Meltdown and Spectre vulnerabilities: Giving insight into reevaluating modern processors

Amna Shahid

Mount Holyoke College

May 10th, 2021

Acknowledgements: My parents

[Turano 2021]

Abstract:

Meltdown and Spectre are computer vulnerabilities which exploit certain mechanisms employed by modern CPUs. They mainly make use of the side effects of two mechanisms, out-of-order execution and speculative execution. The hacker uses them to manipulate the computer instructions being carried out, eventually accessing the targeted memory, resulting in the extraction of data. This is transferred from the hacked machine to the hackers device through a covert side channel, a mechanism which transfers messages secretly.

Meltdown and Spectre show how even though modern processors increase the performance of computers, they bring along new vulnerabilities to be exploited by hackers. Researchers have been able to come up with solutions to prevent Meltdown and Spectre attacks. It is significantly harder to prevent Spectre attacks, compared to Meltdown, since the former affects both the hardware and software. KAISER is an effective technology which protects computers against attacks such as Meltdown. But we still need to come up with a permanent solution against all such computer attacks.

Introduction:

New mechanisms built for modern processors to increase their performance and speed have ended up causing vulnerabilities in these systems. The side effects of two of these, out-of-order execution and speculative execution, are exploited in Spectre and Meltdown. Meltdown acts by breaking down the barrier between user mode and kernel mode, leading to kernel address spaces being accessed, and thus allowing access to physical address spaces and the memory.

Spectre attacks misuse speculative execution to carry out certain instructions and access memory which would otherwise have not been accessed if the correct instructions were carried out. Caches which store the page translation tables are accessed through covert channels during attacks such as Spectre and Meltdown. The data can then be easily read and information such as passwords etc can be stolen by the hacker.

Quite recently new vulnerabilities in latest CPUs have been discovered. Researchers should think about changing their approach while building CPUs to protect computers from all possible attacks, rather than having to come up with a unique solution each time a new vulnerability is discovered.

Mechanisms used in attacks:

Cache:

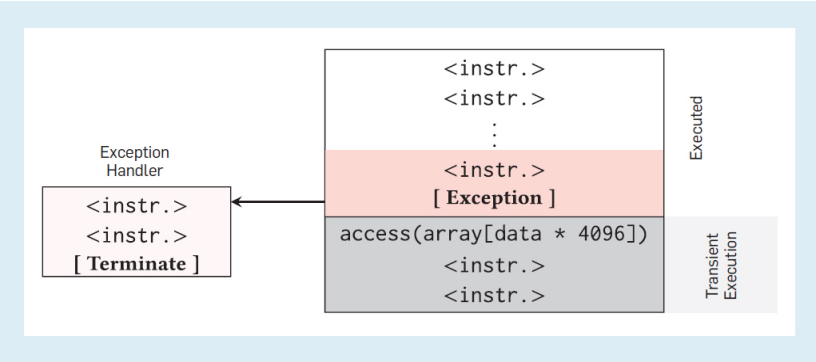
A cache is used to store recurrently used data, including storing certain address spaces' translation tables which are being commonly used. This way they can quickly be loaded and used from the cache, whenever required, increasing the speed of the computer. But caches also allow information from the computer to be leaked through a side channel, once the attacker is able to access them.

Out-of-order execution:

Out-of-order execution is "an optimization technique that allows maximizing the utilization of all execution units" [Lipp et al. 2020]. A specific example would be an execution unit carrying out an instruction requiring file input/output. The execution unit is free while the file is being read into memory. If there are other instructions waiting to be carried out, the unit will start executing another instruction instead of being idle while waiting for the file data to be read and stored in a register to then carry out the next command for that instruction. All execution units use this mechanism, allowing several instructions to be carried out in parallel and thus making computers fast.

Since out-of-order execution causes instructions to run in parallel, at times an instruction may be getting carried out at the same time as the instruction preceding it, or even if the

preceding instruction hasn't yet occurred. This is one of out of order execution's side effects; the once in a while execution of some instructions that were not meant to be executed. This can lead to the case of instructions after an exception being executed, without the exception being raised, since the instructions above haven't yet finished execution. This is shown in the diagram below as well.



[Lipp et al. 2021]

An exception is raised when the OS catches an error in the system and needs to get rid of the specific job which caused it, before harm is caused to the system. This is one of the protective mechanisms of the system to prevent the kernel's privileged mode from being accessed. But out-of-order execution causes this mechanism to be compromised. It "can change the microarchitectural state in a way that leaks information" [Lipp et al. 2020]. An instruction that was never meant to be executed, called transient instruction, can allow the system to be exploited. Memory is accessed for whatever data is required for that instruction, and once the data is cached it might be accessed by a malicious program through a side channel.

Speculative execution:

In speculative execution the computer predicts the next instruction to be executed, based on the previous instructions being carried out. Due to out-of-order execution, the processor may execute a certain instruction whose preceding instructions have yet to be executed. The results from those instructions might be required to carry out that specific instruction, but since they are not yet available we might speculate based on similar instructions executed before.

