ISSN 2347 - 3983
Volume 8. No. 3, March 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter21832020.pdf
https://doi.org/10.30534/ijeter/2020/21832020

Mitigating Risk of Spectre and Meltdown Vulnerabilities

Wong Jin Kee[1], Mohd Fadzil Abdul Kadir[1*], Fauziah Ab Wahab[1], Aznida Hayati Zakaria@Mohamad[1], Mohamad Afendee Mohamed[1], Ahmad Faisal Amri Abidin@Bharun[1]
[1]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut Campus, Malaysia

ABSTRACT

Speculative execution (Spectre) and Meltdown is a chip attempting to predict the future in order to improve the system performance which involves multiple logical branches, it will start working out the math for all those branches before the program even has to decide between them. Normally, it works together with caching which is to speed up the memory access by filled with the data that will need some and often the output of the speculative execution is stored here. The speculative execution and caching in the operating system improve the over- all system and operating system performance through the prediction of data or resources to be used and cache memory is for quick access of data or resources. The problem arises where this function could allow potential attackers to get access to data they should not have access by exploiting the Spectre attacks and Meltdown. Spectre attacks and meltdown open up the possibilities for dangerous attacks which involved the breach of security and confidentiality of the user. Various techniques and patches have been introduced to mitigating the Spectre attack and meltdown. In this paper, we present the view various variants of attack from the speculative execution with its mitigation techniques.

Key words: Spectre Attack; Meltdown; Speculative Execution; Attack Mitigation.

1. INTRODUCTION

Speculative Execution [1, 2, 3] is a technique used by a processor to increase the performance by predicting the likely outcome and execute the predicted outcome instruction prematurely as shown in Figure 1. The processor will start to calculate the math for all the branches while the program is still deciding between them. It is also an optimization technique where the processor performs tasks that may not be required. In the simple term, the work is done before the process determine whether it is actually needed as this is to prevent the delay of the system performance if the work is carried out after known that it might be needed. For example, if the program decide that the process A is true then compute function X; if A is false then compute function Y, therefore, the processor with speculative execution will start working out and compute both the outcome of whether A is true or false which is X and Y in parallel before the it knows whether A is true of false [1, 2]. Once after the A has made the decision whether it is true or false, it already has a result ahead which this is to speed up the processing time.

For a program instruction to be processed in the processor it might takes several clock cycles before the outcome is known, rather than wasting the cycle by putting it them in idle, the processor speculatively execute the program to product possible outcomes [4]. For any variation, the processor may start to learn the function that frequently used or accessed by the program and the program will speculatively execute the function during the idle time even without the acknowledgement of the user [1, 4, 5].

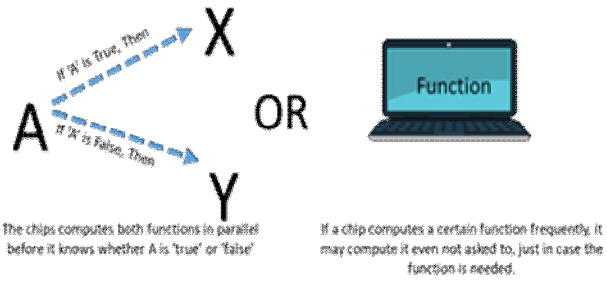

Figure 1: Definition of speculative execution.

1.1 Caching

As shown in Figure 2, caching is also a technique used to speed up the process from aspect of how the memory is accessed. For a process from the processor to access the memory in RAM to fetch data it might take a relatively long time compared to access the cache memory.

The RAM located on a separate chips but the cache memory is a small amount of memory located in the processor or also known as CPU memory which it can be accessed quicker. The cache memory is stored with all the data and information that needed by the processor that will be needed soon or often. Besides, the results of the speculative execution are always stored in this memory and this is a reason that speculative execution will be performing faster [1, 2].

