Application Note

BCM20737

WICED Secure Over-the-Air Firmware Upgrade

MMPWICED-Smart-AN100-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

July 30, 2014

Revision History

Revision	*Date*	*Change Description*
MMPWICED-Smart-AN100-R	07/30/14	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

About This Document

Purpose and Scope

This application note provides detailed information about how to prepare and deploy the Secure Over the Air Firmware Upgrade (SOTAFU) feature for the applications developed using Broadcom Wireless Internet Connectivity for Embedded Devices (WICED, pronounced "wick-ed") Smart™ development system. It is intended for software developers who are using the WICED Smart Development System to create applications for Broadcom Bluetooth Smart devices that are required to support Secure Over the Air Firmware Upgrade.

Note: This document applies to WICED Smart SDK 2.1.0 and later versions.

Note: The SOTAFU feature is available for devices that are based on BCM20737 chips.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	Description
Bold	User input and actions: for example, type exit, click **OK**, press **Alt + C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates optional command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

References

The references in this section may be used in conjunction with this document.

Note: Broadcom provides customer access to technical documentation and software through its Broadcom Support Community website (community.broadcom.com). Additional restricted material may be provided through the Customer Support Portal (CSP) and Downloads (support.broadcom.com).

For Broadcom documents, replace the "xx" in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document (or Item) Name	Number	Source
Broadcom Items		
1] WICED Smart Quick Start Guide	WICED-Smart-QSG2xx-R	WICED Website
Other Items		
2] Bluetooth Specification, v4.1 [Vol 2]	—	Bluetooth SIG

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal. For a CSP account, contact your Broadcom Sales or Engineering support representative.

General WICED support is available to registered users via the Broadcom Support Community website: community.broadcom.com

Note: Broadcom provides customer access to technical documentation and software through its Broadcom Support Community website (community.broadcom.com). Additional restricted material may be provided through the Customer Support Portal (CSP) and Downloads: (support.broadcom.com).

Overview

The SOTAFU feature of the SDK uses RSA public-key cryptography to provide a secure method to upgrade the firmware of devices based on BCM20737 chips. The RSA functions are embedded in the BCM20737 ROM and are available for the applications to use.

The device memory is split into active and passive partitions. The new firmware image is downloaded to the passive partition and verified. If verification is successful then the partitions are swapped and the new image is executed during the next device startup.

This two-step process ensures that interrupted or unverified downloads do not cause problems.

Table 1 shows the format of firmware images intended for applications that support SOTAFU. The image is prepended with application product and version information. The appended digital signature is prepared by calculating an SHA256 hash of the version information and the firmware image file and then signed using the RSASSA-PSS algorithm.

Table 1. Secure Firmware Image Format

Version Information			Firmware Image	Digital Signature
Product ID	**Major**	**Minor**		
2 Bytes	1 Byte	1 Byte	Variable	128 bytes

Verification will fail unless the following conditions are met:

- RSA verification must pass: access to the private key is required to create the new firmware image.
- The product ID of the new image must match that of the existing application.
  This requirement prevents the use of a correctly signed application built for a different product.
- The major version number must be the same or greater than that of the existing application.
  The major version should be increased when security vulnerabilities are fixed. This requirement prevents replacing the application with a version that has known security problems.

Note: The minor version number can be higher or lower than that of the current application to allow the customer to upgrade or downgrade the application.

If verification fails, the new image will be rejected and the existing application will continue to run without interruption.

Preparing the Secure Firmware Image

The WICED Smart SDK provides the tools to create a signed firmware image. Some changes are also required to the application code to support the SOTAFU feature. Detailed information is provided in the following sections. Below is a summary of the required steps to make the application SOTAFU compatible.

1. Create a public and private key pair. Save the private key in a safe location. When an updated version of the application is developed it should be signed with the same private key.
2. Modify the application project to include a file with the public key and the files to support over-the-air upgrade and memory access.
3. Modify the application source code to:
   a. Add the WICED Smart Secure Upgrade service to the GATT database.
   b. Process commands and data to perform the download.
4. Build the application.
5. Sign the application using the private key.

Below is a summary of the steps to prepare the next version of the application.

1. If security vulnerability is being fixed, increment the major version and optionally set the minor version to zero (0). Otherwise, increment the minor version.
2. Build the new version of the application.
3. Sign the application using the same private key that was used to sign the original version of the application.

Note: The WICED Smart SDK 2.1 contains a sample application that supports the SOTAFU feature (`ota_secure_firmware_upgrade`).

