Tutorial: Uncovering and Mitigating Side-Channel Leakage in Intel SGX Enclaves

Jo Van Bulck and Frank Piessens

imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium
{jo.vanbulck,frank.piessens}@cs.kuleuven.be

Abstract. The inclusion of the Software Guard eXtensions (SGX) in recent Intel processors has been broadly acclaimed for bringing strong hardware-enforced trusted computing guarantees to mass consumer devices, and for protecting end user data in an untrusted cloud environment. While SGX assumes a very strong attacker model and indeed even safeguards enclave secrets against a compromised operating system, recent research has demonstrated that considerable private data (e.g., full text and images, complete cryptographic keys) may still be reconstructed by monitoring subtle side-effects of the enclaved execution.
We argue that a systematic understanding of such side-channel leakage sources is essential for writing intrinsically secure enclave applications, and will be instrumental to the success of this new trusted execution technology. This tutorial and write-up therefore aims to bring a better understanding of current state-of-the-art side-channel attacks and defenses on Intel SGX platforms. Participants will learn how to extract data from elementary example applications, thereby recognizing how to avoid common pitfalls and information leakage sources in enclave development.

Keywords: Side-channel · Enclave · SGX · Tutorial.

1 Introduction

Trusted Execution Environments (TEEs), including Intel SGX, are a promising new technology supporting secure isolated execution of critical code in dedicated *enclaves* that are directly protected and measured by the processor itself. By excluding vast operating system and hypervisor code from the trusted computing base, TEEs establish a minimalist hardware root-of-trust where application developers solely rely on the correctness of the CPU and the implementation of their own enclaves. Enclaved execution hence holds the promise of enforcing strong security and privacy requirements for local and remote computations.

Modern processors unintendedly leak information about (enclaved) software running on top, however, and such traces in the microarchitectural CPU state can be abused to reconstruct application secrets through side-channel analysis. These attacks have received growing attention from the research community and significant understanding has been built up over the past decade. While information leakage from side-channels is generally limited to specific code or

data access patterns, recent work [11,10,4,8,5,9] has demonstrated significant side-channel amplification for enclaved execution. Ultimately, the disruptive real-world impact of side-channels became apparent when they were used as building blocks for the high-impact Meltdown, Spectre, and Foreshadow speculation attacks (where the latter completely erodes trust on unpatched Intel SGX platforms [7]).

Intel explicitly considers side-channels out of scope, clarifying that "it is the enclave developer's responsibility to address side-channel attack concerns" [2]. Unfortunately, we will show that adequately preventing side-channel leakage is particularly difficult — to the extent where even Intel's own vetted enclave entry code suffered from subtle yet dangerous side-channel vulnerabilities [3]. As such, we argue that side-channels cannot merely be considered out of scope for enclaved execution, but rather necessitate widespread developer education so as to establish a systematic understanding and awareness of different leakage sources. To support this cause, this tutorial and write-up present a brief systematization of current state-of-the-art attacks and general guidelines for secure enclave development.

All presentation material and source code for this tutorial will be made publicly available at https://github.com/jovanbulck/sgx-tutorial-space18.

2 Software Side-Channel Attacks on Enclaved Execution

We consider a powerful class of software-only attacks that require only code execution on the machine executing the victim enclave. Depending on the adversary's goals and capabilities, the malicious code can either be executing interleaved with the victim enclave (interrupt-driven attacks [11,10,4,8,9]), or launched concurrently from a co-resident logical CPU core (HyperThreading-based resource contention attacks [5]). In the following, we overview known side-channels.

Memory Accesses. Even before the official launch of Intel SGX, researchers showed the existence of a dangerous side-channel [11] within the processor's virtual-to-physical address translation logic. By revoking access rights on selected enclave memory pages, and observing the associated page fault patterns, adversaries controlling the operating system can deterministically establish enclaved code and data accesses at a 4 KiB granularity. This attack technique has been proven highly practical and effective, extracting full enclave secrets in a single run and without noise. Following the classic cat-and-mouse game, subsequent proposals to hide enclave page faults from the adversary led to an improved class of stealthy attack variants [10] that extract page table access patterns without provoking page faults. It has furthermore been demonstrated [8] that privileged adversaries can mount such interrupt-driven attacks at a very precise instruction-level granularity, which allows to accurately monitor enclave memory access patterns in the time domain so as to defeat naive spatial page alignment defense techniques [2,8].

A complementary line of SGX-based Prime+Probe cache attacks exploit information leakage at an improved 64-byte cache line granularity [6]. Adversaries first load carefully selected memory locations into the shared CPU cache, and afterwards measure the time to reload these addresses to establish code and data

evictions by the victim enclave. As with the paging channel above, these attacks commonly exploit the adversary's control over untrusted system software to frequently interrupt the victim enclave and gather side-channel information at a maximum temporal resolution [8]. This is not a strict requirement, however, as it has been demonstrated that even unprivileged attacker processes can concurrently monitor enclave cache access patterns in real-time [6].

