Exploring Intel Software Guard Extensions (SGX)

Date: April 28, 2016

Investigators:

1. Praveen Keshavamurthy
   Masters in Computer Science
   Email: keshavamurthy.p@husky.neu.edu

2. Aboobacker Rizwan Payapura
   Masters in Computer Science
   Email: rizwan@ccs.neu.edu

General Field: The project falls in the realms of Cloud Security, where the resources are shared among many virtual machines. The adversary model which SGX targets is complete compromise of operating system, which is indeed a main threat model for cloud hosted applications.

Keywords: cloud security, software guard extensions(SGX), isolated containers, hardware security feature, resource sharing

Project Description: Intel SGX, which provides hardware security features is a major step towards a better security in cloud computing, but it is mostly untested by researchers. So the exact limitations and capabilities of this new technology is still not very clear. We will be exploring Intel SGX by adapting SQLite database to run inside the secure container provided by Intel SGX technology.

Project Responsibilities:

1. Praveen Keshavamurthy: Lead Designer, Software Developer
2. Aboobacker Rizwan Payapura: Lead Researcher, Software Developer

Total Budget: $16200

Deliverables:

- Sample Intel SGX application.
- Adaptation of SQLite library to Intel SGX.
- Guidelines to develop an application with Intel SGX.

Executive Summary
Exploring Intel Software Guard Extensions (SGX)

Date: April 28, 2016

Investigators:

1. Praveen Keshavamurthy
   Masters in Computer Science
   Email: keshavamurthy.p@husky.neu.edu

2. Aboobacker Rizwan Payapura
   Masters in Computer Science
   Email: rizwan@ccs.neu.edu

Keywords: cloud security, software guard extensions(SGX), isolated containers, hardware security feature, resource sharing

Secure remote computation, offloading processing to a remote computer owned and maintained by an untrusted party with some integrity and privacy guarantees, has remained an unsolved problem. This is mainly due to the hierarchical security model that aims only to protect the privileged code from untrusted code and does not fare well to protect user data from access by privileged code. Although complete homomorphic encryption solves the problem for a limited family of computations, it has an impractical performance overhead. Intel SGX is the latest technology which aims to solve the secure remote computation problem by leveraging trusted processors on remote computers.

Intel SGX is an architecture extension designed to increase the security of software through an “inverse sandbox” mechanism. At its root, Intel SGX is a set of new CPU instructions, using which legitimate software can be sealed inside an enclave (secured memory) and protected from attack by the malware, irrespective of the privilege level of the latter. It also provides a novel hardware assisted mechanism to allow secure remote attestation and sealing to application software. Intel SGX technology is the need of the hour for applications related to Data Rights Management (DRMs) and in designing a novel architecture for trusted cloud. As Intel SGX was available to select group of researchers until recently, there is a lack of current research and programs that evaluate the technology with respect to performance impacts on existing state-of-the-art solutions.

In this project, we dissect the security features of Intel SGX by adapting SQLite Library to use Intel SGX technology. We will benchmark the SQLite DB performance overhead due to the use of Intel SGX in secure software execution path. We will also come up with Intel SGX application development guidelines document and list of challenges/limitations posed by Intel SGX based on the experience we obtain by adapting SQLite library.

Table of Contents

Introduction

There is a lot of trust involved in a cloud infrastructure. From a security perspective, this is a bad thing. But due to the complexity and performance issues, many protocols are designed based on trust. For instance creating an app and hosting it in the cloud platform involves many level of trusts. The app developer needs to trust the Operating System (OS) in the cloud (which have high privilege access), the cloud provider, and worst of all other applications hosted in the same platform. The code base of OS is really huge and written by thousands of developers around the world. There is a very high possibility for a bad code in such a huge code base. Since the OS kernel operates in ring 0 (high privileged), the attacker can get a high privileged access and thus can manipulate the private data of other applications hosted in that platform, which operates in lower privileged user space (ring 3). So trusting such platform is a very risky thing to do. This problem can be solved by reducing the level of trust which will substantially reduce the attack surface. Hardware support is a very important step for achieving this.

Intel Software Guard Extensions (Intel SGX) is designed for this purpose through an inverse sandbox mechanism. It solves the above problem by preserving the confidentiality and integrity of sensitive code and data of the applications without disrupting the ability of legitimate system software to schedule and manage the use of platform resources. Since applications create their own secure containers and the processor makes sure that the access to these containers is limited to only that application, the trusted components of the application are protected. Intel SGX provides a feature which allows remote user to verify that the application is set up securely on the untrusted infrastructure by producing a signed attestation (rooted in the processor) that includes the measurement and other certification.