Private and Public Keys

The WICED Smart SDK includes a utility to create the private and public keys (`WsRsaKeyGen`).

```
C:\rsa\rel>WsRsaKeyGen.exe
 . Seeding the random number generator...
 . Generating the private key... ok
```

The utility generates two files:

- The private key (`rsa.pri`), which is used to sign the firmware image.
- The public key (`rsa_pub.c`), which should be included in the project for the application that will perform SOTAFU.

Modify the Project

In addition to the file with the application code, the makefile.mk for the project should contain three other files:

- `rsa_pub.c` (newly generated public key).
- `ws_sec_upgrade_ota.c`, which contains functions to support the GATT protocol of executing the over-the-air firmware image delivery.
- `ws_upgrade.c`, which contains functions to store and retrieve data from nonvolatile memory.

The last two files are not likely to require modification and can be copied from the `ota_secure_firmware_upgrade` sample projects.

Following is an example of the file list from the `ota_secure_firmware_upgrade` project:

```
APP_SRC = hello_sensor.c ws_sec_upgrade_ota.c ws_upgrade.c rsa_pub.c
```

Source Code to Support SOTAFU

Each application that supports SOTAFU should have:

- Unique product ID and version info
- GATT database section that includes descriptions of the SOTAFU service and characteristics
- Code to process SOTAFU commands and image data

Application Versioning

Each application contains three fields for version control. During the upgrade the application should verify that the download has the same product ID and that the major version did not decrease.

Below are some sample definitions:

```
#define HELLO_SENSOR_APP_ID            0x31AB
#define HELLO_SENSOR_APP_VERSION_MAJOR 1
#define HELLO_SENSOR_APP_VERSION_MINOR 1

const WS_UPGRADE_APP_INFO WsUpgradeAppInfo =
{
       /* .ID =              */ HELLO_SENSOR_APP_ID,
       /* .Version_Major =   */ HELLO_SENSOR_APP_VERSION_MAJOR,
       /* .Version_Minor =   */ HELLO_SENSOR_APP_VERSION_MINOR,
};
```

Use the existing product ID when a new version of an application is developed.

Increase the major version when a serious problem is fixed. Examples of serious problems include crashes under certain conditions and other security vulnerabilities. Because image verification requires that the major version number be the same or greater than that of the existing version, this prevents downgrading to a prior version that has vulnerability.

Change the minor version if no critical problem was fixed.

Changes to the GATT Database

To include SOTAFU the application GATT database shall publish a special Broadcom vendor-specific service.

Definitions for the handles and UUIDs are included in the `ws_sec_upgrade_ota.h` header file.

Typically the application will need to include the following snippet in the GATT database after the last application service. The snippet can be added as-is with the exception of the last two lines, which contain the product ID and version information.

```
// Handle 0xff00: Broadcom vendor specific WICED Smart Secure Upgrade Service.
// The service has 3 vendor specific characteristics.
PRIMARY_SERVICE_UUID128 (HANDLE_WS_UPGRADE_SERVICE, UUID_WS_SECURE_UPGRADE_SERVICE),

// Handle 0xff01: characteristic WS Control Point, handle 0xff02 characteristic
// value. This characteristic can be used by the client to send commands to this
// device and to send status notifications back to the client. Client has to enable
// notifications by updating Characteristic Client Configuration Descriptor
// (see handle ff03 below).
CHARACTERISTIC_UUID128_WRITABLE     (HANDLE_WS_UPGRADE_CHARACTERISTIC_CONTROL_POINT,
                                     HANDLE_WS_UPGRADE_CONTROL_POINT,
                                     UUID_WS_SECURE_UPGRADE_CHARACTERISTIC_CONTROL_POINT,
                                     LEGATTDB_CHAR_PROP_WRITE |
                                     LEGATTDB_CHAR_PROP_NOTIFY |
                                     LEGATTDB_CHAR_PROP_INDICATE,
                                     LEGATTDB_PERM_WRITE_REQ,
                                     3),
       0x00,0x00,0x00,

// Handle 0xff03: Characteristic Client Configuration Descriptor.
// This is a standard GATT characteristic descriptor.  2 byte value 0 means that
// message to the client is disabled.  Peer can write value 1 to enable
// notifications.
CHAR_DESCRIPTOR_UUID16_WRITABLE     (HANDLE_WS_UPGRADE_CLIENT_CONFIGURATION_DESCRIPTOR,
                                     UUID_DESCRIPTOR_CLIENT_CHARACTERISTIC_CONFIGURATION,
                                     LEGATTDB_PERM_READABLE | LEGATTDB_PERM_WRITE_REQ,
                                     2),
       0x00,0x00,

// Handle 0xff04: characteristic WS Data, handle 0xff05 characteristic value
// This characteristic is used to send next portion of the FW.
CHARACTERISTIC_UUID128_WRITABLE     (HANDLE_WS_UPGRADE_CHARACTERISTIC_DATA,
                                     HANDLE_WS_UPGRADE_DATA,
                                     UUID_WS_SECURE_UPGRADE_CHARACTERISTIC_DATA,
                                     LEGATTDB_CHAR_PROP_WRITE,
                                     LEGATTDB_PERM_VARIABLE_LENGTH |
                                     LEGATTDB_PERM_WRITE_REQ,
                                     20),
       0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
       0x00,0x00,0x00,0x00,

// Handle 0xff06: characteristic Application Info, handle 0xff07 characteristic value
// Client can read value of this characteristic to figure out which application id is
// running as well as version information.
CHARACTERISTIC_UUID128       (HANDLE_WS_UPGRADE_CHARACTERISTIC_APP_INFO,
                              HANDLE_WS_UPGRADE_APP_INFO,
                              UUID_WS_SECURE_UPGRADE_CHARACTERISTIC_APP_INFO,
                              LEGATTDB_CHAR_PROP_READ, LEGATTDB_PERM_READABLE, 4),
       HELLO_SENSOR_APP_ID & 0xff, (HELLO_SENSOR_APP_ID >> 8) & 0xff,
       HELLO_SENSOR_APP_VERSION_MAJOR, HELLO_SENSOR_APP_VERSION_MINOR,
```

