

Programming Assignment 1 - Creating an Intel SGX Enclave

- Due 2 Mar 2023 by 23:59
- Points 100
- Submitting a media recording
- Available 14 Feb 2023 at 23:59 - 2 Mar 2023 at 23:59

This assignment was locked 2 Mar 2023 at 23:59.

Assignment 1- Setting Up an Intel SGX Enclave

version 1.0 (last modified Oct 8, 2019)

Part 1

You may run on machine with Intel SGX compatible CPU, or Virtual machine using the simulation mode. If running on AMD hardware, SGX will not run on your computer even in simulation mode on Virtual Machine.

The preferred platform for this assignment is Linux, although there are ways to make the assignment work on other Operating Systems (OSes). Refer to the following notes:

On Windows OS

SGX works on Windows, however the files differ greatly from the Linux files and do not use makefiles. If you want to create an enclave on Windows OS, you must run on Linux virtual machine (VM) on Windows , and run SGX using simulation mode. Create a VM using Virtual Box, then follow Linux Instructions.

Mac

SGX does not work on Macs. Therefore one must use a VM and simulation mode. These VM should be Linux. Create VM using Parallel Box or Virtual Box, then follow Linux Instructions.

Linux

Resources

[Developer Guide](https://download.01.org/intel-sgx/linux-2.5/docs/Intel_SGX_Developer_Guide.pdf)  (https://download.01.org/intel-sgx/linux-2.5/docs/Intel_SGX_Developer_Guide.pdf)


[Installation Guide](https://download.01.org/intel-sgx/linux-2.5/docs/Intel_SGX_Installation_Guide_Linux_2.5_Open_Source.pdf)  (https://download.01.org/intel-sgx/linux-2.5/docs/Intel_SGX_Installation_Guide_Linux_2.5_Open_Source.pdf)

[Install Files](https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.5-release)  (https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.5-release)

[Eclipse Plug](https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip)  (https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip) [i](https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip)  [n](https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip) 

(https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip)

Installing SGX Sample

The Example commands and links below are for Installation on Ubuntu 18.04. The command may be slightly different for other Linux versions. The installation files can be downloaded from the Install Files [link](https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.5-release)  (<https://01.org/intel-softwareguard-extensions/downloads/intel-sgx-linux-2.5-release>).

If you are going to be running on Non-Compatible Hardware or in a VM, only follow the Instructions for the SDK, skip the PSW and driver. (steps 4 & 5)

1. Download the Needed Installation files

- Intel SGX driver - [bin](https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/sgx_linux_x64_driver_f7dc97c.bin)  (https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/sgx_linux_x64_driver_f7dc97c.bin)
- Intel SGX PSW - [libsgx-enclave-common_2.5.101.50123-bionic1_amd64.deb](https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/libsgx-enclave-common_2.5.101.50123-bionic1_amd64.deb)  (https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/libsgx-enclave-common_2.5.101.50123-bionic1_amd64.deb)
- Intel SGX SDK - [5.100.49891.bin](https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/sgx_linux_x64_sdk_2.5.100.49891.bin)  (https://download.01.org/intel-sgx/linux-2.5/ubuntu18.04-server/sgx_linux_x64_sdk_2.5.100.49891.bin)

2. Install the following Prerequisites (refer to the Installation Guide for non Ubuntu OS)

- `sudo apt-get update`
- `sudo apt-get upgrade`
- `sudo apt-get install libssl-dev libcurl4-openssl-dev libprotobuf-dev`
- `sudo apt-get install build-essential python`

3. Install SGX Driver Package

- `sudo chmod +x sgx_linux_x64_driver_f7dc97c.bin`
- `sudo ./sgx_linux_x64_driver_f7dc97c.bin`

