Foreshadow Attack and its Variants

180050067 Neel Aryan Gupta
180050081 Pulkit Agrawal
180050111 Tathagat Verma

Introduction

- Computer system security fundamentally relies on memory isolation, like kernel address is protected from user access
- In 2018, Foreshadow Attack was publicly announced along with its variants and the mitigations for it
- Foreshadow is a practical software-only microarchitectural attack that allows overcoming memory isolation completely and decisively dismantles the security objectives of past SGX and VMM implementations
- This attack can be launched by unprivileged adversaries (i.e without root access)

Background

- **Speculative Execution:** Modern processors use branch predictor to guess the next instruction and execute it speculatively. If the prediction is correct, performance improves substantially, otherwise, the changes to the architectural state have to be rolled back.
- **Flush and Reload:** The Flush and Reload attack is used to deduce the byte which was accessed last by the victim as it would be in the cache. The attack is based on the fact that the time taken to retrieve data from L1 cache is drastically less than that of main memory.

Intel SGX

- Intel SGX is a set of instructions used for improving the security of application code and data.
- It creates a trusted execution environment(TEE) inside the memory that prevents users' sensitive data from being revealed or modified.
- SGX uses secure enclaves to protect information from processes running at higher privilege levels.
- Each enclave has its own private CPU & memory state which is only accessible from inside the enclave. It cannot be reached by any software running at any privilege level.
- Any attempt to directly access private pages from outside the enclave results in **abort page semantics**: reads return the value -1(0xff), and writes are ignored.

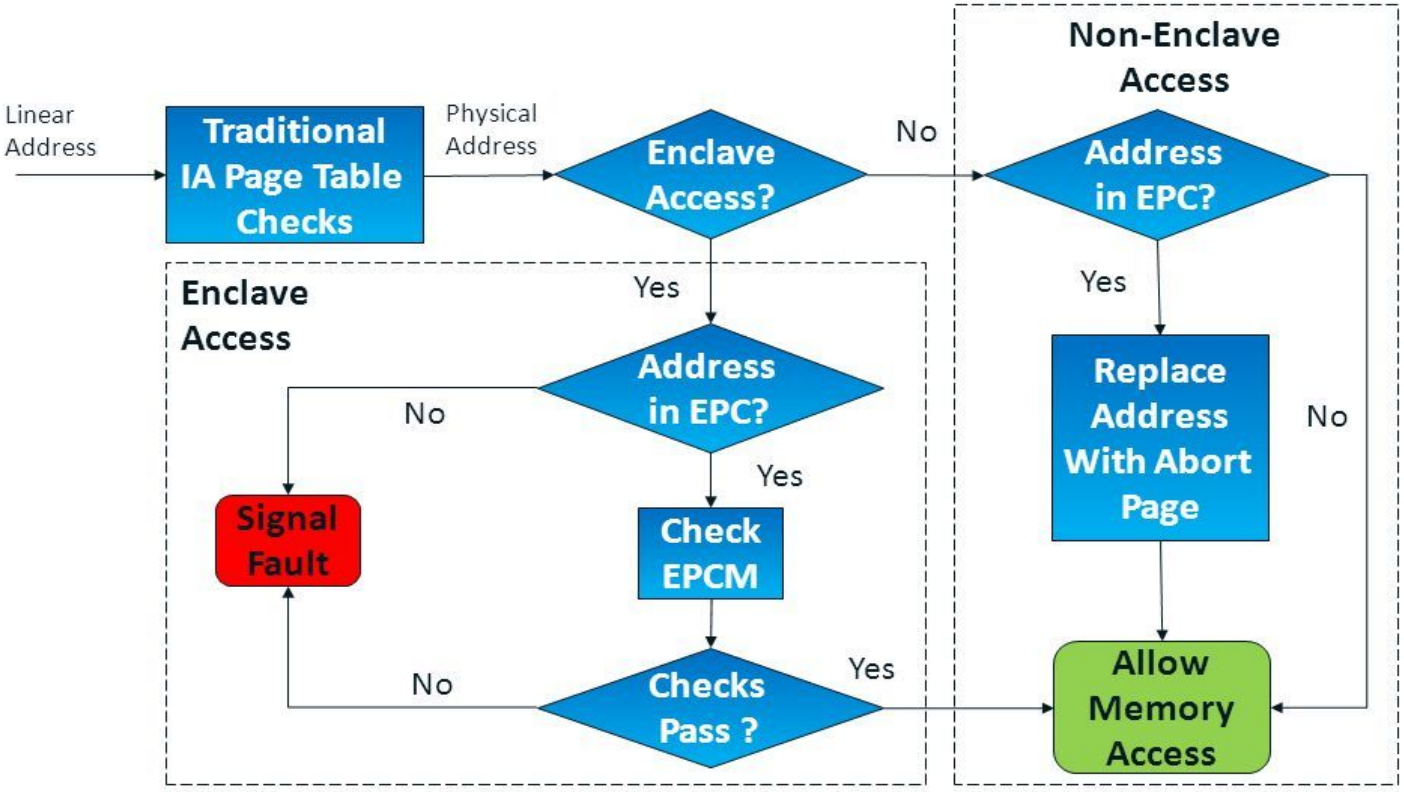
Meltdown Attack

- Foreshadow uses the Meltdown attack as a subroutine with some extra steps. So, here we discuss the basics of Meltdown.
- In Meltdown Attack, we try to read some kernel address from user space as shown in Line 3. This would raise an exception.

```
char probe_array[256 * STEP];
clflush(probe_array);
secret = *kernel_address;
access(probe_array[secret * STEP]);
```

