The Spectre Vulnerability

The effect of the Spectre vulnerability is that an attacking program can read data in memory it does not have permission to access. The vulnerability comes from the interaction of two techniques: speculation and caching.

Speculation

Processors frequently make decisions. To speed up processing, processors predict the outcomes of those decisions, then act on those predictions before the decisions are made. If the predictions are wrong, then the incorrect actions are rolled back (i.e. undone) to maintain program correctness.

Caching

Main memory is slow compared to processors, so processors keep frequently accessed data in a small, fast memory called a cache. Caches are divided into blocks of data. The first access to a block brings it from memory into the cache, so the first access is slow. Later accesses are fast because the data is in the cache.

Spectre

The attacker wants to know the value of secret data in memory it doesn't have permission to access. The attacker runs code to intentionally predict incorrectly. Based on the prediction, the attacker's code accesses the secret data, then treats that data as a number to select and access one of many blocks the program does have permission to access. The processor discovers the incorrect prediction and rolls back the incorrectly executed code. However, the access to the selected block caused it to be brought from memory into the cache; this doesn't affect the correctness of the program, it just makes access to one block faster. Then the attacker, who has permission to access all the blocks that could have been selected, accesses each one and measures the time it takes to do so. The block that can be accessed quickly is the one selected by the secret data. Knowing the number of that block tells the attacker the value of the secret data. The reason this works is because the processor does not verify whether the attacker has permission to access the secret data until the incorrect prediction is discovered.

Discussion

Q: Why doesn't the processor prevent the program from reading the secret data speculatively?
A: The processor doesn't verify whether the program has permission to access the data until the prediction is verified. By that point the access has happened. The secret data is then hidden from the program, but the cache has already been accessed. One option would be to verify every access at the time it is made rather than when the prediction is verified, but that would hurt performance.

Q: Why doesn't the processor take the block out of the cache that doesn't belong there?
A: A mechanism for detecting these kinds of accesses would be complex and potentially hurt performance. Often blocks are brought into the cache based on incorrect predictions, but these blocks might have been accessed sooner or later anyway, so the processor leaves them there.