Sectools: FuseBlower Tool

User Guide

80-NM248-3 E

November 10, 2014

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	March 2014	Initial release.
B	April 2014	Rearrange 4.2 and 4.3, add chapter 4.3.4
C	May 2014	Added MSM8939 support, updated USER.xml with new settings
D	July 2014	Added support for sec.dat version 2 header
E	November 2014	Updated usage params, added -p for platform and -m for mode.

Contents

1 Introduction 6

1.1 Purpose 6
1.2 Scope 6
1.3 Conventions 6
1.4 References 6
1.5 Technical assistance 7
1.6 Acronyms 7

2 FuseBlower Tool Overview 8

2.1 Key features 8
2.2 System diagram 9

3 FuseBlower Tool Components 11

4 Configuration and Usage 12

4.1 Prerequisites 12
4.2 FuseBlower tool configuration file 12
4.3 FuseBlower tool usage 13
4.3.1 Generate sec.dat and perform self-validation 15
4.3.2 Validate a sec.dat against config files 16
4.3.3 Example commands 17
4.3.4 Loading sec.dat 18

Figures

Figure 2-1 FuseBlower generate sec.dat with preconfigured XMLs 9
Figure 2-2 FuseBlower validate sec.dat against given XMLs 10

Tables

Table 1-1 Reference documents and standards 6
Table 1-2 Acronyms 7

1 Introduction

1.1 Purpose

To use only authenticated images on target, secure control e-fuses must be blown. The FuseBlower tool helps users – both Qualcomm engineers and OEMs – to easily manipulate fuse data, and validate and generate sec.dat to blow e-fuse on target.

1.2 Scope

This document provides detailed instructions on how to use the FuseBlower tool to configure fuse and generate sec.dat.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

1.4 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm® Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1

1.5 Technical assistance

Send email to sectools.support@qti.qualcomm.com for any FuseBlower related issues.

1.6 Acronyms

For definitions of terms and abbreviations, refer to [Q1]. Table 1-2 lists terms that are specific to this document.

Table 1-2 Acronyms

Acronym	Definition
FEC	Forward error correction
QFPROM	Qualcomm Fuse-Programmable Read-Only Memory

2 FuseBlower Tool Overview

The FuseBlower tool is a standalone tool developed in Python. Its main functionality is providing the ability to easily create a sec.dat file, which is used to blow e-fuses on a target.

IMPORTANT!

- OEM's are advised **not** to disable any bits in QC.xml file
- Do **not** modify QC.xml file without a recommendation from Qualcomm Customer Engineering, or you may risk blowing fuses in ways that the target behaves abnormally or prevents the target from booting or loading signed images properly
- OEM's may modify OEM.xml to include more customer-specific fuse settings
- It is **not** recommended to use sec.dat to blow sensitive information, such as private keys (i.e., ck, cpk0, cpk1), as the information is in the clear and will stay in the emmc
- The Secondary Derivation Key is set to RANDOM value, since TZ will generate a random value for QFPROM_RAW_SEC_KEY_DERIVATION_KEY and blow the fuses
- It is recommended for OEM's to apply sec.dat as the final step in configuring e-fuses as sec.dat will disable write permissions to certain fuse regions defined in the config files.
- FEC and FEC_EN will automatically be calculated and applied to OEM_PK_HASH and SEC_BOOT fuse regions.
- Default fuse settings and values from USER.xml and OEM.xml config files take priority over values in QC.xml.

2.1 Key features

The FuseBlower tool has the following features:

- Configurable fuse data: config files to add/remove/modify fuses
- Configurable target dependent data: chipset-specific config files
- Unified format for Qualcomm and non-Qualcomm configuration (XML)
- Calculate OEM_PK_HASH to fuse values
- Automatically set fuses required for secure boot authentication
- Calculate FEC values and enable FEC for OEM_PK_HASH and SEC_BOOT regions
- Create a sec.dat file to be loaded on a target
- Support for sec.dat version 2 header
- Create a common sec.dat file containing multiple segments
- Software validation comparing output and input data model

2.2 System diagram

Figure 2-1 illustrates using the FuseBlower tool to generate sec.dat.

