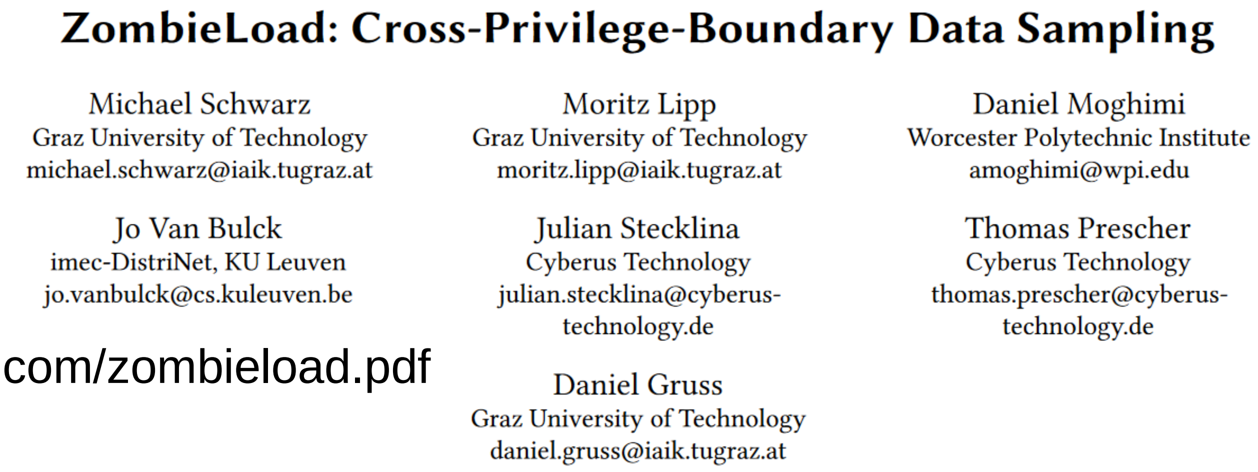
ZombieLoad

Toby Nelson

Exploit Discovered and Published

- In May 2019, an international team of researches published a disclosure and report detailing a new hyperthreading attack:



ZombieLoad: Cross-Privilege-Boundary Data Sampling

Michael Schwarz
Graz University of Technology
michael.schwarz@iaik.tugraz.at

Moritz Lipp
Graz University of Technology
moritz.lipp@iaik.tugraz.at

Daniel Moghimi
Worcester Polytechnic Institute
amoghimi@wpi.edu

Jo Van Bulck
imec-DistriNet, KU Leuven
jo.vanbulck@cs.kuleuven.be

Julian Stecklina
Cyberus Technology
julian.stecklina@cyberus-technology.de

Thomas Prescher
Cyberus Technology
thomas.prescher@cyberus-technology.de

Daniel Gruss
Graz University of Technology
daniel.gruss@iaik.tugraz.at

https://zombieloadattack.com/zombieload.pdf

Introduction (Microarchitecture)