4. Install SGX PSW Package

- `sudo dpkg -i ./libsgx-enclave-common_2.5.101.50123-bionic1_amd64.deb`

5. Install SGX SDK

- `sudo chmod +x sgx_linux_x64_sdk_2.5.100.49891.bin`
- `sudo ./sgx_linux_x64_sdk_2.5.100.49891.bin`
- When the installer ask **“Do you want to install in current directory? [yes/no]”** Enter **“no”**
- `/opt/intel` – as the location to Install the SDK
- `source /opt/intel/sgxsdk/environment`

6. Edit the `.bashrc` file to include the `source /opt/intel/sgxsdk/environment` commands

- `nano ~/.bashrc`
- Add line - `source /opt/intel/sgxsdk/environment`

Installing Eclipse Plug-In

1. Install Eclipse, this can be done through the Ubuntu Software store
2. Download the Eclipse Plug In [File !\[\]\(511a36c244659513b679df9c639945de_img.jpg\) \(https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip\)](https://download.01.org/intel-sgx/linux-2.5/sgx_eclipse_plugin_2.5.100.49891.zip)
3. Go to **Help menu > Install New Software**. Click the **Add** button for the **Work with** field to open the **Add Repository** dialog
4. Enter *Intel (R) SGX Archive* in the **Name**. Click the **Archive...** button and select the location of the downloaded archive
5. Press **OK** to add the Archive
6. In the **Install** dialog, select the **Intel(R) Software Guard Extension Plugin** checkbox and proceed as normal.
7. Go to **Window Menu -> Preferences**. Select the **Intel SGX Preference** page
8. Enter the path for the SGX SDK in the **Intel SGX SDK Directory** field

Testing Intel SGX Installation

1. Navigate into the Intel SGX SDK Sample Code (`/opt/intel/sgxsdk/SampleCode/SampleEnclave`)
 - `cd '/opt/intel/sgxsdk/SampleCode/SampleEnclave'`
2. Run the Sample Enclave
 - `sudo make SGX_MODE=SIM SGX_DEBUG=0`
 - `./app`

Part 2

Resources

In Eclipse Go To **Help -> Help Contents** and Select the **Intel SGX Plugin Developer Guide**

Creating Sample Project

1. **File -> New Project**, In the **C/C++ projects with Intel SGX Enabled** select the **C++ Project**
2. Enter a Project name and Select the **Intel SGX Tool** chain, and click **Finish**

3. Right Click on the Project, in the project explorer and navigate down to **Intel Software Guard Extension Tools -> Add Intel SGX Enclave**, name the enclave **Main** and hit **OK**
4. Build the project using the hammer, first select the drop down next to the hammer and Select **Intel SGX Simulation Debug**
5. Navigate into the enclave in the terminal, and run `./sample`

Defining New Trusted Function

1. In the trusted folder, Open **Main.cpp** and **Main.edl**
2. Create a function in **Main.cpp** that receives a pointer..... (to define a function)
3. Create a function prototype in **Main.edl** for the function that was just created, in the trusted subsection. Add the needed control statements to ensure the function can do what is needed without possibly leaking secure data.
4. Call this function from **sample.cpp** in the untrusted folder, this should be done in main after the enclave is created and before it is closed.

Calling Untrusted Function from a Trusted Function

1. In the untrusted folder, Open **sample.cpp**
2. Create a function in **sample.cpp** that receives a pointer..... (to define a function)
3. Create a function prototype in **Main.edl** for the function that was just created, in the untrusted subsection. Add the needed control statements to ensure the function can do what is needed without possibly leaking secure data.
4. Call this function from **Main.cpp** in the trusted folder, from within the **ecall_Main_sample()** function.

