EmSPARK™ Security Suite

Evaluation Kit - Getting Started Guide for Jetson Xavier NX Developer Kit

Date March 18, 2022 | Version 1.0

TABLE OF CONTENTS

1. INTRODUCTION

This **Getting Started Guide** is an overview of the prerequisites to use the **EmSPARK™ Suite Evaluation Kit**, description of the Kit contents, and installation instructions on a Jetson Xavier NX Developer Kit. After completing the installation process in this guide, please see the `CORELOCKR_LIBRARIES_GUIDE.pdf` which provides an overview of the CoreLockr™ APIs. The security features of CoreTEE™ in the Secure Domain and the CoreLockr™ APIs in the Rich OS are supported after flashing the device. The installation also includes the root filesystem which has preinstalled the EmSPARK™ components for the Rich OS.

The EmSPARK™ Evaluation Kit demonstrates some of the credentials provisioned on the device. The firmware image in `EmSpark_Xavier-NX-Dev_Eval_[release].tar.gz` has preconfigured keys and certificates that typically a customer would configure in personalization data manifests when building the firmware. Other keys are device-specific generated during the first boot when they are not available.

To demonstrate security scenarios using such credentials for verification, the provided example applications include the private key files associated with public keys provisioned on the device. Client applications using the CoreLockr™ APIs can use the provisioned credentials for operations executed in the Trusted Execution Environment (TEE). In the case of the generated device-specific keys, Device Keys, the example applications illustrate how to use them in the TEE while the Rich OS has not access to the device key private attributes.

Devices flashed with the EmSPARK™ Security Suite are also EmPOWER™ enabled. EmPOWER™ is a SaaS solution that provides the essential cloud services needed to secure, provision, update and manage intelligent edge devices. For information about EmPOWER™ please contact Sequitur Labs.

The secure boot and update processes follow the NVIDIA Jetson flow. Please refer to the NVIDIA Jetson documentation for detail.

1.1. Prerequisites

This guide assumes that the following hardware and software are available:

- EmSPARK Suite, `EmSpark_Xavier-NX-Dev_Eval_[release].tar.gz`
  - Firmware image and flashing tool
  - EmSPARK CoreLockr Libraries for client application development
- Jetson Xavier NX Developer Kit
- Linux system which is the building environment (currently Ubuntu 16.04 LTS or 18.04 LTS) **[1]**
- Micro USB cable to connect the device to the Linux system
- MicroSD card, minimum 16GB, where the root filesystem is installed
- Either a serial console or peripherals attached to the device
  - FTDI UART cable such as FTDI TTL-232R-3V3 to connect to the serial terminal and to be able to see flashing progress
  - Connected peripherals to the device (HDMI, keyboard, and mouse)

1.2. EmSPARK™ Suite Package Contents

Download the Evaluation Kit package. Expand the package in a Linux environment:

```
tar -zxvf EmSpark_Xavier-NX-Dev_Eval_[release].tar.gz
```

The following directory structure is created:

- `flash` contains the firmware image ready for flashing. The firmware image has the EmSPARK components and is preconfigured with sample keys and certificates. This image is equivalent to the image that customers would produce with custom components in a development or production environment.
- `corelockr_package` contains the CoreLockr™ APIs. These are C libraries for development of client applications running in the Rich OS (Linux) and work with the Trusted Applications (TAs) running in the Trusted Execution Environment (CoreTEE™). The Kit includes:
  - `corelockr`, the CoreLockr libraries, TAs, example applications and API documentation
  - `coretee_dev_kit`, the toolchain and the client API for building the example applications
  - `CORELOCKR_LIBRARIES_GUIDE.pdf`, an overview of the CoreLockr libraries and tutorial to build and execute the example applications
  - `COPYRIGHT.txt`, the copyright notice
- `RELEASE_NOTES.txt`, information about the release
- `COPYRIGHT.txt`, the copyright notice
- `GETTING_STARTED.pdf`, this guide

1.3. Terminology

CoreLockr™	Sequitur's native C APIs for development of client applications that execute in the Rich OS and perform security-specific functions in the TEE

CoreTEE™	Sequitur's Trusted Execution Environment (TEE), or secure OS, enabled by ARM's TrustZone™ architecture.
Manifest (SLIP)	Encrypted component containing customer personalization data such as keys and certificates. They are installed on device along with the device firmware.
TEE	Trusted Execution Environment or secure OS, enabled by ARM's TrustZone™ architecture.

