SIERRA WIRELESS	Integrity Measurement Architecture Application Note 2174255-1.0

Author:	Sierra Wireless	Date:	October 09, 2019

APN Content Level	BASIC		INTERMEDIATE		ADVANCED	✓	Confidentiality	Public		Private	✓

Hardware Compatibility	Product Line	AirPrime	Series	WP76xx

Software Compatibility	ALL	Document Type	Application Note	✓	Technical Note	

1 Introduction

1.1 Purpose

This document describes the Integrity Measurement Architecture (IMA) feature on the AirPrime WP76xx platform and explains how to use this feature with the Sierra Wireless SDK.

1.2 Scope

This document explains the Integrity Measurement Architecture (IMA) mechanism, the implementation of IMA on the WP76xx, how to enable IMA before target image build, and basic test verification.

1.3 Glossary of Terms

Term	Definition
.der	Usually in binary DER form, but Base64-encoded certificates are common too
.pem	Base64 encoded DER certificate
.x509	Files with X509 format
CA	Certificate Authorities
CER	Canonical Encoding Rules
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
IMA	Integrity Measurement Architecture
SDK	Software Development Kit
X509	A standard that defines the format of public key certificates

1.4 References

[1] AirPrime WP8548/WP75xx/WP76xx/WP77xx AT Command Reference

Reference number: 4118047

[2] Integrity Measurement Architecture (IMA)

Links:

- https://wiki.gentoo.org/wiki/Integrity_Measurement_Architecture
- https://sourceforge.net/p/linux-ima/wiki/Home/

1.5 Document Conventions

Document Conventions	Description
`Monospace`	Message output to Terminal or file content.
`Bold Monospace`	Command line Input, commands to Target, Windows or Linux Terminal

Command Line Prompt	Description
`[user@x86-Linux]$`	The command run on an x86 Linux server
`[C:\path\name]$`	The command run in Windows
`root@swi-mdm9x28-wp:$PATH#`	The command run on the Sierra Wireless module

2 Integrity Measurement Architecture (IMA) Introduction

The Linux IMA subsystem is responsible for calculating the hashes of files and programs before they are loaded and supports reporting on the hashes and validates if they adhere to a predefined list.

It introduces hooks within the Linux kernel to support measuring the integrity of files that are loaded (including application code) before it is executed or mapped to memory. The measured value (hash) is then registered in a log that can be consulted by administrators.

2.1 IMA Appraisal Introduction

Since kernel 3.7, an additional patch called the IMA appraisal patch has been merged within the IMA subsystem so it can even register the measured value as an extended attribute, and after subsequent measurement(s), validate this extended attribute against the measured value and refuse to load the file (or execute the application) if the hash does not match. In that case, the IMA subsystem allows files and applications to be loaded if the hashes match (and will save the updated hash if the file is modified) but refuse to load it if it doesn't. This provides some protection against offline tampering of the files.

Files store a hash or a digital signature in their extended attribute (security.ima).

2.2 IMA and IMA Appraisal Workflow

The following figure shows the IMA and IMA appraisal workflow.

