Investigating the Impact of Spectre Vulnerability Patches on AMD CPU System Performance

How does the Spectre vulnerability patch affect the performance of systems operated by AMD CPUs?

Computer Science Extended Essay
Word Count: 3948
klk611

Table of Contents

Introduction

In early 2018, a significant CPU vulnerability named 'Spectre' was brought to light by a collaborative effort of researchers from institutions including Google Project Zero, Cerberus Technology, and Graz University [2]. Spectre represents a deep security challenge as it exposes the potential for attackers to access and read CPU processes. This vulnerability threatens millions of users as well as businesses, notably the healthcare sector, by allowing the theft of sensitive data such as passwords and financial information [14]. This security flaw has sent shockwaves across the cybersecurity world, placing analysts and engineers in a state of constant alarm and response.

The impact of the Spectre patch on system performance has been the topic of numerous reports, with varying viewpoints on the extent of its influence. The varied nature of these reports reflects the difficulty of analyzing the patch's effects, as they depend on factors such as unique hardware configurations, the type of workloads, and the methodology used in the assessments. This disparity in observations highlights the need for a more complex and contextualized understanding of how the Spectre patch interacts with various system contexts, driving a deeper investigation into the complexities of its impact on performance across a range of computing scenarios, which poses the research question: How does the Spectre vulnerability patch affect the performance of systems operated by AMD CPUs?

Thus, the objective of this essay is to systematically evaluate various scenarios and workloads, with an emphasis on AMD systems. The goal is to carefully examine the effects of the Spectre patch through different benchmarking tests, such as daily user workloads, mathematical calculations and graphics rendering, to determine whether the Spectre patch affects personal computers' performance.

The discovery of Spectre is significant as the flaw affects devices that contain Intel CPU chips, ARM, and AMD CPUs produced after 1995 [2]. This indicates that a wide collection of devices are affected by this vulnerability, from huge multinational institutes like Google and Microsoft to the average user. The term 'Spectre' derives its name from its exploitation of 'Speculative Execution,' which deceives CPUs into executing commands that typically fall outside their range. This approach grants attackers access to confidential data residing in the memory of other concurrently running programs [2].

Spectre's origins can be traced back to flaws in the architecture of central processing units (CPUs). Recognizing the fundamental nature of the issue, the only true and long-term remedy is a total redesign of the hardware itself. Undertaking such a massive makeover, on the other hand, is a challenging process, and major chip manufacturers like Intel and others have the challenge of executing significant changes inside their existing hardware frameworks. The complexities involved in creating and introducing revised CPU designs necessitate a significant investment of time and resources.

In the meantime, as a practical response to the situation's urgency, developers and cybersecurity professionals quickly gathered to develop and split patches aimed at avoiding and limiting possible attackers' exploitation of the Spectre flaw. While these patches are critical for reinforcing systems against immediate threats, they are merely stopgap measures until more complete and hardware-oriented solutions can be adopted. The inherent contrast between the need for immediate response and the difficulty of altering basic hardware components highlights the complicated issues that the tech industry has in navigating the terrain of hardware vulnerabilities.

Had the researchers not identified this vulnerability, malicious attackers could have carried out a 'day-zero attack.' In such a scenario, the attack takes place when hackers exploit the flaw before developers have a chance to address it, potentially releasing chaos worldwide, reminiscent of the impact seen during the WannaCry ransomware outbreak [4].

Background Information

Within this section, I will include background information on the microarchitectural elements included in modern high-speed CPUs. The purpose is to shed light on how these elements support improved performance; and second, to examine the possible channels by which they can unintentionally expose data from programs that are now in operation. This investigation explores the inner workings of contemporary processors and their consequences for system security, offering knowledge of the fine balance between performance optimization and the unintentional disclosure of sensitive information.

Speculative Execution:

Frequently, the CPU needs to know what instructions or code it needs to execute next in a program. The CPU waits for the next instruction to be fetched or decoded too long, limiting CPU performance and speed. This occurs when out-of-order execution reaches a conditional branch instruction that depends on the oncoming instructions whose execution is not completed yet [2].

