

[Xiangyi's Blog](#)  
[Articles for research, development and something else.](#)  
[Archive](#) [Categories](#) [Tags](#) [About](#) [Poems](#)  
Hugo Theme [Diary](#) by [Rise](#)  
Ported from [Makito's Journal](#).

© Copyright by Xiangyi Meng 2021.

- CATALOG -

- [enable\\_sw\\_sgx.cpp](#)



[Archive](#) [Categories](#) [Tags](#) [About](#) [Poems](#)

- CATALOG -

- [enable\\_sw\\_sgx.cpp](#)

## Enable Software Controlled SGX in Ubuntu

2020-09-25 13:45 [Programming](#) [C](#) [SGX](#)

Intel® Software Guard Extensions (SGX) is a hardware-based isolation and memory encryption mechanism provided by modern Intel® CPUs. Normally, it is disabled in the BIOS by the manufacture of your motherboard. In order to use it, the SGX option in the BIOS must be set to **Enable** or **Software Controlled**.

By setting the option to **Enable**, all of the SGX instructions and resources are available to applications, making it easy to deploy SGX related program on your machine. However, in some motherboards, the only available options in the BIOS are **Software Controlled** and **Disable**. According to the [official document of Intel](#), **Software Controlled** indicates that *Intel SGX can be enabled by software applications, but it is not available until this occurs (called the “software opt-in”)*.

So now the question is, how to make **this** occurs? Actually, two functions are provided for us to enable SGX by program:

```
sgx_status_t sgx_is_capable(int *sgx_capable);
sgx_status_t sgx_cap_enable_device(sgx_device_status_t *sgx_device_status);
```

**sgx\_is\_capable** determines whether the system is *capable* of executing Intel SGX instructions under the current operating environment (Is secure boot disabled? Does your CPU support SGX?...) The return value is an indicator showing that whether the inquiry is successful. If it returns **SGX\_SUCCESS**, the function successfully queried the SGX support characteristic of your machine and stored it into **sgx\_capable** (of course, 1 means yes, 0 means no). A return of it is **SGX\_ERROR\_NO\_PRIVILEGE** means the user has no root/administrator privilege. Any other return value means that the SGX capability of this machine could not be determined.

(Notes: Your Ubuntu must be installed and booted in UEFI mode)

**sgx\_cap\_enable\_device** attempts the software opt-in for the SGX and set the final state of SGX in **sgx\_device\_status**. This is a mandatory procedure for **Software Controlled** motherboards to fully enable the SGX. The meaning of the return value is identical to that of **sgx\_is\_capable**. And the inquiry result will be stored into **sgx\_device\_status** only if the software opt-in is attempted.

The meaning of the output value is listed here:

- SGX\_ENABLED = 0
- SGX\_DISABLED\_REBOOT\_REQUIRED = 1, /\* A reboot is required to finish enabling SGX \*/
- SGX\_DISABLED\_LEGACY\_OS = 2, /\* SGX is disabled and a Software Control Interface is not available to enable it \*/
- SGX\_DISABLED = 3, /\* SGX is not enabled on this platform. More details are unavailable. \*/
- SGX\_DISABLED\_SCI\_AVAILABLE = 4, /\* SGX is disabled, but a Software Control Interface is available to enable it \*/
- SGX\_DISABLED\_MANUAL\_ENABLE = 5, /\* SGX is disabled, but can be enabled manually in the BIOS setup \*/
- SGX\_DISABLED\_HYPERV\_ENABLED = 6, /\* Detected an unsupported version of Windows\* 10 with Hyper-V enabled \*/
- SGX\_DISABLED\_UNSUPPORTED\_CPU = 7, /\* SGX is not supported by this CPU \*/

Now using these functions, we can write the following sample code to fully enable the software controlled SGX:

### enable\_sw\_sgx.cpp

```
#include <stdio.h>
#include "../common/inc/sgx_capable.h"

int main()
{
    int is_sgx_capable = 0;
    sgx_device_status_t status;

    sgx_is_capable(&is_sgx_capable);
    printf("is_sgx_capable: %d\n", is_sgx_capable);

    sgx_cap_enable_device(&status);
    printf("status: %d\n", (int)status);

    return 0;
}
```

In order to compile the above code, we need to tell the compiler where the two functions are. Actually, they are declared in `linux-sgx/common/inc/sgx_capable.h`, and their implementations are in `linux-sgx/sdk/libcapable/linux/`. Here `linux-sgx` is [this](#) Github repository.

As the libcapable rely on some other libraries in the repo, we need to compile it to a dynamic link library first by simply running

```
$ make
```

inside the `linux-sgx/sdk/libcapable/linux/` directory. Then there should be a `libsgx_capable.so` library in this directory. Now save your `enable_sw_sgx.cpp` inside the same directory and compile it with the following command:

```
$ gcc enable_sw_sgx.cpp -o enable_sw_sgx -L. -lsgx_capable
$ sudo LD_LIBRARY_PATH=. ./C
```

Now the output should be

```
$ sudo LD_LIBRARY_PATH=. ./enable_sw_sgx
is_sgx_capable: 1
status: 1
```

Let's give a review of the meaning of `sgx_device_status`. 1 is SGX\_DISABLED\_REBOOT\_REQUIRED, which means **A reboot is required to finish enabling SGX**. And after rebooting your machine, if no error occurs, you may run `enable_sw_sgx` again and the output should be

```
$ sudo LD_LIBRARY_PATH=. ./enable_sw_sgx
is_sgx_capable: 1
status: 0
```


This indicates that SGX is fully enabled in your system. Now you can start any work with SGX enabled.

Last modified on 2020-09-25

- Next
- [Combinations of Binary Number Given the Number of 1 and 0](#)
- [Previous](#)
- [How to Fix Failed to Initialize NVML Error in Ubuntu 20.04](#)

0 comments

Anonymous ▾



Leave a comment

Markdown is supported

Login with GitHub

Preview

Be the first person to leave a comment!

Hugo Theme [Diary](#) by [Rise](#)  
Ported from [Makito's Journal](#).

© Copyright by Xiangyi Meng 2021.