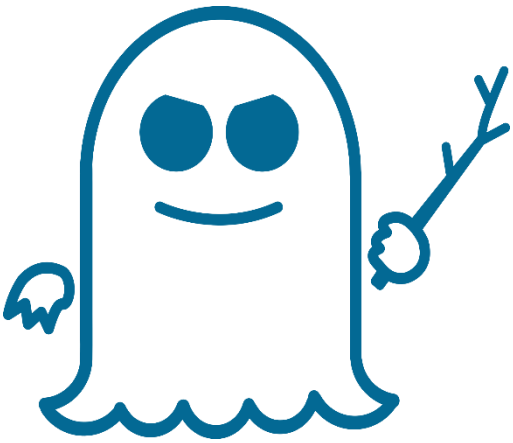
EPQ Dissertation

Responsible disclosure of security vulnerabilities

The industry debate following the experience of Meltdown and Spectre



Shepherd, Lydia
30-06-2018

Table of Contents

Abstract

The problems with the current system of disclosing security vulnerabilities in the technology industry were highlighted by the leaking of a series of security vulnerabilities, later called Spectre and Meltdown. Spectre and Meltdown are serious flaws in the silicon of many modern day processors including those of Intel, AMD and Arm. Thus, it was a problem that required responsible disclosure in order to allow mitigations to be created and to avoid public panic. However, the leaking of Spectre and Meltdown sparked a debate within the technology industry over whether responsible disclosure is the correct method for dealing with security vulnerabilities.

Zero Day disclosure is one alternative to responsible disclosure and involves releasing the details of the vulnerability as soon as it is discovered. This would allow all companies to have an equal chance to mitigate the problem, rather than only a few, select companies being told. However, this could also allow malicious users to exploit the vulnerabilities before they can be fixed. Whilst this debate is still ongoing within the technology industry, this paper hopes to set out an evaluation of Spectre and Meltdown and their role in this debate.

Introduction

In June of 2017 a series of security vulnerabilities in most modern-day computer processors was discovered by a team at Google Project Zero (Walden, et al., 2018). These vulnerabilities were later dubbed ‘Spectre’ and ‘Meltdown’. Due to the seriousness of these vulnerabilities, Intel, along with AMD and Arm, set an 8-month embargo period on the vulnerabilities in order to give them time to create fixes. Considering that these vulnerabilities are inherent to the chip design, it was impossible for them to fix them completely. Any fixes would require changes to be made to the silicon of the chip or the microcode that runs on it. Instead they had to attempt to mitigate the problems, with these mitigations negatively affecting the speed of the processor.

Whilst all the companies who were aware of these vulnerabilities worked to have patches ready for release, a week before the embargo date was due the vulnerabilities were leaked. This sparked a massive debate within the tech industry over how security vulnerabilities should be dealt with in general and whether Spectre and Meltdown specifically were correctly dealt with. A key part of the debate is whether it is fair to leave some companies out of the embargo but not others. For example, many smaller companies were not informed of Spectre and Meltdown and therefore could not create mitigations until it was already public.

1. Spectre and Meltdown

Spectre and Meltdown are the names used to refer to the two main classes of vulnerabilities discovered in June of 2017 by Jann Horn from Google Project Zero, Werner Haas and Thomas Prescher from Cyberus Technology, Daniel Gruss, Moritz Lipp, Stefan Mangard and Michael Schwarz from Graz University of Technology and Paul Kocher. There are in fact two variants of the Spectre attack, CVE-2017-5753 and CVE-2017-5715 (Horn, 2018). Common Vulnerabilities Enumeration (CVE) are unique codes given to vulnerabilities in order to differentiate between different types of the same vulnerability. By looking at the CVEs you can see that Spectre variant 2 was discovered first and then Spectre variant 1 and

Meltdown were discovered at the same time (they have numbers 5753 and 5754). They can both be exploited using side-channel attacks such as cache timing and power level reading. A side-channel attack exploits weaknesses in the hardware rather than the software which is why Spectre and Meltdown are so hard to fully fix.

