SC60 Secure Boot User Guide

Smart LTE Module Series

Rev. SC60_Secure_Boot_User_Guide_V1.0

Date: 2017-10-24

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2017-10-24	Barret YUAN	Initial

Contents

Table Index

Figure Index

1 Introduction

This document mainly introduces what is secure boot and how to use the secure boot function of Quectel SC60 module.

2 Secure Boot Overview

Secure boot refers to the bootup sequence that establishes a trusted platform for secure applications. It starts as an immutable sequence that validates the origin of the code using cryptographic authentication so only authorized software can be executed. The bootup sequence places SC60 in a known security state and protects it against binary manipulation of software and reflashing attacks.

A secure boot system adds cryptographic checks to each stage of the bootup process. This process asserts the authenticity of all secure software images that are executed by SC60. This additional check prevents any unauthorized or maliciously modified software from running on the module. Secure boot is enabled through a set of hardware fuses. For the code to be executed, it must be signed by the trusted entity identified in the hardware fuses.

NOTE

Secure boot is not guaranteed without blowing an eFuse. Licensees must blow certain eFuses.

To sign the images, a trusted vendor uses their private key to generate a signature of the raw code that they want to use, and adds this to the device alongside the software binary. The device contains the corresponding public key of the vendor, which can be used to verify that the binary has not been modified and that it was provided by the trusted vendor in question.

Images (the format in which the code is packaged) can be signed using the licensee's own code signing system. Signed images include the code signature and the certificate chain. The certificates carry the public keys used to decrypt the certificate and image signatures. Secure boot supports 2048 bit exponent 3 or 65537 public RSA keys for the certificate and image signatures. The certificate signatures support the PKCS v1.2 standard format for SHA1 or SHA256, while the code signature follows a proprietary algorithm.

The certificate chain can have three certificates: attestation certificate → attestation Certificate Authority (CA) certificate → root certificate; or two certificates: attestation certificate → self-signed root certificate. For the two-certificate chain model, the root of the certificate chain must be self-signed, which need not be the case for the three-certificate chain model. Version 1, 2, or 3 X.509 certificate format is supported.

The boot up of a device comprises a multiple-stage process. Each image in the stage performs a specific function, and each image is verified by the previous image (e.g., Primary Boot Loader (PBL) → Secondary Boot Loader (SBL) → ARM® TrustZone). The root of trust (the most trusted entity that kicks off this process) is the PBL, which is firmware and therefore already trusted. Before the next image in the

boot up sequence is executed, that image is first authenticated to ensure that it contains authorized software. For example, control passes to the SBL only after it has been successfully authenticated by the PBL. Since the SBL is now trusted, it can be trusted to authenticate the image. The images further establish the security of the device through their functionality.

3 Set up Compiling Environment

This chapter takes SC60 to illustrate how to set up compiling environment. Customers should download the files of corresponding version, not the wrong version.

1. Install open source tools Python and OpenSSL, or use Quectel's compile_tools tools.

Open source tools Python and OpenSSL can be downloaded from their respective official websites. Quectel's compile_tools can be downloaded from FTP. After downloading, extract *compile_tools.tar* to *C:* root directory.

2. Download tool prebuilt_for_QFIL

Download file *SC60_Android_XXX_prebuilt_for_QFIL_XXX.rar* from FTP and unzip it. Then change the Read Only files into Read-Write ones.

3. Download firmware SC60xx_prebuilt

Download file *SC60xxTAR02A01V01H16G_prebuilt.zip* from FTP and unzip it. Then use the unzipped files to overwrite prebuilt_for_QFIL.