2. INSTALLATION PROCEDURE

To flash the device:

1. Start with the board powered off
2. Insert the microSD card in the module
3. Connect the micro USB port to the Linux host computer
4. If connecting the serial console to see the flashing process:
    a) On the board, the serial console is located on J14, the button header - pin 3 = RX, pin 4 = TX
    b) Connect the Linux host machine to the serial port
    c) On the Linux host machine open a serial terminal

```
115200 bps
No parity
8 bits
1 stop bit
No flow control
```

5. Put the Xavier NX Development board into forced Recovery Mode **[1]**:
    a) Power off the board
    b) Jump the Recovery Mode pins on the button header (pins 9 and 10 on J14)
    c) Power on the board
    d) Remove jumper
    e) To confirm the device is in recovery mode, on the Linux host computer, the `lsusb` command lists

```
ID 0955:7e19 NVidia Corp.
```

6. Flash the device:
    a) On the Linux host computer change to ~/flash
    b) Execute `sudo ./flash_device.sh`

During flashing, the Linux terminal prints messages signifying the process on the device. Flashing the image may take several minutes, and eventually prints messages like these:

```
[ 180.8176 ] [.............................................] 100%
[ 186.1234 ] Writing partition kernel-dtb with kernel_tegra194-p3668-all-p3509-0000_sigheader.dtb.encrypt
[ 186.1433 ] [.............................................] 100%
[ 186.1489 ] Writing partition kernel-dtb_b with kernel_tegra194-p3668-all-p3509-0000_sigheader.dtb.encrypt
[ 186.1872 ] [.............................................] 100%
[ 186.1924 ] Writing partition manifest with manifests.img
[ 186.2282 ] [.............................................] 100%
[ 186.2328 ] Writing partition APP with system.img
[ 186.2630 ] [.............................................] 100%
[ 844.2325 ]
[ 844.2356 ] tegradevflash_v2 --write BCT br_bct_BR.bct
[ 844.2370 ] Bootloader version 01.00.0000
[ 844.2390 ] Writing partition BCT with br_bct_BR.bct
[ 844.2396 ] [.............................................] 100%
[ 844.3819 ]
[ 844.3860 ] tegradevflash_v2 --write MB1_BCT mb1_cold_boot_bct_MB1_sigheader.bct.encrypt
[ 844.3872 ] Bootloader version 01.00.0000
[ 844.3893 ] Writing partition MB1_BCT with mb1_cold_boot_bct_MB1_sigheader.bct.encrypt
[ 844.3904 ] [.............................................] 100%
[ 844.5939 ]
[ 844.5967 ] tegradevflash_v2 --write MB1_BCT_b mb1_cold_boot_bct_MB1_sigheader.bct.encrypt
[ 844.5978 ] Bootloader version 01.00.0000
[ 844.5997 ] Writing partition MB1_BCT_b with mb1_cold_boot_bct_MB1_sigheader.bct.encrypt
[ 844.6008 ] [.............................................] 100%
[ 844.8045 ]
[ 844.8086 ] tegradevflash_v2 --write MEM_BCT mem_coldboot_sigheader.bct.encrypt
[ 844.8099 ] Bootloader version 01.00.0000
[ 844.8117 ] Writing partition MEM_BCT with mem_coldboot_sigheader.bct.encrypt
[ 844.8123 ] [.............................................] 100%
[ 846.0722 ]
[ 846.0767 ] tegradevflash_v2 --write MEM_BCT_b mem_coldboot_sigheader.bct.encrypt
[ 846.0792 ] Bootloader version 01.00.0000
[ 846.0824 ] Writing partition MEM_BCT_b with mem_coldboot_sigheader.bct.encrypt
[ 846.0834 ] [.............................................] 100%
[ 847.3435 ]
[ 847.3437 ] Flashing completed

[ 847.3438 ] Coldbooting the device
[ 847.3479 ] tegrarcm_v2 --ismb2
[ 847.3526 ]
[ 847.3548 ] tegradevflash_v2 --reboot coldboot
[ 847.3557 ] Bootloader version 01.00.0000
[ 847.3581 ]
```

The following are some of the processes that take place during flashing:

- Initialize CoreTEE
- CoreTEE installs personalization data contained in manifests and generates unique device identification
  - Load manifests containing keys and certs
  - Decrypt encrypted manifests
  - Generate device keys and certs
  - Generate OEM Device Key
  - Generate EmPOWER Device Key
  - Generate OEM Device CSR and OEM Device Certificate
  - Generate EmPOWER Device CSR and EmPOWER Device Certificate
- CoreTEE re-encrypts the manifest that contains keys and certs
  - Include in the manifest the newly generated device keys and certificates
- Write manifests to NVM

When the serial console is connected to the device, the console prints messages of the process during flashing and booting.

