Hardening Intel SGX Applications: Balancing Concerns

Kobe Vrancken
imec-DistriNet, KU Leuven
Heverlee, Belgium
kobe.vrancken@cs.kuleuven.be

Frank Piessens
imec-DistriNet, KU Leuven
Heverlee, Belgium
frank.piessens@cs.kuleuven.be

Raoul Strackx
imec-DistriNet, KU Leuven
Heverlee, Belgium
raoul.strackx@cs.kuleuven.be

Abstract

Today's technologies such as Intel SGX enable developers to protect applications from more privileged security layers. Applications can be developed with significantly smaller TCBs than before. Two main techniques have emerged to harden applications for these new security architectures. One focuses on partitioning applications in pieces-of-application-logic thereby isolating security-sensitive parts in their own enclaves. The other moves full unmodified applications to a single enclave tied together with a library OS. In this paper we argue that both techniques should be treated as separate security mechanisms each providing their own balance in terms of security, performance and usability.

CCS Concepts • Security and privacy → Software security engineering;

Keywords Intel SGX, Trusted computing base

1 Introduction

Applications typically require a non-malicious operating system (OS) to guarantee correct execution. Operating systems can comprise millions of lines of code (LOC). Vulnerabilities in this codebase can lead to applications being compromised. To reduce this problem Intel SGX provides confidentiality and integrity in a region of memory called the Enclave Page Cache (EPC). Programs are loaded into this memory in units called enclaves. The Quoting Enclave (QE), provided by Intel, can generate cryptographic proofs that enclaves were initialized and called exactly as specified by the enclave providers. The verification of this proof by a local/remote party is called local/remote attestation. A fixed set of predefined entry points have to be used in order to pass execution to and make use of the services provided by an enclave. The

combination of the mechanisms described above allows the integrity or correct execution of an enclave program to be independent of the underlying OS or hypervisor.

There are two main approaches when designing applications for this new architecture. One approach isolates security sensitive pieces-of-application-logic (PALs) [4] from the rest of the application. The rest of the application is considered untrusted and can make use of all OS features. As an alternative, recent systems [2] have emerged that allow full (unmodified) applications (FApps) to run within enclaves. This is made possible by loading a library OS inside an enclave which intercepts system calls. The system calls are either handled by the library OS directly or the parameters and return values are sanitized and passed on to the real OS.

Both approaches harden applications against attacks by leveraging new security architectures. However, we argue that they should be treated as separate security mechanisms, that make different assumptions about the code they protect and provide different protection guarantees.

2 Security

The main use case of FApps is to run full applications securely on the infrastructure of cloud providers without requiring a complete redesign. They aim to isolate cloud applications from each other and from potentially malicious or infected cloud providers. The TCB of these applications remains large, with complete library OSes and full applications residing within the enclave. In the past researchers have placed complete SQL and Apache servers inside of enclaves [2].

In contrast, PALs require a complete redesign of applications so that security-critical parts can be isolated. Since the resulting enclaves will be smaller it becomes feasible to formally verify them. On top of that they could easily be written in memory-safe (compilable) languages such as Rust. These methods minimize the TCB and attack surface of the applications but require a much harder and more intrusive manual process. Small security-sensitive pieces of applications such as the password storage component in a password manager are perfect use cases for this type of design.

Low-level vulnerabilities FApps typically expose a large interface. Low-level exploits such as buffer overflows can still be leveraged by malicious cloud providers to circumvent the security measures just like local applications can use these to exploit one another. Techniques such as fine-grained Address

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SysTEX'17, October 28, 2017, Shanghai, China

© 2017 Association for Computing Machinery.

ACM ISBN 978-1-4503-5097-6/17/10...\$15.00

<https://doi.org/10.1145/3152701.3152711>

Space Layout Randomization (ASLR) within enclaves have been explored [5] as effective probabilistic defences against these types of attacks. However, due to the large size of the TCB it becomes very hard to completely eliminate them.

Since PALs have small TCBs, can be written in memory safe languages and can be formally verified, they can offer strong protection against these types of vulnerabilities.