1.1 Speculative Execution and Branch Prediction

Both Spectre and Meltdown rely on the fact that most modern processors use speculative execution to speed up processing time (Bright, 2018). Speculative execution allows processors to execute an instruction before it knows whether it will need to be executed or not. (Upton, 2018). If the result of the speculative execution is not required, it will be discarded without ever being stored in main memory. This allows the processor to execute programs much faster as it can keep all of its cores busy (many modern processors are multicore). However speculative execution can cause a change to the low-level architecture such as loading data into the cache when the data should not have been loaded (Bright, 2018). The cache is a small amount of very fast memory that is located on the processor chip itself and stores copies of the recently used memory locations (Upton, 2018). For example, if a program requires memory address 6 it will likely need the memory addresses surrounding 6 soon. So, the processor loads the entire block of memory addresses into cache, allowing it to retrieve data from those addresses faster, thus reducing processing time. If a program requires a lot of different memory addresses, it will take longer to execute due to the processor having to retrieve the data from the much slower main memory. The problem with *speculatively* loading data into the cache is that malicious code could potentially infer what data has been loaded into the cache which could include personal information such as passwords, depending on what program is currently running.

For this to happen, branch prediction is also required. Branch Prediction allows the processor to make a ‘guess’ at which branch it should take when it reaches a condition in the code, in order to avoid a stall (Upton, 2018). A conditional branch is an ‘if’ statement in programming terms and requires the processor to decide whether a given condition is true or false. In order to make the best guess, a branch predictor uses previous statistics to see which branch is taken more often. However, this can be exploited by an attacker who could train the branch predictor to make bad predictions by crafting a series of branches (Upton, 2018).

1.2 Meltdown CVE-2017-5754

Meltdown allows an unprivileged user program to read kernel memory and physical memory locations (Kocher, et al., 2018). All Intel processor chips are vulnerable to Meltdown because they do not use user privilege levels when speculatively executing commands. Arm claim that only some of its chips are vulnerable to Meltdown and AMD claims that none of its chips are vulnerable to Meltdown because they don’t speculatively execute around the kernel (Bright, 2018).

1.2.1 Memory Addressing

In order to fully understand Meltdown, you must understand the use of memory management in computers. Different areas in physical memory map to different types of memory. For example, Ring 3 (user applications) has its own section of physical memory which is separate to Ring 0 (kernel memory). This is done by using virtual addresses which are mapped to physical addresses using page tables. The Translation Lookaside Buffer (TLB) is a cache

which stores the most frequently used memory addresses so that the processor doesn't have to search the whole table every time. This process is carried out by the Memory Management Unit (MMU) on the CPU. It makes it so that the Ring 3 and Ring 0 data cannot overlap and read each other. In modern systems, the Ring 0 is mapped along with Ring 3 as an optimization so that the page tables do not need to be updated every time a request is made. This is only made possible by privilege levels so that if a user program tries to access kernel data it is blocked (Allen, 2018) (Bright, 2018).

However, Intel chips allow user applications to speculatively execute code to read kernel data which loads it into the cache. The request is successfully blocked once the processor realizes it should not allow access to the kernel data but if careful cache timing attacks are used, the data which was read from the kernel could be inferred.

1.2.2 Mitigations

The only mitigation for Meltdown is to use KPTI (Kernel Page Table Isolation) which separates Ring 0 and Ring 3 memory. This causes a performance loss due to the page tables having to be changed every time a request is made and the TLB having to be flushed more often (Allen, 2018).

It was when this patch was made for Linux that people first got an idea of how serious the problem was. Since Linux is open source, the patches had to be made in public view and so when KPTI was added in the patch, users were confused why they would add something that would negatively affect the performance of the OS. For programs that don't call the kernel much, a performance drop of 2-3% is expected. However, for programs that almost exclusively call the kernel, the performance drop may be as much as 50%. (Bright, 2018)

1.3 Spectre

Spectre allows data to be leaked from arbitrary memory in the cache by tricking the computer into speculatively executing code it shouldn't be able to during the program execution. There are two different variants of the Spectre attack which both allow data to be inferred from the cache. (Kocher, et al., 2018)

1.3.1 Cache Timing Attacks

Different areas in main memory have a fixed slot in the cache in which data from that area can be loaded into for processing.

