Understanding Spectre and Meltdown

By Nick Kostalas

August 2020

Dangerous new security flaws

In January 2018, the public learned of two important new security flaws affecting modern computers. Named Spectre and Meltdown, these flaws are hard to fix, and they expose nearly every computer on the planet to attacks that can result in a severe data breach. This article will help you understand how these vulnerabilities work, what makes them so novel and dangerous, and what's being done to mitigate their impact.

Flaws in the physical device

Spectre and Meltdown are hardware flaws. They exploit features built into the silicon chip at the heart of a computer: the central processor, or CPU. The processor takes coded instructions from programs located in the computer's memory, executes the code, and outputs the results. Processors operate much faster than computer memory can send them instructions, so chip designers build in optimization techniques to make efficient use of their extra speed. One of these optimization techniques is called *speculative execution*, and it's what makes Spectre and Meltdown possible.

Speculative execution defined

When a processor using speculative execution receives a batch of instructions from a program, it looks for ways to execute the code faster. One method is to guess which way a conditional line of code (like an IF–THEN statement) will branch, and then calculate the result ahead of time. This is called *branch prediction*. Another method is to reorder lines of code so that instructions that are ready to execute are prioritized over instructions that need to wait for further information. This method is called *out-of-order execution*. Branch prediction and out-of-order execution are each a kind of speculative execution, and both methods have proven so efficient that they're found on practically every CPU manufactured since around 1995.

Speculative execution breaks the rules

The flaw in processor design that Spectre and Meltdown take advantage of is this: in order to maximize speed gains, speculative executions are allowed to break the rules. In normal, non-speculative performance, when a processor tries to execute a line of code that uses an illegal variable or requests access to restricted data, the execution fails—it's canceled before it can cause an error or access secret data. But during speculative execution, the processor is allowed to calculate a result using an illegal variable, and will succeed in temporarily accessing restricted data.

Hackers use the cache as a back door

The processor doesn't completely abandon all security measures: after a speculative execution produces its results, but before those results are ever output from the CPU, the processor checks to see if it broke any rules. If it did, it discards the results of any illegal calculations. This prevents any restricted data from being sent directly to the user or the program. But there's a big hole in this plan: when a speculative execution accesses restricted data, it stores that data in the cache. When the processor discards the results of an illegal operation, the data is wiped from the CPU—but it remains in the cache.

Indirect attacks reveal hidden data

Users and programs can't take restricted data directly from the cache, but an attacker can use indirect methods to learn what's inside. One such method is called a *timing attack*. A timing attack exploits the fact that data stored inside the cache can be retrieved measurably faster than data stored in main memory (where all data is kept by default). By asking the computer to retrieve different pieces of data in sequence, and taking note of which data arrives in much less time than the rest, an attacker can easily see which of those pieces of data came from the cache. If the attacker has caused a piece of restricted data, such as a password, to be stored in the cache, they can recover it quickly and accurately with a timing attack.

Basic structure of an attack

Putting it all together, an attack using Spectre or Meltdown might look like this:

1. The attacker uses speculative execution to force the CPU to access restricted data.
2. The CPU discards the data, withholding it from the user—but leaving a trace in the cache.
3. The attacker then uses indirect methods to recover the data from the cache.

Spectre is harder to fix

Compared to Meltdown, Spectre attacks are more difficult to carry out, and expose a smaller portion of the computer's memory. Spectre can access all restricted data within the target program's memory, which is a big problem, but the problem stops there—Spectre can't steal secrets from other programs in the same system. However, researchers have found more than a dozen different ways to put together a Spectre attack, each requiring a different security defense. This makes it particularly hard to fix.

Meltdown exposes more data

By contrast, Meltdown attacks are easier to carry out, and they expose a much greater portion of the computer's memory. Meltdown allows the attacker to ignore a foundational principle of computer security called *address space isolation*, which is used to designate which parts of memory a given user or program has permission to access. By breaking this security feature, Meltdown gives the attacker access to all data within the kernel—the part of a computer with full access to, and full control over, all data in the entire system.

A hard problem to solve

Taken together, Spectre and Meltdown are a serious and persistent threat to modern data security. The feature they exploit—speculative execution—is found in almost every modern computing device, and is considered to be indispensable. Hardware fixes are not yet feasible, while software fixes are incomplete and tend to significantly reduce processing speed. It will take years to eliminate the Spectre and Meltdown vulnerabilities from newly-manufactured computer chips, and devices with the flaws already built in might never be completely protected against them.

First attempts to defend against the attacks

In late 2017, before the public learned of the existence of Spectre and Meltdown, chip manufacturers and software companies began releasing defensive patches. The patches have succeeded in mitigating some of the impact of these vulnerabilities, but they don't represent a complete solution. Researchers have proposed more comprehensive fixes, some of which have been implemented, but many of which are still in development more than two years after the discovery of the flaws.

Current proposals to address the flaws

Some of the proposed fixes aim to prevent attackers from loading restricted data into the cache; others make it more difficult for attackers to recover that data from the cache. For example, most operating systems now use a technique that greatly reduces the amount of kernel space accessible to an unprivileged user. This is an important safety measure, but it comes at a significant cost to processor speed, and it helps only against Meltdown attacks. Many browsers have implemented changes that reduce the user's ability to finely measure processor response times, making it more difficult to probe the cache with a timing attack. Other proposals include compartmentalizing the cache in such a way that limits the ability of programs to access data that doesn't belong to them.

An ongoing issue

Spectre and Meltdown shocked the computer security world when they were revealed in 2018. The stark truth is that some of the most important advances in processor speed in the last quarter century have come at the expense of some of the most foundational concepts of data security—and because it took us decades to figure this out, we find ourselves in a highly compromised position today. A robust solution to the pervasive vulnerability caused by Spectre and Meltdown might be possible, but as of today, we aren't close to finding it.