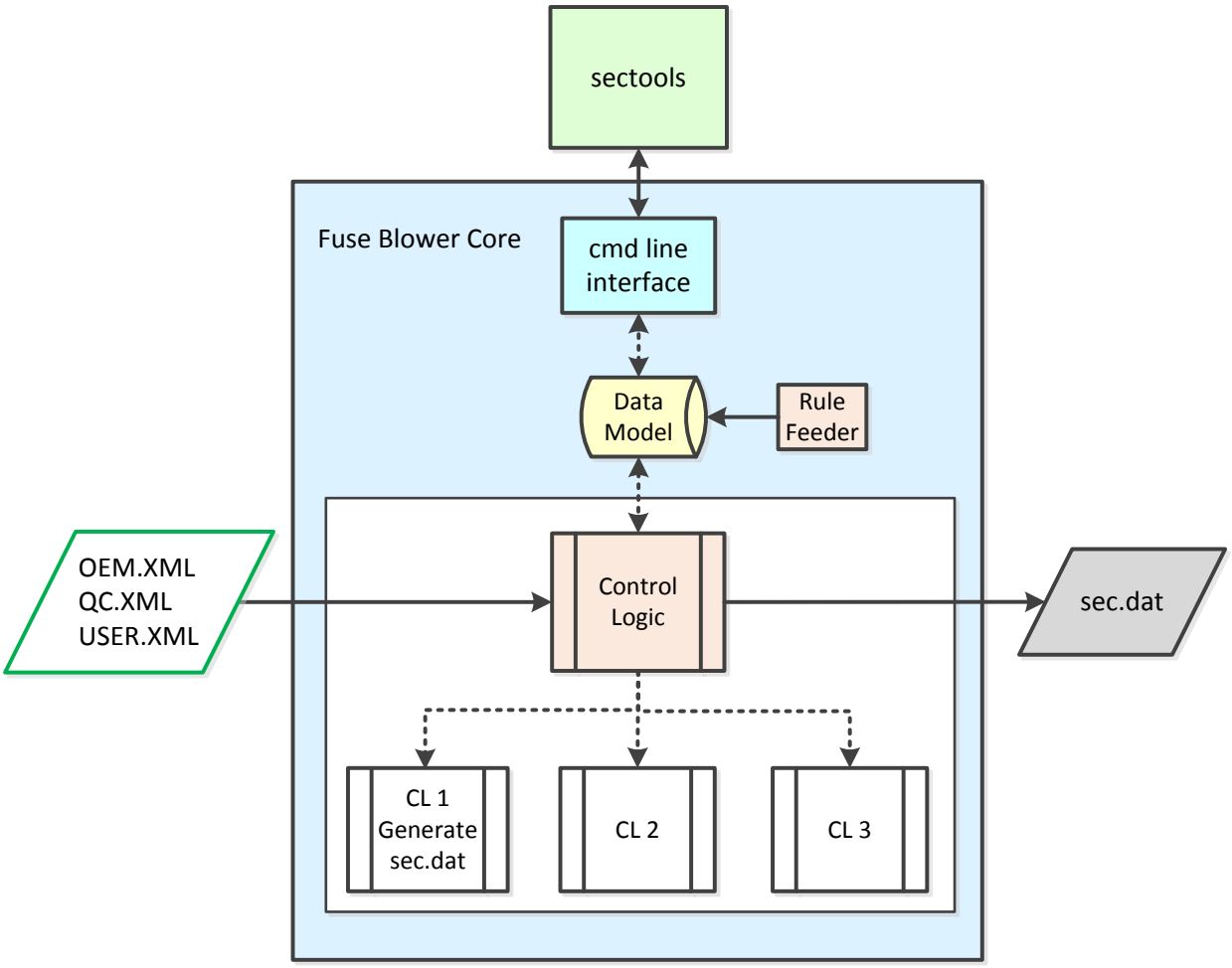

Figure 2-1 FuseBlower generate sec.dat with preconfigured XMLs

Figure 2-2 illustrates using the FuseBlower tool to validate sec.dat against config file.