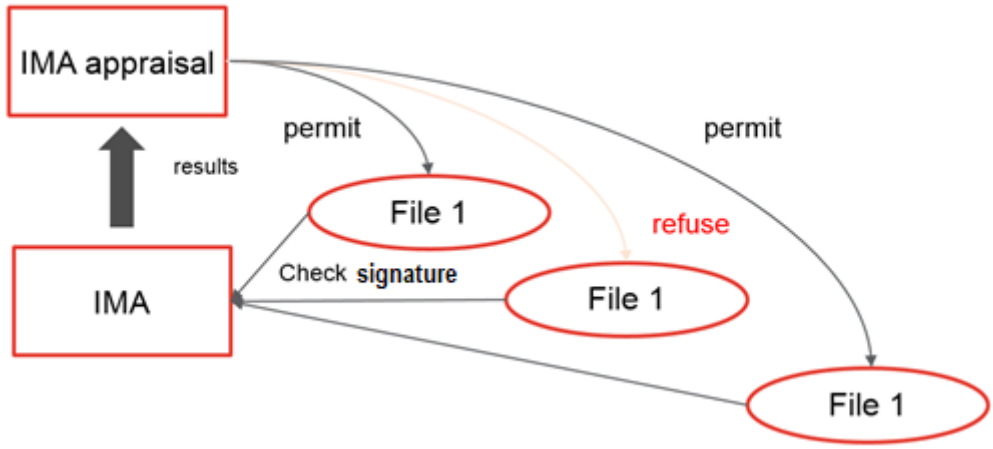

Figure 1. IMA and IMA Appraisal

2.3 IMA Protect Mechanism

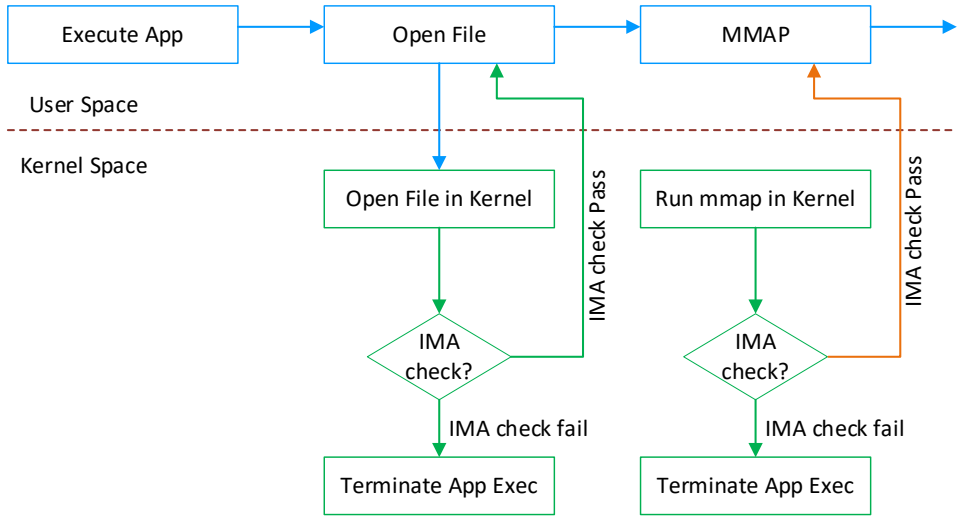

Figure 2. IMA Checking Mechanism

Note: While it is not possible to draw all IMA check points in the figure above, the two check points in this diagram are shown in order to illustrate the IMA protect mechanism.

When a file is opened or memory map (mmap) is run, and there are hooks in the Linux kernel, IMA check will start according to the policy configuration. If hash checking passes, it continues to run; else, the execution is terminated.

3 IMA Sierra Wireless Implementation

This chapter mainly illustrates basic IMA mechanism and Sierra Wireless implementation, including how the RSA keys are generated and their usage.

3.1 Overview of IMA on Sierra Wireless Modules

The whole architecture of IMA on the Sierra Wireless side includes:

- Build environment setup – this is the introduction requirement about the development system. Refer to section 4.1 Environment Setup for details.
- Key generation – details the key generation process and where the key is used. Refer to section 3.2 IMA Key Generation for details. This also includes information on how to generate the needed keys with Sierra Wireless tools. Refer to section 4.2 Keys Generation with Sierra Wireless Tools on Host Build Server for details.
- Yocto Build process – describes how to build the Yocto system with IMA enabled. Refer to section 4.4 Yocto Build Process for details.
- Legato build process – describes how to build the Legato framework and applications with IMA enabled. Refer to section 4.5 Legato Build Process for details.
- Fuse configuration – describes how to write the IMA fuse bit. Refer to section 5 Target Board Boot Configuration for details.

