Data Sampling on MDS-resistant 10th Generation Intel Core (Ice Lake)

Daniel Moghimi

Worcester Polytechnic Institute, Worcester, MA, USA

Abstract

Microarchitectural Data Sampling (MDS) is a set of hardware vulnerabilities in Intel CPUs that allows an attacker to leak bytes of data from memory loads and stores across various security boundaries. On affected CPUs, some of these vulnerabilities were patched via microcode updates. Additionally, Intel announced that the newest microarchitectures, namely Cascade Lake and Ice Lake, were not affected by MDS. While Cascade Lake turned out to be vulnerable to the ZombieLoad v2 MDS attack (also known as TAA), Ice Lake was not affected by this attack.

In this technical report, we show a variant of MSBDS (CVE-2018-12126), an MDS attack, also known as Fallout, that works on Ice Lake CPUs. This variant was automatically synthesized using Transynther, a tool to find new variants of Meltdown-type attacks. Based on the findings of Transynther, we analyze different microcodes regarding this issue, showing that only microcode versions after January 2020 prevent exploitation of the vulnerability. These results show that Transynther is a valuable tool to find new variants, and also to test for regressions possibly introduced with microcode updates.

1 Introduction

In May 2019, a class of CPU attacks, under the umbrella of microarchitectural data sampling (MDS), showed that Meltdown-style attacks are still possible on Intel Core microarchitecture [1, 11, 12, 14]. MDS fundamentally exploits the same root cause as Meltdown [9] and Foreshadow [12]. MDS attacks showed that Meltdown-type attacks are not limited to the L1 cache but that they can generally leak stale data from various internal CPU buffers [10]. In contrast to Meltdown [9] or Foreshadow [12], the attacker has only limited control of what data can be leaked, as these internal buffers only contain *memory loads* and *stores* from different execution contexts, regardless of any architectural security boundary.

Fallout [1], or microarchitectural store buffer data sampling (MSBDS), is an MDS variant leaking recent data stores from the store buffer. MSBDS has been shown in attacks on the kernel space as well as on SGX [1]. In addition to the leakage component, MSBDS can also be turned around easily to transiently inject values into a victim, an attack technique known as Load Value Injection (LVI) [13].

To mitigate these vulnerabilities, Intel released software workarounds and microcode updates for affected CPUs [4]. One of the most recent microarchitectures, Ice Lake, is reported to be unaffected by MDS attacks [5, 7]. Intel has also explicitly listed Ice Lake processors as not being vulnerable to LVI-SB [13], which exploits MSBDS for Load Value Injection [6].

To better analyze the MDS vulnerabilities, and potentially find new variants, Moghimi et al. [10] presented Transynther. Transynther mutates the basic primitives of existing Meltdown-type attacks to automatically generate and evaluate new subvariants. In the original publication [10], Transynther did not only reproduce existing attacks but also discovered a ZombieLoad [11] variant (Medusa) that targets the write-combining mechanisms. In this report, we show that Transynther also found a variant of MSBDS that works on the allegedly unaffected Ice Lake microarchitecture.

Responsible Disclosure. We have reported this finding to the Intel Product Security Incident Response Team (iPSIRT) on March, 27, 2020. On May 5 2020, iPSIRT completed the triage of our proof of concept, and they replied that the mitigation for MSBDS was not ported correctly to the Ice Lake microarchitecture. As a result, Ice Lake required a microcode patch, which they developed as part of their late November 2019 microcode version 0x5C. In the May 2020 update of Intel's specification update for the 10th Generation Intel Core Processor Family, a new errata, 057, has been added. This errata mentions that the `MDS_NO` bit in `IA32_ARCH_CAPABILITIES` control registers were incorrectly set [8]. Intel requested an embargo until July 13, 2020, to allow enough time for OEMs and their customers to deploy these patches. The report was awarded a bug bounty.

2 Fallout (MSBDS) on Ice Lake

In this section, we present the MSBDS variant automatically synthesized by Transynther [10]. This MSBDS variant is the only known MDS attack that works on the Ice Lake microarchitecture. We analyze the leakage rates of this variant, and also the affected microcode versions.

Discovery. We ran Transynther on a Core i5-1035G1 CPU with the latest microcode that is shipped with Ubuntu 18.04, version 0x48. After running for about 5000 iterations, Transynther reported store-to-load-forwarding leakage due to 4K aliasing of store addresses with a faulty memory load. This behavior was initially exploited in Fallout to bypass KASLR and leak cryptographic keys from the kernel space [1].

Based on the generated proof-of-concept, we produced a minimal working example to analyze the auto-generated proof of concept that triggers MSBDS manually.

Classification. We noticed that MSBDS on Ice Lake only works with memory load operations that suffer a permission failure due to accessing privileged memory (cleared US bit), or accessing a memory page with wrong protection keys [2]. Based on the systematization of Canella et al. [2], and to the best of our knowledge, we conclude that Ice Lake is only vulnerable to Meltdown-US or Meltdown-MPK attacks.

Modified Cache State. One of the observations from Transynther is that the leakage rate increases significantly if the target store address is flushed from the cache. We observe the same behavior for other instructions that modify the cache state. Specifically, executing `lock incl` on the store address leads to an even higher leakage rate than flushing the store address using `clflush`. Listing 1 shows our simplified proof of concept that demonstrates MSBDS on the Ice Lake microarchitecture. Uncommenting Line 16 or 17 modifies the cache state of the store address, resulting in a faster leakage. If we do not modify the cache state, we observe a very slow leakage of approximately 1 B/s. As we can see in Table 1, with approximately 750 B/s, the leakage rate is significantly higher when using `lock inc` instruction (Line 19) to modify the cache state.