As part of this project, our goal is to answer the following key questions related to Intel SGX technology.

- What applications are suitable for using Intel SGX technology.
- What are the guidelines for the design decisions to port an application to use Intel SGX.
- How does Intel SGX impact the application performance.
- What are the challenges and limitations while trying to develop an application for Intel SGX.

For achieving our goal we will,

- Explore Intel SGX Architecture.
- Explore Intel SGX SDK by developing an application to run inside an enclave and make use of the Intel SGX features.
- Port SQLite library to use Intel SGX features.
- Benchmark the DB performance impact due to Intel SGX

Why SQLite library? To find a good cloud use case for adapting to Intel SGX, we went through different interesting applications like adapting Cloud Infrastructures, TOR networks, Datacenter

Applications, Kernel-based Virtual Machine (KVM). But due to the following limitations of Intel SGX at this time we had to discard many of those:

- Intel SGX supports only C/C++.
- Platform Software for Intel SGX is available only for Windows. So as of now applications can be developed only for Windows.
- Limitations in memory. Maximum run time memory reserved for SGX enclaves is 128 MB and maximum heap size is 96 MB.

We found that, given the above limitations SQLite is a perfect use case for adapting to Intel SGX. It is a C library available for Windows. Also the SQLite memory footprint ceils at 3 MB, irrespective of the size of the database file being accessed. This is in line with the Intel SGX enclave memory limitations. Moreover SQLite has a cipher version that performs transparent and on-the-fly encryption. Using SQLCipher, an application uses the standard SQLite API to manipulate tables using SQL. SQLCipher encryption engine can be developed using Intel SGX.

Literature Review

Intel SGX is the latest in the long list of trusted execution technologies. Some of the previous technologies include IBM 4765 Secure Coprocessor, ARM's TrustZone, Trusted Platform Module (TPM), Trusted Execution Technology (TXT), etc. Intel SGX has evolved from all these previous implementations of trusted execution environment. Intel SGX directly inherits the software attestation and Root of Trust Measurement technologies from Intel's TPM and TXT. It inherits inverted cache map feature from Bastion architecture to give secure access to memory. Also, Intel SGX has a very robust design which includes DRAM Encryption & Integrity protection and an on chip tamper resistant chip which holds the processor secret key.

Intel SGX technology was only available to select group of research workers until recently. The previous works on Intel SGX include adapting OS library to support unmodified legacy applications, including SQL Server and Apache, on a commodity OS (Windows) and commodity hardware. Using trusted SGX processors as a building block, a prototype was developed by adapting Hadoop distribution framework where in the simple map and reduce jobs were bounded within an enclave, thereby reducing the trusted computing base (TCB) and hence the privacy and integrity of the data. This work has also benchmarked the performance saying average runtime overhead due to Intel SGX being 8% with read/write integrity.

All the research work on Intel SGX fails to answer the following questions.

- What applications are suitable for using Intel SGX technology?
- What are the main design decisions to port an application to use Intel SGX?
- How does Intel SGX impact the application performance?