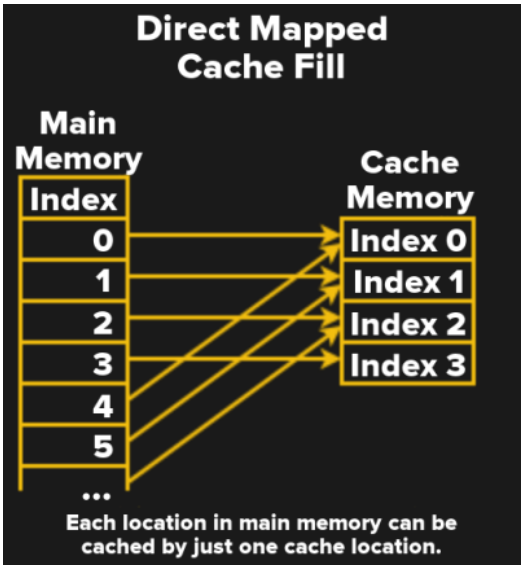


Figure 1: A diagram showing how main memory is mapped to cache

This makes it possible to time how long it takes for some data to be read or moved. If the data you request is in the cache then it will be very quick to execute it. Whereas if the data you request is not in cache, it has to be loaded into cache from main memory which is much slower so the request takes significantly longer (Upton, 2018) (Allen, 2018). Some example code would be:

READ clock

MOV R1, data

READ clock

This gets the time before and after it has copied the value ‘data’ to Register 1 so that it can tell if the value was already in cache or not. Cache timing is used to exploit both variants of spectre.

1.3.2 Spectre Variant 1 CVE-2017-5753

Variant 1 takes advantage of conditional branching in code to allow the data to be leaked. A conditional branch is when a condition has to be met in order for the code to branch. Spectre requires a branch with only two paths which is a very specific type of code for the user (not the attacker) to run so there are limited opportunities to exploit it.

The first step is to ‘poison’ the branch prediction so it will predict wrongly and speculatively run the code the attacker wants it to. Next the malicious code runs and fills up the cache with data so that when the user code runs, the attacker can use cache timing. The user code is then allowed to run and due to the poisoned branch prediction it will speculatively execute the wrong code which could contain important data. Then the attacker times the cache and sees which area of the cache took the longest to update so that they know which area of main memory the data came from. (Allen, 2018)

1.3.3 Spectre Variant 2 CVE-2017-5715

The second variant of Spectre works in a very similar way to the first but it takes advantage of indirect branching. Indirect branching can make a jump to anywhere in the binary code by supplying a pointer to the address where the data can be found. This makes Spectre variant 2 easier to exploit because there are more instances of this type of branch in user run code.

As before, the first step is to poison the branch predictor so that it will predict the path that is required. The difference this time is that the cache is flushed so it is empty before the user is allowed to run the poisoned code and speculatively execute the wrong data. Now the attacker times which cache channel is fastest to update in order to infer the data. (Allen, 2018)

Inferring just one small area of memory using this attack might sound useless but computers can run this attack multiple times a second. Therefore, it can very quickly build up a picture of what the main memory looks like so that they can know where to find the most important data such as passwords.

1.3.4 Mitigations

Spectre is much harder to mitigate than Meltdown because it requires the architecture of the entire processor chip to be changed. The patches that were released for it allow the branch prediction cache (the area of cache used to store the common branch predictions) to be flushed. While mitigating against attack, this causes a performance loss of up to 10% due to the branch predictor not having common branch predictions to use.

2. Dealing with Security Vulnerabilities

Dealing with security vulnerabilities within one's product is an important part of any tech company's job. However, it has always caused debate within the industry as to how they should be dealt with and whether the current method used by most companies is both effective and fair enough.

When a person discovers a vulnerability in a product, they have many options of what to do with the information. It can entirely depend on how and why they found the vulnerability. For example, Google Project Zero, one of the groups who discovered Spectre and Meltdown, are paid by Google to search for bugs and follow the route of responsible disclosure. However, it is also possible that someone with malicious intentions will discover the bug and choose to sell it, use it for extortion or release it publicly for fame and glory.