1.2 Protected Memory

In computer security, protected memory as shown in Figure 3, is part of its fundamental concepts where the speculative execution and caching is part of the protected memory [2]. With protected memory, no process will be allowed to access the protected data without permission, this is to allow the program to keep the data in privacy from some of the users and also allow the program from accessed to other program's data. For a program to access a protected data it has to undergo a process called – privilege check to determine whether or not this process is allow to fetch the data [1, 2]. Since privilege check is also a process where it takes time for it to be processed and while it is taking its time to compute the process will be idle. While the process is being examined if the process is allowed to access the data, the speculative execution will start to execute its task by starting to work with the data even it is not granted with permission [1].

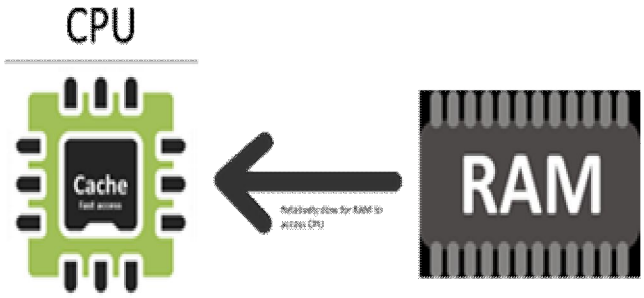

Figure 2: Definition of caching

However, it is still safe and secure as the speculative execution and cache is still being protected at the hardware level. Since the results of the speculative execution will be stored in the cache memory, the process will not be allowed to access and see the data until it passes the privilege check, if it does not pass through the check then the data is discarded. This is where the problem begins, the protected data is kept in the cache memory after speculatively executed even the process has not been granted permission to access it and also due to cache memory can be accessed relatively faster than other memory like RAM, the process will attempt to access the memory to locate whether the data has been cached [6]. The data can be predicted and deduced by identifying the location of the data through an attempt to access the data. If the data has been cached, the attempts will be rejected more quickly so the location of the data could be easily identified. This is also known as side-channel attack [7]. Side-channel attack is type of attack based on information gained from a physical level of a cryptosystem. It is neither classified as brute force nor attempts to breach a cryptosystem by social engineering with legitimate access. Side-channel attack is based on the assumption by observing the algorithm being executed on a processing device [6, 7].

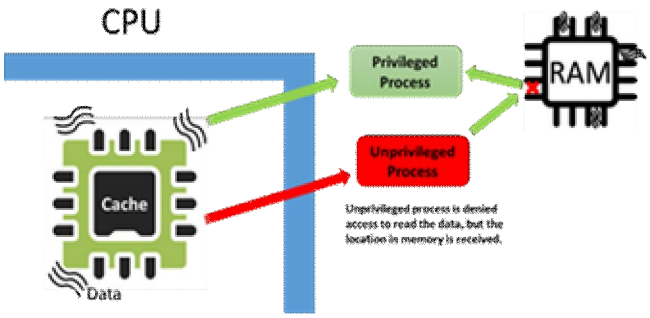

Figure 3: Definition of protected memory

2. MELTDOWN AND SPECTRE ATTACKS

Speculative execution has been running on many modern CPUs for more than 20 years, which means these vulnerabilities have existed for more than 20 years but there is no one raise up the issues of these vulnerabilities. Speculative execution is used to speed up the system performance based on the prediction of the unknown future outcomes of branches [8]. If the speculative execution is efficiently executed, it will eventually improves the parallel computations in hardware thus it can improve the performance of serial execution of the program [9]. Recently, this method has been widely used in high performance CPU designs and structure as these hardware or machines have strong predictive ability for future executions [10]. Speculative execution in the microarchitectures supposed to be highly invisible to the architecture of the programmers that develop its software, the instruction that are predicted wrongly should be cancelled so preventing its architectural outcomes from being revealed to the program. The consequences of the speculative execution is effect on the cache memory where this allows side-channel attack to attack the memory which consists of the speculatively executed outcomes and the attackers will able to extract the content (sensitive data or private personal data) from cache memory by manipulating the speculation execution to trigger a cache-timing side channel back to the attacker [11]. There are five variants of speculative execution attack which affect affected many modern processors by Intel, AMD and ARM [12, 13].