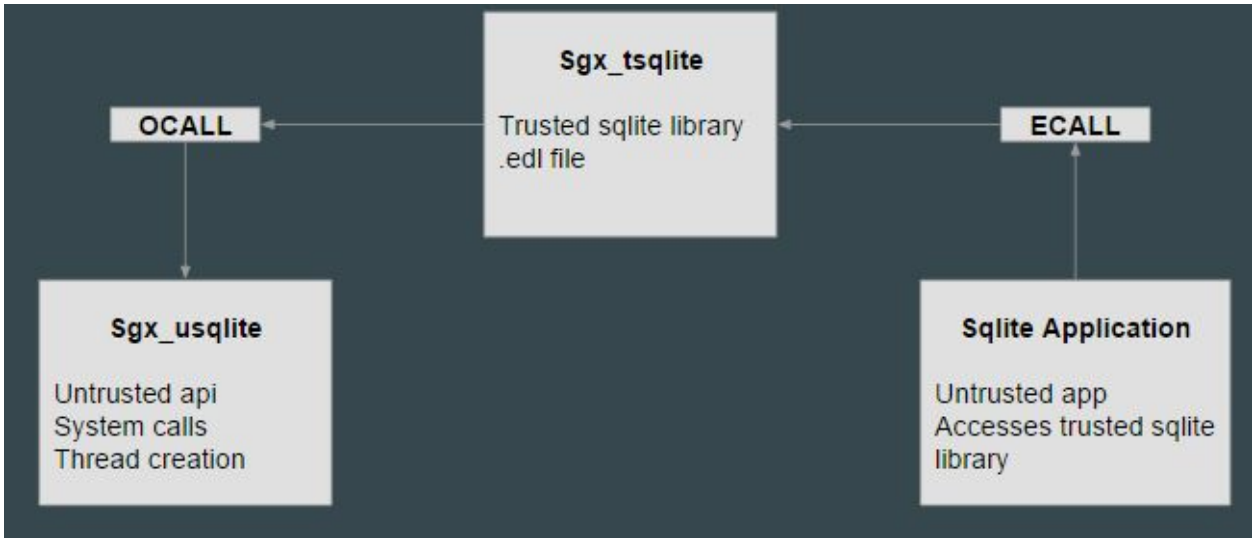
This project tries to dissect Intel SGX features by adapting widely used SQLite DB library. Using such an approach, we try to identify the possible issues one might face while adapting the code to Intel SGX and also possible solutions for the same. There are no such comprehensive developer guide as of now. If successful, one can safely use SQLite DB library with Intel SGX thereby being sure that all the data are completely safe, irrespective of the adversary. Such a proposition has a huge stake in the real world, where the need of minimal trust surface is ever increasing.

Methods and Procedures

We employed a constructive research approach, where we try to find solutions to different problems which might arise when trying to port an application to Intel SGX. We had two approaches to adapt SQLite code.

1. Identify the SQLite code which performs the crux of database operations. Try to move those logic into secure intel SGX enclaves
2. Move the entire code into Intel SGX library so that complete DB runs inside secure enclave and identify the code that Intel SGX does not support.

As part of this project, we finalized on second approach considering the steep curve involved in understanding the complete SQLite design, required for the first approach.



The major tasks involved in this approach of porting SQLite Code to Intel SGX are:

1. Segregate the SQLite code to be run outside the Intel SGX enclave such as OS system call, thread creation and non-sgx library dependencies.
2. Create both trusted and untrusted SQLite Libraries.
3. Develop a sample SQLite DB application which uses both these libraries in creation and maintenance of the DB.
4. Adapt the above mentioned changes to SQLite Cipher code by making sure the interaction between the trusted and untrusted libraries are completely secure.
5. Run the DB Benchmark measure to identify the overhead due to the use of Intel SGX technology.

Date	Goals	Status
10th Feb	Understand high level architecture of Intel SGX	Done
24th Feb	Develop simple application to explore SGX build environment and build tools.	Done
9th March	Compile successfully sgx_tsqlite.dll	Done
23rd March	Compile successfully sgx_usqlite.lib	Done
6th April	Build a sample sqlite application using the above two libraries	Done
13th April	Test the sqlite application and make sure the application behaves for all SQL commands	OpenDatabase() successful. SQL Queries are not working.
27th April	Adapt the above changes to SQLCipher	Not Done
27th April	Measure the DB Performance impacts	Not Done
27th April	Document the findings and report the results obtained	Not Done – Only documented the issues faced during the above compilation.
25th April	Final Report Submission and a Demo	Done

The adapted SQLite code has following limitations

1. None of the Async SQLite APIs shall be exposed to the user application.
2. Only single database connection is expected to work with the ported code.
3. The global configuration data of SQLite library are duplicated i.e., both in sgx_tsqlite.dll and sgx_usqlite.lib.

The following describes the boundaries of the project.

1. We will not be running the ported SGX application on any cloud infrastructure as the SGX hardware is only available on client devices.
2. Both Intel SGX SDK and Platform SW are available only for Windows. Hence we will be concentrating on adapting windows application.

