Jenna Berlinberg

CMSI 401

10/6/2020

Unstoppable SpectreRSB

The cyberthreat known as Spectre was first discovered in January 2018, and as of July 2018, three variants had been identified. Spectre exploits a vulnerability in the processor during speculative execution. Speculative execution is included in most every modern processor, and without it, the speed and performance of the processor is significantly decreased. This process allows the pipeline to continue working without stopping despite having no tasks provided by predicting the possible next instruction. If it was an incorrect assumption, the data is discarded and the processor executes the new command. Since the other option is having the pipeline idle, there is no loss involved. If the process predicts correctly, then the pipeline continues with the computation it already began, thus increasing performance.

When designing this process, the designers realized that the processor may temporarily access memory that should be separate from the hardware, but they did not consider it to be a big concern. After speculation, all the data, including the access to that memory, will be dumped anyways, so it does not matter that it is accessed. What they did not consider is that by accessing and tracking the speculatively accessed data, a malevolent actor could find the address of the sensitive memory. This is how Spectre gets access to sensitive information in computing systems.

In response, Google and Intel released patches to help protect against the early versions of the threat. Google's Retpoline adds protection to the regions involved in deciding speculative execution. However, the variant discovered in July 2018, known as SpectreRSB, is invulnerable to Retpoline and the other patches available at that time. This is because SpectreRSB attacks using the return stack buffer- hence the RSB in the name. It adds false addresses or removes addresses in the return stack buffer. A team of 2 computer science doctorate students and 2 computer science professors at UC Riverside wrote a paper on the discovery of this threat and possible solutions. They recommend that all processors vulnerable to this sort of attack use the patch RSB refilling. This patch adds an extra address into the return stack buffer, which should prevent SpectreRSB from successfully attacking it. It is also noted that all Intel processors produced moving forward will have this patch pre-installed. However, older processors, which are still in frequent use, need to incorporate this patch

The SpectreRSB situation reflects a greater issue in terms of balancing attention to details that contain potential vulnerability with the limited resources of time and energy. The fact that the memory was being accessed inappropriately was not completely off the radar of the computer designers, but they either did not consider how an adversary could exploit this, or they thought it was not high enough priority to dedicate significant energy to preventative measures pre-production. The other option would have been to drop speculative execution altogether, but the consequences of that would be a severe detriment to the modern standard of computing power. They thought, "the addresses of the memory would be dumped anyways; it will be as if it never accessed it at all." Unfortunately, someone with a high enough level of skill and patience can exploit the fact that it was accessed at all by following the trail. When viewing the trail from one location became impossible with the implementation of Retpoline, they tailed a lower level

worker in the form of the return stack buffer until they found an even better vantage point. The sophistication required for this attack is high, but the information it can access is imperative. Engineers need to ask themselves if their information is sensitive enough and the threat high enough to dedicate time to filling in as many holes as possible.

In cybersecurity, this is a frequent issue in terms of both hardware and software, particularly to deal with particularly sensitive information. There is a chart commonly used by professionals known as the risk assessment that compares the likelihood of the threat to the possible consequences. A threat such as SpectreRSB that requires a level of skill not easily acquired is a lot less likely to happen, especially to a piece of software that does not retain any particularly important information. Going through all this effort just to receive an email address or two is a waste of the attackers time. They are much more likely to use more common attacks, which are what the engineers working on cybersecurity at the company would want to implement prevention tactics on the most. However, for an entity that contains highly sensitive information with very advanced and determined adversaries, such as military offices or bitcoin banking services, a complicated and devastating threat such as SpectreRSB is likely to be high on their list of threats they should address. As much as we might want to demand high security measures on every single possible hole that the brightest hacker in the world could exploit, attention to such details can lead to an extreme increase in time before deployment and overexert employees. By the time you finish patching up every hole, you have missed deadlines and the list of threats with higher probability has grown.

That being said, attention to detail is exceptionally important when it comes to the hardware that will be contained in all computing everywhere, such as the processors affected by SpectreRSB. Everything from a student's laptop to the machines controlling nuclear plants might

be vulnerable to an attack such as this. The security of such a universal piece of architecture needs to be of the highest priority; at this low of a level, every single vulnerability should be addressed before deployment. These reactive patches should instead be proactive features of the hardware in the first place, which they now are in processors past July 2018. I would not expect this level of detail to cyberthreats on the very basic website I play mahjong on, but I very much would mind if this oversight resulted in massive world consequences.