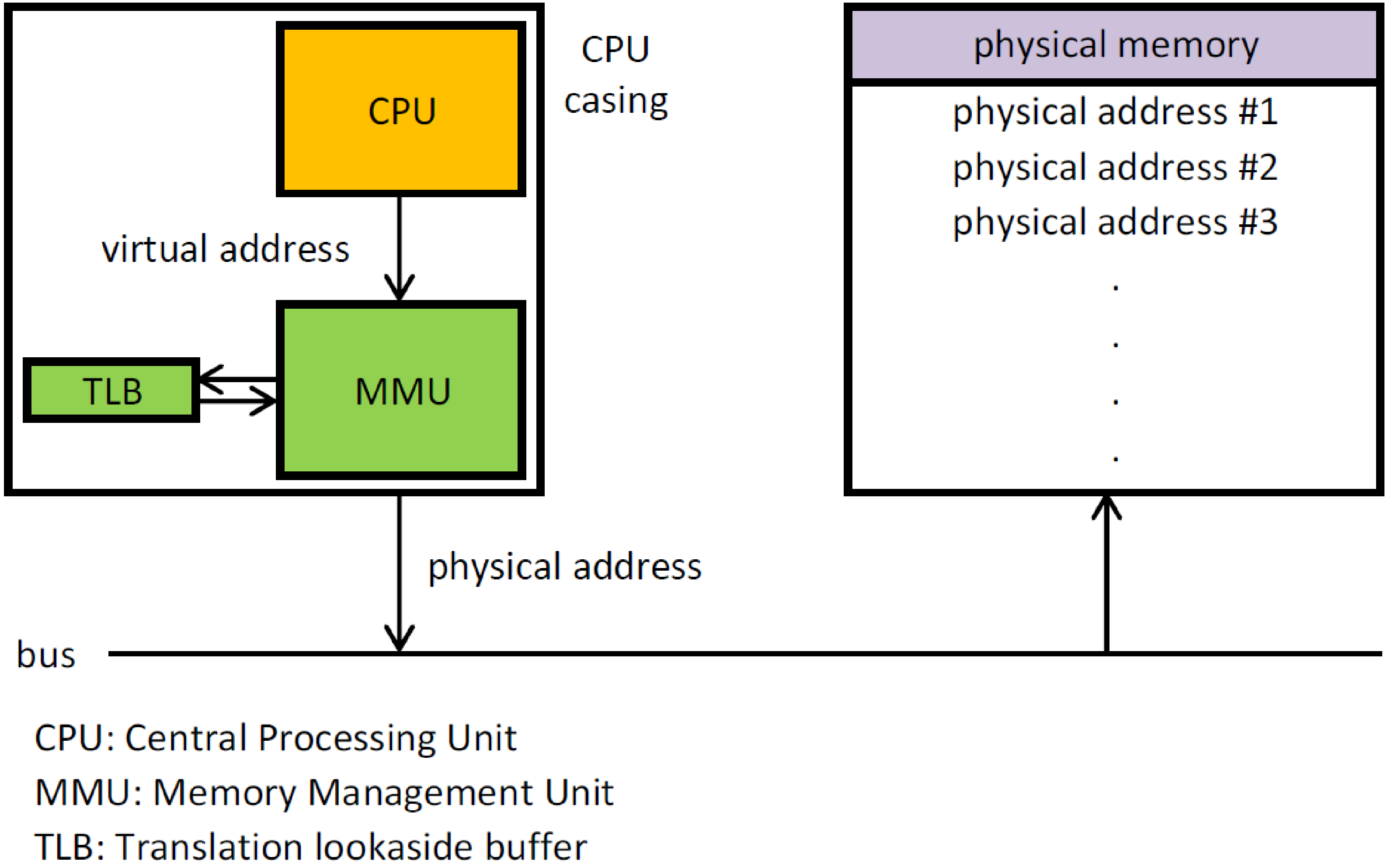
- Modern CPUs will often complete instructions out of order (for example branch prediction, or computing unrelated values while waiting on a cache miss)
- To achieve this, instructions are executed until an exception is raised, after which the cpu rolls back all following instructions.
- There were traditionally little security consideration over this as it was not considered a useful attack vector prior to Meltdown

Intro (Microarchitecture)

- One microarchitecture optimization is to complete independent instructions following a load while you wait for the load to come back from a cache miss.
- To accomplish this, modern Intel cpus use a load and store buffer, collectively, the memory order buffer, as well as a line fill buffer where retrieved or stored values are cached.
- The line fill buffer is never intended to be accessible outside the cpu.

A Data Load

- When a load operation is dispatched, an entry is created in the load buffer and the reorder buffer.
- From the linear address provided by the CPU, the upper 36 bits are handed off to the MMU to translate to physical memory address. The lower 12 bits just index the cache set.



36 bits are handed to the mmu to retrieve the physical address, when retrieved from TLB it is immediate, otherwise the page table must be walked.

https://en.wikipedia.org/wiki/Memory_management_unit#/media/File:MMU_principle_updated.png

A Cache Miss:

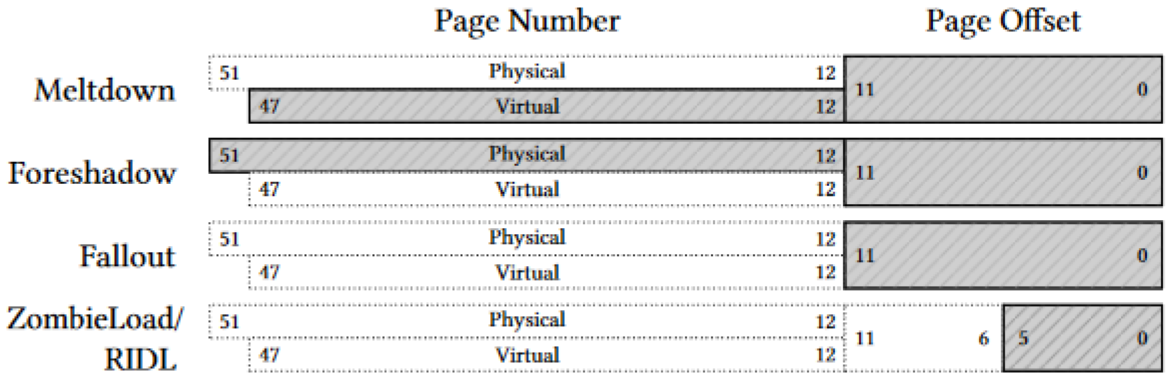
- When the physical location is found, if it is not in l1 cache, the load buffer entry remains, and a fill-buffer reservation is made. When the data returns to the fill buffer, the load buffer and fill buffer entries can be cleared.
- Further, if there is a store that does not reference an l1 value, it is also stored in fill-buffer and gains an entry in store buffer.