Variant 1 : Bounds Check Bypass (CVE-2017-5753)
Variant 2 : Branch Target Injection (CVE-2017-5715)
Variant 3 : Rogue Data Cache Load (CVE-2017-5754)
Variant 3a : Rogue System Register Cache (CVE-2018-3640)
Variant 4 : Speculative Store Bypass (CVE-2018-3639)

Google's Project Zero team has posted an article about the detailed technical information on these three variants of the newly discovered security issues involving speculative execution in the starting of year 2018 [14]. In the post, Google provide a summary of speculative execution and how each of the five variants being implemented [12]. According to the post by Google, there is no specific fix of each of the attack

variants but each of the attack requires its own mitigation, patches or protection against this attack.

2.1 Variant 1: Bound Check Bypass (CVE-2017-5733)

This variant is the vulnerability which affects a certain sequences in the compiled applications at which these sequences must be addressed on a per-binary basis. Bounds checking features have been built into most of the binaries and this variant allows the malicious code to evade it. Bound checking is a method used to detect whether a variable is falls within some bounds before it is being used [12]. This is to ensure that the number of variables of an array index is fits into a given bounds (range checking and index checking). Usually, it is time consuming to perform bound checking for every usage therefore the CPU will speculatively execute the instructions after the bounds check. The speculative execution allows access to memory that the process could not access to. Once the bounds check has been failed, it will discard the outcome that was produced speculatively [12, 15].

However, the changes to the system which is the changes to the state of cache memory will be observed by the malicious code [12]. The attackers will use the malicious code to detect the changes and read or retrieve the data that was speculatively accessed. In the kernel, this affected the systems as the extended Berkeley Packet Filter (eBPF) that receive the packet filters from the user space code, just-in-time (JIT) will gather the packet filter code and execute it within the context of the kernel. The JIT compiler will be using bounds checking to restrict the memory that the packet filter can access [12]. This allows an attacker to use Variant 1 speculation to evade the bounds checking features by repeatedly calling the eBPF program with an out-of-bounds offset which will target the a data to leak [15].

Google's Project Zero team has carried out a research and in their proofs of concept (PoCs) they demonstrated the operation of Variant 1 in userspace, testing on the Intel Haswell Xeon CPU, AMD FX CPU, AMD PRO CPU and ARM Cortex A57. From this PoCs, they have tested that it has the ability to read the data from the mis-speculated execution but it is within the same processes without overlap the privilege boundaries [15]. One of the consequences of Variant 1 is that the system will have difficulties in running untrusted code as it will restrict the access of untrusted code into the memory [12].

2.2 Variant 2: Branch Target Injection (CVE-2017-5715)

This variant is the vulnerability which concern about the leakage of information [12]. In this variant, the attacker may use one of the processes to affect the other process's speculative execution behaviour of code [10]. In modern processors, it uses the prediction of destination for indirect jumps, the program may start the speculative execution code for the predicted program. Prior research also shown there are possibilities for a separated security contexts code to influence each other via branch prediction. In simple term, this variant allows the interference from victim to attacker and vice versa as shown in Figure 4 [15].

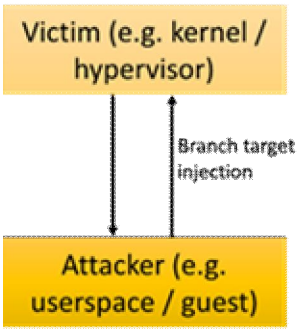

Figure 4: Interference between victim & attacker in Variant 2

According to the whitepaper published by Intel on February 2018, this variant allows the indirect branch predictor to use the information from the previously executed branches to predict the location of the future indirect branches. Intel also shows the example of indirect call in compiled languages like C and C++ in the paper. Programmers will use the function in the compiled languages to perform indirect calls [16]. The indirect calls can be executed by passed a comparison function using sort functions where the call from the sort ( ) is likely to be an indirect call, Figure 5 shows the indirect call of calling from inside sort ( ).

```
int compare(int a, int b)
{
    return a < b;
}
sort(array, &compare);
```

Figure 5: The indirect call from inside sort ( ) which can passed a comparison function [16].