With Speculative Execution, the CPUs can preserve their current register state, and use various prediction mechanisms to determine which instructions are most likely to be executed next. This prediction is based on past execution patterns, branch history, and other factors [2]. If the CPU prediction is correct, then the result of speculative execution will be utilized, rendering a performance advantage over idling during the wait. If the prediction is incorrect, the CPU will abandon the speculative execution by reverting its register state continuing the right path. On modern CPUs, speculative execution can plan out hundreds of instructions. The size of the CPU's reorder buffer often determines the limit [13].

An analogy suitable for speculative execution is imagining a hiker lost in the woods. There is a fork in the road outlining two paths, with one path leading the hiker back home, and the other will not. Rather than the hiker wait until another hiker shows up and indicates directions, the hiker picks the path that will most likely bring them home. At some point, the hiker encounters a sign, and if that sign informs the hiker that it is the correct path, then they continue to go down it. However, if the trial sign indicates that it is the wrong way, then the hiker will simply go back to the fork and go the other path [5].

Branch Prediction:

Branch prediction is the process of predicting the outcome of conditional branches (e.g. if-else statements or loops) in a system before the outcome is determined. The main purpose is to keep the CPU pipeline full by minimizing stalls caused by branch instructions [2]. A digital circuit used to perform this operation is known as a “branch predictor”. It is a significant component of modern CPU architectures, such as the x86.

There are several CPU components used to predict the outcome of branches, including the Branch Target Buffer (BTB) [2]. BTB is used to keep a mapping from addresses of recently executed branch instructions to destination addresses, which the CPUs can use BTB to predict future addresses even before decoding the branch instruction. The destination is often encoded in the instruction and the condition is determined at runtime, so conditional branches do not require target address recording. Processors keep track of recent branch results to improve predictions.

The Return Stack Buffer (RSB) retains a copy of the most recently used portion of the call stack. A call stack is a data structure used in computer programming and software execution. The call stack manages the flow of program execution, especially when functions or subroutines are called [2].

Out-of-order execution:

The out-of-order execution improves processor performance in modern processors by allowing instructions later in the program's sequence to be performed in parallel with, and often even before, earlier instructions.

Processors use micro-ops, which are micro-level instructions that imitate the instruction set of the architecture. These micro-ops are generated by decoding high-level instructions. When all of the micro-ops for a certain instruction and all preceding instructions have been accomplished, such instructions can be retired [2].

This implies that they commit their changes to registers and other architectural states, freeing up space in the reorder buffer. As a result, instructions are retired in the order in which they appear in the execution sequence of the program [2].

Microarchitectural Side-Channel Attacks:

Microarchitectural components such as speculative execution, improve performance by predicting future instructions based on past behavior. When multiple programs execute on the same hardware, changes in the microarchitectural state can inadvertently affect other programs, resulting in unintended information leaks.

Microarchitectural attacks originally focused on timing variations and data leaks. Over time they extended to various other microarchitectural components, including instruction caches, lower-level caches, branch target buffers (BTB), and branch history. These assaults make use of the microarchitecture of a processor's subtle and unintentional interactions and side effects. Microarchitectural assaults are under the category of side-channel attacks, which acquire data from a system's physical implementation, such as its hardware components, rather than directly focusing on cryptographic algorithms or software flaws [2].

An example of a side channel attack is using the Flush+Reload technique and its variant Evict+Reload. It is used to leak sensitive information by evicting a cache line from a shared cache. To ascertain whether the victim accessed the line between the eviction and probing phases, the attacker timed the execution of a memory read at the evicted cache line. The Spectre attack utilizes this particular method.

Return-Oriented Programming:

By using a method called return-oriented programming (ROP), an attacker can influence arbitrary behaviour in a program whose control flow is diverted without injecting any code [2]. Control flow is the order in which the computer executes statements in a script [11]. This is achieved by linking together small code segments, known as gadgets, which are present in the victim's code. The gadget carries out a specific computation before returning control. This allows the attacker to perform intricate actions within the victim's program.

Spectre Attack:

Spectre emerges in two basic variants: one takes use of conditional branch mispredictions, while the other poisons indirect branches. In the former situation, attackers use microarchitectural side-channel assaults to get access to CPU memory, whereas the latter uses the previously mentioned Return-Oriented Programming (ROP) approaches. The attack below is a general idea of how Spectre exploits speculative execution.

