Getting Started with OpenEnclave

*September 4, 2017*

# Introduction

This document provides a step-by-step tutorial to begin using the OpenEnclave SDK. It explains how to obtain, build, and install the SDK. It also describes how to build a few simple enclave applications.

# Licenses

Microsoft plans to release the OpenEnclave SDK under the MIT license, included here in the source distribution.

<https://github.com/Microsoft/openenclave/blob/master/LICENSE>

OpenEnclave builds on various third-party packages. It modifies and redistributes **libunwind** and in addition downloads other third-party packages on-the-fly during the build process. Licensing details for all third-party packages shown in the table below.

|  |  |
| --- | --- |
| **Package** | **License** |
| dlmalloc | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/dlmalloc/LICENSE> |
| musl libc | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/musl/COPYRIGHT> |
| OpenSSL | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/openssl/LICENSE> |
| libcxx | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/libcxx/LICENSE> |
| libcxxrt | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/libcxxrt/LICENSE> |
| libunwind | <https://github.com/Microsoft/openenclave/blob/master/3rdparty/libunwind/LICENSE> |

# Obtaining the source distribution

OpenEnclave is available from Github. Use the following command to download the source distribution.

# git clone https://github.com/Microsoft/openenclave

This creates a source tree under the directory called **openenclave**.

# Quick Start

Chapters 5 through 7 discuss prerequisites, building, and installing in some detail. This chapter explains how to perform these steps quickly when one wishes to install OpenEnclave into the default location (**/opt/openenclave**). If this suffices, then perform the steps below, skip those chapters and proceed to chapter 8.

## Prerequisites

Execute the following command from the root of the source tree to install the prerequisites (required packages, the SGX driver, and the SGX AESM service).

# make prereqs

## Building

To configure for installation into the default location and to build, type the following command.

# ./configure

# make

## Installing

The following command install OpenEnclave in the default location (**/opt/openenclave**).

# make install

# Prerequisites

The following are prerequisites for building and running OpenEnclave.

* Intel® X86-64bit architecture with SGX1 or SGX2
* Ubuntu Desktop-16.04-LTS 64bits
* Various packages: build-essential, ocaml, automake, autoconf, libtool, wget, python, libssl-dev, libcurl4-openssl-dev, protobuf-compiler, libprotobuf-dev, build-essential, python, libssl-dev, libcurl4-openssl-dev, libprotobuf-dev, uuid-dev, libxml2-dev, cmake, pkg-config
* Intel® SGX Driver (**/dev/isgx**)
* Intel® SGX AESM Service (from the Intel® SGX SDK)

Once Linux and the various packages are installed, it is necessary to install the **SGX driver** and the **SGX AESM service**. These can be obtained from the following Github repositories.

* <https://github.com/01org/linux-sgx-driver>
* <https://github.com/01org/linux-sgx>

Both contain detailed instructions about building and installing these pieces. As a convenience, OpenEnclave provides a script for downloading, building and installing both the driver and the AESM service. From the root of the OpenEnclave source tree, type the following command:

# make prereqs

After this completes, verify that the AESM service is running as follows.

# service aesmd status

Look for the string “active (running)”, usually highlighted in green.

# Building

To build the OpenEnclave SDK, type the following command from the root of the source tree.

# ./configure

.

.

.

Configured for x86\_64-ubuntu-linux-gnu

This configures for installation in the default location (**/opt/openenclave**). Configure provides options to install components in alternative locations. Use the --help option to display options for doing this. Once configured, just type make to build everything.

# make

This builds the entire OpenEnclave SDK, creating the following files.

|  |  |
| --- | --- |
| **Filename** | **Description** |
| lib/host/liboehost.a | Library for building host applications |
| lib/enclave/liboeenclave.a | Core library for building enclave applications |
| lib/enclave/liboelibc.a | C runtime library for enclave |
| lib/enclave/liboelibcxx.a | C++ runtime library for enclave |
| bin/oesign | Utility for signing enclaves |
| bin/oegen | Utility for generating ECALL and OCALL stubs from IDL |

Now that everything is built, try running the tests.

# make tests

# Installing

To install the OpenEnclave SDK, type this command.

# make install

Created /opt/openenclave/lib/openenclave

Created /opt/openenclave/include/openenclave

Created /opt/openenclave/share/openenclave

Created /opt/openenclave/share/openenclave/enclave.mak

Source /opt/openenclave/share/openenclave/environment to initialize the OpenEnclave environment

By default, all files are installed under **/opt/openenclave**. Source the given **environment** script to update the **PATH** and to define environment variables used by makefiles.

The following table shows where key components are installed.

|  |  |
| --- | --- |
| **Path** | **Description** |
| /opt/openenclave/lib/openenclave/enclave | Enclave libraries |
| /opt/openenclave/lib/openenclave/host | Host libraries |
| /opt/openenclave/include/openenclave/enclave | Enclave includes |
| /opt/openenclave/include/openenclave/host | Host includes |
| /opt/openenclave/bin | Programs |
| /opt/openenclave/share/openenclave | Data files |

# Samples

Above the samples were installed here: **/opt/openenclave/share/openenclave/samples**. Copy these to another location. For example:

# cp -r /opt/openenclave/share/openenclave/samples /home/john/samples

Next source the environment script as follows:

# source /opt/openenclave/share/openenclave/environment

Finally, change to the new samples directory and build and run the samples.

# cd /home/john/samples

# make

# make run

.

.

.

If these samples run without an error, then OpenEnclave is installed and working correctly.

# Uninstalling

To uninstall OpenEnclave, use the **oeuninstall** script (in the installed bin directory). For example, to run it from the default location do this.