Creating Another Untrusted File

1. In the untrusted folder, create a new C++ file
2. Create a function in the new C++ file, if we were just going to call the function from another untrusted C++ file we would add the prototype to the header file, as we are going to add it to **Main.edl** and we don't want conflicting definitions
3. Copy the includes from **sample.cpp**, into the new C++ file
 - `#include "Main_u.h"`
4. Create a function prototype in **Main.edl** for the function that was just created, in the untrusted subsection. Add the needed control statements to ensure the function can do what is needed without possibly leaking secure data.
5. Call the new function from both the trusted **Main.cpp** and untrusted **sample.cpp**
6. Edit the **sgx_u.mk** file to compile both untrusted C++ file (sample.cpp and the newly created one)
 - edit line 47, to match the following

App_Cpp_Files := \$(UNTRUSTED_DIR)/sample.cpp \$(UNTRUSTED_DIR)/<new file>.cpp

- If you are having issues building you app, try cleaning before building, this can be done by going **Project -> Clean**

Creating Another Trusted File

1. In the trusted folder, create a new C++ file with a matching header file
2. Create a function in the new C++ file, if we were just going to call the function from another trusted C++ file we would add the prototype to the header file, as we are going to add it to **Main.edl** and we don't want conflicting definitions
3. Copy the includes from Main.cpp, into the new C++ file
 - *#include "Main_t.h"*
4. Create a function prototype in **Main.edl** for the function that was just created, in the trusted subsection. Add the needed control statements to ensure the function can do what is needed without possibly leaking secure data.
5. Call the new function from both the trusted **Main.cpp** and untrusted **sample.cpp**
6. Edit the **sgx_t.mk** file to compile both untrusted C++ file (sample.cpp and the newly created one)
 - edit line 47, to match the following

Main_Cpp_Files := \$(TRUSTED_DIR)/sample.cpp \$(Trusted_DIR)/<new file>.cpp

- If you are having issues building you app, try cleaning before building, this can be done by going **Project -> Clean**

Adding a Trusted Static Library

1. Right Click on the Project, in the project explorer and navigate down to **Intel Software Guard Extension Tools -> Add Trusted Static Library** name the library **lib1** and hit **OK**, this library by default has a sample call that we can use for testing called **ecall_lib1_sample()**
2. Call **ecall_lib1_sample()** from the trusted **Main.cpp**, and the untrusted **sample.cpp**
3. Import the libraries edl, **lib1.edl**, into **Main.edl**, allowing us to call the function from main

from "../trustedlib_lib1/static_trusted/lib1.edl" import;*

4. Edit the enclaves **sgx_t.mk** file line 120 add the following to the requirements for **Main.so**
../trustedlib_lib1/liblib1.sgx.static.lib.a
5. Edit the **Makefile** in the **sgx** folder, so that it compiles the static make files before all the rest, move line 13 above line 11



Submission Instructions:


Please demo your project. by screen capturing your program source code and execution with narration, ensuring to include the following items:

1. EDL File with Function Prototypes
2. New Functions Created in
 - New Trusted File
 - New Untrusted File
 - Trusted File
 - Untrusted File
3. The Trusted Static Library Has been Created
4. Compiling the Program
5. Running the Program

Free Screen Recording Tools

Mac – [Screenshot Tool](https://support.apple.com/en-us/HT208721)  (https://support.apple.com/en-us/HT208721) (Preinstalled)

Windows – [PowerPoint](https://support.office.com/en-us/article/record-your-screen-in-powerpoint-0b4c3f65-534c-4cf1-9c59-402b6e9d79d0)  (https://support.office.com/en-us/article/record-your-screen-in-powerpoint-0b4c3f65-534c-4cf1-9c59-402b6e9d79d0) has a screen recording option, or [Windows Game Bar](https://betanews.com/2019/01/14/windows-10-screen-recorder-utility/)  (https://betanews.com/2019/01/14/windows-10-screen-recorder-utility/) (Preinstalled)

Linux – [Kazam](https://launchpad.net/kazam)  (https://launchpad.net/kazam), there are a lot out there this is just one I have used

Other Useful Links:

[SGX Emulation](https://www.intel.com/content/www/us/en/developer/articles/training/usage-of-simulation-mode-in-sgx-enhanced-application.html)  (https://www.intel.com/content/www/us/en/developer/articles/training/usage-of-simulation-mode-in-sgx-enhanced-application.html)