Setup Phase:

The attacker manipulates the processor's behaviour to cause an inaccurate speculative prediction. Preparing a covert channel for extracting the victim's data, using techniques like Flush+Reload or Evict+Reload.

Information Transfer Phase:

The processor speculatively executes instructions that move confidential data from the victim's context to a microarchitectural covert channel. The attacker then triggers this transfer by having the victim perform specific actions or leveraging the speculative execution of their code.

Data Recovery Phase:

The attacker recovers the sensitive data using Flush+Reload or Evict+Reload, allowing the attacker to deduce the victim's information.

Current Mitigation:

The current technique for addressing the Spectre vulnerability varies across different technology companies. These methods are primarily designed to combat the two main Spectre versions, conditional branch misprediction and poisoning of indirect branches. The goal is to improve system security and prevent potential attacks from exploiting these vulnerabilities.

Intel and ARM, for example, have advised a mitigation technique that involves inserting a serializing instruction within the code, particularly near array limits and array access operations [6]. These serializing instructions are crucial in imposing a precise execution order within the CPU. They require that all flag and register changes made by previous instructions be completed before the subsequent instruction can be performed. Potential vulnerabilities connected with conditional branch mispredictions can be efficiently avoided by introducing this sequence.
In a similar spirit, AMD has proposed its solution to the Spectre vulnerability. AMD has proposed that mitigation for Spectre Variant 1 (conditional branch mispredictions) can be accomplished through operating system updates. This highlights the importance of software-based solutions in enhancing security against this specific version [6].

AMD has taken a stance that acknowledges the difficulty in exploiting Variant 2, which relies on the poisoning of indirect branches. Nonetheless, Microsoft still recommends installing an operating system update as a preventive measure [6]. This strategy is consistent with the general industry view that proactive software upgrades are critical in reducing potential risks.

Experimentation

To determine the effect of the Spectre vulnerability in AMD systems, variables were chosen into consideration to keep the results accurate. The decision to keep constant over two important variables—the CPU and the operating system—was the most important of these. This instance involved the establishment of a reliable and consistent setup with the AMD Ryzen 7 3700x CPU and Windows 11 Home Edition.

Method of Testing:

The effectiveness of the system will be evaluated using four various benchmarking programs. Because the Spectre patch affects how raw CPU instructions are executed, two performance scenarios—one with and one without the patch—will be compared using the Passmark PerformanceTest. Through various activities, including mathematical operations, compression, encryption, and Streaming SIMD Extensions (SSE), this thorough benchmark evaluates the system's performance [7].

The benchmarking programs PC Mark 10 and Geekbench 6 will be used to emulate real-world user workloads accurately. PC Mark 10 includes numerous workload stress tests, including online surfing, document processing, video and photo editing, and video conferencing [7]. On the other hand, Geekbench 6 assesses single-core and multi-core performance for a variety of tasks, including email management, photography, and concurrent music listening [8].

Cinebench R24 performs an outstanding job of accurately assessing the rendering performance of the CPU. Cinebench R24 tests the CPU capabilities of a computer by utilizing the power of Redshift, Cinema 4D's default rendering engine [9]. Using every processor core available, this test renders a 3D scene to evaluate the performance of the CPU. It assesses things like multi-core processing power, rendering speed, and overall CPU performance.

The tool InSpectre was used as a critical component of the methodology for the experiment. InSpectre is critical to the testing process since it allows for the selective disabling of the Spectre vulnerability patch. This intended deactivation enables a controlled and comparative investigation of system performance with and without the patch. The use of InSpectre as part of the experimental framework adds a level of accuracy and flexibility to the investigation, allowing researchers to investigate the complex dynamics of system behaviour under varied security setups. This intentional manipulation of the Spectre vulnerability patch status provides the scenario for a thorough study, providing significant insights into the complex interplay between security measures and overall system performance. [10].

Hypothesis:

Since the Spectre patch affects how instructions are executed by serialization, the CPU throughput should be affected significantly in a negative notion. CPU throughput is the amount of work completed in a unit of time. Daily workload performance should also be slightly negatively impacted by the patch.