Probabilistic defences such as fine-grained ASLR that were used for FApps do not offer strong enough guarantees for PALs. Similarly formally verifying and completely rewriting FApps to memory-safe languages is not realistic. Techniques developed for FApps do not necessarily translate to PALs and vice-versa. Both methods have their own security objectives and require their own techniques and defences.

Side-channels The main vulnerabilities discovered in SGX-protected applications to this date come in the form of side-channels [7]. These types of attacks can lead to significant leakage of secrets and are not trivial to defend against. The focus of this line of research so far has mainly targeted FApps.

PALs can suffer from the same type of vulnerabilities. Good defences against side-channels are essential if PALs are to be adopted as a way to provide high levels of security. Low-level security defences that incur significant overhead such as Page Access Oblivious computations [6] can be used with acceptable overhead mainly thanks to the small enclave sizes of PALs. These type of defences would incur too much overhead for FApps. This again highlights the different focus needed in research for both of these hardening methods.

3 Performance

The execution of enclave instructions and the access of enclave memory happens at the same speeds as their non-enclave counterparts. However, whenever there is a context-switch between enclave mode and regular applications significant overhead can be observed.

For FApps it is only necessary to perform a context-switch when the library OS needs to rely on the regular OS for specific services. PALs on the other hand are inherently divided into trusted and untrusted parts. When the application logic switches between these parts a context-switch has to be performed. The authors of Glamdring [3], an annotation-based system that aides with partitioning applications into PALs, have compared the performance of PALs generated by their system and FApps. They observe that there seems to be a trade-off between TCB size and performance. Partitioning allows for smaller TCBs but this comes with a hit to the performance of the application.

It is however acceptable for PALs to incur significant overhead if this increases security. Hence, defences for these types of enclaves can be allowed to have substantially higher performance costs. Since these overheads only occur in small security-sensitive parts, it can still result in only small performance hits for complete applications.

Conversely, for FApps using defence techniques with large overheads would likely make the applications unusable. When complete applications are placed inside enclaves, the performance penalties incurred by this protection should not be excessive. Users will expect applications to perform similarly to when they are executed locally.

4 Conclusion

FApps offer an additional layer of security with relatively low performance and implementation costs. Applications can be hardened with very little disadvantages. This allows developers to move applications to untrusted cloud infrastructures with similar security guarantees as when they are executed locally.

PALs provide strong security guarantees at the lowest levels. Defences for PALs can afford to incur higher levels of overhead. Here, security is the main concern.

Both security mechanisms come with different balances in terms of usability, security and performance. Future research in SGX should take these balances into account and design defences accordingly.

Acknowledgements. This work was partially supported by the Research Fund KU Leuven and the TearLess research project. Raoul Strackx is supported by a grant of the Research Foundation – Flanders (FWO).

References

- [1] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumar, Dan O’Keeffe, Mark Stillwell, et al. 2016. SCONE: Secure Linux Containers with Intel SGX.. In *OSDI*. 689–703.
- [2] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2015. Shielding applications from an untrusted cloud with haven. *ACM Transactions on Computer Systems (TOCS)* 33, 3 (2015), 8.
- [3] Joshua Lind, Christian Priebe, Divya Muthukumar, Dan O’Keeffe, Pierre-Louis Aublin, Florian Kelbert, Tobias Reiher, David Goltzsche, David Eysers, Rüdiger Kapitza, et al. 2017. Glamdring: Automatic application partitioning for Intel SGX. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, Santa Clara, CA.
- [4] Jonathan M McCune, Bryan J Parno, Adrian Perrig, Michael K Reiter, and Hiroshi Isozaki. 2008. Flicker: An execution infrastructure for TCB minimization. In *ACM SIGOPS Operating Systems Review*, Vol. 42.
- [5] Jaebaek Seo, Byoungyoung Lee, Seongmin Kim, Ming-Wei Shih, Insik Shin, Dongsu Han, and Taesoo Kim. 2017. SGX-Shield: Enabling address space layout randomization for SGX programs. In *Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA.
- [6] Rohit Sinha, Sriram Rajamani, and Sanjit A Seshia. 2017. *A Compiler and Verifier for Page Access Oblivious Computation*. Technical Report. Technical Report UCB/EECS-2017-124, EECS Department, University of California, Berkeley.
- [7] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. 2015. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 640–656.