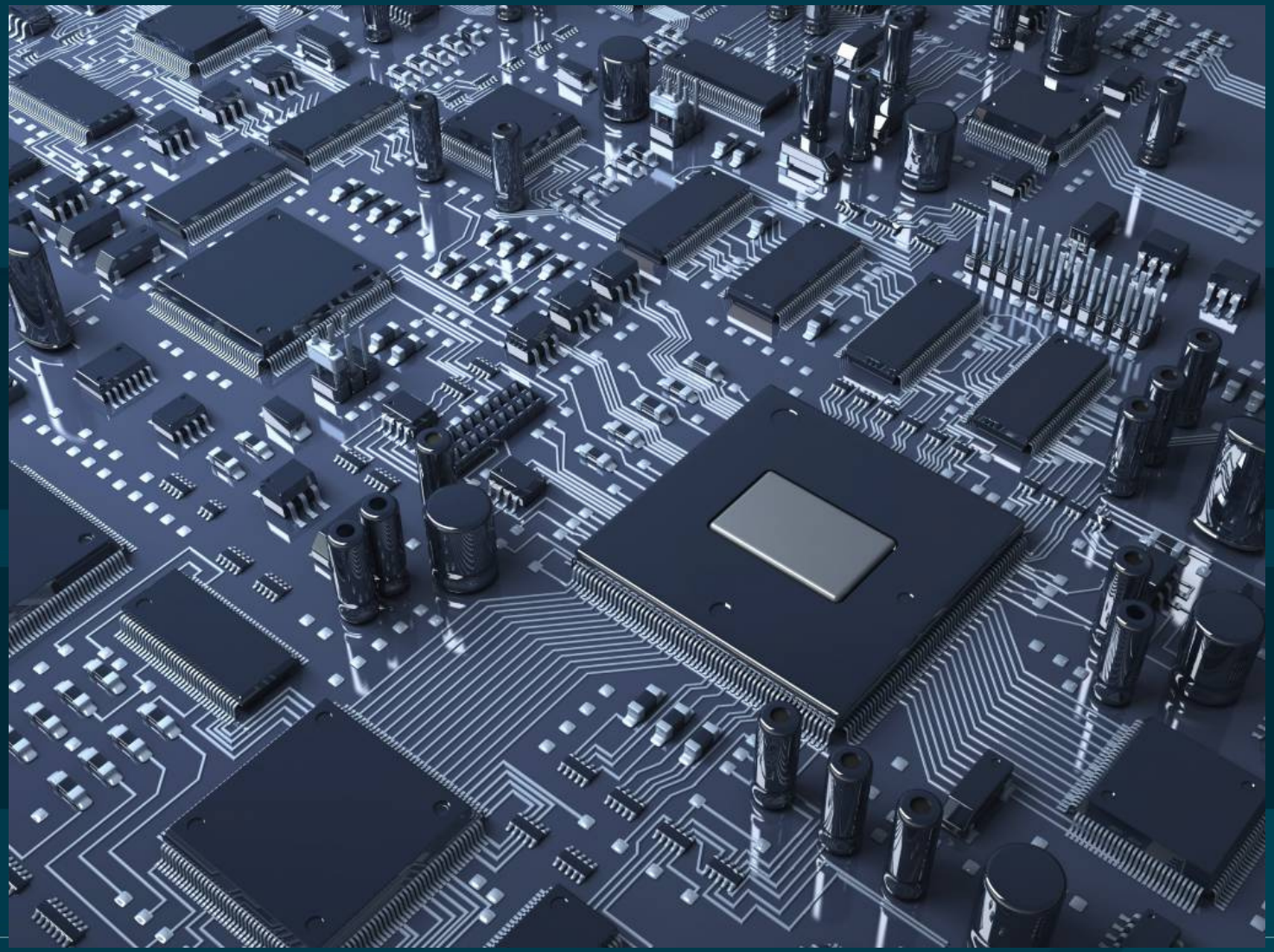
- Indirect branches – look at the contents of a location and jump to the address in the contents, not the location itself.
- Example - return from a subroutine (ret); pops a value from the stack and jumps to that location.
- The attack poisons an indirect branch, executes the indirect branch speculatively, leaving its legacy in the cache.
- Even more tricky to do because it needs to be tailored for individual systems.
- No known attacks exist in the real world

Spectre mitigation

- Much of this needs to be in microcode.
- Retpoline (return trampoline) – change a ret instruction to a series of instructions to pop values off the stack and populate the program counter.
- Compilers also need an update.
- This is major kernel and compiler surgery.

The bad news – there are no simple fixes

- Every mitigation so far has been a workaround.
- And they all come with a performance tradeoff.
- For now - either cripple some of the chip optimizations or accept the security risk.