Results

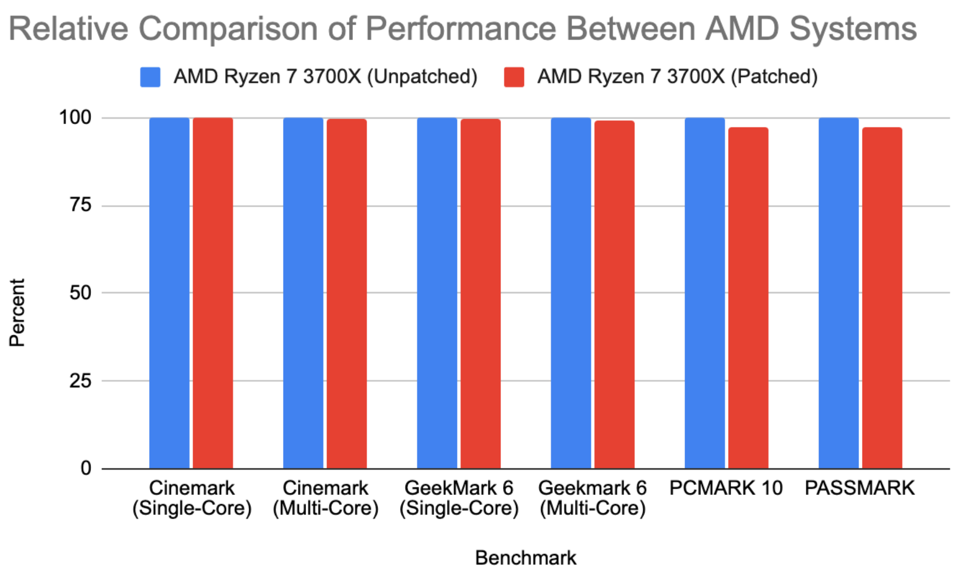
Figure 1 shows the raw data taken from the benchmarking programs with and without the patch:

Figure 1:

	Non-Patched Ryzen 7 3700X	**Patched Ryzen 7 3700X**
Cinemark (Single-Core)	72	72
Cinemark (Multi-Core)	705	704
Geekbench 6 (Single-Core)	1631	1634
Geekbench 6 (Multi-Core)	8219	8274
PCmark 10	6655	6483
Passmark	5267	5123

Figure 2 shows the data as a relative comparison:

Figure 2:



Analysis:

The raw CPU throughput analysis showed a slight but detectable drop of 2.73%. Even though this drop might not be considered very noteworthy, it still indicates a decrease in processing speed overall.

Notable performance reductions were also shown when a typical daily user workload was emulated, which is frequently a more accurate representation of real-world usage. Specifically, when PCMARK 10 benchmarking was applied, the system's overall performance decreased by 2.58%. This decline suggests a slight negative effect on various activities such as web browsing, video conferencing, application launches and document editing.

The Geekbench 6 benchmark tests provided information on the system's response in terms of single-core and multi-core performance. The findings showed that single-core performance had increased by a marginal but noticeable 0.18%, while multi-core performance had increased by a somewhat more noticeable 0.66%. These results completely contradicted my hypothesis, that the patch slightly increased performance in a portion of the daily user workload. Random variables of the system can be a culprit to the small performance boost.

The performance evaluation's reassuring finding is that the fix did not affect rendering activities. There were no appreciable variations in rendering performance between single-core and multi-core usages in systems with and without the patch, according to the Cinebench R24 benchmark test results.

Interpretation:

Carefully examining the trial findings reveals that there was no difference in system performance between using and not using the Spectre vulnerability patch. Even if a few observations did show slight performance losses, it's crucial to remember that these variations were insignificant enough to cause a noticeable decline in either daily user workloads or raw CPU throughput.

These results are significant since they imply that the Spectre vulnerability fix had little to no impact on AMD CPU performance. The conclusion is comforting for both system administrators and regular users. Users should not expect a noticeable decrease in performance as their systems will continue to operate smoothly and effectively. The impact on typical workloads and system responsiveness is minimal.