4. Modify *partition.xml*

non-hlos, *tz*, *rpm*, *sbl* and other files of the firmware are not written in *partition.xml* by default. When using secure boot, there is a need to reflash signed firmware image, and thus customers have to modify *common/build/partition.xml* to fill in the corresponding filename.

```
<!--       they will instead be placed side by side at the beginning of the disk -->
<partition label="modem" size_in_kb="86016" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="true" filename="NON-HLOS.bin"/>
<partition label="fsc" size_in_kb="1" type="57B90A16-22C9-E33B-8F5D-0E81686A68CB" bootable="false" readonly="false" filename=""/>
<partition label="ssd" size_in_kb="8" type="2C86E742-745E-4FDD-BFD8-B6A7AC638772" bootable="false" readonly="false" filename=""/>
<partition label="sbl1" size_in_kb="512" type="DEA0BA2C-CBDD-4805-B4F9-F428251C3E98" bootable="false" readonly="false" filename="sbl1.mbn"/>
<partition label="sbl1bak" size_in_kb="512" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename="sbl1.mbn"/>
<partition label="rpm" size_in_kb="512" type="098DF793-D712-413D-9D4E-89D711772228" bootable="false" readonly="false" filename="rpm.mbn"/>
<partition label="rpmbak" size_in_kb="512" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename="rpm.mbn"/>
<partition label="tz" size_in_kb="2048" type="A053AA7F-40B8-4B1C-BA08-2F68AC71A4F4" bootable="false" readonly="false" filename="tz.mbn"/>
<partition label="tzbak" size_in_kb="2048" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename="tz.mbn"/>
<partition label="devcfg" size_in_kb="256" type= "F65D4B16-343D-4E25-AAFC-BE99B6556A6D" bootable="false" readonly="false" filename="devcfg.mbn"/>
<partition label="devcfgbak" size_in_kb="256" type= "F65D4B16-343D-4E25-AAFC-BE99B6556A6D" bootable="false" readonly="false" filename="devcfg.mbn"/>
<partition label="dsp" size_in_kb="16384" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename="adspso.bin"/>
<partition label="modemst1" size_in_kb="1536" type="EBBEADAF-22C9-E33B-8F5D-0E81686A68CB" bootable="false" readonly="false" filename=""/>
<partition label="modemst2" size_in_kb="1536" type="0A288B1F-22C9-E33B-8F5D-0E81686A68CB" bootable="false" readonly="false" filename=""/>
<partition label="DDR" size_in_kb="32" type="20A0C19C-286A-42FA-9CE7-F64C3226A794" bootable="false" readonly="true"/>
<partition label="fsg" size_in_kb="1536" type="638FF8E2-22C9-E33B-8F5D-0E81686A68CB" bootable="false" readonly="true" filename=""/>
<partition label="sec" size_in_kb="16" type="303E6AC3-AF15-4C54-9E9B-D9A8FBECF401" bootable="false" readonly="true" filename="sec.dat"/>
<partition label="splash" size_in_kb="11264" type="20117f86-E985-4357-B9EE-374BC1D8487D" bootable="false" readonly="false" filename=""/>
<partition label="aboot" size_in_kb="1024" type="400FFDCD-22E0-47E7-9A23-F16ED9382388" bootable="false" readonly="true" filename="emmc_appsboot.mbn"/>
<partition label="abootbak" size_in_kb="1024" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="true" filename="emmc_appsboot.mbn"/>
<partition label="boot" size_in_kb="65536" type="20117F86-E985-4357-B9EE-374BC1D8487D" bootable="false" readonly="true" filename="boot.img"/>
<partition label="recovery" size_in_kb="65536" type="9D72D4E4-9958-42DA-AC26-BEA7A90B0434" bootable="false" readonly="true" filename="recovery.img"/>
<partition label="devinfo" size_in_kb="1024" type="1B81E7E6-F50D-419B-A739-2AEEF8DA3335" bootable="false" readonly="true" filename=""  sparse="true"/>
<partition label="system" size_in_kb="3145728" type="97D7B011-54DA-4835-B3C4-917AD6E73D74" bootable="false" readonly="true" filename="system.img" sparse="true"/>
<partition label="cache" size_in_kb="262144" type="5594C694-C871-4B5F-90B1-690A6F68E0F7" bootable="false" readonly="false" filename="cache.img" sparse="true"/>
<partition label="persist" size_in_kb="32768" type="6C95E238-E343-4BA8-B489-8681ED22AD0B" bootable="false" readonly="false" filename="persist.img" sparse="true"/>
<partition label="misc" size_in_kb="1024" type="82ACC91F-357C-4A68-9C8F-689E1B1A23A1" bootable="false" readonly="false" filename="" />
<partition label="keystore" size_in_kb="512" type="DE7D4029-0F5B-41C8-AE7E-F6C023A02B33" bootable="false" readonly="false" filename="" />
<partition label="config" size_in_kb="32" type="91b72d4d-71e0-4cbf-9b8e-236381cff17a" bootable="false" readonly="false" filename="" />
<partition label="oem" size_in_kb="262144" type="7db6ac55-ecb5-4e02-80da-4d335b973332" bootable="false" readonly="false" filename="" />
<partition label="limits" size_in_kb="32" type="10A0C19C-516A-5444-5CE3-664C3226A794" bootable="false" readonly="true"/>
<partition label="mota" size_in_kb="512" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename=""/>
<partition label="dip" size_in_kb="1024" type="4114B077-005D-4E12-AC8C-B493BDA684FB" bootable="false" readonly="false" filename=""/>
<partition label="mdtp" size_in_kb="32768" type="3878408A-E263-4B67-B878-6340B35B11E3" bootable="false" readonly="false" filename="mdtp.img"/>
<partition label="syscfg" size_in_kb="512" type="098DF793-D712-413D-9D4E-89D711772228" bootable="false" readonly="false" filename=""/>
<partition label="mcfg" size_in_kb="4096" type="EBD0A0A2-B9E5-4433-87C0-68B6B72699C7" bootable="false" readonly="false" filename=""/>
<partition label="lksecapp" size_in_kb="128" type="A11D2A7C-D82A-4C2F-8A01-1805240E6626" bootable="false" readonly="true" filename="lksecapp.mbn"/>
```

