MELTDOWN AND SPECTRE: A REVIEW

Meet Parekh[1], Nand Gondha[2], Kashyap Sojitra[3]

[1]BE Student, I.T Dept., V.V.P. Engineering College, Rajkot, GTU
[2]BE Student, I.T Dept., V.V.P. Engineering College, Rajkot, GTU
[2]BE Student, I.T Dept., V.V.P. Engineering College, Rajkot, GTU
Email: [1]parekhmeet97@gmail.com, [2]nand12398@gmail.com, [3]kkashyapsojitra@gmail.com

Abstract: - In modern processors, Spectre and Meltdown exploit critical vulnerabilities. These hardware based vulnerabilities allow programs to steal data which is currently processed on the computer. While programs are typically not permitted to read data from other programs, a malicious program can get access of information stored in the memory of other running programs by exploiting Meltdown and Spectre which include passwords stored in a password manager or browser , emails, personal pictures and also private documents. Meltdown and Spectre work on mobile devices, personal computers and also in the cloud. It might be possible to steal other customers' data depending on the cloud provider's infrastructure. This paper reviews the References [1],[2] and [8] briefly and provides an overall insight for the Spectre and Meltdown attacks , their working and possible solutions.

Keyword: out-of-order execution, speculative execution

1. Introduction

Memory isolation is one of the central security features of today's operating systems. Operating systems ensures that user applications cannot access each other's memories and also prevents them from reading or writing kernel memory. This isolation allows running multiple applications on personal devices or executing processes of multiple users on a single machine in the cloud and so it is a cornerstone of our computing environments. On modern processors, the isolation between the kernel and user processes is generally realized by a supervisor bit of the processor that defines whether a memory page of the kernel can be accessed or not. This hardware feature allows operating systems to map the kernel into the address space of every process and it makes the transitions from the user process to the kernel very efficient , e.g., for interrupt handling. Consequently, in practice there is no change of the memory mapping when switching from a user process to the kernel.

Out-of-order execution is an important performance feature of today's processors for overcoming latencies of busy execution units, e.g., a memory fetch unit needs to wait for data arrival from memory. Instead of stalling the execution, modern processors run operations *out-of-order i.e.,* they look ahead and schedule subsequent operations to idle execution units of the processor. However, such operations often have unwanted side-effects, e.g., timing differences can leak information from both out-of-order and sequential execution. It was observed that out-of-order memory lookups influence the cache, which in turn can be detected through cache side channel. Hence on the micro architectural level, there is an exploitable security problem.

Speculative execution is a technique used by high-speed processors for increasing performance by guessing likely future execution paths and prematurely executing the instructions in them. For example when the program's control flow depends on an uncached value located in the physical memory, it may take several hundred clock cycles before the value becomes known. Rather than wasting these cycles by idling, the processor guesses the direction of control flow, saves a checkpoint of its register state, and proceeds to speculatively execute the program on the guessed path. When the value eventually arrives from memory the processor checks the correctness of its initial guess. If the guess was wrong, the processor discards the incorrect speculative execution by reverting the register state back to the stored checkpoint, resulting in performance comparable to idling. In case the guess was correct, however, speculative execution results are committed, yielding a significant performance gain as useful work was accomplished during the delay. From a security perspective, speculative execution involves executing a program in possibly incorrect ways. However, as processors are designed to revert the results of incorrect speculative execution on their prior state to maintain correctness, these errors were previously assumed not to have any security implications.

The above discussed important processes or features of operating system are being exploited by two security flaws:

a) Meltdown:

It breaks the most fundamental isolation between Operating system and User applications. This attack allows programs to access the memory and information of other programs in the operating system. If a computer system has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information on that system since it will be prone to attacks. This is applicable to both personal and cloud computers. Every Intel processor which implements out-of-order execution is potentially affected, which is every processor since 1995 except Intel Itanium and Intel Atom before 2013. According to ARM [4], some of their processors are also affected.

b) Spectre:

Spectre attacks trick the processor into speculatively executing instructions sequences that should not have executed during correct program execution. As the effects of these instructions on the nominal CPU will be eventually reverted, we call them transient instructions. By carefully choosing which transient instructions are speculatively executed, leaking information from within the victim's memory address space is possible. Spectre affects almost every system: Personal Computers, Cloud Servers as well as Smartphone's. Specifically, all the modern processors capable of keeping many instructions in flight are potentially vulnerable to Spectre. Spectre has been verified to have affected non-Intel processors, including AMD [5] and ARM [4] processor.

