Will Suitor
CSCI Multicore - Paper

As computing systems are becoming incorporated into contemporary technology, cybersecurity is growing in importance and complexity. It now has to account for the embedded computers in cars or airplanes, things where the crucial calculations need to happen within a certain time frame every time. These systems are no longer simple processors but complicated multicore machines that need to ensure critical computations always have priority while also dealing with numerous other tasks. The unpredictable latency and mandatory performance of these tasks makes protecting them all the more difficult.

Because of the strict requirements of the systems, it does not take much to interrupt them. Recent research at the University of Kansas seems to suggest that some of these computers are particularly susceptible to basic Denial-of-Service attacks. DoS attacks are malicious programs that intend to overwhelm a system or resource and render it unavailable for some period of time, usually through a flood of unnecessary tasks and requests. And, as these systems have become more widespread, DoS attacks can gain entry through some downloaded third party apps.

The current solution is to have the cores and their memory strictly partitioned. The main idea is to isolate the crucial tasks to certain cores and allow them to perform those tasks without interruption. This could work if there was no shared memory. However, it is impractical to have a multicore system without any shared memory, so most share at least one low-level cache and main memory. Still, the partitioning works in some sense. The DoS attacks have no way to directly impact any critical systems but through shared memory. So, DoS attacks want to do what they can to make the critical systems use of shared memory as difficult as possible. To do this, they target a few features of the shared memory architecture.

The caches in these (and most modern systems) systems are non-blocking caches. A non-blocking cache can continue to service read requests even after multiple cache misses. It won't block the cache while waiting on one miss request and simply fill the request when it can. When a cache miss occurs, its information is put into the miss status holding register (MSHR) until it can be serviced. When the MSHR is full, the entire cache becomes locked and does not service any requests until it is no longer full. Similarly, the cache becomes locked if the writeback buffer becomes full. The writeback buffer stores new writes that have not yet been committed to main memory but are visible for reading. This buffer is utilized because it is faster to read from a cache than to read all the way from main memory. So, for DoS attacks, the MSHR and writeback buffer are ideal targets.

These systems also utilize a feature known as prefetching. Because accessing main memory is so much slower than accessing caches, there are many intermediaries that try to lower that latency and require less trips to main memory. Prefetching does not reduce the number of trips necessarily but does reduce the amount of time spent waiting for that memory. Through memory access patterns, prefetching preemptively stores memory from main memory in the cache. Normally this process is quite useful, but it can be problematic if it wastes time bringing memory that is unneeded or even overwrites memory that might be needed later in the program.

A basic thought to protect against DoS attacks lies in the idea that in-order processing is less susceptible to memory conflicts than out-of-order processing. An in-order system executes instructions in sequential or atomic order. When an instruction needs access to something that isn't currently available, an in-order processor will sit and twiddle its thumbs until that resource becomes available. On the other hand, most modern processors use an out-of-order system for instructions. While an in-order processor is waiting, an out-of-order processor will find some instruction it can complete in the meantime. Then, it when it can complete the initial instruction, it will and will reorder all the instructions into the correct, sequential order. While in-order computation is slightly antiquated, if it was not susceptible to memory conflicts, it could serve as a solution for some of these low level embedded systems. It was believed that it would be less susceptible because it appeared that an out-of-order system might violate temporal locality for a well-written program, or at least be requiring movement between memory locations. However, this research has found that to be false and the order of instruction largely unimportant.

Regardless of the order of processing, DoS attacks target these systems through both MSHR and the writeback buffer because when they're full, the whole cache is locked. When the whole cache is locked, crucial programs that needs to be completed within a certain time are forced to wait until both of them aren't full. However, if the malicious program keeps running, everytime it unlocks, it will just lock again once the DoS program makes another request to either write or read. The effect of this is an instruction that should only take a few cycles can end up taking hundreds. Systems with aggressive prefetching are particularly susceptible to DoS attacks. They will see how the DoS program is accessing or writing to memory and try to make it easier by bringing that memory to the cache. However, with DoS attacks, this will create a huge problem as the prefetching will almost constantly be writing and overwriting memory unnecessarily. So, through third party apps, DoS attacks can have significant impacts on the critical functions of these embedded systems. In a car, for instance, these attacks could be so powerful as to cause a crash.

The solution these researchers propose is to be implemented at the operating system. They want to essentially have two counters for each core that tracks how many reads and how many writes it is performing. Each of these counters would have a limit on the rate each core could be performing reads and writes. The threshold for reads would be much higher than that for the write which should allow processor to function without significant performance impact for programs that are not malicious. However, DoS attacks would have to be slowed down to the point where they would be unable to affect critical systems, meaning neither the MSHR nor the writeback buffer would be continuously full.

What is most remarkable about this research is the depth of understanding it requires to fend off DoS attacks versus what it requires to write them. The example malicious program that the researchers were using was a single for loop. To write that attack, all you need to know is the general size of memory on the system, what size the cache line is and you can write a program that would crash a car. To best defend it, you need to fundamentally alter the operating system of the car. You need to know everything about the system and how it works. This kind of research really highlights the relevance and difficulty of cybersecurity in the modern world.

Works Cited

Bechtel, Michael, and Heechul Yun. “Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention.” 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 4 Mar. 2019, doi:10.1109/rtas.2019.00037.