In the paper, it also added that indirect calls will be frequently executed with calls to object functions in C++ as in Figure 6.

```
Vehicle *car = new Car();
car->drive();
```

Figure 6: The indirect calls executed with the calls to object in C++ [16].

2.3 Variant 3: Rogue Data Cache Load (CVE-2017-5754)

This variant is also known as meltdown as it will "melts" the security boundaries which are enforced by the hardware [1]. The attackers will be able to violate the security of the hardware and access to the memory even in user mode [17]. The attack allows a user mode process to access as if is in the kernel mode via virtual memory. In some situations, the speculative execution may access memory virtually to the current executing processor, for example, a user mode may access to the memory as if it is in the kernel mode using speculative execution. In the recent post by Google, they have

running the Intel Haswell Xeon CPU in the normal user mode but they can successfully read kernel memory from the processor under some predefined condition. Google believed that this precondition is the targeted kernel memory must be in the L1D cache [15]. However, this variant of attack is only works with the specific types of Intel processors [15].

2.4 Variant 3a: Rogue System Register Cache (CVE-2018-3640)

Variants 1 and 2, known as Spectre and Variant 3, known as Meltdown which is the processor vulnerabilities proposed by the Google Project Zero researched early of January 2018 [18]. Recently, Microsoft and Google researchers have discovered Variant 3a and Variant 4 which is similar to Spectre variant that takes advantages of speculative execution to retrieve unauthorized data via side-channel attack. Variant 3a also known as Rogue System Register Read is a variation of Meltdown that allows the attacker to access to the system by local access to read sensitive data and system parameters by using the side-channel analysis. Kernel Address Space Layout Randomization (KASLR) could be bypass once the attacker successfully exploited this vulnerability. This variant may affect AMD, ARM and Intel CPUs [19].

2.5 Variant 4: Speculative Store Bypass (CVE-2018-3639)

Storing data and loading data from memory addresses is part of the processes in the processor. There are buffering requests to store and load the data from the memory addresses which uses the speculative execution to make sure the storing and loading of data execute as quickly as possible to increase the system performance. Before the memory writes, the processor will check if any addresses used in the load was part of the recent store to the same addresses to avoid error, if so, the speculative data will gets thrown out [20]. The problem is, this speculative occurs in a shared unsecured area so it is possible for unauthorized users to see it. This allows unauthorized disclosure of information to an attacker by local user access via a side-channel analysis also known as Speculative Store Bypass or SpectreNG. By tricking the processor, attackers can steal data like passwords, and credit card numbers undetected [18].

3. MITIGATION TECHNIQUES

The vulnerabilities attack at the hardware level and it cannot be patched or there is no specific fix for each of the vulnerabilities but there are techniques introduced by the vendors to mitigate the vulnerabilities. By mitigating means the vulnerabilities will be minimized or to prevent it from being severe but not vulnerabilities will not be removed completely. Many major processors vendor such as Intel, AMD and ARM have re- ported that the processors have being affected more than one variant of vulnerabilities. The expert around world and the processor vendors has being working around the vulnerabilities to research and product the effective techniques and patches to mitigate the attacks. Microsoft, Apple, Google, Firefox and Samsung has released operating system patches for most version of their machine to prevent this attack from attacking their users, their patches was announced publicly at the starting of January 2018 [21]. Table 1 is the summary of the mitigation techniques to mitigate the various variants of the vulnerabilities.

Table 1: Mitigation Techniques for Meltdown and Spectre Attacks [10]

Type of Mitigation	Technique
Branch avoidance in favour of conditional moves. Mitigate: Variant 1	Conditional move instruction permits conditional behaviour. No involve branches. Reasonable form of mitigating. Not use of data-value speculation.
Speculation barriers and code behaviours to limit speculation. Mitigate: Variant 1, Variant 2	Speculation barrier instruction is used to prevent further speculative loads. Load Fence (LFENCE) used as a barrier to eliminate all outstanding speculation, barrier can be placed follow by software bounds check but be-fore memory access. On AMD processors, LFENCE is placed before an indirect jump. Indirect Branch Prediction Barrier (IBPB) to limit indirect branch prediction after an explicit barrier. Indirect Branch Restriction Set (IBRS) to restrict branch speculation. STOBP to protect hyperthreads from branch-predictor manipulation. Conditional Speculation Dependency Barrier. (CSBD) to prevent speculative loads, stores, instructions prefetches and indirect branches. Return Trampoline (Retpoline) techniques is in introduced by Google to cause indirect jump to prevent branch prediction on Intel and AMD processors [22].
Explicit flushes of architec tural and microarchitectural state.	Architecture does not include portable interfaces to flush the branch-predictor state.