What the industry is doing about it

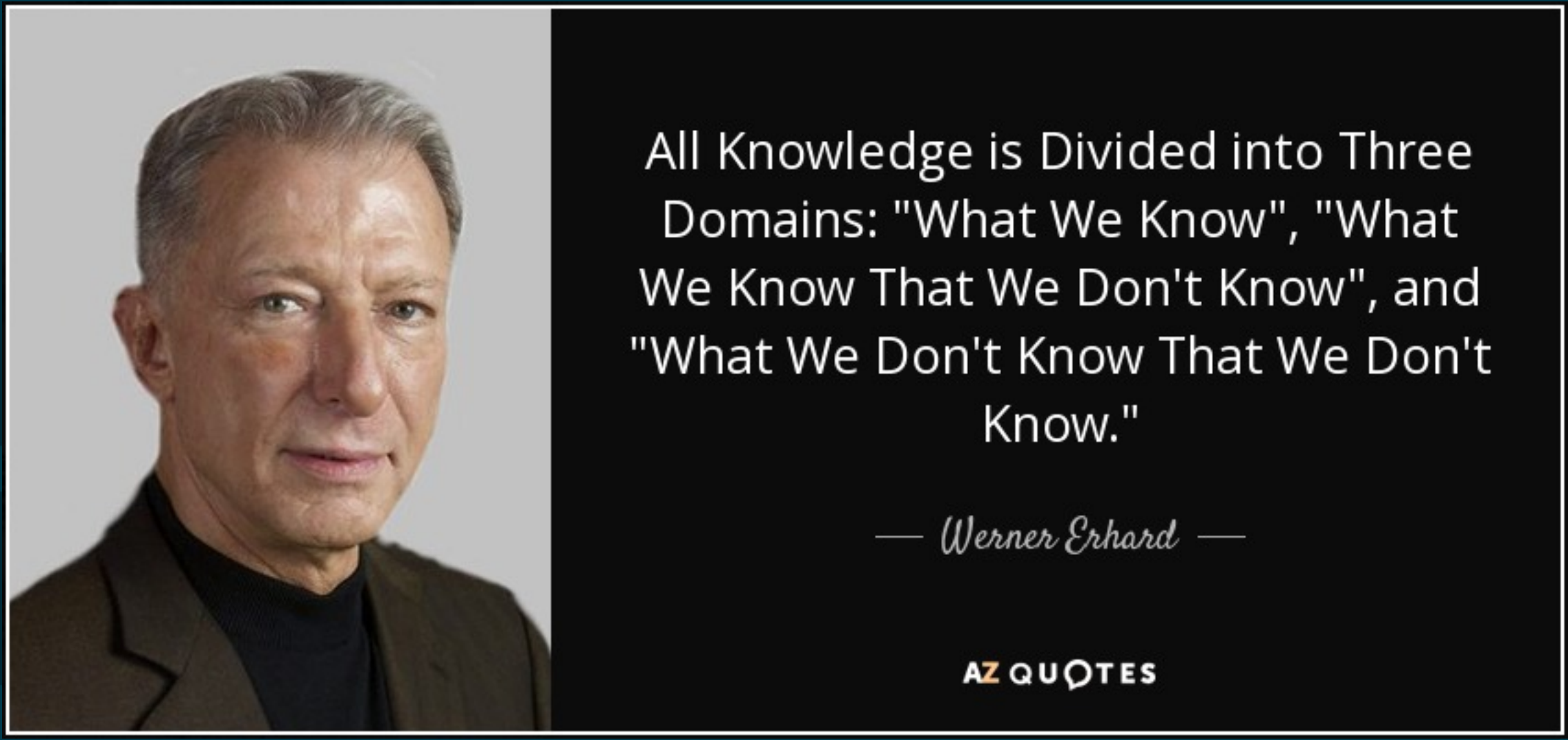
- Intel tried to rush a microcode update in early January, 2018. That didn’t work out so well.
- Lots of kernel developers and chip architects continue to burn lots of midnight oil developing workarounds.
- This is not an Apple vs. Microsoft, or Android vs. Apple, or Linux vs. Windows, or VMware vs. Hyper-V vs. RHV fight.
- It’s an industry-wide problem and we’re all in this together – chip, system, software, and service vendors, security researchers, and end user customers.

The most important thing we can do

More we can do

- If it connects to the Internet, make sure it has a provision for updates. And a commitment from the vendor to provide them for a long time.

- It will take years to cycle through current hardware generations and fix this in silicone. Expect more workarounds and difficult patching tradeoffs.


All Knowledge is Divided into Three Domains: "What We Know", "What We Know That We Don't Know", and "What We Don't Know That We Don't Know."
— Werner Erhard —
AZ QUOTES

Red Hat notes as of March 21, 2018

- Red Hat labeled these vulnerabilities as important, not critical.
- Retpoline RHEL 7 kernels delivered March 6.
- Retpoline RHEL 6 kernels delivered March 13.
- RHEL5.11 and 5.9z retpoline packages are built and undergoing QA now. Target availability date is early April.

If you do this...

Sooner or later, you'll end up like this

For more information, see:

- https://meltdownattack.com/ - has an FAQ and links to the original academic papers.
- https://www.youtube.com/watch?v=zuBw1HFJMsM – Stanford University, EE380: Computer Systems Colloquium Seminar Exploiting modern microarchitectures: Meltdown, Spectre, and other hardware attacks Speaker: Jon Masters, Redhat.
- https://www.youtube.com/watch?v=2kCDPCgjlJ4&t=3s – Jon Masters' at Fosdem 2018, Exploiting modern microarchitectures Meltdown, Spectre, and other hardware attacks
- https://access.redhat.com/security/vulnerabilities/speculativeexecution - Red Hat article with links to several specific articles.