Hypothesized Source of the Leak

- When the load instruction encounters a fault, a microcode assist flushes the instruction pipeline
- This flush allows all pending instructions to complete
- The researchers hypothesize that to avoid slowdown during faults, the instructions are allowed to match fill buffer entries with only a partial match to the physical address. This can result to load operations being completed with stale entries. Thus, they speculate the fault is not architectural (i.e. it is possibly to build a hyperthreaded cpu without this leak) but implementation specific.
- The fill buffer is shared among both logical cores of a hyperthreaded core

Microarchitectural Data Sampling

- Because the attack is believed to rely on stale cache lines sitting in the fill buffer, the only control the attacker has is the byte offset in the cache line (functionally a 6 bit address)



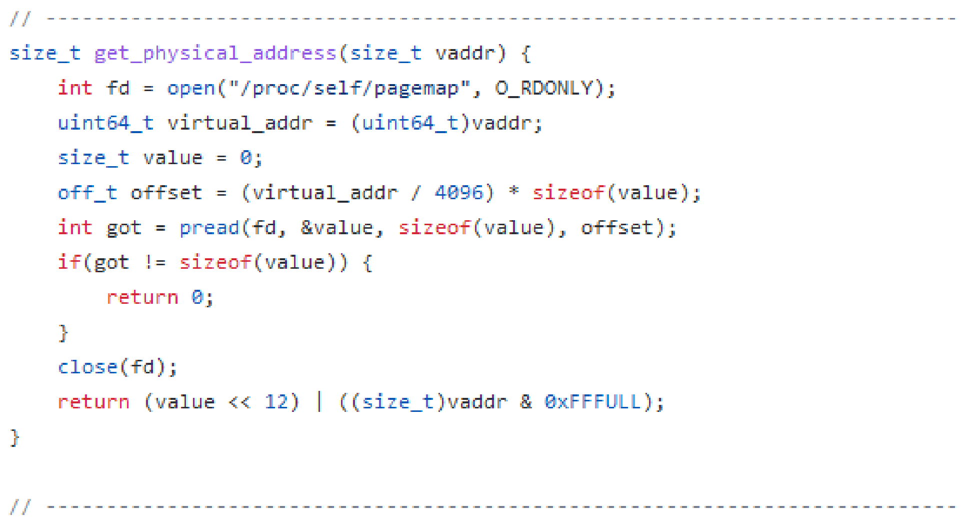
https:// zombieloadattack.com/ zombieload.pdf

MDS (cont)

- Due to the limited targeting capabilities, ZombieLoad can be seen as a more traditional side-channel attack vector compared to Meltdown-style attacks. The attacker can read values and track the time, but they must find a way to relate those to secret values.
- For example, the researchers were able to extract secret keys from a ‘side-channel resistant’ AES implementation in OpenSSL 3.0. To accomplish this, they monitored a cache line they knew the implementation would set, which does not leak secrets directly but helps to synchronize the attack. Repeated instances of the attack are necessary to control for noise and imprecision in this process. In implementations where there are not measurable signals near the sensitive information, more sophisticated methods using timestamps may be necessary.

Vulnerability

- At the date of publication, the researchers described three variants:
  - Variant 1 : This relies on being able to associate a kernel address with a userspace address. One method has been addressed in the Linux kernel 4.15 and backported to 4.4.110, called Kernel Page Table isolation. When the attacker is privileged, both linux and windows are vulnerable to this attack. If they are not, only versions of linux without KPTI available/enabled are vulnerable.



```
// ---------------------------------------------------------------------------
size_t get_physical_address(size_t vaddr) {
    int fd = open("/proc/self/pagemap", O_RDONLY);
    uint64_t virtual_addr = (uint64_t)vaddr;
    size_t value = 0;
    off_t offset = (virtual_addr / 4096) * sizeof(value);
    int got = pread(fd, &value, sizeof(value), offset);
    if(got != sizeof(value)) {
        return 0;
    }
    close(fd);
    return (value << 12) | ((size_t)vaddr & 0xFFFULL);
}

// ---------------------------------------------------------------------------
```

https://github.com/IAIK/ZombieLoad/blob/master/attacker/variant1_linux/cacheutils.h