Afterwards when the preceding instruction is finally executed and our guess is correct, a lot of time is saved and we can move on, if not then the computer tries to reverse the changes. The prediction might be wrong when the computer checks the instruction's result against the

predicted value. In that case the registers stored contents for that instruction are removed and the parts involved, registers, memory etc, are reset.

Speculative execution relies on previous results by similar instructions. If we continuously follow the same path after a certain instruction it is fed into the mechanism for speculative execution. That path is taken again when we make a prediction regarding that specific instruction, since that was our best bet. Similar to out-of-order execution, microarchitectural changes are made when speculative execution occurs.

Spectre:

In Spectre, speculative execution is used by the hacker to carry out their instruction and leak data from memory. The hacker might first carry out correct instructions so that a certain path is continuously taken by those instructions. This has caused microarchitectural change; it has been fed into the system how a certain path is mostly true for a specific kind of instruction. Therefore now the hacker carries out the incorrect instruction, similar to that kind.

It is a transient instruction; they are “speculatively-executed instructions that are destined to be squashed” [Yan et al. 2021]. The hacker finds the virtual address space for the memory they want to access and use that to form the transient instructions. Due to speculative instruction the same path is taken, but this time it is incorrect. Through this the hacker accesses a certain part of the address space and the data stored in its memory. The instruction causes data to be loaded to a register and cached. The data will be removed from the register once the computer finds out that the instruction is incorrect, but the hacker can still leak the data from the cache through a covert side channel. [Kocher et al. 2019]

Meltdown:

Meltdown uses a transient instruction to carry out a series of events resulting in private data stored in the privileged kernel mode to be accessed, if the attack is successful. The attacker uses the transient instruction to get access to a certain target address space, which also has the kernel address mapped in it. The kernel has certain privileges; it can access physical address

spaces and carry out certain instructions that the user cannot. This is because the userspace is isolated from the kernel space in the usermode.

The exception command is distorted by either “exception handling” or “exception suppression” to get the transient instruction executed [Lipp et al. 2020]. Its memory accesses are then exploited to access the kernels address space. When the exception is raised, steps are carried out by the system to delete the memory access but there is a “small time window between the illegal memory access and the raising of the exception” [Lipp et al. 2020]. Thus before the data of the illegal memory access is removed after the exception is raised, the address space is transferred from the cache through a cache covert channel by the attacker.

Along with the kernel address the attacker also gets access to the physical addresses and therefore the data stored there. They can then carry out these steps multiple times to copy the whole memory and thus a lot of valuable data, which can then be misused.

Solutions:

Since Meltdown, Spectre, and other side channel attacks started, researchers have come up with several preventions against these vulnerabilities. It has been easier to come up with implementations to prevent attacks such as Meltdown which only affect the software, but there has been no clear solution against Spectre, which affects both the hardware and software. The methods to prevent Spectre attacks include using combinations of several implementations, but that is still not guaranteed to be fully effective. Researchers have suggested several solutions over the years, such as ConTExT , and 007 [Schwarz et al. 2020; Wang et al. 2020].

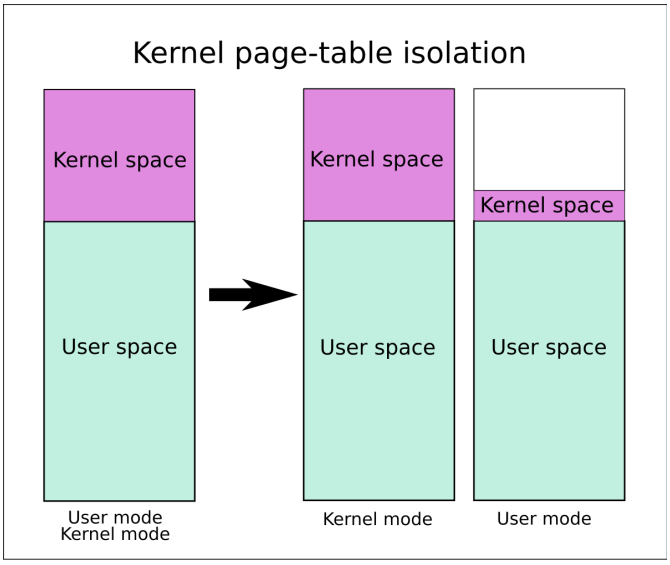
KASLR:

KASLR was an initial technique used to prevent occurrence of attacks such as Meltdown, by randomizing the address space locations. This purpose was to make it hard for the attacker to obtain access to one address space and then the other addresses, through it. But KASLR is weak because it can be easily manipulated by the user since they can try all the different possible addresses and find one which is correct. Once that is achieved they can carry out the steps multiple times to gain access to all the address spaces. They can eventually gain access to the

kernel space and physical addresses. Therefore KASLR is no longer used on its own. [Lipp et al. 2020;Gruss et al. 2017]

KAISER Patch:

The KAISER Patch "implements a stronger isolation between kernel and user space" [Lipp et al. 2020]. As shown in the diagram below, by implementing KAISER the userspace in the user mode does not contain the kernel space and therefore doesn't contain the physical memory either. Attacks such as Meltdown cannot be successful through this this since getting illegal access to the kernels privileged mode through the user space was its exploitation.