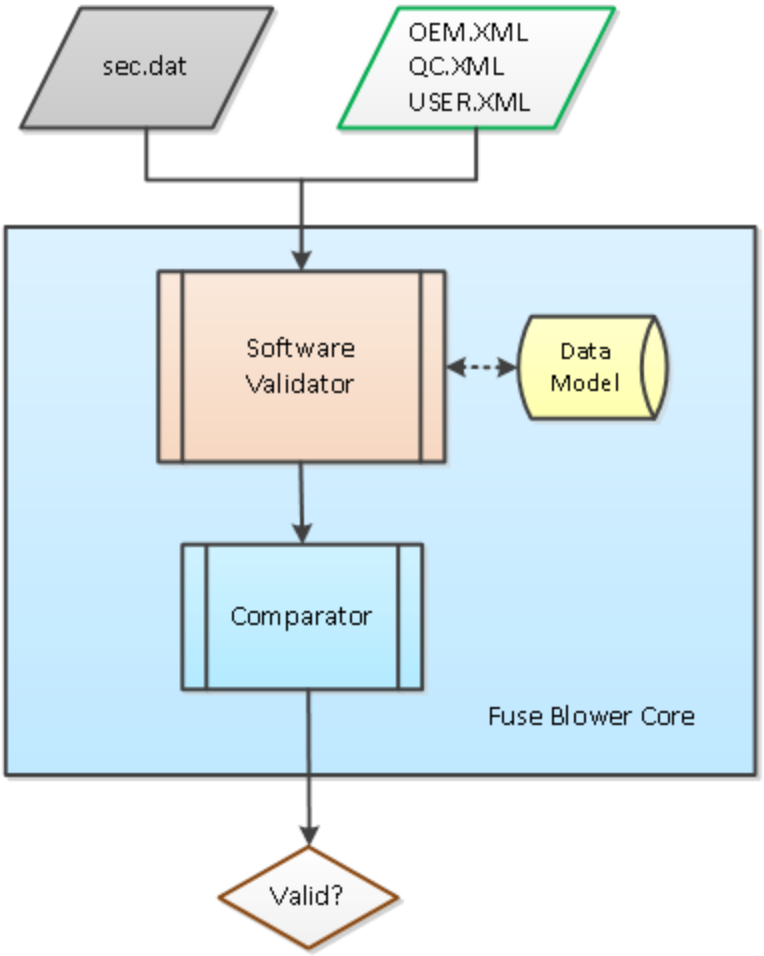

Figure 2-2 FuseBlower validate sec.dat against given XMLs

3 FuseBlower Tool Components

The FuseBlower tool includes the following components/folders that are used to generate and validate sec.dat:

```
<sectools>/
| sectools.py                 (main tool launcher command interface)
|
| -- config/                  (chipset-specific config template directory)
| -- config/<chipset>/        (preconfigured fuse templates directory)
| -- config/xsd               (xsd for config xml)
|
| -- sectools/features/fbc/                   (FuseBlower core)
| -- sectools/features/fbc/fbc.py             (FuseBlower core python script)
| -- sectools/features/fbc/fuseblower.py      (main FuseBlower python script)
| -- sectools/features/fbc/parsegen/fb_config (for OEM/QC/USER config)
|
| -- sectools/features/fuse_common/                   (fuse related common code)
| -- sectools/features/fuse_common/datamodel/         (data model code)
| -- sectools/features/fuse_common/parsegen/fm_config (for Fuse Master layout)
| -- sectools/features/fuse_common/parsegen/secdat    (sec.dat parser)
| -- sectools/features/fuse_common/validator/         (built-in validators)
|
| -- sectools/common/utils                            (common core utilities)
```

4 Configuration and Usage

4.1 Prerequisites

Prerequisites for creating a sec.dat file using the FuseBlower tool are:

- Python 2.6.2 (or later version)
- A target which supports the sec.dat file. Current target support is for APQ 8084, MSM8916, MSM8939, and MSM8994
- For sec.dat version 2 header, the target must also support version 2 header. Current target support is for MSM8994
- Optional: To generate a common sec.dat that contains multiple segments, FuseBlower version 3.0 is required for sec.dat version 2 header support

4.2 FuseBlower tool configuration file

Three configuration files, Qualcomm config file, OEM config file, and USER config file are included and required for each target supported in order to create a sec.dat file with the desired settings for the specified target. The configuration files are chipset-specific and located in the config\<chipset> directory.

