Sahil Gupta

CS-GY 6573

Anthony J Candeias

Overview of how the attack works and functions

Meltdown and spectre attacks are the two biggest hacks discovered yet. They exploit those features which are built into almost every CPU for their better performance and faster execution. They are exploitable by the help of side channel attacks which can dump the kernel memory which is not accessible by the normal user or user program. They can be used to steal confidential and read memory of other processes without any user permissions and privileges.

Meltdown basically took the advantage of the Out-of-Order execution feature built in CPUs to speed the execution and give better performance results. With out of order execution, it reads the secret value or content that is stored in-accessible memory location and then can be dumped using side channel attack. Side channel attack enables the sender instruction to send the secret content via covert channel to the receiver instruction.

Spectre has two variants. Variant 1 exploits Conditional branch miss-prediction and Variant 2 leverages the fact that Indirect branches can be poisoned to access the arbitrary memory from a restricted area. Using these two features, an attacker can access a restricted memory and read the contents into one of the cache lines. Then side channel attack is used to build the covert channel so that an attacker can dump the contents from microarchitecture to system architecture.

Explain how you would detect usage of this exploit throughout the network

The exploits can be detected by monitoring the cache misses or page faults errors. If the cache misses and page faults are too high, we can block the process. This can be explained as all these exploits, weather due to meltdown or spectre, all take advantage of the side channel attack i.e. Flush and reload in most of the cases.

These attack load the secret content value multiplied by cache size into probe array. Probe array is not cache but the secret value is cached somewhere around in between of 256 pages of probe array. The attacker has to check access times for first line of cache for all 256 pages. If the access time is slow this means the bit was ‘1’ and victim’s code was not able to access memory but if it is fast, then the address the saved in cache. These 256 checks will generate too many cache-misses and page faults in short span of time which otherwise is not normally seen in any of the user programs. Mostly, when we observe this kind of behaviour we can block the process right away and stop the attackers code from dumping the whole contents of kernel memory or other user space memory.

Research methodologies that have been used to weaponize the vulnerabilities.

Researchers have proposed various ways to weaponize the vulnerabilities in the lieu of exploiting the meltdown and spectre. Researchers have used various side channel attacks to dump the secret content of kernel from cache. The three main ways of side channel attacks mentioned in the paper are flush + Reload, prime + probe, evict + time. Though The researchers have only provided the proof of concept using flush n reload. .

Some researchers have also tried to weaponise these vulnerabilities using prime + probe side channel attack. They are calling those attacks as MeltdownPrime and SpectrePrime. Where Flush and reload attack just load the secret content value from one side of the covert channel in cache and dump that memory from receiving side of the channel. MeltdownPrime and SpectrePrime uses prime and probe where they leverage the cache invalidation mechanism of CPUs. When the prime and probe attack was used leveraging coherence cache invalidation, it acheived the same accuracy and precision as in original proof-of-concept. These kind of attacks are caused by caused by write requests being sent out speculatively in a system that uses an invalidation-based coherence protocol.

Attack Vector: Both meltdown and spectre are local attacks, that means, which requires attacker to execute malicious code or file on the victim’s computer. This can have achieved by drive-by download attack where the attacker lures in victim to download malicious file on personal computer and then execute the file. Though this cannot be spread through network, some researchers have found out the way to implement the attack through network also. These new variants are called as NetSpectre and NetMeltdown.

Since both of the attacks access kernel memory without any user privileges, Cloud is a big attack surface for these kind of attacks. Single server or device is being shared by many users and program or services, Cloud platforms are very critical to the impact of these kind of attcaks.

Create a remediation plan to provide defensive measures against the threat

Remediation plan for the threats are difficult to deploy in the enterprise network and systems. As these attacks doesn’t exploit any software vulnerability, no software updates or software patches can mitigate the risk of these attacks. So to completely remove these vulnerabilities, there are two options: either purchase new hardware or apply firmware updates.

Big leaders in the processor manufacturers have already released firmware updates which can put a stop at these attacks. As suggested in the research papers of meltdown spectre, the attacks can be thwarted by completely disabling out of order execution feature in the CPU and only allow serialization of the statements. This would be high on the performance of the CPU. So this kind of update is not feasible. Another remediation can be a hard-split of the user space and kernel space so whenever the address belongs to kernel space memory, the execution would be disallowed on compile time. Another best and most efficient approach to remove the threat due to meltdown is use KAISER which user space and kernel space page tables entirely. So, the side channel attacks which are possible even in KASLR, can be thwarted by the KAISER implementation. But this also comes with the performance costs.

Even in the spectre attack the plausible solution for the firmware updates is to prevent speculative execution but that adds very high performance costs to the CPUs. To prevent the branch poisoning the companies have executed the Indirect Branch Restricted Speculation modes and Single Thread Indirect Branch prediction modes. The other solution to prevent indirect branches is retpolines. Retpolines is a code sequence that replaces indirect branches with return instructions. Nothing has been much said about the performance issues of these updates.

But considering the high performance issues with Meltdown patch, we should consider deploying detection and response mechanism for these attacks. For once keeping aside the fact that prevalence of these attacks is very high, its ease of exploitability is very difficult and taking trade-off between performance and risk into consideration, it is recommended to implement incident detection and response mechanism rather than applying the patch until efficient patch gets released.

Provide a comparison between the variants noted in these vulnerabilities

There are three variants of these vulnerabilities: Meltdown, spectre Variant 1 and Spectre Variant 2

Meltdown: Meltdown take advantage of the out of order execution of the instructions. In out of order execution, the series of instructions are executed out of order to give faster results. If subsequent instructions after some instructions are not dependent on the output of the prior instruction, the subsequent instructions get executed first. If the CPU finds or encounters the exception in the prior instruction, all the execution of the subsequent statements gets revert back.

So exploiting the meltdown is a race condition between accessing the data and the interruption of exception. However, the race condition can always be won if the data is close to the core. The another way is exception suppression.

Spectre: Spectre is largely based on speculative execution feature of the CPU. Spectre attacks also cannot achieve privilege escalation as they leak information through speculatively executing the instructions. They cannot be prevented by KAISER. Spectre attacks can only access the memory which the victim process can access. It cannot just access kernel memory like meltdown. So, Spectre is orthogonal to meltdown in this way. The counter measures of the meltdown cannot be applied as the counter measures for the spectre. If you want to prevent both attacks, you have to apply counter measure for both them separately.

Spectre also has following two variants:

Spectre variant 1: The variant 1 exploits the conditional branch miss-prediction. Attacker intentionally mistrain the processor for the condition to be always true. So when attack gives some invalid input, the processor speculatively predicts the condition to be true and execute the true branch instructions. When the CPU actually checks the value and raise an exception, the cache gets already loaded with the secret value.

Spectre Variant 2: this variant takes advantage of Poisoning Indirect Branches. The adversary mistrains the branch predictor with invalid destinations and further the CPU speculatively also allow the execution with the address pointed by an attacker.