- However, due to transient out-of-order execution employed by the modern processors, the contents of this dereferenced memory will be brought into the cache.
- When the exception is raised, all changes made as a part of this out-of-order execution is reversed except clearing it from cache
- Meltdown exploits this by recovering the secret value using FLUSH and RELOAD technique.

SGX Access Control



Foreshadow Attack

Why Meltdown Attack cannot work on SGX?

- Meltdown fails mainly due to the Abort Page Semantics employed by SGX, which does not result in a page fault because the enclave memory is never really accesses.
- Be it a transient instruction or not, if any inaccessible memory is tried to be accessed by an unprivileged process, SGX checks kick in, since the address being accessed lies in the ELRANGE but was not an enclave access.
- This results in getting -1 (0xFF) value instead of the actual value in the enclave memory.

Foreshadow Attack

Foreshadow builds on top of Meltdown:

- It exploits the present bit stored along with the virtual addresses in the page table of the process
- The present bit is used to denote if the page actually resides in the main memory(when the bit is set), or it needs to be fetched from the disk into memory
- The attacker unsets the present bits on the page where the secret resides

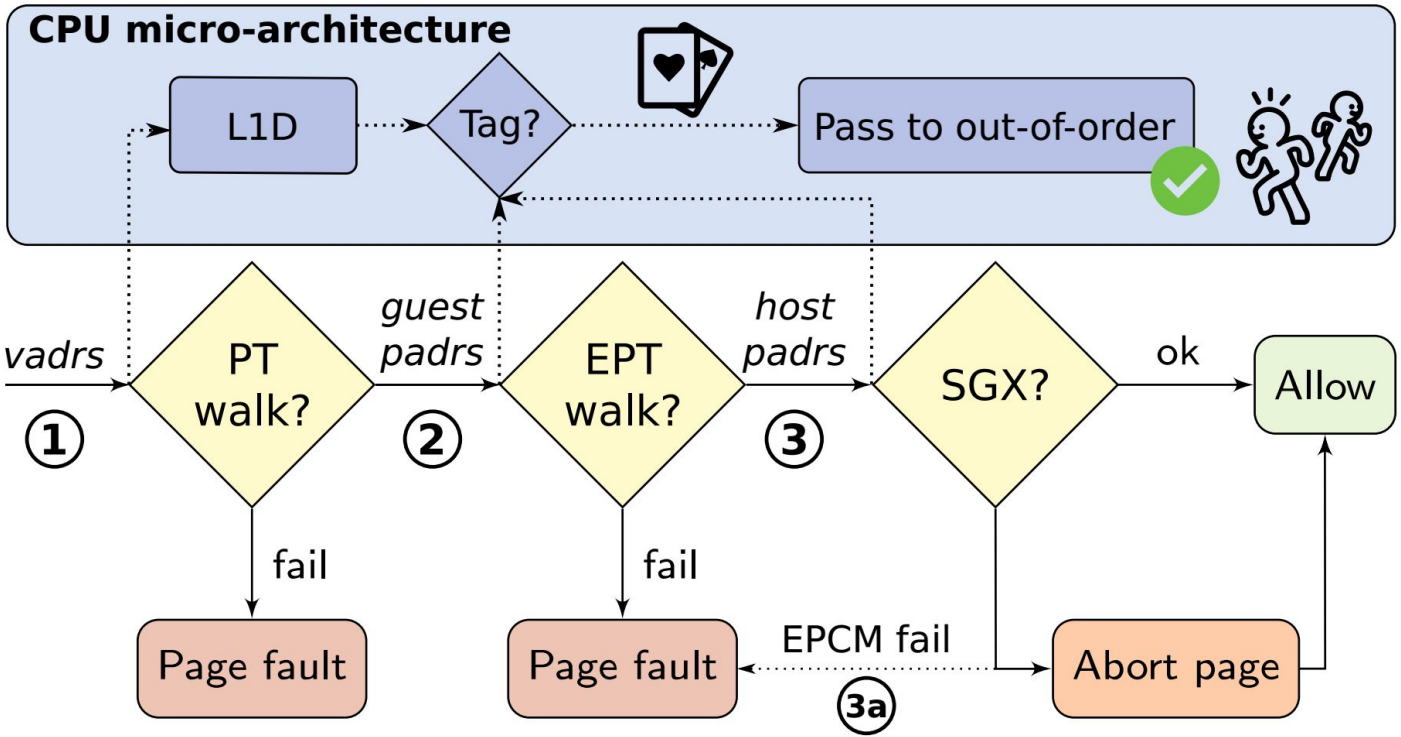
```
mprotect( secret_ptr & ~0xfff, 0x1000, PROT_NONE );
```

- The page table checks occur before the checks implemented by SGX and thus the page fault occurs without those SGX checks which essentially get skipped in the transient execution of the instructions. Hence, now the meltdown attack is applicable.

