Rambus

Security by Design – Meltdown and Spectre

When Jann Horn of Google's Project Zero published a detailed blog post titled "Reading privileged memory with a side-channel", it set off a firestorm of activity as the post confirmed that secret information inside of a computer could be accessed via two different attacks, Meltdown and Spectre. Essentially, both attacks utilize CPU data cache timing to efficiently exploit and leak information out of the system. This could lead to – at worst – arbitrary virtual memory read vulnerabilities across local security boundaries in various contexts. The discovery and disclosure of these attacks was done by a number of security experts, including senior Rambus technology advisor Paul Kocher and senior Rambus security engineer Mike Hamburg.

While companies are furiously working to find solutions to prevent these attacks from being exploited, they demonstrate the importance of Rambus' philosophy of **Security by Design**; focusing on building security into the bedrock of any design, including siloing secure processing from other applications.

	MELTDOWN	SPECTRE
Architecture	Intel, Apple	Intel, Apple, ARM, AMD
Entry	Must have code execution on the system	Must have code execution on the system
Method	Intel Privilege Escalation + Speculative Execution	Branch prediction + Speculative Execution
Impact	Read kernel memory from user space	Read contents of memory from other users' running programs
Action	Software patching	Software patching (more nuanced)

Daniel Miessler 2018

Meltdown

The Meltdown attack takes advantage of a feature on Intel processors called "out-of-order execution" which is used by modern CPUs to speed up the processing of data and applications. Out-of-order execution basically allows a CPU to not be slowed down by processes that take a long time to execute, like a memory fetch, by allowing other operations that will be needed later to be run while the longer process is taking place. Meltdown exploits this by allowing for an attacker to gain access to secret information stored in the computer and exporting it to the outside world, enabling attackers to steal the entire memory contents of computers. Patches to protect this exploit have been seen to slow down CPUs by up to 30%. It is important to note that this is a hardware based attack and can occur on any operating system.

Spectre

The Spectre attack is similar to Meltdown in that it is also a hardware exploit that takes advantage of a process used to increase the speed of modern CPUs, including Intel, ARM, and AMD. Spectre takes advantage of a

feature called “speculative execution”, where the CPU looks at the sequence of processes it’s been asked to perform and tries to predict what the next series of processes are that need to be performed so it can start the processing ahead of time, especially for processes that take a long time to execute. For example, if for the last 10 times processes A, B and C were performed processes D, E, and F were performed right after, the CPU would start working on processes D, E, and F when it sees processes A, B, and C being requested. Attackers can exploit this by having the CPU start a process that would reveal secret information speculatively, and then having the secret information sent to the outside world. Patches for Spectre have been shown to slow down process execution by up to 20%, but these are much more difficult to patch and may not adequately address the exploit. Again, like Meltdown, this is purely a hardware attack and can happen on any operating system.

Security by Design – Secure Processing

Both attacks exploit a feature that is integral to the way that modern CPUs operate and are architected, affecting everything from cloud servers, mobile devices, PCs, and IoT devices. To use an analogy, Meltdown is like having holes in your roof, it can be patched but that patch may still leak in the future. Spectre, on the other hand, is like having a cracked foundation, you must tear the house down to truly fix it. This demonstrates that to truly thwart these attacks, the CPUs will need to be redesigned from the ground up. As Paul Kocher told the *New York Times*,

“Whereas Meltdown is an urgent crisis, Spectre affects virtually all fast microprocessors. We’ve really screwed up. There’s been this desire from the industry to be as fast as possible and secure at the same time. Spectre shows that you cannot have both.”

Both Meltdown and Spectre could have been mitigated using the Rambus philosophy of Security by Design, which means that security is not an afterthought, but an integral part of the design process of any SoC. A major component of that philosophy is to have the secure compute processes, such as those that involve passwords, to be siloed away from the main processor, effectively having a secure processor as a part of the CPU, via an IP core, or as a standalone chip. This allows for secure compute processes to be protected while still allowing for the speed gains of out-of-order and speculative execution to be realized. If the CPUs had been architected with this siloing methodology, Meltdown and Spectre would have been completely ineffective. To this point, Mike Hamburg of Rambus, and one of the discovers of both exploits, stated,

“However, beyond short-term solutions such as patching, the semiconductor industry should seriously consider designing chips that run sensitive cryptographic functions in a physically separate secure core, siloed away from the CPU. This design approach will go a long way in helping to prevent vulnerabilities that can be exploited by Meltdown and Spectre.”

Embracing a hardware-first strategy and implementing the necessary functionality on the SoC level is a key element of fully securing devices and platforms across multiple verticals. As Meltdown and Spectre illustrate, the importance of adopting a hardware-based siloed approach to secure processing at the most basic core level cannot be overemphasized and must be a fundamental part of any SoC design.