2.1 Responsible Disclosure

When the bug is discovered, if the discoverer decides to go privately to the company and tell them about the bug, this is called "responsible disclosure". It allows the company to create fixes for the problem before it goes public, which is clearly in the company's interests. This stops the spread of panic among the users and limits the number of people who could exploit the vulnerability (note there is always a chance a malicious user has discovered the same bug and is already exploiting it in the wild). Some companies will pay a "bug bounty" to anyone who comes to them with a bug. (Allen, 2018)

2.1.1 Embargo Periods

An embargo period is the period of time that the company has to fix the bug before it is released publicly. The length of time depends on the difficulty of the bug to be fixed, a longer period could be given if the bug would be hard to fix. The length of the embargo period is always up for debate and is always discussed with the discoverer. An aggressive embargo can be good for getting the fix out quick due to the time pressure. However, there is always the chance that the fix will be faulty itself when it is released which makes matters worse for the company. A longer embargo may allow time for a well-made fix, but it also allows more time for someone else to discover the bug and the more people who discover the bug, the more likely it is to be exploited, released publicly or sold to malicious people or governments (Allen, 2018). Post embargo transparency requires that any decisions that the company made regarding the vulnerability or bug are publicly released. (Linux Foundation, 2015)

Google Project Zero takes a different view on embargo periods and will always set a 90-day embargo period for any bugs they find. Clearly, for some bugs this is a reasonable amount of time but for others it can cause issues. At the end of the 90-days, no matter what state the fix is in, Google Project Zero will release it publicly on their blog (Allen, 2018). An example of this is a recent bug in Microsoft Edge's Just-In-Time compiler used to load browsers. Google Project Zero released the information about this bug on their blog before Microsoft had created a working fix, despite giving them an extra 14 days to make the fix. In this case, the 90-day embargo (which was lengthened to 104 days) was not long enough for a fix to be created and thus Microsoft had an exploitable bug in the edge browser which was able to be exploited by anyone who reads the Google Project Zero blog (Sharwood, 2018).

For Spectre and Meltdown, Google Project Zero gave a longer embargo than 90-days because Intel realised it was going to be very difficult to mitigate effectively. Originally this was not the case and Google Project Zero were only going to give 90-days. However, the head of Intel went to the head of Google and persuaded them to change the length of the embargo period.

2.1.2 Choosing who to tell

Another choice that a company has to make when a bug is discovered is who should be told about it. It should be clear that all of the companies that are affected by the bug should be told about it so that they can create fixes for their software. This doesn't apply when the bug is specific to one company. However, if it applies to more than one company, the company who is originally informed about it must make the decision on who to inform. During the embargo period the bug must remain hidden from public view so all of the companies who are told about it must be under NDA (Non-Disclosure Agreement).

In the case of Spectre and Meltdown, Intel were the first company to be told about the bug by Google Project Zero, because it was their processors that had Proof of Concept of being exploited. Intel, in turn, appears to have decided to tell major chip and cloud service providers including Arm, AMD and Microsoft. Since Spectre and Meltdown affect so many companies, Intel decided to only tell some of them. (Allen, 2018)

2.2 Zero Day Disclosure

Zero day disclosure is when the person who discovers a bug goes public straight away without warning the company involved (Allen, 2018). This can cause many issues for

companies because it creates panic among users and also makes it more likely to be exploited by malicious people.

2.2.1 Reasons for Zero Day Disclosure

Most of the time, people do zero day disclosure because they want to get something out of it. A common reason is that they want the fame and glory that is associated with discovering a bug first. However, it is possible to prove that you found the bug first without zero day disclosure. Often, people will tweet a hash of the description of the bug so that when it goes public they have proof that they knew about it before.

Some people use zero day disclosure or very short embargo periods because they stand to gain from it. A recent example of this are the vulnerabilities discovered in AMD chips. The person who discovered them, gave AMD a 1-day embargo period before setting up a website with all of the information on the vulnerabilities. (CTS Labs, 2018)

3. Evaluating Effectiveness

Clearly, the current method of dealing with security vulnerabilities has a few issues that were made clear by Spectre and Meltdown. The main issues are:

- Are long embargo periods worthwhile?
- How should a company decide who to tell and who not to tell?