Findings

Intel SGX does not provide an end to end secure solution. Application developer should make sure the design chosen will leverage the Intel SGX feature to deliver a secure solution. There are many restrictions on the proxy routines, as outlined in the following section, which serve as a boundary between the trusted library and untrusted application. Intel SGX build environment makes use of sgx_edge8r and sgx_sign tools to generate the proxy routines and signed dll. In general, SGX provides an environment where the author shall be sure the application is setup on the target as expected by means of remote attestation

We ported the SQLite code to make use of the SGX features. From the experience of compiling SQLite code with SGX, we can infer the following:

- sgx_edge8r.exe is used to generate proxy routines based on the .edl file
- sgx_sign.exe is used to sign the dll based on the enclave configuration file and author's 3072 bit RSA private key with public exponent of the key set to 3.
- sgx_emmt is a tool to measure the real usage of protected memory by the enclave at runtime. Currently the enclave memory measurement tool provides the following two functions:
    1. Get the stack peak usage value for the enclave.
    2. Get the heap peak usage value for the enclave.
    - Accurate stack and heap usage information for the enclaves can be used to make full use of the limited protected memory.
- Main mode of shipment of SGX binary is signed dll format.
- The SGX DLL should not depend on any runtime dlls. It can be linked to static SGX library, if required.
- Max run time memory reserved for SGX enclaves is 128 MB. This brings in a restriction on number of active enclaves at any point of time in the system.
- Max heap size per enclave is limited to 96MB. This brings in a limitation on the dynamic memory allocation to the enclaves.
- Function pointer cannot be passed across SGX proxy routines.
    - This will limit any async operation in general.
    - We could not adapt SQLite API's which supported async routines. (e.g.: sqlite3_exec())
    - As of now we have only exposed SQLite blocking APIs as part of sgx_tsqlite.dll
- Threads cannot be created inside the enclave.
    - Thread creation code were moved to the untrusted lib.
    - Wrapper routines were created in untrusted lib as entry point to the threads.
- System calls are not supported inside the Enclave.
    - All the system calls were moved to untrusted library.
    - Adapted SQLite code in the trusted library to make use of SGX OCALLS instead of system calls.

- Pointer arguments used in functions which are declared in .edl files should have associated size argument.
    - Opaque pointers are not allowed - used [user_check] for such arguments
    - All void* arguments should have an explicit size argument.
- Adapted sgx_tsqlite code to pass size argument for all the OCALLS.
    - Size can be a direct value or an argument
    - One can also specify SizeFunc which in turn returns the size

Issues

Following are the issues we faced in this project:

- Following limitations made it very difficult to come up with an appropriate cloud use case which can be adapted to Intel SGX:
    - Intel SGX supports only C/C++.
    - Platform software is available only for Windows.
    - Maximum memory inside an enclave is limited to 128 MB and maximum heap size is 96 MB.
- Visual Studio Premium 2012 doesn't support Intel SGX Debugger.
    - We used logging as an alternative for debugging. This was a time consuming alternate since the code base for SQLite was very huge.
- Multiple copies of global config variable.
    - A global config variable is used in SQLite implementation which is used for maintaining the mutex state. Since the SQLite code inside the enclave cannot access the variables outside and vice versa, we were forced to create copies of the global config variable both inside and outside the enclave. This can cause synchronization issues.
- We were not able to port all the required SQLite APIs as a trusted call. As a result we had to use a workaround APIs to execute a select query, for instance.

Conclusions and Recommendations

Intel SGX provides a trusted execution environment by executing the trusted code and data inside an enclave and guaranteeing no adversary can access the enclave. Due to the wide adoption of the x86 architecture, its Software Guard Extensions (SGX) for trusted execution potentially has a tremendous impact on software security, enabling a wide range of applications to enhance their security and privacy properties. But the current limitations and restrictions in implementing Intel SGX applications make it very difficult to develop or adapt applications to Intel SGX.

Intel SGX does not provide an end to end secure solution. Application developer should choose a design which will leverage the Intel SGX feature to deliver a secure solution. The design should be such that only the secure data processing code should be part of enclave and application design should have a secure architecture as part of non-enclave code.

Adapting an already existing non Intel SGX application to Intel SGX is not a straightforward process. The communication between the code inside the enclave and the code outside the enclave of the application behaves differently than expected in order to guarantee the security of the code and data in enclave. Restriction on memory in the enclave also need to be considered while adapting. So the implementation of the application need to be modified which require a good understanding of the architecture of the application.