# source /opt/openenclave/bin/oeuninstall

This script silently removes all installed components.

# Developing a simple enclave (echo)

This chapter shows how to develop a simple enclave called **echo**. The next chapter explains how to use this enclave in a host application. This example is included in the installed samples directory (see **/opt/openenclave/share/openenclave/samples/hello**).

## The ECALL

The **echo** enclave implements a single **ECALL** named **EnclaveEcho()**, which is called by the host (in the next chapter). This function has the following signature.

OE\_ECALL void EnclaveEcho(void\* args);

The **args** parameter can be whatever the host and the enclave agree on. In this case **args** is a pointer to a zero-terminated string. The **OE\_ECALL** macro exports the function and injects it into a special section (**.ecall**) in the ELF image. When the host loads the enclave, it builds a table of all **ECALLs** exported by the enclave.

## The Echo enclave Listing

Here’s the full listing for the echo enclave (**enc.c**):

#include <openenclave.h>

OE\_ECALL void EnclaveEcho(void\* args)

{

OE\_CallHost("HostEcho", args);

}

Notice **EnclaveEcho()** performs an **OCALL**, calling the host’s **HostEcho()** function with the same arguments.

## Compiling enc.c

This sample includes a makefile for building this enclave, but to be more instructive, this chapter shows how to build components from scratch. First, we define the **INCLUDES** make variable as follows.

INCLUDES=-I/opt/openenclave/include/openenclave/enclave

This is the directory that contains the **<openenclave.h>** header, as well as C headers files.

Next, we define the **CFLAGS** make variable as follows.

CFLAGS=-O2 -nostdinc -fPIC

Finally, we compile the source file.

gcc -c $(CFLAGS) $(INCLUDES) enc.c

This produces **enc.o**.

## Linking the enclave

Next, we link the enclave to produce **echoenc.so**. First, we define the **LDFLAGS** make variable.

LDFLAGS=\

-nostdlib \

-nodefaultlibs \

-nostartfiles \

-Wl,--no-undefined \

-Wl,-Bstatic \

-Wl,-Bsymbolic \

-Wl,--export-dynamic \

-Wl,-pie \

-Wl,-eOE\_Main

The **-eOE\_Main** option requires some explanation (see the **ld** man page about other options). This option specifies the name of the entry point for the enclave. The linker stores the virtual address of the **OE\_Main()** function in the ELF header (**Elf64\_Ehdr.e\_entry**) of the resulting binary. When the enclave is instantiated by the host, this entry point is copied to each **TCS** (Thread Control Structure) in the image. When the host invokes the **SGX EENTER** instruction on a TCS, the hardware fetches the entry point from the TCS and jumps to that address and the **OE\_Main()** function begins to execute.

Next, we define the **LIBRARIES** make variable.

LIBRARIES=-L/opt/openenclave/lib/openenclave/enclave -loeenclave

The **LIBRARIES** make variable specifies the enclave library, which contains the enclave intrinsics, including the **OE\_Main()** entry point. Note that this sample uses neither a C or C++ runtime library. Other samples will show how these are used.

Finally, we link the enclave.

gcc $(LDFLAGS) $(LIBRARIES) enc.o -o echoenc.so

## Signing the enclave

The final step in creating an enclave is to sign it with the **oesign** tool. This tool takes the following parameters.

# oesign

Usage: oesign ENCLAVE CONFFILE KEYFILE

The **CONFFILE** argument is the name of a configuration file that defines enclave settings, such as stack size, heap size, and the maximum number of threads (TCSs). Here is a sample:

# echo.conf

Debug=1

NumHeapPages=1024

NumStackPages=1024

NumTCS=16

The **KEYFILE** argument is a private RSA key used to sign the enclave.

A self-signed private key can be generated using OpenSSL as follows.

# openssl genrsa -out private.pem -3 3072

Then the public key can be generated from this key as follows.

# openssl rsa -in private.pem -pubout -out public.pem

Finally, we sign the enclave as follows.

# oesign echoenc.so echo.conf private.pem

Created echoenc.signed.so

# Developing a simple host (echohost)

Next, we develop a host to run the **echoenc.signed.so** enclave that we developed in the previous chapter. The listing follows.

#include <openenclave.h>

#include <stdio.h>

OE\_OCALL void **HostEcho**(void\* args)

{

if (args)

{

const char\* str = (const char\*)args;

printf("%s\n", str);

}

}

int **main**(int argc, const char\* argv[])

{

OE\_Result result;

OE\_Enclave\* enclave = NULL;

if (argc != 2)

{

fprintf(stderr, "Usage: %s ENCLAVE\_PATH\n", argv[0]);

return 1;

}

result = **OE\_CreateEnclave**(argv[1], OE\_FLAG\_DEBUG, &enclave);

if (result != OE\_OK)

{

fprintf(stderr, "%s: OE\_CreateEnclave(): %u\n", argv[0], result);

return 1;

}

result = **OE\_CallEnclave**(enclave, "EnclaveEcho", "Hello");

if (result != OE\_OK)

{

fprintf(stderr, "%s: OE\_CallEnclave(): %u\n", argv[0], result);

return 1;

}

**OE\_TerminateEnclave**(enclave);

return 0;

}

This host performs the following tasks:

* Defines an **OCALL**: **HostEcho()**
* Instantiates an enclave: **OE\_CreateEnclave()**
* Calls into the enclave: **OE\_CallEnclave()**
* Terminates the enclave: **OE\_TerminateEnclave()**

After building the host application, we are ready to run the host.

# host/echohost ./enc/echoenc.signed.so

Hello