The debate within the industry is whether this method should change and if so, how great of a change should occur.

Customers would like bug free software and often dislike hearing about bugs when they become public. At the same time, customers feel entitled to know when there is a bug in the software, because it could directly affect them. This fickle nature of customers makes knowing the ‘right’ way of dealing with bugs and vulnerabilities very difficult for the technology industry.

3.1 Argument for Zero Day Disclosure

Spectre and Meltdown brought responsible disclosure as a method of dealing with vulnerabilities into question. Many people within the industry felt that they needed to know about these vulnerabilities from day one, not 6 months later. This is especially relevant since Intel only decided to tell some of the bigger companies about it, not all of the smaller companies. Was it fair to not tell a lot of companies about these vulnerabilities even though they were also affected in the same way? It comes down to the fact that there is no fair way of writing a pre-disclosure list without including everyone on it and therefore removing the requirement of having a list in the first place. Choosing which companies to tell and not to tell could be seen as ‘playing God’. The companies who are not informed of the vulnerability get a nasty surprise when it goes public and they have to fix it very quickly under public scrutiny. Therefore, the fairest thing to do is to tell everyone about the vulnerability at the same time.

Another advantage to zero day disclosure is that it forces the hand of the company to fix the bug or mitigate the vulnerability. The company might want to keep the bug or vulnerability secret so that they don't lose credibility for having bad or insecure software. An example of this is the fact that Intel's share prices dropped significantly after Meltdown went public in January (Priday, 2018). As far as users are concerned, if the bug or vulnerability isn't public then it doesn't exist so they believe that the software is safe. This means that the company doesn't necessarily have to fix the bug if they don't want to. This, in turn, is good for malicious users who know about the bug or vulnerability and want to exploit it. Since the company isn't fixing the bug, the malicious user can selectively exploit it without any issues. However, the more people that know about the bug, the more likely that it will go public. (Dunlap, 2018)

3.2 Argument for Responsible Disclosure

As with anything, there is always a balance of how long the embargo period should be and when it should go public. Some companies might have incentives to never publicly disclose the bug such as avoiding financial loss. Most companies think that having an embargo period for creating a fix is a good idea because it will shorten the period of time that the customers are publicly vulnerable. While the bug or vulnerability is not public then the malicious users will be being very careful how much they exploit it so as not to draw attention to it. However, when it goes public, the malicious users know that it will be fixed so they try to exploit as many people as possible. Therefore, having an embargo period decreases the likelihood of the bug or vulnerability being exploited a lot in the wild. (Dunlap, 2018)

3.3 Evaluation of Spectre and Meltdown

For vulnerabilities as complicated and difficult to fix as Spectre and Meltdown, a long embargo period seemed like the best option so that the companies involved could create mitigations. However, by the time that patches were released, there were still many bugs within the patches that lead to problems for the users. Many reported their computers freezing and having problems with their anti-virus software being interfered with by the patch (Walden, et al., 2018). For example, Intel tried to release some new microcode[1] to mitigate the problems, but it was broken so they had to retract it and spend more time fixing it. To not have created fully working patches by the end of a long embargo is surely a fail on the part of the companies. Would a longer embargo have helped creating more complete patches? (Shenoy, 2018)

Another issue with how Spectre and Meltdown were dealt with was the fact that it leaked one week early. Whilst it was only one week early, and it didn't greatly impact the companies' response, the fact that it leaked is serious. Leaking of bugs is rare because since everyone is under NDA people have a good reason to not talk about it. In general, people will either keep it secret or leak it straight away. Sometimes, companies do not make it onto pre-disclosure lists because they refuse to be under NDA. For example, the maintainer of OpenBSD, an

[1] Microcode is the low-level code that runs on the processor itself.

open source operating system, appears to have been not informed about Spectre and Meltdown despite being directly affected. This could be due his previously expressed views and would mean that he only found out about Spectre and Meltdown when they went public so could only then start creating patches for his system (Allen, 2018). In a recent live stream, he showed his distaste at not being included in Intel's NDA group, but is this surprising when he obviously has such strong views on the topic. (De Raadt, 2018) Therefore, it cannot be based on the fact that it was a long embargo period.

