Intel SGX

Praveen Keshavamurthy
Aboobacker Rizwan

Why Intel SGX??

- Creating an app and hosting it in the cloud platform involves many levels of trust
  - Operating System, cloud provider, other applications in the cloud.
- Provides a huge attack surface with potential bad code for adversaries.
- Secure remote computation is an unsolved problem
- Hardware support reduces this trust level
  - Need to trust only the processor which substantially reduces the attack surface.
- OS being compromised is a powerful adversary.
- Are user-space applications immune to such compromise??
- Homomorphic Encryption - Performance Overheads.

Intel SGX - Solution to OS Adversary!!

- Secure remote computation by leveraging trusted processors to provide Trusted Execution Environment
- Applications run inside an Enclave in Processor Reserved Memory
- Enclave Application code executes in ring 3
- Enclave measurements (MRENCLAVE)
  - Secure hash over inputs to ECREATE EADD EXTEND
- Remote attestation for authenticating an enclave based on its measurement.

Research Activity till date on Intel SGX..

1. Microsoft Research
    a. Adapted library OS and benchmarked unmodified applications like SQL Server & Apache
    b. Secure implementation of Hadoop Framework using Intel SGX in mappers and reducers.
        i. Verifiable Confidential Cloud Computing framework.
        ii. Encrypt map and reduce functions and upload to the cloud
    c. Observing and Preventing Leakage in MapReduce
        i. Using large I/O buffers
    d. Moat: Verifying Confidentiality of Enclave Programs
        i. Due to vulnerabilities in the application
2. OpenSGX - Hypervisor adapted to emulate SGX feature
    a. Emulating at the instruction level by extending open source emulator QEMU
    b. TOR Networks was adapted using OpenSGX

Goals Of the Project

- Exploring Intel SGX Architecture
- Explore Intel SGX SDK by developing an application
- To create development guidelines for Intel SGX
- Adapt SQLite3 library to Intel SGX
- Benchmark the DB performance impact due to Intel SGX

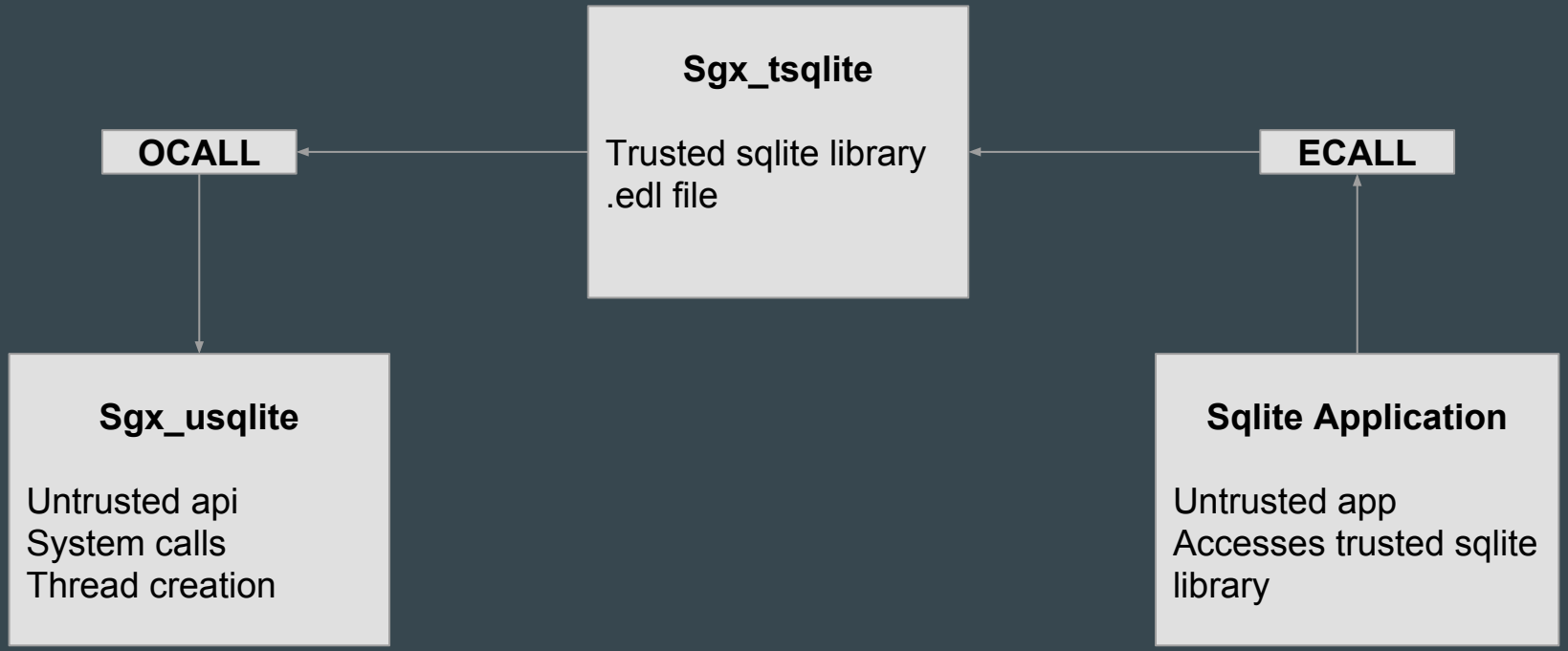
Intel SGX - Software Availability Status

1. Both SDK and Platform SW available for Windows
2. As of now intel SGX supports only C/C++ programs.
3. Expected Linux support in late Q2/Q3 2016 - Intel ISA Extension forum
4. Production Licensing is only issued for commercial softwares - Intel SGX Product License
5. A premature version of Linux PSW available for Intel SGX - https://github.com/jethrogb/sgx-utils
6. Microsoft Visual Studio Professional 2012 supports Intel SGX SDK, Intel SGX tools and Debugger.
7. Microsoft Visual Studio Premium 2012 supports Intel SGX SDK and Intel SGX tools except for SGX debugger.

Why SQLite3

- SQLite is a C library which the Intel SGX supports.
- SQLite is available for Windows.
- SQLite memory footprint ceils at 3 MB
  - Verified with varied database sizes.
  - Less than the maximum enclave size.
- SQLite has a cipher version that performs transparent and on-the-fly encryption
  - SQLCipher encryption engine can be developed using Intel SGX.
- SQLite is shipped as a dll which is the delivery mode for Intel SGX enclave.

Design Approach



Implementation Details

- Port the entire SQLite code into the Enclave dll.
- Segregate the OS call and move it to untrusted library.
- Move thread creation logic to the untrusted library.
- Create .edl file for:
  - Exposing the trusted API's in the Enclave to the application
  - Proxy routines for accessing untrusted library from the enclave
- Create a minimal SQLite application to verify SQLite SGX adaptations.

Challenges

- 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. (eg: sqlite3_exec())
  - As of now we have only exposed SQLite blocking api's 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.

Challenges (continued)

- 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.
- Visual Studio Premium 2012 doesn't support Intel SGX Debugger.
  - Used logging as an alternative for debugging.

Final status

- Exploring Intel SGX Architecture ✓
- Explore Intel SGX SDK by developing an application ✓
  - Explored the SGX build tools- sgx_edge8r, sgx_sign
- To create development guidelines for Intel SGX ✓
- Adapt SQLite3 library to Intel SGX ✘
  - Successful compilation of sgx_tsqlite, sgx_usqlite and the sqlite application. ✓
  - Successfully opened the database and the corresponding db file creation on disk ✓
  - Failed to execute database query due to Disk I/O error. ✘
- Benchmark the DB performance impact due to Intel SGX ✘

SGX Application Development Guidelines...

- Main mode of shipment of SGX binary is signed dll format.
- Author should use 3072-bit RSA key with public exponent set to 3 for signing.
- 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.
- SGX is not an end to end 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.

Future Scope

- Alternate Design Approach - Porting only the essential code into trusted library.
  - Pros
    - The challenges faced can be eliminated by implementing the logic outside enclave.
    - The enclave code will have very minimal dependency on the untrusted library.
  - Main Challenge:
    - Needs a complete understanding of the SQLite Architecture and Implementation.

- SQLiteCipher can be adapted to Intel SGX using the above mentioned approach.

DEMO

REFERENCES

- Git repo for the project: https://github.com/arizwanp/intel_sgx.git
- Victor Costan and Srinivas Devadas. Intel SGX Explained. https://eprint.iacr.org/2016/086.pdf
- Intel Software Guard Extensions. [Online] https://software.intel.com/sites/default/files/332680-002.pdf
- Intel Software Guard Extensions Programming Reference [Online] https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf
- Andrew Baumann, Marcus Peinado, and Galen Hunt, Microsoft Research. Shielding Applications from an Untrusted Cloud with Haven
- 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.