Meltdown differs from Spectre attacks in two main ways. First, unlike Spectre, Meltdown does not use branch prediction for achieving speculative execution. Instead, it relies on the observation that when an instruction causes a trap, following instructions that were executed *out-of-order* are aborted. Second, Meltdown exploits a privilege escalation vulnerability executed specific to Intel processors [6], due to which speculatively executed instructions can bypass memory protection.

2. Background

a) Out-of-order execution:

Out-of-order execution is an optimization technique that allows to maximize the utilization of all execution units of a CPU core to its maximum capacity. Instead of processing instructions in sequential manner, the CPU executes them as soon as all required resources become available. So even if the execution unit of the current operation is occupied, other execution units can run ahead. Thus, instructions can be run in parallel as long as their results follow the architectural definition.

b) Speculative Execution:

Often, the processor does not know the future stream of instructions of a program. For example, this occurs when *out-of-order* execution reaches a conditional branch instruction whose direction depends on preceding instructions whose execution has not yet completed. During such cases, the processor can make save a checkpoint containing its current register state, make a prediction as to the path that the program will follow, and speculatively execute instructions along the path. If the prediction turns out correct, the checkpoint is not needed and instructions are retired in the program execution order. Otherwise, when the processor determines that it followed the wrong path, it abandons all pending instructions along the path by reverting to its previous state from the checkpoint and execution resumes along the correct path. Abandoning instructions is performed so that changes made by instructions outside the program's execution path are not made visible to the program. Hence, it maintains the logical state of the program as if execution has followed the correct path.

c) Branch Prediction:

Speculative execution requires the processor to make guesses as to the likely outcome of branch instructions. Better predictions improve performance by increasing the number of speculatively executed operations that can be successfully committed. And for this several processor components are utilized.

3. Attack overview

a) Spectre attack overview:

Spectre attacks induce a victim to speculatively perform operations that would not occur during program execution and which leaks the victim's confidential information via a side channel to the adversary. In most cases there are three phases in which the attack is performed. First phase is the setup phase, in which the adversary performs operations that train the processor in such a way that it will later make an erroneous speculative prediction. During this phase the adversary can also prepare a side channel that will be used for extracting the victim's information, e.g., by performing the flush or evict portion of a flush + reload attack or evict + reload attack. During the second phase, the processor speculatively executes the instructions that transfer confidential information from the victim context into the micro architectural side channel. While speculative execution can potentially expose sensitive data via a broad range of side channels, the examples given cause the speculative execution to read memory value at an attacker chosen address then perform a memory operation that modifies the cache state in a way that exposes the value. For the final phase the sensitive data is recovered. For this type of attacks using flush + reload or evict + reload, the recovery proves consists of timing how long reads take from memory addresses in the cache lines being monitored.

b) Meltdown attack overview:

Meltdown attack consists of 3 steps:

1) First, the content of an attacker-chosen memory location, which is inaccessible to the attacker, is loaded into the register.
2) Second, a transient instruction accesses a cache line based on the secret content of the register.
3) Finally, the attacker uses Flush + Reload to determine the accessed cache line and hence the secret stored at the chosen memory location.

By repeating these steps for different memory locations, an attacker can dump the kernel memory, including the entire physical memory. This type of attack could also be implemented entirely in higher level languages like C.

4. Literature Review

Spectre attacks train a victim's processor to speculatively perform operations that would not occur during the correct program execution and in turn leak the victim's confidential information. Reference [1], discusses in detail Spectre Attacks along with explaining various terminologies that are required for understanding these attacks. The authors also published their own results for exploiting speculative execution and the techniques that were used by them. Implementation for exploiting the speculative execution flaw in languages like C and JavaScript are also discussed with the code snippets by the authors.

