Google: Software is never going to be able to fix Spectre-type bugs

reader comments

125 with 79 posters participating

Share this story

Researchers from Google investigating the scope and impact of the Spectre attack have published a paper asserting that Spectre-like vulnerabilities are likely to be a continued feature of processors and, further, that software-based techniques for protecting against them will impose a high performance cost. And whatever the cost, the researchers continue, the software will be inadequate—some Spectre flaws don't appear to have any effective software-based defense. As such, Spectre is going to be a continued feature of the computing landscape, with no straightforward resolution.

The discovery and development of the Meltdown and Spectre attacks was undoubtedly the big security story of 2018. First revealed last January, new variants and related discoveries were made throughout the rest of the year. Both attacks rely on discrepancies between the theoretical architectural behavior of a processor—the documented behavior that programmers depend on and write their programs against—and the real behavior of implementations.

Specifically, modern processors all perform speculative execution; they make assumptions about, for example, a value

being read from memory or whether an if condition is true or false, and they allow their execution to run ahead based on these assumptions. If the assumptions are correct, the speculated results are kept; if it isn't, the speculated results are discarded and the processor redoes the calculation. Speculative execution is not an architectural feature of the processor; it's a feature of implementations, and so it's supposed to be entirely invisible to running programs. When the processor discards the bad speculation, it should be as if the speculation never even happened.

Footsteps left behind

What the Meltdown and Spectre researchers found is that the speculative execution it isn't entirely invisible and that, when the processor discards the speculated results, some evidence of the bad speculation is left behind. For example, speculation can change the data held in the processor's cache. Programs can detect these changes by measuring the time to read values from memory.

With careful construction, an attacker can make the processor speculate based on some value of interest and use the cache changes to disclose what that speculated value actually was. This becomes particularly threatening in applications such as Web browsers: a malicious JavaScript can use data revealed in this way to learn about the memory layout of the process it's running in, then use this information to leverage other security flaws to execute arbitrary code. Browser developers have assumed that they can construct safe sandboxes within the browser process, such that scripts can't learn about the memory layout of their containing process. Architecturally, those assumptions are sound. But reality has Spectre, and it blows those assumptions out of the water.

The Meltdown attack, faced by chips from Intel, Apple, and other manufacturers building certain standard ARM designs, was

a particularly nasty variant of this. It allowed a malicious program to extract data from the operating system kernel. In the immediate aftermath of the discovery of Meltdown, changes were made to operating systems to hide most of their data from such malicious programs. Intel has made specific changes to its processors to address Meltdown, so its most recent processors no longer need to activate these operating-system changes.

An apt name

But Spectre—which is best thought of as a particular *style* of attack, with many different variants and iterations—has proven more insidious. A variety of software techniques has been devised to either prevent the processor from executing sensitive code speculatively or limit the information that can be disclosed through speculative execution.

Google's research found that these software measures leave a lot to be desired. Some measures, such as blocking all speculation after loading values from memory, protect against many attacks but are far too debilitating to use in practice. The researchers were experimenting with modified versions of the V8 JavaScript engine from Chrome, and indiscriminate use of this technique made performance drop to between one third and one fifth of what it was without mitigation. Other mitigations were less punitive—for example, protecting array accesses from a certain kind of disclosure had a 10 percent performance cost.

But in every case there were trade-offs; no mitigation protected against all Spectre variants, so a mix of techniques has to be used, and for techniques that can't be used indiscriminately, there's a big challenge in even identifying where mitigations should be applied. Moreover, Google devised a general-purpose Spectre-family attack that could not be defeated with any of the known mitigation techniques.

An important element of Spectre attacks is a timing system to measure those cache changes. One of the ideas that people have had to counter Spectre is to make the clocks available to applications less accurate. The working theory is that, if you need to measure cache differences that are a few nanoseconds in length, a clock that has a resolution of, say, milliseconds will be too coarse. The researchers devised a technique for amplifying small timing differences, and this amplification can defeat any attempt to make the timers coarser.

No end in sight

As such, the company concluded that we just can't depend on software fixes to guard against Spectre. Hardware mitigation might be possible, but this is presently an open question—unlike Meltdown, which had a clear resolution, Spectre seems to be far more intrinsic to speculative execution. And ditching speculative execution isn't much of an option either; it's a feature of every high-performance processor, and with good reason—it provides a substantial performance advantage.

For now, then, applications that try to construct secure environments will have to rely on the guarantees that *are* made by hardware—the protection between processes. For example, Chrome has been changed to not allow content from multiple domains to run within the same process. This still doesn't protect the Chrome sandbox itself from attack by scripts, but it does mean that one script can't attack content from other domains.

All in all, the research shows that Spectre was aptly named. It's going to haunt both software and hardware developers for years to come, and there's no clear end in sight.

https://arstechnica.com/?p=1462619