- The following is a subset of messages printed in the serial console alluding to processes during flashing:

```
[0216.490] I> Writing manifest partition.
...
[0002.115] I> Initializing CoreTEE...
[0002.118] I> Opening manifest
[0002.121] I> Initializing slip: 1 @ 0x00000000
I/TC:   Initializing slip: 1
I/TC:   Slip 1: SW ver = 0    HW ver = 0
I/TC:   Creating Device Key and Cert
I/TC:   Creating EmPower Device Key and Cert
[0002.364] I> CoreTEE Manifest initialization: 0x00000001
[0002.365] I> Writing Manifests back to NVM
[0002.375] I> Manifest written back to NVM
[0002.375] I> Opening manifest
[0002.376] I> Initializing slip: 2 @ 0x00010400
...
I/TC:   Initializing slip: 2
I/TC:   Slip 2: SW ver = 0    HW ver = 0
I/TC:   Diversifying manifest: 2
[0002.401] I> CoreTEE Manifest initialization: 0x00000001
[0002.402] I> Writing Manifests back to NVM
[0002.413] I> Manifest written back to NVM
[0002.414] starting app coretee_init_app
[0002.415] I> CoreTEE Initialized
```

- The following is a subset of messages during device boot, which does not modify the manifests, e.g.

```
[0002.103] I> Initializing CoreTEE...
[0002.106] I> Opening manifest
[0002.109] I> Initializing slip: 1 @ 0x00000000
[0002.114] I> HW Version: 0x00
[0002.128] I> OP: 0xb2000010
I/TC:   Initializing slip: 1
I/TC:   Slip 1: SW ver = 0    HW ver = 0
I/TC:   Device Key and Cert exist
[0002.154] I> CoreTEE Manifest initialization: 0x00000000
[0002.154] I> Manifests not modified
[0002.155] I> Opening manifest
[0002.155] I> Initializing slip: 2 @ 0x00010400
[0002.155] I> HW Version: 0x00
[0002.167] I> OP: 0xb2000010
I/TC:   Initializing slip: 2

I/TC:   Slip 2: SW ver = 0    HW ver = 0
[0002.182] I> CoreTEE Manifest initialization: 0x00000000
[0002.182] I> Manifests not modified
[0002.182] starting app coretee_init_app
[0002.183] I> CoreTEE Initialized
```

3. USING THE SYSTEM

After booting into Linux, on the device serial console or with connected peripherals:

- The user is prompted to enter access credentials
  - User: `seq`
  - Password: `seq`
- The device is configured to acquire an IP address.
- To execute the certificate management operations, the date on the device must be current. If the device is offline, please configure the date.

After installation, the `seq` user has access to the CoreLockr APIs. The filesystem setup has the `coretee` group that has read and write permissions to `/dev/tee0`, e.g.

```
seq@tegra-seqlabs:~$ ll /dev/tee0
crw-rw---- 1 root coretee 244, 0 Mar  5 05:48 /dev/tee0
```

The `seq` user belongs to the `coretee` group, e.g.

```
seq@tegra-seqlabs:~$ groups seq
seq : seq adm sudo audio video gdm lightdm coretee
```

For a user to execute applications using the CoreLockr APIs that communicate with CoreTEE, the user must be in the `coretee` group, or to be `root`. However, any access scheme is acceptable, provided the user calling CoreTEE services has read/write access to `/dev/tee0`.

Note: The CoreLockr Secure Certificates API function that modifies the certificates installed during flashing requires `root` privileges. The command updating these certificates installed during flashing writes back to the manifest partition. Writing to this partition is restricted to `root`. Additional information is provided in `EMSPARK_KEYS_CERTS.pdf`, part of the **EmSPARK Development Kit**.

4. BIBLIOGRAPHY

[1] "NVIDIA Jetson Linux Driver Package Software Features," NVIDIA, [Online]. Available: https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3261/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/quick_start.html#. [Accessed 4 March 2022].

CHANGE HISTORY

DATE	VERSION	RESPONSIBLE	DESCRIPTION
March 18, 2022	1.0	Julia Narvaez	Produced document for release.