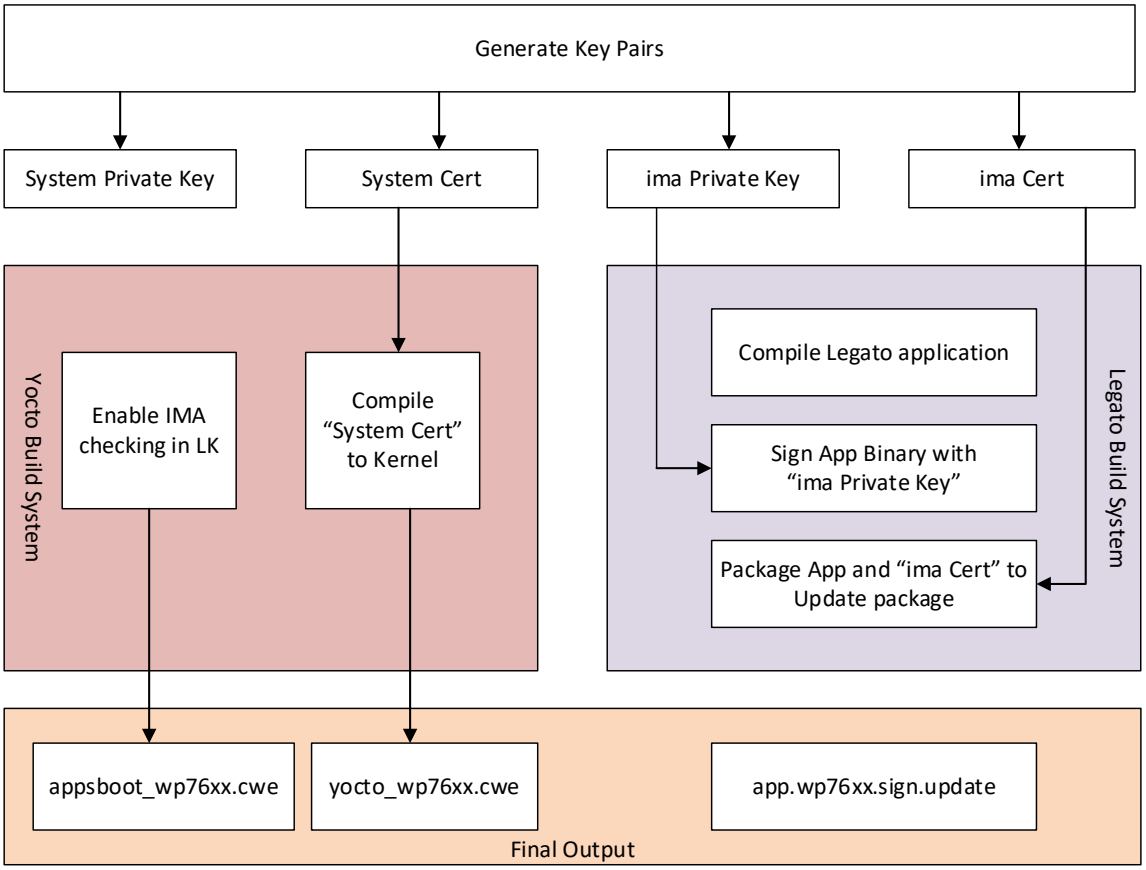

Figure 3. IMA Overview Diagram of Build Stage

3.2 IMA Key Generation

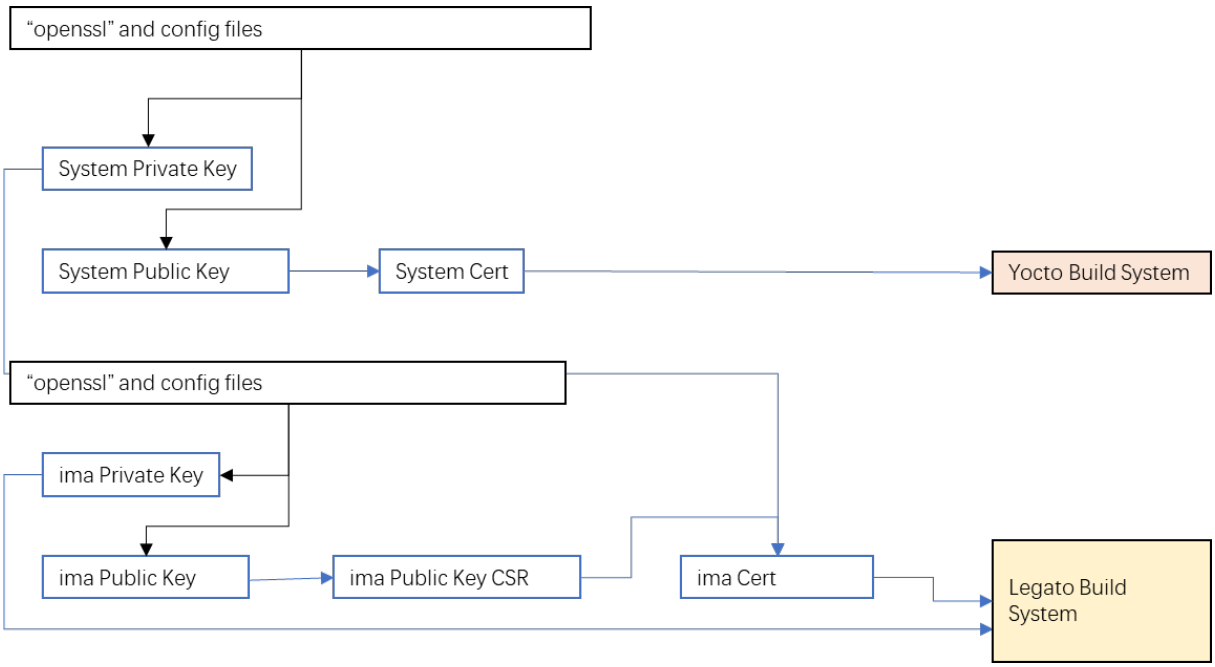

Figure 4. IMA Key Generation

As shown in the figure above, generated system and ima keys are paired by the “openssl” tool on the host.

There is a default mapping name for each key as described below.

“System Private Key”: ima-local-ca.priv

“System Cert”: ima-local-ca.x509

“ima Private Key”: privkey_ima.pem

“ima Public Key CSR”: csr_ima.pem

“ima Cert”: x509_ima.der

System Private Key is only used to sign the ima Public Key CSR. This key should be saved to a safe place afterwards.

System Cert is embedded to the Linux kernel, and its purpose is to verify the ima Cert when it is imported to the IMA system.

Ima Private Key is only used to sign user files (data files and executable files) which will be run or stored on the WP76xx during cross compile. This key should also be saved in a safe place.

Ima Cert is imported by Legato and installed to the WP76xx file system when the Legato application is installed.

3.3 IMA Startup on Target Module

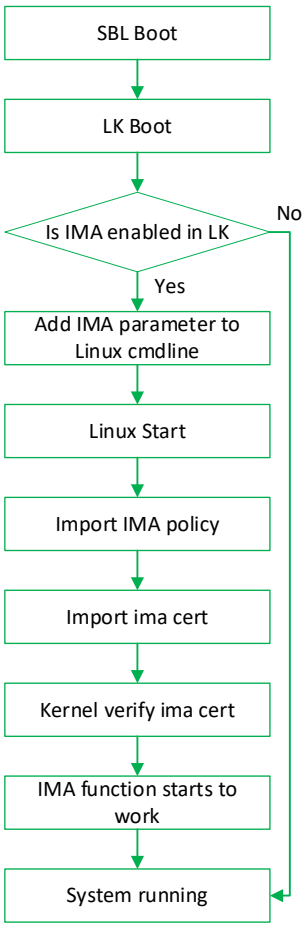

Figure 5. IMA Startup Flow Chart

The Linux kernel already has the system Cert during the Linux kernel build, so when Linux boots up, it imports the IMA policy, imports the ima Cert, and then verifies it with the System Cert since ima Cert is signed by system Private Key. If the ima Cert is valid, then the IMA sub-system is ready to verify other files with ima Cert.

4 Configuration and Build

This chapter explains how to setup the build environment, and how to customize configuration and build process.

4.1 Environment Setup

Build Host Requirement:

1. Setup build environment as per Sierra Wireless SDK requirement.
2. Install “openssl” if it’s not installed.
3. Install “evmctl” package.
4. Install “bsdtar” package

4.2 Keys Generation with Sierra Wireless Tools on Host Build Server

To generate any kind of IMA enabled images, the user would need ".system" and ".ima" private/public key pairs. Key pair generation is a separate process, and IMA-enabled builds would assume that the required key pairs exist and are available.