NOTE

Please make sure *partition.xml* exactly matches with the firmware version.

5. Download sec_boot_tools

Download file *sec_boot_tools_SC60_SG30.zip* from FTP and unzip it. Then use the unzipped files to overwrite prebuilt_for_QFIL.

4 Configure OEM Key

The flow to configure OEM key is illustrated below:

1. Run as administrator to run the following command, and then a new key will be generated in *common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs*.

```
build_sc60_secboot.bat newkey
```

2. Backup these keys.

Table 1: OEM Key List

Keys	Path	Description
qpsa_rootca.key	*common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs*	Root CA private key
qpsa_rootca.cer	*common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs*	Root certificate

qpsa_attestca.key	*common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs*	Attestation CA private key
qpsa_attestca.cer	*common/tools/sectools/resources/data_prov_assets/Signing/Local/oem_certs*	Attestation CA certificate

5 Sign Zap

Use the following command to sign zap images:

1. Run the command below, and then the new image a*_zap.* will be generated in the path below: */common/tools/sectools/secimage_output/8953/gfx_microcode*.

```
build_sc60_secboot.bat zap
```

2. Copy the new image a*_zap.* to Android source code and then build the Android image. Android source code is available from the path below: *LA.UM.5.6/LINUX/android/vendor/qcom/proprietary/prebuilt_HY11/target/product/msm8953_64/system/etc/firmware/a*_zap.**.

3. Copy Android image to *prebuilt_for_QFIL_xxx/LINUX/android/out/target/product/msm8953/*.

NOTE

If a*_zap.* is not signed, then system cannot work. So there is a need to use the signed image to rebuild Android image. Path of a*_zap.* is shown below: *LA.UM.5.6/LINUX/android/vendor/qcom/proprietary/prebuilt_HY11/target/product/msm8953_64/system/etc/firmware/a*_zap.**.

6 Generate Secure Images

The steps to generate secure images are illustrated below:

1. Use the following command to sign all images:

```
build_sc60_secboot.bat
```

```
I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829>build_SC60_secboot.bat
Logging to I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\sectools\se
cimage_output\SecImage_log.txt


    SecImage launched as: "common/sectools/sectools.py secimage -m . -p 8953 -sa"

Config path is set to: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common
\sectools\config\8953\8953_secimage.xml
Chipset is set to: 8953
WARNING: OEM ID is set to 0 for sign_id "sbl1"
WARNING: OEM ID is set to 0 for sign_id "prog_emmc_firehose_ddr"
WARNING: OEM ID is set to 0 for sign_id "prog_emmc_firehose_lite"
WARNING: OEM ID is set to 0 for sign_id "validated_emmc_firehose_ddr"
WARNING: OEM ID is set to 0 for sign_id "validated_emmc_firehose_lite"
WARNING: OEM ID is set to 0 for sign_id "qsee"
WARNING: OEM ID is set to 0 for sign_id "devcfg"
WARNING: OEM ID is set to 0 for sign_id "appsbl"
WARNING: File not found in meta build: uefi
WARNING: OEM ID is set to 0 for sign_id "adsp"
WARNING: OEM ID is set to 0 for sign_id "mba"
WARNING: OEM ID is set to 0 for sign_id "modem"
WARNING: OEM ID is set to 0 for sign_id "rpm"
WARNING: OEM ID is set to 0 for sign_id "wcnss"
WARNING: OEM ID is set to 0 for sign_id "venus"
```

