Software Side-Channel Attacks Against Intel SGX Enclaves

CS 658 Research Paper

Colin Howes
University of Waterloo
200 University Ave. W
Waterloo, ON N2L 3G1
chowes@uwaterloo.ca

ABSTRACT

Intel Software Guard Extensions (SGX) is a hardware-backed security extension of the Intel architecture that allows user-level software to run securely in an environment where all other software on the host system is untrusted. SGX uses secure enclaves running in processor reserved memory combined with a software attestation scheme to provide confidentiality and integrity guarantees to users wishing to execute security-sensitive software on an untrusted remote system. SGX is vulnerable to a number of software side-channel attacks, which leverage performance measurements to determine memory access patterns and derive secrets from software executing in secure SGX enclaves. SGX vulnerabilities to software side-channel attacks, possible countermeasures, and directions for future research are discussed.

1 INTEL SGX

Intel Software Guard Extensions (SGX) is an extension to the Intel architecture designed to allow programs to run securely in an environment where all software on the host machine is potentially untrusted [1]. SGX was designed to address the problem of *secure remote computation*, that is, secure execution of software on a remote system controlled by an untrusted party [1]. Under this threat model, *all* software on a remote system is potentially malicious, including privileged software such as the operating system and hypervisor. Thus SGX must provide protection to user-level programs from untrusted software running at user level as well as at higher privilege levels [2].

SGX provides confidentiality and integrity guarantees through the use of trusted hardware that sequesters a user-level program into a secure container called an *enclave*, and then proves to a user through *software attestation* that he or she is running unmodified software protected in an enclave by secure hardware [1]. In order to provide software attestation, the host machine provides a cryptographic signature certifying an enclave's measurement hash, which includes a measurement of its contents after it is loaded and initialized [2]. The remote user can then verify this signature against an endorsement certificate provided by the hardware manifacturer, and can refuse to load his or her data into an enclave whose measurement hash differs from the expected value [1, 2]. Under this threat model, the remote user need only trust the hardware manufacturer responsible for providing the secure hardware and endorsement certificate.

1.1 SGX Enclaves

The SGX security model is centred around maintaining security-sensitive information in secure containers isolated from the rest of an untrusted system. An SGX enclave accomplishes this through the addition of a set of microcode instructions and the use of a contigious range of protected memory called Processor Reserved Memory, accessible only via these SGX microcode instructions from within an enclave [2]. Enclave contents and metadata are stored in a subset of Processor Reserved Memory, which is split into 4 KB pages. Management of enclave pages is delegated to the untrusted host OS or hypervisor, and memory management decisions made by system software are tracked in order to verify that enclave pages can only be accessed by enclave code running in the enclave associated with a specific enclave page [1, 2].

2 SOFTWARE SIDE-CHANNEL ATTACKS

Side-channel attacks are a class of attacks that leverage information about the physical properties of a system in order to carry out an attack. While physical side-channel attacks against modern hardware are difficult and costly, requiring advanced tools and physical access to the victim machine, software side-channel attacks are inexpensive to deploy and can be executed by anyone with remote or local access to a system [2]. The software side-channel attacks discussed here exploit hardware and software implementation details to acquire information about memory access patterns, which can be used to infer secrets from an otherwise secure system [3–6].

Cache timing attacks are used to infer memory access information by exploiting timing differences. Briefly, cache timing attacks take advantage of an otherwise desirable property of caching - accessing data stored in a chance is considerably faster than accessing data stored in memory.

An attacker can take advantage of this difference in access speed by filling the last level cache with data, thereby evicting data used by other processors, and measuring the time it takes to access this data after a context switch. The attacker can determine which regions of memory were accessed by the victim based on the time it takes to access the data it had previously placed in the cache. In effect, the attacker can infer information about the victim's memory usage patterns at page level granularity, which can be used to extract secrets such as encryption keys.

Page fault attacks can be employed by a malicious operating system to determine when a program is accessing specific pages. Page fault attacks are similar to cache timing attacks, in that an attacker moves a large number of pages into memory and leverages kernel-level information about page faults to determine which pages are used by a victim process.

3 ATTACKS AGAINST SGX ENCLAVES

Both cache timing attacks and page fault attacks have been demonstrated against SGX enclaves in proof-of-concept attacks. While enclave data is protected in processor-reserved memory, memory management is deferred to the untrusted OS and hypervisor.

Kernel level attacks...

User level attacks... Moreover, since even privileged services are oblivious to the contents and behavior of SGX enclaves, the protections enjoyed by enclave programs may also be used by attackers to disguise malware. A host OS has no means to monitor its SGX enclaves for malicious behaviour, and as such cannot protect honest user-level process from malicious enclave behaviour.

4 COUNTERMEASURES

A number of countermeasures have been proposed, both for future development of secure hardware, and additional security measures that can be employed by security conscious developers working with the current implementation of SGX.

5 FUTURE DIRECTIONS

6 CONCLUSIONS

REFERENCES

[1] Intel Corporation. 2016. Intel Software Guard Extensions. (2016). https://software.intel.com/en-us/sgx

[2] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint Archive* 2016 (2016), 86. https://pdfs.semanticscholar.org/2d7f/3f4ca3fbb15ae04533456e5031e0d0dc845a.pdf

[3] Johannes Gtzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Mller. 2017. Cache Attacks on Intel SGX. ACM Press, 1–6. https://doi.org/10.1145/3065913.3065915

[4] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clmentine Maurice, and Stefan Mangard. 2017. Malware Guard Extension: Using SGX to Conceal Cache Attacks. *arXiv:1702.08719 [cs]* (Feb. 2017). http://arxiv.org/abs/1702.08719 arXiv: 1702.08719.

[5] Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena. 2015. Preventing Your Faults From Telling Your Secrets: Defenses Against Pigeonhole Attacks. *arXiv:1506.04832 [cs]* (June 2015). http://arxiv.org/abs/1506.04832 arXiv: 1506.04832.

[6] Y. Xu, W. Cui, and M. Peinado. 2015. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. In *2015 IEEE Symposium on Security and Privacy*. 640–656. https://doi.org/10.1109/SP.2015.45