Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks

Continuous Code Re-Randomisation at Runtime for Intel SGX Enclaves

Mohammad Mahhouk

7. September 2020

1 Introduction

Hardware-based Trusted Execution Environments (TEEs), like ARM TrustZone [4, 23], RISC-V Keystone [15] and Sancus [21], have become a focus of interest regarding their protection of security-sensitive applications running in untrusted environments. Intel Software Guard Extensions (SGX) [1, 10, 18] is currently the most advanced hardware-based (TEE) technology and readily available in commodity Intel CPUs since the Skylake architecture. Furthermore, it promises the confidentiality and integrity of sensitive computations and data against strong adversary models, such as compromised operating systems or malicious cloud providers. To achieve the promised security guarantees, Intel SGX provides TEEs, so-called enclaves, that isolates the security-sensitive code and data in special memory regions encrypted by a hardware unit, the Memory Encryption Engine (MEE) [12]. Consequently, only the enclave can access and process its contents.

Some research works have used SGX TEEs to strengthen the weak trust in cloud infrastructures by implementing system prototypes that contain a complete library OS and enable running unmodified legacy applications in trusted enclaves such as Haven [5] and Graphene-SGX [31]. Others have utilised SGX to secure the computations on the outsourced sensitive data in the cloud like the distributed MapReduce framework VC3 [25]. As of today, Microsoft provides one of the first TEE in the cloud based on Intel SGX [2].

2 Motivation

Lately, multiple studies have demonstrated several attacks on SGX that break down its offered security guarantees. They mainly took advantage of SGX's strict threat model and created numerous return-oriented programming (ROP) [16, 29] and side channel [7, 8, 17] attacks to exploit memory corruption vulnerabilities inside a SGX enclave and leak its sensitive contents. To mitigate the impact of such attacks, some researchers have implemented

various defensive mechanisms for SGX enabled applications [11, 13, 22, 28]. For example, SGX-Shield [26], SGX-Armor [27] and DR.SGX [6] have introduced different Address Space Layout Randomisation (ASLR) implementations into SGX TEEs. Alternatively, Obfuscuro [3] is a cryptography-based defence technique, like oblivious RAM mechanisms [24, 30], for Intel SGX that protects from sensitive data leakage by access pattern-based [9] and timing-based side channel attacks [7, 19].

Nonetheless, recent research works have presented powerful classes of attacks, like JIT-ROP [29] and controlled channel attacks [14, 20, 32], that can circumvent the aforementioned defence techniques by either derandomising the shuffled memory layout or observing enclave access patterns without resorting to page faults at runtime. Furthermore, all presented SGX defence mechanisms either require special compilers or hardware modifications due to the limitations of SGX or they add a significant runtime overhead to the enclaved applications.

Very recently, Intel has released the novel SGXv2 extensions that lifted the initial enclave restrictions regarding its fixed size and the inability to modify the access permissions of its pages at runtime. As a result, they allowed the development of more flexible and complex applications, such as an in-enclave JIT-compiler or loading code modules at runtime, as well as new defence strategies.

3 The Talk's Content

At start, we briefly present the *sgx-dl* framework which, to our knowledge, is the first project built upon the novel SGXv2 instructions. It enables dynamic loading and hot patching of function code inside an enclave at runtime. Moreover, it does not demand any special compilers or hardware modifications and only utilises the SGX SDK functionalities and requires the adoption of a particular programming model.

Subsequently, we present a new security feature for SGX enclaves, namely periodic ASLR, as an extension of the *sgx-dl* framework. Different from the traditional ASLR techniques, it periodically randomises the locations of the dynamically loaded function code and data inside the enclave at runtime while adapting the shuffling frequency to the application's workload between each two consecutive shuffles. Thus, it significantly increases the generated noise during the previously mentioned strong attacks, such as JIT-ROP, page-fault and some timing based and cache-based side channel attacks. Furthermore, we illustrate the design decisions taken to achieve the aforementioned security feature. At the end, we demonstrate an analysis regarding the security and performance overhead induced by the periodic ASLR feature.

In conclusion, our approach improves the enclave protection against the modern ROP and side channel attacks with lower execution overhead costs and minimal necessary efforts compared to existing defensive mechanisms. The conducted macro-benchmark, STANlite's built-in performance benchmark (*Speedtest1*), has shown an average runtime overhead of 2.26% per test experiment compared to the native execution without periodic ASLR. Meanwhile, the performance overhead of the micro-benchmarks have shown some fluctuations between 4% and up to 50% in some cases based on the utilised randomisation technique.

Literatur

[1] *Intel Software Guard Extensions (Intel SGX).* https://software.intel.com/sites/default/files/332680-002.pdf. Accessed: 2020-01-27.

[2] *Azure Confidential Computing.* https://azure.microsoft.com/de-de/solutions/confidential-compute/, 2019. Accessed: 2020-05-08.

[3] Ahmad, A., B. Joe, Y. Xiao, Y. Zhang, I. Shin und B. Lee: *OBFUSCURO: A Commodity Obfuscation Engine on Intel SGX.* In: *NDSS*, 2019.