[Wikipedia 2021]

Even though "user pages are not mapped in kernel space" , there are still some privileged memory locations that are mapped into user space to carry out the instructions occurring in the processor at the current time [Gruss et al. 2017]. These are "only a minimal number of pages mapped both in user space and kernel space", but the pointers to these address spaces can still be exploited by the attacker, and they will get access to the other address spaces through them [Gruss et al. 2017]. KAISER is still quite effective in preventing side channel attacks, including Meltdown, which affect hardware. especially when combined with KASLR, the randomization of address space layout.

KAISER has been successful in preventing side channel attacks, including Meltdown and to a certain limit Spectre, in Linux kernel, Intel(intel page 9), Windows, and OS X (including macOS) systems [Intel 2018 ;Gruss et al. 2017;Lipp et al. 2020]. KAISER has a low overhead cost as well, making it an ideal solution against side channel attacks [Gruss et al. 2017].

Conclusion:

Out-of-order execution is a modern technique which first came with the new processor, POWER1, that was introduced by IBM in 1990 [Gara et al. 2011]. Speculative execution was introduced in the new microprocessors by Intel, in the 1990s [Hruska 2021]. Both mechanisms smartly schedule instructions, making modern processors efficient while execution.

They improved computer performance significantly but also brought with them new vulnerabilities to be exploited, the most well known being Meltdown and Spectre. Both “have in common that they exploit that the kernel address space is mapped in user space as well”, exactly the problem which the KAISER patch solves to a great extent against Meltdown [Gruss et al. 2017].

Hackers will continue to find new attacks as new technology is developed, for example recently three new vulnerabilities of Spectre have been discovered in the latest processors [Riley 2021]. Such new vulnerabilities are more sophisticated and can prove to be of greater harm because they use advanced techniques which allow the hacker to leak sensitive data quickly. For example, the Spectre vulnerabilities are harder to combat because they exploit speculative execution at a later stage. This is an unusual new development employed by hackers. Researchers are finding it hard to come up with a prevention against this without “roll(ing) back critical performance innovations in most modern Intel and AMD processors” [University of Virginia 2021].

Just as new technology enables hackers to find vulnerabilities to exploit within it, similarly new defenses can be developed just as quick. In the past implementations have been developed fast but this is an ongoing cycle which will continue to cause harm because of the privacy and data violations that occur when successful attacks are carried out. Therefore instead

of waiting for new attacks to take place and then coming up with the defenses for them, researchers need to figure out how to make a secure CPU from the ground up.

Works Cited:

Gara, A. et al., 2011. IBM Power Architecture. *Encyclopedia of Parallel Computing*, pp.900-907.

Gruss, D., Lipp, M., Schwarz, M., Fellner, R., Maurice, C. and Mangard, S., 2017. KASLR is Dead: Long Live KASLR. *Lecture Notes in Computer Science*, pp.161-176.

Hruska, J., 2021. What Is Speculative Execution? - ExtremeTech. *ExtremeTech*.

Intel, 2018. *Newsroom.intel.com*.

Kocher, P. et al., 2019. Spectre Attacks: Exploiting Speculative Execution. *2019 IEEE Symposium on Security and Privacy (SP)*.

Lipp, M. et al., 2021. *Figure 2.*

Lipp, M. et al., 2020. Meltdown. *Communications of the ACM*, 63(6), pp.46-56.

Riley, D., 2021. New Spectre vulnerabilities discovered on Intel and AMD processors - SiliconANGLE. *SiliconANGLE*.

Schwarz, M., Lipp, M., Canella, C., Schilling, R., Kargl, F. and Gruss, D., 2020. ConTExT: A Generic Approach for Mitigating Spectre. *Proceedings 2020 Network and Distributed System Security Symposium*.

The ITS Crew, 2021. Spectre & Meltdown: Newly Discovered Vulnerabilities that Affect Almost All Computing Devices. *ITS Tactical*.

Turano, G., 2021. *Meltdown e Spectre*.

University of Virginia School of Engineering and Applied Science, 2021. Computer scientists discover new vulnerability affecting computers globally. ScienceDaily.

Wang, G., Chattopadhyay, S., Gotovchits, I., Mitra, T. and Roychoudhury, A., 2020. oo7: Low-overhead Defense against Spectre attacks via Program Analysis. *IEEE Transactions on Software Engineering*, pp.1-1.

Wikipedia, 2021. *Kernel page-table isolation.*

Yan, M., Choi, J., Skarlatos, D., Morrison, A., Fletcher, C. and Torrellas, J., 2021. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy.