Local attestation

Peng Xu

May 7, 2019
1. What is attestation?
   ▶ Attestation is a mechanism for software to prove its identity
   ▶ Attestation is the process of demonstrating that a piece of software has been established on a platform
1. What is attestation?
   - Attestation is a mechanism for software to prove its identity
   - Attestation is the process of demonstrating that a piece of software has been established on a platform

2. Why we need attestation?
   - To prove to a remote party that your operating system and application software are intact and trustworthy
1. What is attestation?
   - Attestation is a mechanism for software to prove its identity
   - Attestation is the process of demonstrating that a piece of software has been established on a platform

2. Why we need attestation?
   - To prove to a remote party that your operating system and application software are intact and trustworthy

3. How can we implement attestation by Intel SGX?
   - Local (Intra-platform) attestation
     - a mechanism for creating a basic assertion between enclaves running on the same platform
   - Remote (Inter-platform) attestation
     - a mechanism that provides the foundation for attestation between an enclave and a remote third party
SGX Application - Local Attestation

Figure: local_attestation

https://software.intel.com/en-us/node/702983
SGX Application - Remote Attestation

Figure: remote_attestation

https://software.intel.com/en-us/node/702987
1. Attestation application structure
   ▶ Enclave(secure_world)
   ▶ App(Non-secure_world)
   ▶ Makefile
   ▶ Include
SGX Application - Attestation

1. Attestation application structure
   - Enclave(secure_world)
   - App(Non-secure_world)
   - Makefile
   - Include

2. Basic concepts
   - Public key/Private key
   - Signature
   - Service provider
   - etc.
1. App
   ▶ App.cpp
     ▶ main() - three modules
1. **App**
   ▶ **App.cpp**
     ▶ **main()** - three modules

2. **Enclave**
   ▶ **Enclave*.h/Enclave*.cpp**
   ▶ **Enclave*.edl**
   ▶ **Enclave*_private.pem**
   ▶ **Enclave* configures**
SGX Application - Local attestation

1. App
   ▶ App.cpp
     ▶ main() - three modules

2. Enclave*
   ▶ Enclave*.h/Enclave*.cpp
   ▶ Enclave*.edl
   ▶ Enclave*_private.pem
   ▶ Enclave* configures

3. LocalAttestationCode
   ▶ EnclaveMessageExchange*.h/*.cpp
   ▶ LocalAttestationCode.edl
SGX Application - Local attestation

1. App
   - App.cpp
     - main() - three modules

2. Enclave*
   - Enclave*.h/Enclave*.cpp
   - Enclave*.edl
   - Enclave*_private.pem
   - Enclave* configures

3. LocalAttestationCode
   - EnclaveMessageExchange*.h/*.cpp
   - LocalAttestationCode.edl

4. Untrusted_LocalAttestation
   - UntrustedEnclaveMessageExchange*.h/*.cpp
1. App
   ▶ App.cpp
   ▶ main() - three modules

2. Enclave*
   ▶ Enclave*.h/Enclave*.cpp
   ▶ Enclave*.edl
   ▶ Enclave*_private.pem
   ▶ Enclave* configures

3. LocalAttestationCode
   ▶ EnclaveMessageExchange*.h/*.cpp
   ▶ LocalAttestationCode.edl

4. Untrusted_LocalAttestation
   ▶ UntrustedEnclaveMessageExchange*.h/*.cpp

5. Include

6. Makefile
1. EnclaveInitialize

2. Logical functionalities
   - Enclave* _test_create_session(…)
   - Enclave* _test_enclave_to_enclave_call(…)
   - Enclave* _test_message_exchange(…)
   - Enclave* _test_close_session(…)

3. EnclaveDestroy
Enclave*/Enclave*.edl

1. from * import *
2. trusted
   - public uint32_t test_create_session(...);
   - public uint32_t test_enclave_to_enclave_call(...);
   - public uint32_t test_message_exchange(...);
   - public uint32_t test_close_session(...);
   - How to handle the return value in caller side?
Enclave*/Enclave*.edl

1. from * import *
2. trusted
   - public uint32_t test_create_session(...);
   - public uint32_t test_enclave_to_enclave_call(...);
   - public uint32_t test_message_exchange(...);
   - public uint32_t test_close_session(...);
   - How to handle the return value in caller side?
3. Caller and callee mapping
   - Enclave*_test_create_session(...) => test_create_session(...)???
   - Parameters mapping
     - Caller: func(e1_enclave_id, &ret_status, e1_enclave_id, e2_enclave_id)
     - Callee: func(sgx_enclave_id_t src_enclave_id, sgx_enclave_id_t dest_enclave_id)
1. function implementation
   ▶ uint32_t test_create_session(...){logic block}
   ▶ uint32_t test_enclave_to_enclave_call(...)
   ▶ uint32_t test_message_exchange(...)
   ▶ uint32_t test_close_session(...)

2. logic blocks
1. trusted
   - public uint32_t session_request(...);
   - public uint32_t exchange_report(...);
   - public uint32_t generate_response(...);
   - public uint32_t end_session(...);

2. untrusted
   - uint32_t session_request_ocall(...);
   - uint32_t exchange_report_ocall(...);
   - uint32_t send_request_ocall(...);
   - uint32_t end_session_ocall(...);
1. ECALL
   - `sgx_status_t Enclave*_session_request(...);`
   - `sgx_status_t Enclave*_exchange_report(...);`
   - `sgx_status_t Enclave*_generate_response(...);`
   - `sgx_status_t Enclave*_end_session(...);`

2. OCALL
   - `uint32_t session_request_ocall();`
   - `uint32_t exchange_report_ocall();`
   - `uint32_t send_request_ocall();`
   - `uint32_t end_session_ocall();`
Makefile

1. SDK setting
2. App compiling setting
3. Enclave compiling setting
   - (SGX_EDGER8R) --use-prefix --trusted
     ../Enclave1/Enclave1.edl
   - (SGX_EDGER8R) --trusted
     ../LocalAttestationCode/LocalAttestationCode.edl
   - (SGX_EDGER8R) --use-prefix --untrusted
     ../Enclave1/Enclave1.edl

4. Edger8r setting explain
   - --use-prefix: Prefix the untrusted proxy with the enclave name
   - --trusted: Generate trusted proxy and bridge routines only
   - --untrusted: Generate untrusted proxy and bridge routines only

https://download.01.org/intel-sgx/linux-2.3.1/docs/IntelSGXDeveloperReferenceLinux2.3.1OpenSource.pdf
Makefile

1. SDK setting

2. App compiling setting

3. Enclave compiling setting
   - (SGX_EDGER8R) –use-prefix –trusted
     ../Enclave1/Enclave1.edl
   - (SGX_EDGER8R) –trusted
     ../LocalAttestationCode/LocalAttestationCode.edl
   - (SGX_EDGER8R) –use-prefix –untrusted
     ../Enclave1/Enclave1.edl

4. Edger8r setting explain
   - –use-prefix: Prefix the untrusted proxy with the enclave name
   - –trusted: Generate trusted proxy and bridge routines only
   - –untrusted: Generate untrusted proxy and bridge routines only
   - https://download.01.org/intel-sgx/linux-2.3.1/docs/Intel_SGX_Developer_Reference_Linux_2.3.1_Open_Source.pdf
Question?

Questions?