The Qualcomm/OEM config files are supplied as base configuration for each target to generate the sec.dat. Do not modify the XML file or you may risk blowing fuses on a target in ways that the target behaves abnormally or prevents the target from being able to boot or use signed images properly.

Starting with v2.3, FuseBlower also supports sec.dat version 2 header and is specified by the <file_version> tag in the USER config file:

```
<secdat>
      <file_info>default dat file</file_info>
      <file_version>2</file_version>
      <fuse_version>1</fuse_version>
</secdat>
```

The USER config file is structured in the module, as shown below, where each entry can be set with ignore=False to be used for creating sec.dat (i.e., if ignore=True, it will not be taken).

```
<module id="SECURITY_CONTROL_CORE">
      <entry ignore=”true” or "false">
            <description></description>
            <name></name>
            <value></value>
      </entry>
</module>
```

The following entries are defined in the USER config file for easy configuration. The FuseBlower tool will combine the USER config file with the Qualcomm/OEM config file to create a complete fuse data model and write into sec.dat file.

```
root_cert_hash: contains the OEM public key hash as set by OEM
root_cert_file: SHA256 signed root certificate to generate root hash
[1]oem_hw_id: OEM hardware ID in hex (0x0000)
[1]oem_product_id: OEM product ID in hex (0x0000)
mrc_cert_count: number of root certificates hashed into OEM_PK_HASH
mrc_pk_hash_index: index to select root certificate
[2]SEC_BOOT1_PK_Hash_in_Fuse: set if PK HASH for SEC_BOOT1 is in fuse
SEC_BOOT1_rom_pk_hash_index: index to select which of 16 keys in ROM to
                             use for SEC_BOOT1 if hash is NOT in fuse
SEC_BOOT1_use_serial_num: use serial num for secure boot authentication
[2]SEC_BOOT2_PK_Hash_in_Fuse: set if PK HASH for SEC_BOOT2 is in fuse
SEC_BOOT2_rom_pk_hash_index: index to select which of 16 keys in ROM to
                             use for SEC_BOOT2 if hash is NOT in fuse
SEC_BOOT2_use_serial_num: use serial num for secure boot authentication
[2]SEC_BOOT3_PK_Hash_in_Fuse: set if PK HASH for SEC_BOOT3 is in fuse
SEC_BOOT3_rom_pk_hash_index: index to select which of 16 keys in ROM to
                             use for SEC_BOOT3 if hash is NOT in fuse
SEC_BOOT3_use_serial_num: use serial num for secure boot authentication
apps_debug_disabled: disable APPS debug
watchdog_enable: prevents the Watch Dog from being disabled by the GPIO
[3]allow_bus_dcvs: allowing bus DCVS SW feature will require and set
                APPS_NIDEN_DISABLE to 0

      [1]certain signing mechanisms require these values to match what was
       defined in the signing parameters
      [2]user must ensure root_cert_hash/file is set if set to true
      [3]only for targets that support enabling of bus DCVS via
      APPS_NIDEN_DISABLE
```

The following parameters, when specified, have additional side effects of having other fuses blown:

- root_cert_hash/root_cert_file
- REGION23_FEC_EN in the FEC_EN region is blown (for targets that support FEC_EN region, otherwise, FEC_EN bit is part of the row that contains FEC value)

NOTE: The FEC is enabled by default for OEM_PK_HASH, SEC_KEY_DERIVATION, and SEC_BOOT regions.

4.3 FuseBlower tool usage

The supported command line options for the FuseBlower tool are:

```
sectools.py fuseblower
--oem_config_path=<OEM config file path>
--qc_config_path=<QC config file path>
--user_config_path=<user config file path>
--secdat=<sec.dat file path>
--platform=<platform>
--mode=<mode>
```

```
--output_dir=<destination directory>
--generate
--validate
--version
--help
-d (for debug)
```

Where:

- <OEM config file path> is the path to the OEM config file which contains OEM-specific fuse data.
- <QC config file path> is the path to the Qualcomm config file which contains Qualcomm-only fuse data.
- <user config file path> is the path to the USER config file which has user friendly format for OEM to easily configure OEM-specific fuse data.
- <platform> is the name of the platform for the set of config files to be used (for example, 8994 as <platform> will use config files from the following default directory: .\config\8994).
- <mode> selects the segment type to generate as supported by FuseBlower, i.e., secboot or rot (default mode if none is specified: secboot).