Write Handler

In the write callback the application should watch for the client data packet for the SOTAFU. The following code snippet shows additional processing in the `write_handler` function.

```
// To Support Secure Over the Air Firmware Upgrade write requests passed to the
// corresponding handles need to be addressed to ws_upgrade functions
if ((len > 0) &&
    (handle == HANDLE_WS_UPGRADE_CONTROL_POINT))
{
    return (ws_upgrade_ota_handle_command (attrPtr, len));
}
else if ((len == 2) &&
         (handle == HANDLE_WS_UPGRADE_CLIENT_CONFIGURATION_DESCRIPTOR))
{
    return (ws_upgrade_ota_handle_configuration (attrPtr, len));
}
else if ((len > 0)  &&
         (len <= WS_UPGRADE_MAX_DATA_LEN) &&
         (handle == HANDLE_WS_UPGRADE_DATA))
{
    return (ws_upgrade_ota_handle_data (attrPtr, len));
}
```

`ws_sec_upgrade_ota.c` provides the source code for the `ws_upgrade_ota_` functions: typically it will not need to be modified.

Build the SOTAFU Image

Applications should be built for the BCM920737TAG_Q32 platform to use the RSA library available in the ROM code of the BCM20737 devices.

When a target application is built a separate binary file for the over-the-air upgrade is created in the build directory. For example, when the `ota_secure_firmware_upgrade-BCM920737TAG_Q32` is successfully built the `ota_secure_firmware_upgrade-BCM920737TAG_Q32-rom-ram-Wiced-release.ota.bin` file is created in the `WICED-Smart-SDK\build\ota_secure_firmware_upgrade-BCM920737TAG_Q32-rom-ram-Wiced-release` directory.

Sign the SOTAFU Image

Execute the `WsRsaSign` utility, passing the name of the file that contains the private key, the image file name, the product ID, the major version, and the minor version numbers.

The product ID, the major version, and the minor version specified in the command should be identical to that in the source code in the `WsUpgradeAppInfo` structure and in the GATT database. The command below is used to sign the application with ID = 0x3a19, major version = 1 and minor version = 1.

Note: The following procedure should be executed in a secure environment because the private key is used.

```
C:\rsa\rel>WsRsaSign.exe rsa.pri ota_secure_firmware_upgrade-BCM920737TAG_Q32-rom-ram-Wiced-
release.ota.bin 3A19 1 1

       . Seeding the random number generator...
       . Reading private key from 'rsa.pri'
       . Generating the RSA/SHA-256 signature
       . Done (created "ota_secure_firmware_upgrade-BCM920737TAG_Q32-rom-ram-Wiced-
              release.ota.bin.signed")
```

The signed output file is used to perform the upgrade.

Upgrade Firmware

The `ota_secure_firmware_upgrade\peerapps` directory contains the source code and the executables to perform the upgrade. Use the appropriate file for the Windows, Linux, or MAC OS environment, passing the name of the signed firmware file as a parameter. For example, for Windows use:

```
C:\rsa\rel>WsSecOtaUpgrade.exe ota_secure_firmware_upgrade-BCM920737TAG_Q32-rom-ram-Wiced-
release.ota.bin.signed.
```

The application displays the progress of the operation and the status when the procedure is complete.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com

MMPWICED-Smart-AN100-R July 30, 2014