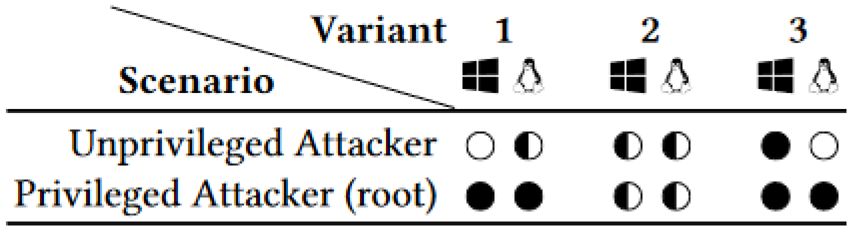
Vulnerability (cont)

- The variant the researchers dubbed Variant 2, relies on an instruction set called Intel TSX which is an extension supported by recent Intel CPUs. This variant does not require privilege and works across platforms, but it only leaks the bytes corresponding the the load/store command which left behind the address, not the entire cache line.

Vulnerability (cont)

- Finally, for variant 3, the access bits of files can be cleared to trigger a microcode which leaks the data. For Linux, this exploit requires privilege escalation, but at the time of publication the researchers found Windows seemed to do this periodically automatically meaning no elevation of privilege was necessary.

Table 2: **Overview of different variants to induce zombie loads in different scenarios.**



Variant / Scenario	1		2		3	
	🪟	🐧	🪟	🐧	🪟	🐧
Unprivileged Attacker	○	◐	◐	◐	●	○
Privileged Attacker (root)	●	●	◐	◐	●	●

Symbols indicate whether a variant can be used in the corresponding attack scenario (●), can be used depending on the hardware configuration as discussed in Section 5.1 (◐), or cannot be used (○).

https://zombieloadattack.com/zombieload.pd

Suggested Countermeasures

- “Co-Scheduling” : The scheduler could ensure processes running in a kernel context do not share a physical core with processes running in a userspace context. Similar measures would be applied for virtual machine vs host machine and user vs user

Countermeasures (cont).

- Flushing Buffers: This is covered to an extent under existing mitigations for other microarchitecture attacks. Processors with updated microcode only leak ~0.1b/s (provided the necessary instruction is called). However, alternatives for cpus which have not updated microcode must rely on code sequences which are not reliable, for example the the i7 8650 was found to still be vulnerable with this mitigation.

Countermeasures (cont)

- For software, loading a secret directly from memory should be avoided. If it is loaded in separate parts and combined in registers, it makes it less likely for the attacker to recover and distinguish the complete secret from noise.

Countermeasures (cont).

- Selective Feature Deactivation: The variants relied on features such as Intel TSX instruction set extension, ensuring access bits are always set properly by the Kernel, and disabling virtualization in the bios so an attacker can not create a virtual machine for easy access to a kernel-user page alias.

Aftermath

- Intel described the exploit as ‘low to medium severity’ as ‘Exploiting the MDS vulnerabilities outside the controlled conditions of a research environment is a complex undertaking.’
- The researchers later responded:

  ‘On January 27th, 2020, an embargo ended showing that the mitigations against MDS attacks released in May 2019 are insufficient. With L1D Eviction Sampling, an attacker can still mount ZombieLoad to leak data that is being evicted from the L1D cache.

  We disclosed this issue to Intel on May 16th, 2019. However, as microcode updates containing the necessary fixes are not yet available, we are not releasing any proof-of-concept code.’
- This seems to be referring to the 0.1b/s leak they described after microcode updates are applied and used. ‘We can still observe leakage from kernel values accessed on the same logical core. However, the leakage rate drops from multiple kilobytes per second to less than 0.1 B/s. Our hypothesis is that we can leak data which is evicted from L1 to L2 after issuing the VERW instruction. As the VERW instruction does not flush dirty L1-cache lines, these can be easily leaked if the attacker partly evicts the L1.’

Aftermath(cont)

- In an email to Wired in 2020, Cristiano Giuffrida, one of the researchers to discover an a closely related microarchitecture data sampling attack, RIDL, had this to say "These issues aren't trivial to fix. But after eighteen months, they're still waiting for researchers to put together proofs-of-concept of every small variation of the attack for them? It’s amazing. We don’t know the inner workings of Intel's team. But it’s not a good look from the outside."
- I did not find any articles discussing real events of ZombieLoad being used in a black hat context.

References

- https://zombieloadattack.com/zombieload.pdf
- https://www.intel.com/content/www/us/en/architecture-and-technology/mds.html
- https://zombieloadattack.com/#faq
- https://www.wired.com/story/intel-zombieload-third-patch-speculative-execution/