Conclusion

Cyber Security is no longer just relevant to the technology industry and thus there needs to be a change in the way security vulnerabilities are handled. Security vulnerabilities now affect most industries as well as governments and state infrastructures such as the NHS. This means that cyber-crime is a bigger issue than ever and it is especially important that when bugs and vulnerabilities are discovered they are handled correctly and with care in order to limit the damage done. As vulnerabilities can also impact company performance (stock prices, valuations etc.) it is also important that the disclosure process is equitable as possible, although arguably this should be a secondary consideration, with protection of public and private data being paramount. Responsible disclosure may not seem the fairest way of handling vulnerabilities and bugs, but it works to keep as many people as possible safe from exploitation.

Spectre and Meltdown highlighted deficiencies in the current process of dealing with vulnerabilities, but this does not mean that it was handled wrong. It is a good example to look back on and think about what changes can be made to the current system to prevent something like that happening again.

Bibliography

Allen, M., 2018. *Spectre and Meltdown* [Interview] (21 February 2018).

Bright, P., 2018. *"Meltdown" and "Spectre": Every modern processor has unfixable security flaws.* [Online]
Available at: https://arstechnica.com/gadgets/2018/01/meltdown-and-spectre-every-modern-processor-has-unfixable-security-flaws/
[Accessed 19th January 2018].

Bright, P., 2018. *What's behind the Intel design flaw forcing numerous patches?.* [Online]
Available at: https://arstechnica.com/gadgets/2018/01/whats-behind-the-intel-design-flaw-forcing-numerous-patches/
[Accessed 19 January 2018].

CTS Labs, 2018. *Severe Security Advisory.* [Online]
Available at: https://amdflaws.com/
[Accessed 23rd March 2018].

De Raadt, T., 2018. *Speculating About Intel.* Ottawa: BSDCan.

Dunlap, G., 2018. *Dealing with Security Vulnerabilties* [Interview] (14 March 2018).

Fabio, A., 2018. *Spectre and Meltdown: How Cache Works.* [Online]
Available at: https://hackaday.com/2018/01/15/spectre-and-meltdown-how-cache-works/
[Accessed 20th April 2018].

Horn, J., 2018. *Reading privileged memory with a side-channel.* [Online]
Available at: https://googleprojectzero.blogspot.co.uk/2018/01/
[Accessed 20th April 2018].

Kocher, P. et al., 2018. *Meltdown and Spectre.* [Online]
Available at: https://spectreattack.com/
[Accessed 17 January 2018].

Kocher, P. et al., 2018. *Meltdown and Spectre.* [Online]
Available at: https://spectreattack.com/
[Accessed 18 January 2018].

Linux Foundation, 2015. *Xen Security Problem Response Process.* [Online]
Available at: https://www.xenproject.org/security-policy.html
[Accessed 2nd March 2018].

Priday, R., 2018. *Intel top brass smacked with sueball for keeping schtum about chip flaws.* [Online]
Available at: https://www.theregister.co.uk/2018/01/11/intel_ceo_cfo_sued_spectre_meltdown/
[Accessed 27 March 2018].

Sharwood, S., 2018. *Google reveals Edge bug that Microsoft has had trouble fixing.* [Online]
Available at: https://www.theregister.co.uk/2018/02/20/google_reveals_edge_bug_that_microsoft_has_had_trouble_fixing/
[Accessed 21 February 2018].

Shenoy, N., 2018. *Intel Security Issue Update: Addressing Reboot Issues.* [Online]
Available at: https://newsroom.intel.com/news/intel-security-issue-update-addressing-reboot-issues/
[Accessed 13 June 2018].

Upton, E., 2018. *Why Raspberry Pi isn't vulnerable to Spectre or Meltdown.* [Online]
Available at: https://www.raspberrypi.org/blog/why-raspberry-pi-isnt-vulnerable-to-spectre-or-meltdown/
[Accessed 30th January 2018].

Walden, G., Latta, R. E., Blackburn, M. & Gregg, H., 2018. *Letter to Tech Companies on Meltdown and Spectre Vulnerabilities.* [Online]
Available at: https://energycommerce.house.gov/wp-content/uploads/2018/01/Meltdown-Spectre-Letters.pdf