Factors influencing performance:

The effect of the instruction serialization that the patch implemented is one logical explanation for this subtle drop in speed. In essence, serialization is the act of processing instructions in a sequential manner as opposed to a parallel manner. In this case, the patch may cause instructions to momentarily stall in the pipeline. To put things in perspective, pipelining is a basic mechanism in contemporary CPUs that allows instructions to be queued and prioritized for effective execution [12]. The little variations in performance that were noticed might have been caused by the patch's modification of this pipeline behaviour.

Another possible factor that contributes to influencing performance is the background processes in the system. The number and kind of background processes running at the same time might have a considerable impact on CPU processing performance. The CPU's processing power is noticeably diminished when the system is overburdened with background tasks. This could manifest as increased CPU usage, longer reaction times, and, most importantly, increased system memory consumption.

As these background operations eat memory resources, the available memory for active tasks becomes constrained. This can lead to an increased dependency on virtual memory, which can slow down the system's performance. When confronted with an overwhelming number of background processes competing for system resources, the system may experience delays and decreased responsiveness.

Security vs Performance Trade-off:

Using the results from the experiment, it can be stated that turning off the Spectre patch will only give a marginal performance boost, compared to the usage of the Spectre patch.

It appears that the performance benefits of disabling the Spectre patch are negligible, especially in light of the benefits associated with using it. It is crucial to stress that the Spectre vulnerability is not impervious to exploitation, even though it may be difficult for hostile actors to take advantage of on its own. This contrast emphasizes how crucial the Spectre fix is to protect the system from future security lapses.

The trade-off between a minor performance boost and the possible danger of data leakage owing to a security flaw in the CPU is a difficult balancing act. The patch's principal goal is to protect sensitive data and maintain system integrity, ensuring a secure computing environment for both individuals and companies. The marginal performance improvement obtained by disabling the patch may be desirable in some cases, but it should be used with caution.

Limitations:

A limitation in this experiment is a lack of CPU diversity, causing the results to be limited to only one CPU. The Ryzen 7 3700X was used to experiment with the Spectre patches. This CPU was released in 2019, making it a modern CPU. Older CPUs can be used to determine the effect of the Spectre patch in older AMD systems. Server CPUs, such as the AMD EPYC series, can also be used to determine the effect of the Spectre patch on server-based performance. To diversify even more, Intel CPUs could also be introduced and experimented with for any performance degradation.

Conclusion

To answer the research question “How does the Spectre vulnerability patch affect the performance of systems operated by AMD CPUs?” it was concluded that the patch overall negatively impacted the performance of the system marginally, with the anomaly of the slight increase in performance shown in Geekbench 6. The hypothesis was correct as to the negative impact on performance but was not correct in the strength of the impact, which was remarkably small.

In the context of this essay, performance evaluation included three critical aspects: CPU throughput, execution of daily user workloads involving a variety of tasks, and rendering performance. The primary goal of this research was to determine whether the Spectre vulnerability fix had any visible impact on AMD computers.

Benchmarking tools were used to thoroughly examine the performance. PCMark 10 and Geekbench 6 were used to evaluate daily user workload performance, while PassMark was used to measure CPU throughput across a variety of workloads. Cinebench was used as the test rendering performance in both single-core and multi-core utilization scenarios.

The findings revealed a complex picture. PCMark 10 showed a decline in daily user workload performance, however, Geekbench 6 showed a modest increase in user workload performance, regardless of single-core or multi-core usage. PassMark found a little loss in CPU throughput, however, Cinebench revealed no discernible difference in performance comparing computers with and without the patch.

This comprehensive evaluation emphasizes that, while the patch did make some performance changes, the difference between systems with and without the patch was relatively minor. The measured performance drop was mostly unnoticeable and of little concern to the ordinary everyday user and system managers.

Two main explanations can explain the minor drop in system speed. The patch's implementation of instruction serialization, in which instructions are performed sequentially rather than in parallel, may have created occasional delays in the instruction pipeline. This change to the pipelining mechanism, which is a core feature of current CPUs, may explain the modest performance variances noticed.

Background processes running concurrently within the system have a significant impact on system performance. The quantity and kind of background jobs can have a substantial impact on CPU processing capability. Excessive background process activity can result in greater CPU usage, slower response times, and higher system memory consumption. This, in turn, could lead to a larger reliance on virtual memory, slowing system performance. When a large number of background tasks compete for system resources, system delays and responsiveness may occur.