Script “ima-gen-keys.sh”, which is provided by Sierra Wireless, is used to generate the key pairs which will be needed in build and running system. “ima-gen-keys.sh” reads the configuration file and calls the “openssl” command to generate the key pairs.

Although item and file names used throughout this document can be changed, it is recommended that the names be kept the same until developers or readers of this document become more familiar with the IMA subsystem, key generation and build procedures.

1. Until the user becomes familiar with the keys, create a folder tree as follows to get started.

   ```
   [user@x86-Linux]$ export KEY_ROOT=/path/of/yours
   [user@x86-Linux]$ mkdir $KEY_ROOT
   [user@x86-Linux]$ mkdir $KEY_ROOT/config
   [user@x86-Linux]$ mkdir $KEY_ROOT/system
   [user@x86-Linux]$ mkdir $KEY_ROOT/ima
   ```

2. Update the configuration files for key generation.

   Copy the key related configuration file to the folder specified above, then open and change the content.

   ```
   [user@x86-Linux]$ cp …/legato/3rdParty/ima-support-tools/samples/ima-local-ca.genkey \
   $KEY_ROOT/config
   [user@x86-Linux]$ cp …/legato/3rdParty/ima-support-tools/samples/ima.genkey \
   $KEY_ROOT/config
   ```

   The user can define the correct RSA bits, organization name, common name and e-mail address in the configuration file.

   For more parameters to change, please read the Sierra Wireless script and openssl document/usage for details.

3. Generate key pairs.

   The ".system" and ".ima" key pair can be generated in one command or separately by two commands, both of which are described below. Either method can be used.

Note: *In this step, “-e <expire days>” parameter for ima-gen-keys.sh command must be specified to ensure the correct expire time. If it is not specified, the default expire time is 365 days.*

- Generate both ".system" and ".ima" key pair. Script “**ima-gen-keys.sh**” is in directory legato/3rdParty/ima-support-tools.

  ```
  [user@x86-Linux]$ ./ima-gen-keys.sh -a \
  ```

```
-c $KEY_ROOT/config/ima-local-ca.genkey \
-n $KEY_ROOT/system/ima-local-ca.priv \
-o $KEY_ROOT/system/ima-local-ca.x509 \
-m $KEY_ROOT/config/ima.genkey \
-p $KEY_ROOT/ima/privkey_ima.pem \
-q $KEY_ROOT/ima/csr_ima.pem
```

- Generate ".system" and ".ima" key pair separately.

  Because of a small bug in the script “ima-gen-keys.sh”, please create folder “system” and “ima” first. Example:

  ```
  [user@x86-Linux]$ cd ${KEY_ROOT}
  [user@x86-Linux]$ mkdir system
  [user@x86-Linux]$ mkdir ima
  ```

  a. Generate ".system" and ".ima" key pair separately.

  ```
  [user@x86-Linux]$ ./ima-gen-keys.sh -s \
  -c $KEY_ROOT/config/ima-local-ca.genkey \
  -o $KEY_ROOT/system/ima-local-ca.x509 \
  -n $KEY_ROOT/system/ima-local-ca.priv
  ```

  Three files are then generated:

  ```
  [user@x86-Linux]$ ls -l system
  total 12
  -rw-r--r-- 1 user user 1289 Jun  5 19:06 ima-local-ca.pem
  -rw-r--r-- 1 user user 1704 Jun  5 19:06 ima-local-ca.priv
  -rw-r--r-- 1 user user  912 Jun  5 19:06 ima-local-ca.x509
  ```

  b. Create ima public and private keys; use default PEM format, only generate ima key pair.

  ```
  [user@x86-Linux]$ ./ima-gen-keys.sh -i \
  -m $KEY_ROOT/config/ima.genkey \
  -q $KEY_ROOT/ima/csr_ima.pem \
  -p $KEY_ROOT/ima/privkey_ima.pem
  ```

  Three files are then generated:

  ```
  [user@x86-Linux]$ ls -l ima
  total 8
  -rw-r--r-- 1 user user 1013 Jun  5 19:09 csr_ima.pem
  -rw-r--r-- 1 user user 1704 Jun  5 19:09 privkey_ima.pem
  ```

4. Sign ima CSR.

```
[user@x86-Linux]$ ./ima-gen-keys.sh  -g \
                  -q $KEY_ROOT/ima/csr_ima.pem \
                  -m $KEY_ROOT/config/ima.genkey \
                  -n $KEY_ROOT/system/ima-local-ca.priv \
                  -o $KEY_ROOT/system/ima-local-ca.x509 \
                  -r $KEY_ROOT/ima/x509_ima.der
```

   ima Cert (x509_ima.der in the command above) is generated.

5. After the key generation is completed, a file tree like the example below can be seen if a folder structure as shown in Step 1 was initially created.

```
[user@x86-Linux]$ tree
.
├── config
│   ├── ima.genkey
│   └── ima-local-ca.genkey
├── ima
│   ├── csr_ima.pem
│   ├── privkey_ima.pem
│   └── x509_ima.der
└── system
    ├── ima-local-ca.pem
    ├── ima-local-ca.priv
    ├── ima-local-ca.srl
    └── ima-local-ca.x509
```

4.3 Build Configuration

This section describes the configuration parameter related to an IMA build. Most of the IMA related build parameters are set in a file named ima.conf which is located at "…/ legato/3rdParty/ima-support-tools/ima.conf".

All keys and CA generated during the system build will be used so verify that each file's path indicated in ima.conf is correct. Also set "ENABLE_IMA" to "1" to enable the Legato IMA feature.