NOTE: rot mode is only supported on MSM8996 and above.

- <destination directory> is the directory to save sec.dat and FuseBlower run logs. If it is not set, the default output directory, . /fuseblower_output, will be used in the directory where FuseBlower tool runs.

NOTE: If an existing sec.dat is detected that was generated by another tool (for example, 8994 and above supports KeyProvision tool (see 80-NM248-5) in <destination directory>\keyprovision_output) and the sec.dat is a version 2 header type, a <destination directory>\common_output will also be generated containing a sec.dat with both segments.

- <secdat file path> is the sec.dat file path
  - If the -g option is given to generate, the sec.dat passed in will be treated as a sec.dat to be merged with. If the given sec.dat is a version 2 header type and contains a segment generated by another tool (for example, KeyProvision), FuseBlower will generate a common_output sec.dat containing a KeyProvision segment from the passed in sec.dat and FuseBlower segment from the config files. Passing in a sec.dat with just a FuseBlower segment will be ignored as the data in the config files will take priority over the given sec.dat.
  - If -a is given without -g, the passed in sec.dat is validated against the config files.
- If -d for debug is specified, FuseBlower will generate the following files in the /debug directory of the destination path
  - data_model_repr_<feature>.txt : text representation of the merged data from the config files which were used to generate sec.dat
  - secdat_repr.txt: text representation of sec.dat binary in MSB/LSB format

4.3.1 Generate sec.dat and perform self-validation

```
sectools.py fuseblower
--platform=<platform>
--generate
--validate
```

Or, specify each config file:

```
sectools.py fuseblower
--oem_config_path=config\<platform>\<platform>_fuseblower_OEM.xml
--qc_config_path=config\<platform>\<platform>_fuseblower_QC.xml
--user_config_path=config\<platform>\<platform>_fuseblower_USER.xml
--generate
--validate
```

The sec.dat and FuseBlower_log.txt can be found at the default output directory.

4.3.2 Validate a sec.dat against config files

```
sectools.py fuseblower
--platform=<platform>
--secdat=<sec.dat file path>
--validate
```

Or, specify each config file:

```
sectools.py fuseblower
--oem_config_path=config\<platform>\<platform>_fuseblower_OEM.xml
--qc_config_path=config\<platform>\<platform>_fuseblower_QC.xml
--user_config_path=config\<platform>\<platform>_fuseblower_USER.xml
--secdat=<sec.dat file path>
--validate
```

In the stdout, the FuseBlower tool will print a detailed table of the mismatch entries. Differences are noted by an asterisk (*).

```
Differences Details: Following differences were found between config files and sec.dat

  1. Following fuses are in sec.dat but not in config files:

| S.No. |  Address   |       Region_ID       | Operation |       Value        |
|-------|------------|-----------------------|-----------|--------------------|
|   1   | 0xfc4b80e0 | QFPROM_RAW_OEM_CONFIG |    BLOW   | 0x840001ff3fc00000 |

  2. Following fuses are in config files but not in sec.dat:

| S.No. |  Address   |       Region_ID       | Operation |       Value        |
|-------|------------|-----------------------|-----------|--------------------|
|   1   | 0xfc4b80e8 | QFPROM_RAW_OEM_CONFIG |    BLOW   | 0x840001ff3fc00000 |

  3. Following fuses are mismatching between config files and sec.dat:

| S.No. |  DataModel   |  Address   |             Region_ID             |  Operation   |        Value        |
|-------|--------------|------------|-----------------------------------|--------------|---------------------|
|   1   | config files | 0xfc4b81f8 |       QFPROM_RAW_SPARE_REG19      |     BLOW     | *0x0000000000000000 |
|       |   sec.dat    | 0xfc4b81f8 |       QFPROM_RAW_SPARE_REG19      |     BLOW     | *0x000000002f000000 |
|       |              |            |                                   |              |                     |
|   2   | config files | 0xfc4b83c8 | QFPROM_RAW_SEC_KEY_DERIVATION_KEY |    *BLOW     |  0x0000000000000000 |
|       |   sec.dat    | 0xfc4b83c8 | QFPROM_RAW_SEC_KEY_DERIVATION_KEY | *VERIFYMASK0 |  0x0000000000000000 |
```