The paper also discusses different contexts in which Spectre can work along with variations in which Spectre attack can be combined with different known

attacks to exploit the vulnerability. The paper is concluded by exploring few mitigation options and future work that is required to prevent attacker from using the vulnerability. Meltdown is distinct from the Spectre Attacks in several ways notably that Spectre requires tailoring to the victim process's software environment, but applies more broadly to CPUs and is not mitigated by KAISER. Meltdown is "probably one of the worst CPU bugs ever found", said Daniel Gruss, one of the author of Reference [2]. Reference [2] is an in-depth view of the Meltdown attack. The paper explains Meltdown attack with examples along with explaining the terminologies and a building blocks view of the attack. A descriptive working of Meltdown attack is provided in the paper in the form of series of steps. Topics like Information that can be leaked by Meltdown, performance after applying countermeasures and limitations for AMD [5] and ARM [4] processors are also discussed by the authors in the later part of the paper. The paper concludes by discussing KAISER, a mitigation option that the authors insist on implementing in every system vulnerable to Meltdown. The Researchers at Google have discovered that CPU data cache timing can be abused to efficiently leak information out of mis-speculated execution, leading to (at worst) arbitrary virtual memory read vulnerabilities across local security boundaries in various contexts. Reference [3] article by Google's Project zero team discusses the issue of reading privileged memory with a side-channel in 3 variants and that too before the Spectre and Meltdown issues were made public. The first and second variant describes Spectre attack while the third variant describes Meltdown. The authors discuss each variant along with code to provide in-depth knowledge of these attacks and they also described the PoCs(proof of concepts) developed by them in the article.

Meltdown and Spectre have wreaked digital havoc and left a critical mass of confusion in their wake. Not only are they terrifically complex, the fixes that do exist, exists in form of patches. Reference [8] article shows how the fixes that are available do solve the issues in some contexts but cannot be consider as a permanent fix that prevent attackers from exploiting the hardware vulnerabilities

5. Conclusion

Meltdown is a novel software-based side-channel attack exploiting out-of-order execution on modern processors to read arbitrary kernel and physical memory locations from an unprivileged user space program. Without requiring any software vulnerability and independent of the operating system, Meltdown enables an adversary to read sensitive data of other processes or virtual machines in the cloud with up to 503 KB/s, affecting millions of devices.

A fundamental security assumption underpinning all of the software isolation techniques is that the CPU will faithfully execute software, including its safety checks. Speculative execution unfortunately violates this assumption in such a way that it allows adversaries to violate the confidentiality of memory and register contents. As a result a broad range of software isolation approaches are impacted.

These vulnerabilities are considered to be critical. Most operating systems in use are affected, including Linux, MacOS, and Windows. Android is also affected, as are Apple's iOS and tvOS (though not watchOS). The real solution, as reference [7] mentions, is to replace the system's CPU with one that does not show the vulnerability. As that is not practically possible, a software solution has been deployed or it is in the works. The problem is that these fixes seem to have a performance impact, some appreciating it up to 30%. Intel declared that performance degradations are "workload-dependent," and "for the average computer user, should not be significant and will be mitigated over time."

A solution that is feasible for implementation in every system should be found because it will be difficult to protect confidential information from unforeseen attack techniques that are focused on these Hardware flaws. Software solutions can only be applied for the known type of attacks but when different attack types are combined for exploiting the hardware flaws, it is not practical to develop specific software patches every time a new attack is recognized.

6. References

[1] Website/blog:-
https://spectreattack.com/spectre.pdf

[2] Website/blog:-
https://meltdownattack.com/meltdown.pdf

[3] Jann Horn, Project Zero "Reading privileged memory with a side-channel"

[4] Website/blog:-
https://developer.arm.com/support/security-update

[5] Website/blog:-
https://www.amd.com/en/corporate/speculative-execution

[6] Website/blog:-
https://newsroom.intel.com/news/intel-responds-to-security-research-findings/

[7] Website/blog:-
http://www.kb.cert.org/vuls/html/fieldhelp

[8] Website/blog:-
https://www.wired.com/story/meltdown-and-spectre-vulnerability-fix/

[9] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg "Meltdown Paper"

[10] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, Yuval Yarom "Spectre Attacks: Exploiting Speculative Execution"