If filenames are not customized and the folder structure in Step 1 of section 4.2 is followed, only "KEY_ROOT" and "ENABLE_IMA" in ima.conf needs to be changed.

The default content of ima.conf is shown below.

```
################################################################################
#                       System build interface
################################################################################

# May be useful, but mostly used as other paths helper
KEY_ROOT=${HOME}/.config/ima/keys
```

```
# Public certificate which should be added to kernel ".system" keyring.
IMA_LOCAL_CA_X509=${KEY_ROOT}/system/ima-local-ca.x509

# Private IMA key used to sign userland binaries.
IMA_PRIV_KEY=${KEY_ROOT}/ima/privkey_ima.pem

# Public IMA certificate which should be added to kernel ".ima" keyring.
IMA_PUB_CERT=${KEY_ROOT}/ima/x509_ima.der

# IMA kernel command line options
IMA_KERNEL_CMDLINE_OPTIONS="ima_appraise=enforce ima_appraise_tcb"

################################################################################
#                      Legato build interface
################################################################################

# Export all variables to child processes from this point on.
set -a

IMA_PUBLIC_CERT=${IMA_PUB_CERT}
IMA_PRIVATE_KEY=${IMA_PRIV_KEY}

# Immutable files will be protected with this SMACK label.
IMA_SMACK=imaLegato

# Enable or disable IMA support.
ENABLE_IMA=0
```

4.4 Yocto Build Process

Build all in Yocto with IMA support.

```
[user@x86-Linux]$ IMA_BUILD=1 IMA_CONFIG=<path-to>/ima.conf make <build-target>
```

4.5 Legato Build Process

Build the Legato framework with IMA support; this is only valid if only building Legato.

```
[user@x86-Linux]$ cd legato
[user@x86-Linux]$ make ENABLE_IMA=1 \
  IMA_PUBLIC_CERT=$KEY_ROOT/ima/x509_ima.der   \
  IMA_PRIVATE_KEY=$KEY_ROOT/ima/privkey_ima.pem
```

Build the Legato application with IMA support. Example commands for building helloWorld App is given below.

```
[user@x86-Linux]$ ./bin/legs
[user@x86-Linux]$ cd apps/sample/helloWorld/
[user@x86-Linux]$ mkapp -t wp76xx helloWorld.adef -a 1.0.0 -v -S \
           -K $KEY_ROOT/ima/privkey_ima.pem \
           -P $KEY_ROOT/ima/x509_ima.der
```

"helloWorld.wp76xx.signed.update" can then be retrieved; install with update command on target.

5 Target Board Boot Configuration

By default, the IMA functionality is not enabled so customers need to enable it by updating the module with appsboot_rw_ima_wp76xx.cwe

After target bootup is done, check the Linux cmdline parameter to confirm that IMA is enabled. If "ima_appraise=enforce ima_appraise_tcb" is included in the cmdline parameter, this means that IMA is enabled.

```
root@swi-mdm9x28:~# cat /proc/cmdline
console=ttyHSL0,115200 console=ttyHSL1,115200 root=/dev/ram rootfs_ro=true user1_fs=ubifs verity=on
ima_ready=0 lkversion=1.3.0_4d8a528df8 androidboot.serialno=d53ee4de
androidboot.authorized_kernel=true quiet fudge_ro_rootfs=true ima_appraise=enforce ima_appraise_tcb
androidboot.baseband=msm rootfstype=ubifs rootflags=bulk_read
```

6 Test Verification on the Sierra Wireless Development Kit

By default, "ima cert" is not in the file system which means any app is forbidden to run since the digital signature is wrong. When a Legato app is compiled, "ima cert" will be packaged with the application. After the application is installed, "ima cert" is also installed, and the application with a valid signature can run. The following subsections show two test methods – one is to compile and install a Legato app, and the other is to test manually.

6.1 Test Verification via Legato App Upgrade

Compile a Legato app. For example, compile the sample Hello World.

1. Build Legato first.

```
[user@x86-Linux]$ cd legato
[user@x86-Linux]$ export IMA_PRIVATE_KEY=/path/to/your/IMA/keys/privkey_ima.pem
[user@x86-Linux]$ export IMA_PUBLIC_CERT=/path/to/your/IMA/keys/x509_ima.der
[user@x86-Linux]$ ENABLE_IMA=1  make wp76xx
```

2. Build Sample App helloWorld.

```
[user@x86-Linux]$ ./bin/legs
[user@x86-Linux]$ cd apps/sample/helloWorld/
[user@x86-Linux]$ mkapp -t wp76xx helloWorld.adef -a 1.0.0 -v -S \
       -K /path/to/your/IMA/keys/privkey_ima.pem \
       -P /path/to/your/IMA/keys/x509_ima.der
```

"helloWorld.wp76xx.signed.update" can now be retrieved.

3. Update app shows the update process via USB. For SOTA, please follow SOTA upgrade guide.
   a. Download file to target board.

      ```
      [C:\ktemp\ima_keys]$ adb push helloWorld.wp76xx.signed.update /tmp
      ```

   b. Update app from board.

      ```
      root@swi-mdm9x28:/tmp# update helloWorld.wp76xx.signed.update
      ```

   helloWorld App can now be run to and check its output.

6.2 Test Verification Manually

Make sure you have upgraded to the correct version and enabled the IMA feature as described in previous chapters. Run IMA verification and test the functions.

Note: Step 1 and 2 are only used for test or in development stage. For Sierra Wireless customers, this should be run automatically in production stage.

Due to security reasons, it's recommended to load the public key in the initramfs.