We also tested the proof-of-concept on various microcode versions on the Ice Lake CPU. As not all issued microcodes are officially available by CPU vendors, we used a crowd-sourced repository of available microcodes [3]. For our analysis, we applied 10 different compatible microcode versions, i.e., microcodes that match the CPUID of our target CPU. As we can see in Table 1, all Intel microcodes until mid-November are vulnerable to MSBDS, although Ice Lake should fundamentally be resistant against MDS attacks.

```
1  unsigned int c = 0;
2  while(cond){
3      if (! setjmp(trycatch_buf ))
4          s_faulty_load ();
5
6      for (size_t i = 1; i < 256; i++)
7          if (flush_reload ((uint8_t *)&oracles + i*PAGE_SIZE))
8              if (i == 0x41) c++;
9  }
10
11 /** asm.S **/
12     .global s_faulty_load
13 s_faulty_load :
14     lea address_normal+0x4822, %r14      // Store address
15     lea address_supervisor +0x822, %r15  // Load address (4K alias w/ store )
16 //    clflush (%r14)                     // Uncomment to modify cache state
17 //    lock;incl (%r14)                   // Uncomment to modify cache state
18     movb $0x41, (%r14)                   // Store
19     movb (%r15), %al                     // Faulty Load
20     lea oracles , %r13                   // Encode
21     and $0xff, %rax
22     shlq $12, %rax
23     movb (%r13,%rax,1), %al
24     ret
```

Listing 1: Proof of concept for MSBDS on IceLake.

Table 1: List of tested microcodes on a Core i5-1035G1 CPU. For vulnerable microcodes, the leakage rate is much higher if the target store is in a modified state, as it is shown by using cache flush and modification instructions. We ran each experiment for two minutes.

MC Version	MC Date	Vulnerable	Leakage (bytes/s)		
			clflush	*lock inc*	Unmodified
0x32 (stock)	2019-07-05	✓	577.87	754.99	1.58
0x36	2019-07-18	✓	148.24	529.84	0.62
0x46	2019-09-05	✓	130.15	695.80	0.11
0x48	2019-09-12	✓	271.69	620.07	0.59
0x50	2019-10-27	✓	96.54	542.10	0.25
0x56	2019-11-05	✓	145.46	751.40	0.08
0x5a	2019-11-19	✓	532.40	645.32	0.70
0x66	2020-01-09	✗	0	0	0
0x70	2020-02-17	✗	0	0	0
0x82	2020-04-22	✗	0	0	0
0x86	2020-05-05	✗	0	0	0

3 Conclusion

Our report shows that Intel Ice lake client processors with early firmware version are vulnerable to MDS attacks. In discussions with Intel engineers we were told that MSBDS mitigations are present in hardware, but where disabled in early versions of these processors. It is crucial for OEMs and users to apply these latest microcode updates to enable protection against MDS attacks.

More importantly, our analysis using Transynther highlights the importance of automated vulnerability testing and analysis for hardware and microarchitectural vulnerabilities. Although Transynther is an academic prototype, it stills proved to be a valuable tool for automated testing of hardware. The newly reported MSBDS vulnerability would not have gone unnoticed on Ice Lake, if the earlier prototypes of the hardware were tested using such tools. Furthermore, OEMs could have tested these vulnerabilities before shipping consumer laptop with a vulnerable microcode update.

References

[1] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Marina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, and Yuval Yarom. Fallout: Leaking Data on Meltdown-resistant CPUs. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[2] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. A Systematic Evaluation of Transient Execution Attacks and Defenses. In *USENIX Security Symposium*, 2019.

[3] Github. CPUMicrocodes: Intel, AMD, VIA & Freescale CPU Microcode Repositories. https://github.com/platomav/CPUMicrocodes, May 2020.

[4] Intel. Deep Dive: Intel Analysis of Microarchitectural Data Sampling. https://intel.ly/3dLmAq0. Accessed: June 21, 2020.

[5] Intel. Processors Affected by Microarchitectural Data Sampling. https://bit.ly/2D1j7qL. Accessed: June 21, 2020.

[6] Intel. Processors Affected: Load Value Injection. https://bit.ly/3gic0sm. Accessed: May 19, 2020.

[7] Intel. Side Channel Mitigation by Product CPU Model. https://bit.ly/2C1sSV9. Accessed: May 23, 2020.

[8] Intel. 10th Generation Intel Core Processor Families Specification Update. https://intel.ly/31x6BcJ, May 2020.

[9] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In *USENIX Security Symposium*, 2018.

[10] Daniel Moghimi, Moritz Lipp, Berk Sunar, and Michael Schwarz. Medusa: Microarchitectural data leakage via automated attack synthesis. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[11] Michael Schwarz, Moritz Lipp, Daniel Moghimi, Jo Van Bulck, Julian Stecklina, Thomas Prescher, and Daniel Gruss. ZombieLoad: Cross-Privilege-Boundary Data Sampling. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[12] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *USENIX Security Symposium*, 2018.

[13] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. Lvi: Hijacking transient execution through microarchitectural load value injection. In *41th IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[14] Stephan van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. RIDL: Rogue In-flight Data Load. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.