2. Then store the signed images to the default path as below:
   /common/sectools/common/tools/sectools/secimage_output/8953

7 Generate sec.dat

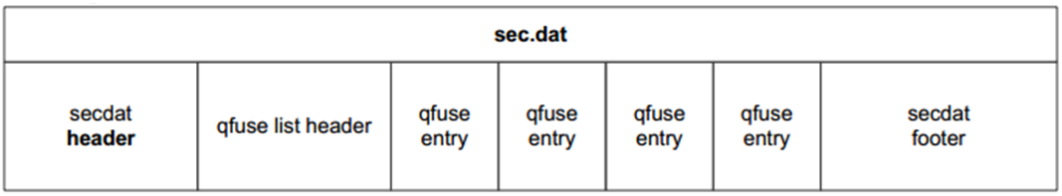

Figure 1: sec.dat Construction

sec.dat contains fuses information that is going to be blown by trustzone. Customers can enable secure boot by blowing OEM special fuses.

The procedure to generate *sec.dat* is shown below:

1. Use the following command to generate *sec.dat*.

```
build_sc60_secboot.bat sec.dat
```

```
I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829>build_SC60_secboot.bat sec.dat
Logging to I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\sectools\fu
seblower_output\FuseBlower_log.txt
DEBUG: Debug logging to I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\commo
n\sectools\fuseblower_output\FuseBlower_log_debug.txt
DEBUG:

    FuseBlower launched as: "common/sectools/sectools.py fuseblower -p 8953 -g -v"

DEBUG: Searching configs corresponding to platform: 8953
Config paths set to:
    OEM: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\sectools\conf
ig\8953\8953_fuseblower_OEM.xml
    QC: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\sectools\confi
g\8953\8953_fuseblower_QC.xml
    UI: None
    USER: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\sectools\con
fig\8953\8953_fuseblower_USER.xml
Output dir is set to: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\
sectools
Secdat generated at: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\s
ectools\fuseblower_output\v2\sec.dat
Secdat generated at: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common\s
ectools\common_output\v2\sec.dat
Secdat path is set to: I:\barret\SC60_Android_7.1.2_prebuilt_for_QFIL_20170829\MSM8953.LA.2.0\common
\sectools\fuseblower_output\v2\sec.dat
```

2. The newly generated *sec.dat* will be found in *common/sectools/common_output/v2/sec.dat*, and please use the new *sec.dat* to replace the old one in *common\sectools\resources\build\fileversion2\sec.dat*. It is recommended to backup the old one

before replacing it with the new one.

3. Load *sec.dat* file using fastboot or QFIL tool.

The command below can be used when using fastboot tool for *sec.dat* file loading:

```
fastboot flash sec <sec.dat file path>
```

For details of loading *sec.dat* file with QFIL tool, please refer to ***Chapter 2*** in ***document [1]***.

8 Flatten a Meta Build

1. Download QFIL tool and install it.
2. From the Tools menu, select “**Flat Meta Build**”.
3. In the “Content XML” field, click “**Browse**”.
4. In the “Flat Build Path” field, click “**Browse**” and navigate to and select the location where the flat build image will be written. Please make sure that you have Administrator access to the folder where the flat build will be written, and the folder must be empty.
5. Select the desired “Product Flavors” and “Device Type”.
6. Click “**OK**” to start flattening the build. When the flattening of the build completes successfully, messages “Flatten Succeed” and “Finish Flatten” will be displayed in the “Status” field.