1. Download ima Cert to target module via "adb push" or other ways.

   ```
   [C:\ktemp\ima_keys]$ adb push x509_ima.der    /tmp
   ```

   Normally, the file signature should be done on the host's x86 server, and the user should never put the IMA private key to target board, but if the user wants to run some test on the target board, the user can download the private key to target board, and perform signature on the target board.

   ```
   [C:\ktemp\ima_keys]$ adb push privkey_ima.pem /tmp
   ```

2. Import ima Cert.

   Check the keyring by usign keyctl, which will also be used in the next step. The keyring is highlighted in red.

   ```
   root@swi-mdm9x28-wp:~# cd /tmp
   root@swi-mdm9x28:/tmp#  keyctl show %keyring:.ima
   Keyring
   918516612 ---lswrv      0     0  keyring: .ima
   ```

3. Import the ima Cert.

   ```
   root@swi-mdm9x28:/tmp#  evmctl import x509_ima.der 918516612
   ```

   ima Cert file can be integrated to the root file system or Legato, and a Linux startup script should be included in the Linux root file system or Legato. Thus, after system boot up, the WP76xx can import the ima Cert automatically.

4. Create test files on the host's x86 server.
   a. Create test scripts.

      ```
      [user@x86-Linux]$ echo "echo Hello, print from shell script without sign" > \
                        ima_test_no_sign.sh
      [user@x86-Linux]$ echo "echo Hello, print from shell script with correct sign" > \
                        ima_test_sign.sh
      [user@x86-Linux]$ echo "echo Hello, print from shell script with wrong sign" > \
                        ima_test_wrong_sign.sh
      [user@x86-Linux]$ chmod 755 *
      ```

b. Sign one script with the correct private key and check.

```
[user@x86-Linux]$ sudo evmctl ima_sign --key ../privkey_ima.pem  ima_test_sign.sh
[user@x86-Linux]$ sudo getfattr -d -m- ima_test_sign.sh
# file: ima_test_sign.sh
security.ima=0sAwIClixtmgEATaDEGkl+KNPg3fh8zkBXTDPset/TPsk85louEH5Wt1lBdC9KxyMipb2SFPEMnakI
zWs3ZSEmXz/WIEA+oD9Lje9SVHcPDYTGmzVYHLz3kHRsYv438/LqmBHNOPChaOdS/cleworCGZJRUYgtgL6lOnc+zc4
x+y95zCd8H86sM0EqgbvK2/byjVWtJlu3qxYtxXqb1v18xO/nF5+2Kan227cRIvQyrEI799q+5Fs1mLM7SS9xaxVhGN
0c6kgwgSWoTBYDc6sYhKsH78UHSefFICcO6mV797wVZ3KXQBVcYPhwhGiCGqmGfl78oxiNbHGJSOjUeShTGP2DCqPYt
sPHog==
$
```

c. Sign one script with the wrong private key and check

```
[user@x86-Linux]$ sudo evmctl ima_sign --key ../a_mismatch_privkey_ima.pem \
ima_test_wrong_sign.sh
[user@x86-Linux]$ sudo getfattr -d -m- ima_test_wrong_sign.sh
# file: ima_test_wrong_sign.sh
security.ima=0sAwIC0o/zFAEARN/4DEviN/xzX96AJxnoO7F+rNxHLJj+/ZqLHoUiBFhfdkkGl4S7HPoLRA2JyOMF
NdNZJs3ArZhOaMiEjEGsSYJdyFWdokTCqSX3RTs6yHF67R6tSQRUWMzUkwI/PoiutFFfQr5KPAJ0ntLhMkK6Scp9hNT
4kqN5n0etDIT+TUc7KXsUph0r7Wj9d0CqAz5eAy3iwazNGEJAJRPXsbgqVE+c+EhSDB8/DBnjRV8zJJgTnfk+uLSRMS
ui7OHd9OVeHgtrKlvzXQoHx7gGGpVNvZdif7oR6zt21/gbsQxbhllFkhRrGLNewcQPx6xlPJbyB4WPGdxz0YyYDwlIZ
1WLeA==
$
```

d. Create tarball using "bsdtar".

```
user@x86-Linux$ bsdtar cvf ima_test.bsdtar *
```

5. Download the test file tarball to target module with “adb push” or other ways.

```
[C:\ktemp]$ adb push ima_test.bsdtar /tmp
```

6. Run basic test.

a. Decompress test files on the target module.

```
root@swi-mdm9x28-wp:/tmp# bsdtar xvf ima_test.bsdtar
x ima_test_no_sign.sh
x ima_test_sign.sh
x ima_test_wrong_sign.sh
root@swi-mdm9x28-wp:/tmp#
```