[4] Alves, T.: *Trustzone: Integrated hardware and software security.* White paper, 2004.

[5] Baumann, A., M. Peinado und G. Hunt: *Shielding Applications from an Untrusted Cloud with Haven.* ACM Trans. Comput. Syst., 2015.

[6] Brasser, F., S. Capkun, A. Dmitrienko, T. Frassetto, K. Kostiainen und A. R. Sadeghi: *DR.SGX: Hardening SGX Enclaves against Cache Attacks with Data Location Randomization*, 2017.

[7] Brasser, F., U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun und A. R. Sadeghi: *Software Grand Exposure: SGX Cache Attacks Are Practical.* In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[8] Bulck, J. V., M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom und R. Strackx: *Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution.* In: *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[9] Bulck, J. V., N. Weichbrodt, R. Kapitza, F. Piessens und R. Strackx: *Telling Your Secrets without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution.* In: *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[10] Costan, V. und S. Devadas: *Intel SGX Explained.* IACR Cryptology ePrint Archive, 2016.

[11] Davi, L., C. Liebchen, A. R. Sadeghi, K. Z. Snow und F. Monrose: *Isomeron: Code randomization resilient to (just-in-time) return-oriented programming.* In: *NDSS*, 2015.

[12] Gueron, S.: *Memory Encryption for General-Purpose Processors.* IEEE Security Privacy, 2016.

[13] Hosseinzadeh, S., H. Liljestrand, V. Leppänen und A. Paverd: *Mitigating Branch-Shadowing Attacks on Intel SGX Using Control Flow Randomization.* In: *Proceedings of the 3rd Workshop on System Software for Trusted Execution*, 2018.

[14] Hund, R., C. Willems und T. Holz: *Practical Timing Side Channel Attacks against Kernel Space ASLR.* In: *2013 IEEE Symposium on Security and Privacy*, 2013.

[15] Lee, D., D. Kohlbrenner, S. Shinde, D. Song und K. Asanović: *Keystone: A framework for architecting tees.* arXiv preprint arXiv:1907.10119, 2019.

[16] Lee, J., J. Jang, Y. Jang, N. Kwak, Y. Choi, C. Choi, T. Kim, M. Peinado und B. B. Kang: *Hacking in Darkness: Return-oriented Programming against Secure Enclaves.* In: *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[17] Lee, S., M. W. Shih, P. Gera, T. Kim, H. Kim und M. Peinado: *Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing.* In: *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[18] McKeen, F., I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue und U. R. Savagaonkar: *Innovative instructions and software model for isolated execution.* Hasp@ isca, 2013.

[19] Moghimi, A., G. Irazoqui und T. Eisenbarth: *CacheZoom: How SGX Amplifies the Power of Cache Attacks.* In: Fischer, W. und N. Homma (Hrsg.): *Cryptographic Hardware and Embedded Systems – CHES 2017*, 2017.

[20] Moghimi, D., J. V. Bulck, N. Heninger, F. Piessens und B. Sunar: *CopyCat: Controlled Instruction-Level Attacks on Enclaves for Maximal Key Extraction.* ArXiv, 2020.

[21] Noorman, J., J. V. Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller und F. Freiling: *Sancus 2.0: A Low-Cost Security Architecture for IoT Devices.* ACM Trans. Priv. Secur., 2017.

[22] Oleksenko, O., B. Trach, R. Krahn, M. Silberstein und C. Fetzer: *Varys: Protecting SGX Enclaves from Practical Side-Channel Attacks.* In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018.

[23] Pinto, S. und N. Santos: *Demystifying Arm TrustZone: A Comprehensive Survey.* ACM Comput. Surv., 2019.

[24] Rane, A., C. Lin und M. Tiwari: *Raccoon: Closing Digital Side-Channels through Obfuscated Execution.* In: *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[25] Schuster, F., M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz und M. Russinovich: *VC3: Trustworthy Data Analytics in the Cloud Using SGX.* In: *2015 IEEE Symposium on Security and Privacy*, 2015.

[26] Seo, J., B. Lee, S. M. Kim, M. W. Shih, I. Shin, D. Han und T. Kim: *SGX-Shield: Enabling Address Space Layout Randomization for SGX Programs.* In: *NDSS*, 2017.

[27] Shih, M.: *Securing Intel SGX against Side-channel Attacks via Load-time Synthesis.* Dissertation, Georgia Institute of Technology, 2019.

[28] Shinde, S., Z. L. Chua, V. Narayanan und P. Saxena: *Preventing Page Faults from Telling Your Secrets.* In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016.

[29] Snow, K. Z., F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen und A. Sadeghi: *Just-In-Time Code Reuse: On the Effectiveness of Fine-Grained Address Space Layout Randomization.* In: *2013 IEEE Symposium on Security and Privacy*, 2013.

[30] Stefanov, E., M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu und S. Devadas: *Path ORAM: An Extremely Simple Oblivious RAM Protocol.* In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013.

[31] Tsai, C. che, D. E. Porter und M. Vij: *Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX.* In: *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017.

[32] Xu, Y., W. Cui und M. Peinado: *Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems.* In: *2015 IEEE Symposium on Security and Privacy*, 2015.