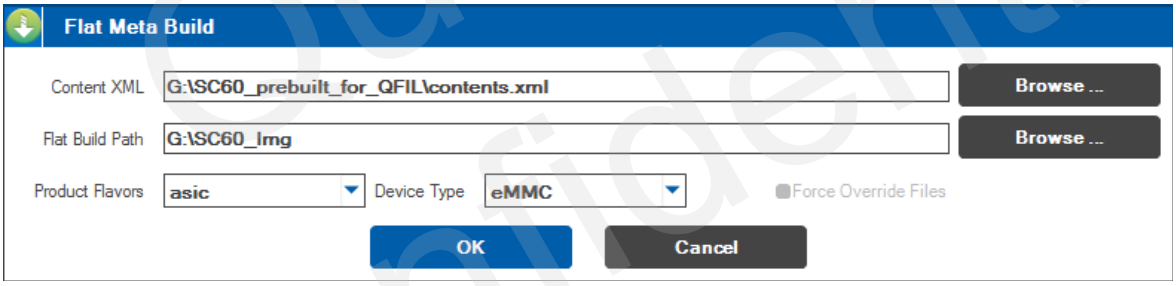

Figure 2: QFIL Flat Meta Build Interface

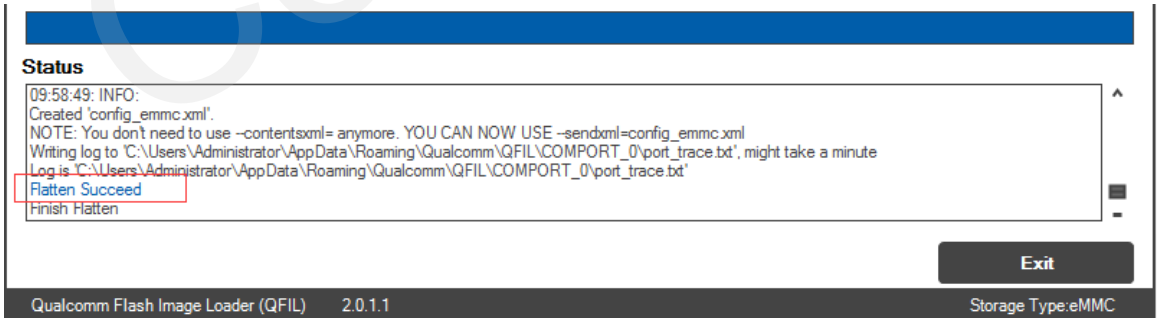

Figure 3: QFIL Flat Meta Build Interface

9 Download Image via QFIL

When using QFIL for image downloading under emergency download mode on S60, it will return an error to indicate download failure.

- **Reason**

The technical reason is that there is no native (.exe/dll) available that would sign digest table, therefore QFIL will not be able to operate in VIP mode.

QFIL team would develop a tool which can support VIP mode when the signing tool that would work in windows environment is ready. Currently it is not ready.

- **Solutions**

1. Use the file in Path 1 to replace the Flat Meta Build image in Path 2.
   Path 1:
   common/tools/sectools/secimage_output/8953/prog_emmc_ddr/prog_emmc_firehose_8953_ddr_quectel.mbn
   Path 2: *emmc/prog_emmc_firehose_8953_ddr.mbn*

2. Make the following file modifications first, then rebuilt *prog_emmc_firehose_8909_ddr.mbn*, and finally sign the image. When using the signed image for downloading, the downloading will be successful.

File Paths:
.BOOT.BF.3.3/boot_images
/core/storage/tools/deviceprogrammer_ddr/src/firehose/deviceprogrammer_initialize.c

```
// In function void deviceprogrammer_init_hw()
/* comment out - start
#ifndef SKIP_SECBOOT_CHECK_NOT_RECOMMENDED_BY_QUALCOMM
      // The check below is used to ensure that only VIP programmer is run on secure boot devices.
      // In other words, signing the non-VIP programmer is NOT recommended.
    // if (FALSE == isValidationMode() && TRUE == isAuthenticationEnabled())
      //{
        // strlcat(err_log, "Secure boot detected. VIP not enabled:fail ", sizeof(err_log));
      //}
#endif + comment out - end */
```

10 Appendix A Reference

Table 2: Related Documents

SN	Document Name	Remark
[1]	Quectel_SC60_Android_Burning_User_Guide	SC60 Android Burning User Guide

Table 3: Terms and Abbreviations

Abbreviation	Description
AP	Application Processor
APPSBL	Applications Boot Loader
APPS PBL	Applications Primary Boot Loader
BP	Baseband Processor
CA	Certificate Authority
HLOS	High-Level Operating System
mba	Modem Boot Authentication
PIL	Peripheral Image Loading
QFIL	Quelcomm Flash Image Loader
QSEE	Qualcomm Secure Execution Environment
RPM	Resource Power Manager
SBL	Secondary Boot Loader
TZ	Trustzone