Mitigate: Variant 3	Recommend to clear untrusted values from registers hen entering privileged mode or sensitive code.
Implicit flushes of microar chitectural state. Mitigate: Variant 3	AMD suggested flushing the branch-predictor table using a well-defined series of function calls. Used during entering to privileged modes to prevent the behaviour of user space to influence kernel branch prediction
Un-sharing user and kernel address spaces Mitigate: Variant 3	Kernel Page Table Isolation (KPTI) consists of Side-channels Efficiently Removed (KAISER) that prevents the mapping to kernel address space when operating in user space [23]. Supervisor Mode Execution Protection (SMEP) in Intel and AMD processor to prevent the influence of speculation of user instruction towards the kernel mode. Supervisor Mode Access Protection (SMAP) used to prevent the speculated kernel to access to the user data.
Mitigate Rogue System Register Read Mitigate: Variant 3a	This vulnerability is only via a microcode or firmware update [24].
Mitigate Speculative Store By pass Mitigate: Variant 4	Patches using Speculative Store Bypass Disable (SSBD) will be introduced by Intel and AMD which this inhibit the Speculative Store Bypass thus elimination the risk of security. Microsoft to develop, release and deploy defence-in-depth mitigations [25].

4. CONCLUSION

In this paper, we presented the review of the current security vulnerabilities which is Meltdown and Spectre attacks. Both the Meltdown and Spectre attack take the benefit of speculative execution features in most of the modern CPU or high-performance computers. The speculative execution will supposed to be a feature to increase the performance of the processors and provide parallel computing to shorten the process pro- cessing time but the Meltdown and Spectre exploitation is abusing it to access to the protected memory or privileged memory to retrieve the protected data. The vulnerabilities arises from the speculative execution can be divided into three variant which is Spectre (Variant 1 and 2) and Meltdown (Variant 3). Later, Variant 3a and 4 has been discovered on 21 May 2018 by Microsoft and Google. Each of the variant uses the advantage of the speculative execution to access to the protected memory by they access via a different ways where Variant 1 is accessed via bounds checking features, Variant 2 is accessed via branch target inject and Variant 3 is accessed to kernel mode via the virtual memory from the user mode. Variant 3a allows the attackers to read the system parameters via local access using side-channel analysis. Variant 4 which is Speculative Store Bypass allows the attacker to read older memory values in the processor's stack or memory location using side-channel attack. Each of the variant opens up the possibilities for dangerous attacks. There are various patches and techniques have been introduced by the vendor and computer expert to mitigate this vulnerability. Each of the variant does not have a specific fix to remove the vulnerability but it can only be managed by the patches to minimize the attack.

ACKNOWLEDGEMENT

Special thanks to the Centre for Research Excellence and Incubation Management (CREIM), Universiti Sultan Zainal Abidin for providing research fund for this research project.

REFERENCES

1. J. Fruhlinger. CSO: Spectre and Meltdown explained: What they are, how they work, what's at risk, https://www.csoonline.com/article/3247868/vulnerabilities/spectre-and-meltdown-explained-what-they-are-how-they-work-whats-at-risk.html, last updated on 2018/01/15.
2. The ITS Crew. Spectre & Meltdown: Newly discovered Vulnerabilities that affect almost all computing devices, https://www.itstactical.com/digicom/security/spectre-meltdown-newly-discovered-vulnerabilities-affect-almost-computing-devices/, last updated on 2018/02/14.
3. J. Hruska. ExtremeTech: What is Speculative Execution, https://www.extremetech.com/computing/261792-what-is-speculative-execution, latest update on 2018/01/10.
4. P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, Y. Yarom. Spectre Attacks: Ex- ploiting speculative execution, Cornell University Library (Jan 2018). https://doi.org/10.1109/SP.2019.00002
5. Giorgi Maisuradze, Christian Rossow, Speculose: Analyzing the Security Implications of Speculative Execution in CPUs, Cornell University Library (January 2018).
6. P. Bright. As predicted, more branch prediction processor attacks are discovered, https://arstechnica.com/gadgets/2018/03/its-not-just-spe