b. Check file extend attributes.

```
root@swi-mdm9x28-wp:/tmp# getfattr -d -m- ima_test_no_sign.sh
# file: ima_test_no_sign.sh
security.SMACK64="_"

root@swi-mdm9x28-wp:/tmp# getfattr -d -m- ima_test_sign.sh
# file: ima_test_sign.sh
security.SMACK64="_"
security.ima=0sAwIClixtmgEATaDEGkl+KNPg3fh8zkBXTDPset/TPsk85louEH5Wt1lBdC9KxyMipb2SFPEMnakI
zWs3ZSEmXz/WIEA+oD9Lje9SVHcPDYTGmzVYHLz3kHRsYv438/LqmBHNOPChaOdS/cleworCGZJRUYgtgL6lOnc+zc4
x+y95zCd8H86sM0EqgbvK2/byjVWtJlu3qxYtxXqb1v18xO/nF5+2Kan227cRIvQyrEI799q+5Fs1mLM7SS9xaxVhGN
0c6kgwgSWoTBYDc6sYhKsH78UHSefFICcO6mV797wVZ3KXQBVcYPhwhGiCGqmGfl78oxiNbHGJSOjUeShTGP2DCqPYt
sPHog==

root@swi-mdm9x28-wp:/tmp# getfattr -d -m- ima_test_wrong_sign.sh
```

```
# file: ima_test_wrong_sign.sh
security.SMACK64="_"
security.ima=0sAwIC0o/zFAEARN/4DEviN/xzX96AJxnoO7F+rNxHLJj+/ZqLHoUiBFhfdkkGl4S7HPoLRA2JyOMF
NdNZJs3ArZhOaMiEjEGsSYJdyFWdokTCqSX3RTs6yHF67R6tSQRUWMzUkwI/PoiutFFfQr5KPAJ0ntLhMkK6Scp9hNT
4kqN5n0etDIT+TUc7KXsUph0r7Wj9d0CqAz5eAy3iwazNGEJAJRPXsbgqVE+c+EhSDB8/DBnjRV8zJJgTnfk+uLSRMS
ui7OHd9OVeHgtrKlvzXQoHx7gGGpVNvZdif7oR6zt21/gbsQxbhllFkhRrGLNewcQPx6xlPJbyB4WPGdxz0YyYDwlIZ
1WLeA==

root@swi-mdm9x28-wp:/tmp#
```

c. Run test script without signature; expect it to fail.

```
root@swi-mdm9x28-wp:/tmp# ./ima_test_no_sign.sh
-sh: ./ima_test_no_sign.sh: Permission denied
root@swi-mdm9x28-wp:/tmp#
```

d. Run test script with the correct signature; expect it to pass.

```
root@swi-mdm9x28-wp:/tmp# ./ima_test_sign.sh
Hello, print from shell script with correct sign
root@swi-mdm9x28-wp:/tmp#
```

e. Run test script with the wrong signature; expect it to fail.

```
root@swi-mdm9x28-wp:/tmp# ./ima_test_wrong_sign.sh
[88170.234253] integrity: Request for unknown key 'id:d28ff314' err -11
-sh: ./ima_test_wrong_sign.sh: Permission denied
root@swi-mdm9x28-wp:/tmp#
```

7 Understand IMA Policy

Related information to understand policy rules, or to update policy, are listed below.

7.1 IMA Policy Format

Default IMA policy with the Sierra Wireless Software Development Kit is as follows:

1. Do not enable IMA function for the following: File System, Squash fs, Proc fs, Sysfs, Debugfs, ramfs, V9FS, BDEVFS, DEVPTS fs, pipefs, sockfs, binfmtfs, securityfs, selinux fs, Usbdevice, Cgroup. Please refer to kernel/include/uapi/linux/magic.h and security/integrity/ima/ima_policy.c to get details for each file system.

Note: For debug purposes, the user can change the policy file. For example, if the user wants to debug some application via NFS without IMA protection, the user can uncomment "# dont_measure fsmagic=0x6969" and "# dont_appraise fsmagic=0x6969" in the IMA policy file.

2. Protect executable files. If an executable file does not have a valid ima signature, it will be forbidden to run.

```
#
# This is default IMA security policy. The real one should not be
# shared with the public.
#
# Magics can be found in kernel/include/uapi/linux/magic.h
# Default can be found in security/integrity/ima/ima_policy.c
# SQUASH_SUPER_MAGIC = 0x73717368
dont_measure fsmagic=0x73717368
```

```
dont_appraise fsmagic=0x73717368
# Magics can be found in kernel/include/uapi/linux/magic.h
# Default can be found in security/integrity/ima/ima_policy.c
# PROC_SUPER_MAGIC = 0x9fa0
dont_measure fsmagic=0x9fa0
dont_appraise fsmagic=0x9fa0
# SYSFS_MAGIC = 0x62656572
dont_measure fsmagic=0x62656572
dont_appraise fsmagic=0x62656572
# DEBUGFS_MAGIC = 0x64626720
dont_measure fsmagic=0x64626720
dont_appraise fsmagic=0x64626720
# TMPFS_MAGIC = 0x01021994
# dont_measure fsmagic=0x01021994
# dont_appraise fsmagic=0x01021994
# RAMFS_MAGIC = 0x858458f6
dont_measure fsmagic=0x858458f6
dont_appraise fsmagic=0x858458f6
# V9FS_MAGIC = 0x01021997
dont_measure fsmagic=0x01021997
dont_appraise fsmagic=0x01021997
# BDEVFS_MAGIC = 0x62646576
dont_measure fsmagic=0x62646576
dont_appraise fsmagic=0x62646576
# DEVPTS_SUPER_MAGIC = 0x1cd1
dont_measure fsmagic=0x1cd1
dont_appraise fsmagic=0x1cd1
# PIPEFS_MAGIC = 0x50495045
dont_measure fsmagic=0x50495045
dont_appraise fsmagic=0x50495045
# SOCKFS_MAGIC = 0x534F434B
dont_measure fsmagic=0x534F434B
dont_appraise fsmagic=0x534F434B
# BINFMTFS_MAGIC = 0x42494e4d
dont_measure fsmagic=0x42494e4d
dont_appraise fsmagic=0x42494e4d
# SECURITYFS_MAGIC = 0x73636673
dont_measure fsmagic=0x73636673
dont_appraise fsmagic=0x73636673
```

```
# SELINUX_MAGIC = 0xf97cff8c
dont_measure fsmagic=0xf97cff8c
dont_appraise fsmagic=0xf97cff8c
# NFS_MAGIC = 0x6969
# dont_measure fsmagic=0x6969
# dont_appraise fsmagic=0x6969
# USBDEVICE_SUPER_MAGIC = 0x9fa2
dont_measure fsmagic=0x9fa2
dont_appraise fsmagic=0x9fa2
# CGROUP_SUPER_MAGIC = 0x27e0eb
dont_appraise fsmagic=0x27e0eb
# Some defaults for measurement
measure func=FILE_MMAP mask=MAY_EXEC
measure func=BPRM_CHECK mask=MAY_EXEC
# Do not measure kernel modules for first IMA release.
# measure func=MODULE_CHECK
# Do not measure all types that have the "logfile" SELinux attribute
# You can use seinfo -alogfile -x to get an overview of all these types
# Remainder of the defaults
# appraise func=MODULE_CHECK appraise_type=imasig
appraise func=MMAP_CHECK appraise_type=imasig
appraise func=BPRM_CHECK appraise_type=imasig
appraise obj_user=M
```