This observation leads to a clear conclusion: keeping the Spectre vulnerability patch is recommended. The trade-off between security and a negligible performance improvement does not justify compromising system security. As a result, the patch's application is consistent with the larger goal of hardening the system's defences against future threats while ensuring that the user experience remains mostly uninterrupted and efficient. The evidence implies that the Spectre fix, in its current form, properly balances security and performance, protecting critical data while providing no significant obstacle to routine computing.

Future Research:

Further extensions to this experiment can be exploring the other processor vulnerability, known as Meltdown. This vulnerability was discovered around the same time Spectre was, but it affects systems in a much different way, by accessing private memory through the kernel system [5]. Exploring and experimenting with this issue can grant more information on how processor flaws can be exploited to steal private data, as well as how to attack works.

Furthermore, as security fixes to address vulnerabilities like Spectre and Meltdown are created and deployed, the critical question of their influence on system performance arises. Another intriguing area for future research is determining the extent to which Meltdown may cause performance reduction. Such research can provide crucial insights into the trade-offs that companies and users must weigh as they negotiate the complicated landscape of system security and performance optimization.

In-depth microarchitectural assessments can also provide a detailed understanding of how these patches interact with the processor's microarchitecture. Such investigations might reveal the complexities of how security measures are implemented at the hardware level and how they affect various system components.

Works Cited

[1] Arm Ltd., “What is a Central Processing Unit? – Arm®,” *Arm | the Architecture for the Digital World*. https://www.arm.com/glossary/cpu

[2] P. Kocher, D. Genkin, D. Gruss, and Y. Yarom, “Spectre Attacks: exploiting speculative execution,” *ResearchGate*, Jan. 2018, [Online]. Available: https://www.researchgate.net/publication/322253254_Spectre_Attacks_Exploiting_Speculative_Execution

[3] A. Staff, “Here’s how, and why, the Spectre and Meltdown patches will hurt performance,” *Ars Technica*, Nov. 09, 2020. https://arstechnica.com/gadgets/2018/01/heres-how-and-why-the-spectre-and-meltdown-patches-will-hurt-performance/

[4] N. Latto, “What is WannaCry?,” *What Is WannaCry?*, Jul. 21, 2022. https://www.avast.com/c-wannacry

[5] “What is Meltdown/Spectre? | Cloudflare,” *Cloudflare*. https://www.cloudflare.com/learning/security/threats/meltdown-spectre/

[6] A. Staff, “Meltdown and Spectre: Here’s what Intel, Apple, Microsoft, others are doing about it,” *Ars Technica*, Nov. 09, 2020. https://arstechnica.com/gadgets/2018/01/meltdown-and-spectre-heres-what-intel-apple-microsoft-others-are-doing-about-it/

[7] “How to read and understand CPU benchmarks - Intel,” *Intel*. https://www.intel.com/content/www/us/en/gaming/resources/read-cpu-benchmarks.html#:~:text=PassMark%20runs%20heavy%20mathematical%20calculations,-to-day%20productivity%20tasks.

[8] “Geekbench 6 - Cross-Platform Benchmark.” https://www.geekbench.com/

[9] “Evaluate your computer’s hardware capabilities | Cinebench from Maxon,” *Maxon*. https://www.maxon.net/en/cinebench

[10] “GRC | InSpectre.” https://www.grc.com/inspectre.htm

[11] “Control flow - MDN Web Docs Glossary: Definitions of Web-related terms | MDN,” Jun. 08, 2023. https://developer.mozilla.org/en-US/docs/Glossary/Control_flow

[12] “Pipelining.” https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/pipelining/index.html
J. Hruska,

[13] “What is speculative execution?,” *ExtremeTech*, Jun. 10, 2022. https://www.extremetech.com/computing/261792-what-is-speculative-execution

[14] “Meltdown’ and ‘Spectre’ guidance.” https://www.ncsc.gov.uk/guidance/meltdown-and-spectre-guidance#:~:text=%27Meltdown%27%20and%20%27Spectre%27%20are%20two%20related%2C%20side,be%20vulnerable%20to%20some%20extent.