Significance of L1 cache

- Experimentally found out that enclave secrets do not reach transient execution stage when not present in L1 cache, making the presence of secret in L1 crucial.
- L1 cache pollution - the enclave secret gets evicted from L1 during the attack. Happens when there are many context switches and kernel code execution (eg. *mprotect*).
- Some measures to prevent enclave secret eviction from L1 :
  - Fault suppression - Intel TSX / high latency misprediction branch
  - Keeping secrets warm (Root) - *wbinvd* syscall / loop
  - Isolating cores (Root) - pin adversary to 1 core
- Bringing data into L1 cache (Root) - *eldu* syscall brings page into L1 cache as plaintext and the microcode implementation does not cause its eviction while exiting the enclave

Page Access Checks



Variants

- As an extension to the original Foreshadow (or Foreshadow-SGX) attack, the analysis made by the researchers at Intel lead to discovery of two more attacks as an extension of the Foreshadow attack :
  - Foreshadow-OS
  - Foreshadow-VMM
- These attacks were collectively called as Foreshadow-NG attacks, and dubbed by Intel as L1 Terminal Fault (L1TF), due to the fact the unauthorized enclave memory accesses could be served if the data resides in the L1 cache.
- Both these attacks follow the same basic principle, but differ in the attack scopes and consequences.

Foreshadow-OS

- The most important part of the vanilla Foreshadow was poisoning the Present bit of the pages.
- When the kernel OS swaps out a page from the DRAM to the disk, the present bit is cleared by itself, which in effect, makes the swapped out page suitable for the FS attack.
- The OS is free to use the remaining bits in the PTE of the swapped out page. If the PTE still contains the old physical address, the attacker can still leak the contents of that page. Even if the page is zeroed out, information can still be leaked at physical address 0.
- Note that the attacker has no control over which pages have been swapped out, but can simply wait for any page to be swapped out, to read its contents.

Foreshadow-VMM

- After performing the traditional page table checks, in case of VMM, Extended Page Table (EPT) is walked to convert the physical address of guest to that of host.
- The key vulnerability for this attack is that the terminal fault during the guest page table walk results in an early out condition in which the guest physical address is directly passed to L1 tag comparison, L1 cache being VIPT.
- The physical address therefore never undergoes EPT translation and an attacker VM on the victim machine can easily read any contents at any physical address of the host machine as the attacker can modify its guest page table PTE’s to point to arbitrary physical address of the host.

Demo

Mitigations

- For SGX, the L1 cache should be flushed every time we cross security boundaries. The microcode patch that performs this is controlled by the kernel parameter `l1tf`.
- For VMM, whenever hypervisor transfers control to guest VMs with the *vmenter* instruction, the hypervisor flushes the L1D cache first. The kernel parameter `kvm-intel.vmentry_l1d_flush` controls this microcode patch
- If HyperThreading is enabled, hypervisors must check that no thread is running on the sibling core in an untrusted VM since L1 cache is shared.
- For OS, PTE entries are 0'ed out while swapping out the page, hence pointing to the 0x00 physical address. Place a dummy page at 0x00 so that secret data is not compromised. Can zero out the page size from PDE and PDPTE so that large pages (2MB/1GB) don't require large dummy pages.

Challenges Faced

- Unavailability of old versions of VMM softwares for Ubuntu 20.04, without the microcode patches for L1TF. Therefore, unable to reproduce the Foreshadow-VMM attack effectively.
- The CPU's available to us, probably contain some hardware patches due to which we were unable to completely turn off the L1TF mitigations, despite the fact that the CPU's used do not contain any hardware patches, as suggested by Intel here.
- The popular vulnerability checker script available here, shows that the CPU microcode still mitigates L1TF.
- The leaked bytes in Foreshadow-SGX are always observed to be 0, which indicates that the L1D cache gets flushed upon the enclave exit.
- Moreover, the absence of TSX support leads to unstable and low accuracy in the results.

REFERENCES

- J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In Proceedings of the 27th USENIX Security Symposium. USENIX Association, August 2018.
- LIPP, M., SCHWARZ, M., GRUSS, D., PRESCHER, T., HAAS, W., MANGARD, S., KOCHER, P., GENKIN, D., YAROM, Y., AND HAMBURG, M. Meltdown. arXiv preprint arXiv:1801.01207 (2018).
- J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution. 2018.
- INTEL. Intel Analysis of Speculative Execution Side Channels, January 2018. Reference no. 336983-001.
- Intel. Description and mitigation overview for L1 Terminal Fault. August 2018.
- Proof of Concept experiment: https://github.com/jovanbulck/sgx-step

Thank You