In summary, the above research results show that enclave code and data accesses on SGX platforms can be accurately reconstructed, both in space (at a 4 KiB or 64-byte granularity) as well as in time (after every single instruction).

Instruction-Level Leakage. It has furthermore been shown that enclave-private control flow leaks through the CPU's internal Branch Target Buffer (BTB) [4]. These attacks essentially follow the general principle of the above Prime+Probe attacks by first forcing the BTB cache in a known state. After interrupting the enclave, the adversary measures a dedicated shadow branch to establish whether the secret-dependent victim branch was executed or not. Importantly, unlike the above memory access side-channels, such branch shadowing attacks leak control flow at the level of individual branch instructions (i.e., basic blocks).

Apart from amplifying conventional side-channels, enclaved execution attack research has also revealed new and unexpected sub-cache level leakage sources. One recent work presented the Nemesis [9] attack that measures individual enclaved instruction timings through interrupt latency, allowing to partially reconstruct a.o., instruction type, operand values, address translation, and cache hits/misses. MemJam [5] furthermore exploits selective instruction timing penalties from false dependencies induced by an attacker-controlled spy thread to reconstruct enclave-private memory access patterns *within* a 64-byte cache line.

Speculative Execution. In the aftermath of recent x86 speculation vulnerabilities, researchers have successfully demonstrated Spectre-type speculative code gadget abuse against SGX enclaves [1]. Recent work furthermore presented Foreshadow [7] which allows for arbitrary in-enclave reads and completely dismantles isolation and attestation guarantees in the SGX ecosystem. Intel has since revoked the compromised attestation keys, and released microcode patches to address Foreshadow and Spectre threats at the hardware level.

3 Enclave Development Guidelines and Caveats

Existing SGX side-channel mitigation approaches generally fall down in two categories. One line of work attempts to harden enclave programs through a combination of compile time code rewriting and run time randomization or checks, so as to obfuscate the attacker's view or detect side-effects of an ongoing attack. Unfortunately, as these heuristic proposals do not block the root information leakage in itself, they often fall victim to improved and more versatile attack variants [10,8,5]. A complementary line of work therefore advocates the more comprehensive constant time approach known from the cryptography community:

eliminate secret-dependent code and data paths altogether. While this approach is relatively well-understood for small applications, in practice even vetted crypto implementations exhibit non-constant time behavior [10,6,5]. In the context of SGX, it has furthermore been shown [11,9] that enclave secrets are typically not limited to well-defined private keys, but are instead scattered throughout the code and hence much harder to manipulate in constant time.

We conclude that side-channels pose a real threat to enclaved execution, while no silver bullet exists to eliminate them at the compiler or system level. Depending on the enclave's size and security objectives, it may be desirable to strive for intricate constant time solutions, or instead rely on heuristic hardening measures. However, further research and raising developer awareness are imperative to make such informed decisions and adequately employ TEE technology.

Acknowledgments. This research is partially funded by the Research Fund KU Leuven. Jo Van Bulck is supported by the Research Foundation – Flanders (FWO).

References

1. Chen, G., Chen, S., Xiao, Y., Zhang, Y., Lin, Z., Lai, T.H.: SgxPectre attacks: Leaking enclave secrets via speculative execution. arXiv:1802.09085 (2018)
2. Intel: Software guard extensions developer guide: Protection from side-channel attacks. https://software.intel.com/en-us/node/703016 (June 2017)
3. Intel: Intel Software Guard Extensions (SGX) SW Development Guidance for Potential Edger8r Generated Code Side Channel Exploits (March 2018)
4. Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In: Proceedings of the 26th USENIX Security Symposium. Vancouver, Canada (August 2017)
5. Moghimi, A., Eisenbarth, T., Sunar, B.: Memjam: A false dependency attack against constant-time crypto implementations in SGX. In: Cryptographers' Track at the RSA Conference. pp. 21–44. Springer (2018)
6. Schwarz, M., Weiser, S., Gruss, D., Maurice, C., Mangard., S.: Malware guard extension: Using SGX to conceal cache attacks. In: Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA (2017)
7. Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: 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)
8. Van Bulck, J., Piessens, F., Strackx, R.: SGX-Step: A practical attack framework for precise enclave execution control. In: Proceedings of the 2nd Workshop on System Software for Trusted Execution. pp. 4:1–4:6. SysTEX'17, ACM (October 2017)
9. Van Bulck, J., Piessens, F., Strackx, R.: Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt logic. In: Proceedings of the 25th ACM Conference on Computer and Communications Security. CCS'18, ACM (2018)
10. Van Bulck, J., Weichbrodt, N., Kapitza, R., Piessens, F., Strackx, R.: Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In: Proceedings of the 26th USENIX Security Symposium. USENIX (August 2017)
11. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In: 2015 IEEE Symposium on Security and Privacy. pp. 640–656. IEEE (2015)