7.2 Policy Rules

Policy rules are as follows:

```
rule format: action [condition ...]

action: measure | dont_measure | appraise | dont_appraise | audit

condition:= base | lsm  [option]
   base:  [[func=] [mask=] [fsmagic=] [fsuuid=] [uid=] [fowner]]
   lsm:  [[subj_user=] [subj_role=] [subj_type=] [obj_user=] [obj_role=] [obj_type=]]
   option: [[appraise_type=]] [permit_directio]

   base:    func:= [BPRM_CHECK][MMAP_CHECK][FILE_CHECK][MODULE_CHECK][FIRMWARE_CHECK]
            mask:= [MAY_READ] [MAY_WRITE] [MAY_APPEND] [MAY_EXEC]
            fsmagic:= hex value
            fsuuid:= file system UUID (e.g. 8bcbe394-4f13-4144-be8e-5aa9ea2ce2f6)
            uid:= decimal value
            fowner:=decimal value
```

```
lsm: are LSM specific
option: appraise_type:= [imasig]
```

From the format, we can see that action will only be taken if the condition is met.

- Action Section
  - `measure & dont_measure`

    “measure” creates an ima template and then stores the template by calling the ima store template. “dont_measure” will not create an ima template.
  - `audit`

    Offer audit information
  - `appraise & don’t_appraise`

    appraise appraises file measurement; if the hash or digital signature is not valid, IMA subsystem will forbid the operation.
- Condition Section
  - Base
    - func

      The valid values of func are `FILE_CHECK, MMAP_CHECK, BPRM_CHECK, MODULE_CHECK` and `FIRMWARE_CHECK`. Set the value for which items to check.
    - mask

      Permission requested for an operation. Valid values are `MAY_READ, MAY_WRITE` or `MAY_EXEC`. For example, if a file is read, it will ask for `MAY_READ`, and if the mask is matched with the rules in policy, the action will be taken.
    - fsmagic

      File system’s super block magic number. Every file system has a magic number. Refer to magic.h in the kernel source code.
    - fsuuid

      UUID in the super block of file systems. The same type of file systems have the same fsmagic. UUID offers a way to distinguish file systems in the same type. Note that not all file systems have UUID.
    - uid

      Linux user id.
    - fower

      Owner of the file.
  - lsm

    IMA can work with SMACK by using the lsm rules. With the support of SMACK rules, it is more flexible to control the IMA behavior.

    The following is an example for IMA only appraising files with `security.SMACK64=M`; other files without “`security.SMACK64=M`” will not be appraised. In other words, operations are always permitted even for those files without a valid hash or a valid digital signature.

    To enable this feature:

    a. Add a SMACK rule at system bootup.

    ```
    echo "_ M rwxa" > /sys/fs/smackfs/load2
    ```

    b. And add this rule in the IMA policy.

    ```
    appraise obj_user=M
    ```

After the IMA policy is exported to the kernel, the IMA subsystem will appraise the files with `security.SMACK64=M` in their extended attribute.

- Option
  - appraise_type

    In circumstances where only files with a valid digital signature are allowed, set `appraise_type=imasig`. For example, if there is a rule in the policy with `appraise func=MODULE_CHECK appraise_type=imasig`.

    If a .ko file does not have a valid digital signature, it cannot be insert into the kernel. This enhances the security of the system since a kernel module cannot be pushed into the system, even if the user trying to do it has root privileges, unless the file has a valid digital signature.

7.3 Updating IMA Policy File

The default policy file is located at “meta-swi/common/recipes-security/ima-policy/files/ima.policy” after the Sierra Wireless SDK is installed.

8 Limitations

Linux command “tar” cannot include file extend attributes so “bsdtar” is added on the module. “bsdtar” is also used to create tarball on the host’s build server.

9 Support

For direct clients: contact your Sierra Wireless FAE.

For distributor clients: contact your distributor FAE.

For distributors: contact your Sierra Wireless FAE.

10 Document History

Revision	Date	History
1.0	October 09, 2019	Creation

11 Legal Notice

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM®.

This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from MMP Portfolio Licensing.

Copyright

Trademarks

Sierra Wireless®, AirPrime®, AirLink®, AirVantage®, WISMO®, ALEOS® and the Sierra Wireless and Open AT logos are registered trademarks of Sierra Wireless, Inc. or one of its subsidiaries.

Watcher® is a registered trademark of NETGEAR, Inc., used under license.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.