ctre-researchers-reveal-more- branch-prediction-attacks/ last update on 2018/03/27.
7. O. Cherednichenko, A.A. Baranov, T.I. Morozova. Side-channel attack, National Min- ing University, pp. 155.
8. I. Sarju. Spectre & Meltdown Processor Vulnerabilities: A Technical Introduction, https://hackernoon.com/spectre-meltdown-processor-vulnerabilities-a-technical-introduc- tion-e3d09d6699a6, last update on 2018/02/03.
9. Trend Micro, Detecting Attacks that Exploit Meltdown and Spectre with performance coun- ters, https://blog.trendmicro.com/trendlabs-security-intelligence/detecting-attacks-that-exploit-meltdown-and-spectre-with-performance-counters/, last update on 2018/03/13.
10. T. Robert, N.M. Watson, J. Woodruff, M. Roe, S. W. Moore, P. G. Neumann. Technical Report: Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks, University of Cambridge (February 2018).
11. D. Page. Partitioned Cache Architecture as a Side-Channel Defence Mechanism (January 2005).
12. M. Linton. Google Security Blog: More details about mitigations for the CPU speculative execution issue, https://security.googleblog.com/2018/01/more-details-about-mitigations- for-cpu_4.html, last updated on 2018/01/04.
13. G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, T. H. Lai, SgxPectre Attacks: Leaking Enclave Secrets via speculative execution, Cornell University Li- brary (February 2018).
14. C. Page. Google claims its Spectre patch results in 'no degradation' to system performance, https://www.theinquirer.net/inquirer/news/3024392/google-claims-its-spectre-patch-results-in-no-degradation-to-system-performance, last update on 2018/01/15.
15. J. Horn. Google's Project Zero: Reading privileged memory with a side-channel, https://googleprojectzero.blogspot.my/2018/01/reading-privileged-memory-with-side.html, last updated on 2018/01/03.
16. Retpoline: A branch target injection mitigation whitepaper, Intel (February 2018).
17. M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg. Meltdown, Graz University of Technology, Cyberus Technology GmbH, Independent, University of Pennsylva- nia and University of Maryland, University of Adelaide and Data61, Rambus, Cryptography Research Division (Jan 2018).
18. S. Khandelwal. New Spectre (Variant 4) CPU Flaw Discovered-Intel, ARM, AMD Affected, https://thehackernews.com/2018/05/fourth-critical-spectre-cpu-flaw.html, last up- date on 2018/5/21.
19. Microsoft. Protect your Windows devices against Spectre and Meltdown, https://support.microsoft.com/en-us/help/4073757/protect-your-windows-devices-against-spectre-meltdown, last update on 2018/07/25.
20. Side-Channel Vulnerability Variants 3a and 4, https://www.us-cert.gov/ncas/alerts/TA18-141A, last update on 2018/05/21.
21. About speculative execution vulnerabilities in ARM-based and Intel CPUs, Apple, https://support.apple.com/en-my/HT208394, last update on 2018/01/29.
22. P. Tuner. Retpoline: a software construct for preventing branch-targ.get-injection, Google, https://support.google.com/faqs/answer/7625886.
23. L. Abbott. KPTI-The new kernel feature to mitigate "meltdown", https://fedoramagazine.org/kpti-new-kernel-feature-mitigate-meltdown/, latest update on 2018/01/05.
24. Microsoft Guidance for Rogue System Register Read, https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/ADV180013, last update on 2018/08/08.
25. Microsoft Guidance for Speculative Store Bypass, https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV180012, last update on 2018/08/08.