Considering the limitations in the proxy routines, we could have used an alternate design approach where rather than trying to include most of the application code inside the enclave, segregate and port only the essential code inside the enclave. Since most of the issues we faced in this project is due to the code inside the restrictive enclave, this approach could solve those issues. This approach will also help in reducing the dependency on the untrusted library since the main logic is implemented outside the enclave. This can guarantee a more secure application. But the main problem with this approach is that the developer need to understand the entire architecture and implementation of the application that is being ported. For application with huge code base like SQLite, this will be a very time consuming part.

The adapting of SQLite database can be extended to adapt SQLCipher library. SQLCipher is a specialized build of the SQLite database that performs transparent and on-the-fly encryption. The library silently manages the security aspects, making sure that data pages are encrypted and decrypted as they are written to and read from storage. SQLCipher running in Intel SGX can be used to securely store the data in the cloud.

The Visual studio projects (Intel SGX application and ported SQLite application), papers related to Intel SGX and the reports are available at:
https://github.com/arizwanp/intel_sgx.git

References

1. Git repo for the project: https://github.com/arizwanp/intel_sgx.git
2. Victor Costan and Srinivas Devadas. “Intel SGX Explained”. [Online] https://eprint.iacr.org/2016/086.pdf
3. Andrew Baumann, Marcus Peinado, and Galen Hunt, Microsoft Research. “Shielding Applications from an Untrusted Cloud with Haven”, in the Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation.
4. Felix Schuster, Manuel Costa, Cedric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich, Microsoft Research. “VC3: Trustworthy Data Analytics in the Cloud”, in Microsoft Research Technical Report MSR-TR-2014-39. http://research.microsoft.com/pubs/210786/vc3-MSR-TR-2014-39.pdf
5. Prerit Jain, Soham Desai, Seongmin Kim, Ming-Wei Shih, JaeHyuk Lee, Changho Choi, Youjung Shin, Taesoo Kim, Brent B. Kang and Dongsu Han, "OpenSGX: An Open Platform for SGX Research", In Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS 2016), 2016.
6. Gehana Booth, Andrew Soknacki, and Anil Somayaji. “Cloud Security: Attacks and Current Defenses”, 8th Annual Symposium on Information Assurance (Asia’13), June 4-5, 2013.
7. Rohit Sinha, Sriram Rajamani, Sanjit A. Seshia and Kapil Vaswani. "Moat: Verifying Confidentiality of Enclave Programs", in CCS '15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.
8. Owen S. Hofmann, Sangman Kim, Alan M. Dunn, Michael Z. Lee, Emmett Witchel. "InkTag: Secure Applications on an Untrusted Operating System", in ASPLOS '13 Proceedings of the 18th International Conference on Architectural support for programming languages and operating systems.
9. Olga Ohrimenko, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Markulf Kohlweiss, Divya Sharma. "Observing and Preventing Leakage in MapReduce", in CCS '15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.
10. Intel SGX Emulation using QEMU. [Online] https://tc.gtisc.gatech.edu/bss/2014/l/final/pjain43.pdf
11. Intel Software Guard Extensions. [Online] https://software.intel.com/sites/default/files/332680-002.pdf
12. Using Innovative Instructions to Create Trustworthy Software Solutions. [Online] https://software.intel.com/en-us/articles/using-innovative-instructions-to-create-trustworthy-software-solutions.
13. Intel Software Guard Extensions Programming Reference [Online] https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

Team

1. **Praveen Keshavamurthy** is pursuing Masters in Computer Science at Northeastern University. His primary area of interest is Networking and Cloud security. He was part of cloud platform development during his tenure in Nokia Networks.

   Tasks done:
   - Segregated the SQLite code to be run outside the Intel SGX enclave such as OS system call, thread creation and non-sgx library dependencies.
   - Created both trusted and untrusted SQLite Libraries.
   - Developed a SQLite DB application which uses both the trusted and untrusted libraries for creation and maintenance of the DB.

2. **Aboobacker Rizwan Payapura** is pursuing Masters in Computer Science at Northeastern University. His primary area of interest is Network Security, Cloud Security and Cryptography. He worked for Platform Infrastructure Team in Akamai Technologies.

   Tasks done:
   - Developed an Intel SGX application which creates a function inside the enclave which can be accessed from an untrusted application by importing the signed dll of the enclave.
   - Developed a SQLite DB application which uses both the trusted and untrusted libraries for creation and maintenance of the DB.
   - Segregated the SQLite code to be run outside the Intel SGX enclave such as OS system call, thread creation and non-sgx library dependencies.