4.3.3 Example commands

- To generate sec.dat (e.g., 8916)

```
sectools.py fuseblower –p 8916 –g
```

Or, specify each config file:

```
sectools.py fuseblower -e config\8916\8916_fuseblower_OEM.xml -q
config\8916\8916_fuseblower_QC.xml -u config\8916\8916_fuseblower_USER.xml
–g
```

- To validate a sec.dat (e.g., 8084)

```
sectools.py fuseblower –p 8084 –s c:\build\sec.dat –a
```

Or, specify each config file:

```
sectools.py fuseblower -e config\8084\8084_fuseblower_OEM.xml -q
config\8084\8084_fuseblower_QC.xml -u config\8084\8084_fuseblower_USER.xml
–s c:\build\sec.dat –a
```

- To generate a version 2 sec.dat ROT segment (e.g., 8996)

```
sectools.py fuseblower –p 8996 –m rot –g
```

Or, specify each config file (ROT segment consists of only USER and OEM xml configs):

```
sectools.py fuseblower -e config\8996\8996_fuseblower_ROT_OEM.xml -u
config\8996\8996_fuseblower_USER.xml –m rot –g
```

NOTE: If there is already an existing version 2 sec.dat in common_output sub-directory inside the main output directory (whether default or specified with -o), FuseBlower will create a common v2 sec.dat in the common_output sub-directory which contains multiple segments.

To generate a common sec.dat containing multiple segments:

A common sec.dat containing multiple segments will be stored in <output directory>\common_output. <output directory> can either be default (current directory where tool resides) or user specified with `-o` option.

- Option 1:

  Run both sec.dat related tools that generate their own segments (order does not matter) with the same output directory. Both tools will check if there is an existing sec.dat from the other tool and will generate a common sec.dat with both segments. This requires that the USER config file used in both tools is of sec.dat version 2 header type.

- Option 2 – KeyProvision (80-NM248-5) example for 8994 and up:
  - Run KeyProvision and generate a KeyProvision sec.dat with a version 2 header.
  - Run FuseBlower and use `-s` to point to the existing sec.dat generated in step a.

```
sectools.py fuseblower -e config\8994\8994_fuseblower_OEM.xml -q
config\8994\8994_fuseblower_QC.xml -u config\8994\8994_fuseblower_USER.xml
-s .\keyprovision_output\sec.dat -g
```

4.3.4 Loading sec.dat

Once the sec.dat file is created, it can be loaded to the "sec" partition on a target using fastboot or emergency download programmer (emmcbld). After loading the file onto the target, the target must be rebooted at which point TZ will blow the e-fuses on the target specified by the sec.dat file. Sec.dat is only used to blow fuses by TZ if the secure boot write permission disable fuse is NOT blown. Once the secure boot write perm disable fuse is blown, sec.dat will be ignored.

Loading sec.dat file using fast boot

The command to load the sec.dat file using fast boot is as follows:

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

Loading sec.dat file using emmcbld

A default sec.dat already exists in the sectools directory (sectools\resources\build). This sec.dat does not contain any fuse information and is used to prevent emmcbld tool related warnings. Users should back up the default sec.dat before replacing it with the generated sec.dat, and then trigger emergency download mode. The sec.dat will be flashed to the "sec" partition along with the other images.

NOTE: For use-cases where the OEM needs to swap the chip on the device, sec.dat should be erased via ***"fastboot flash erase sec"*** prior to swapping the chip so as not to trigger fuse blowing again on the new chip.

NOTE: The OEM should flash signed images prior to or along with flashing sec.dat to ensure that once secure boot fuses are blown and the device is automatically reset, device boot up is normal.