USB 3.2 ENGINEERING CHANGE NOTICE

Title: USB FW Update
Applied to: USB 3.2 Revision 1.0

Brief description of the functional changes:
This ECN describes a way by which embedded USB devices inside the platform can provide the status of the FW image running on the device. This information can in turn be used by the platform SW to detect a device FW update attack.

Benefits as a result of the changes:
USB devices may implement a field FW update capability. However devices may not implement a cryptographically verifiable mechanism to ensure that the FW image on the device is a legal image provided by the device vendor. This may result into a compromised firmware being loaded on the device allowing it to create new attacks on the platform. This specification coexists with the current USB DFU (Device firmware update) specification. Implementation of USB DFU is not mandatory to use the capability defined in this specification. Current USB Type-C Authentication specification does not address this specific attack and requires more complex HW for its full implementation. The ECN adds a new standard methodology hat may be implemented by the USB device such that 1. The device provides a feedback if it supports FW_Status capability using a BOS Descriptor. 2. If the device supports FW_Status capability then it provides a cryptographic hash of its current FW image. 3. This capability would allow higher level SW to ensure that the FW image hash matches a gold image hash value and proceed to use the device with high confidence. Optionally higher level SW may recover the USB device FW to a known gold image. 4. The implementation of these “new commands and the existing FW update commands” must use ROM equivalent code such that their execution is not subject to be attacked by the firmware update process and allow a recovery process in case illegal firmware image gets loaded on the device.

An assessment of the impact to the existing revision and systems that currently conform to the USB specification:
Hardware Implications: There is no hardware implication to existing devices if it encounters a new system which issues the command. If the device does not support the new command then it will respond to the command with a stall response as currently defined. This will allow SW to make appropriate security policy decisions. Software Implications: There is no software implication to existing systems which encounter one of these new devices. The SW will simply not take advantage of the new command and the new capability offered by the device.

An analysis of the hardware implications:

The device must implement capability to compute SHA256 hash of its firmware image and respond to the new commands.

The device must ensure that the new commands and existing FW update commands are always executed from a ROM equivalent FW. The ROM equivalent FW must have access to secure storage so that FW image hash can be pre-computed and returned to the host SW with minimal latency.

An analysis of the software implications:

SW executes the BOS Descriptor command to determine if the device supports Field updateable FW. If the device does support Field updateable FW, then SW will ensure that the hash of the FW image matches the expected gold value of the FW image. If the image hash does not match the expected gold hash, then SW may update the FW so that the device contains the FW image with the expected gold value.

This ECN expects an out of band mechanism by which SW can obtain the expected image gold values from the device vendor. This can be implemented in any number of way such as the device vendor providing the gold image values on a cloud server that the SW can access.

An analysis of the compliance testing implications:

The expected compliance testing change is that the new commands will be added to standard compliance testing. The device response must meet the defined format.
If the device supports FWStatus then the device must demonstrate that the device supports hardened FW update mechanism through interoperability testing. The interoperability testing should include the devices ability to recover to a known good state from after an incomplete FW update.

Actual Change

INSTRUCTIONS: In this section of the ECN, any paragraph starting with the word "INSTRUCTIONS" contains instructions for modifying the text of the USB specification. Any other paragraphs contain new text that should be inserted in the USB specification.

INSTRUCTIONS: In Table 9-5 two new entries are added:

GET_FW_STATUS	1AH
SET_FW_STATUS	1BH

INSTRUCTIONS: Add two new sections between current 9.4.13 and 9.4.14 with the following description. The current 9.4.14 section will become 9.4.16

9.4.14 Set_FW_Status

The default power on state of the device is that it allows FW update. This write command is used to request the device to disallow or allow further firmware upgrades. The state is retained by the device until the next power on or receipt of the FLR command. This state survives the device entering selective suspend state.

Field	Byte	Size	Value	Description
bmRequestType	0	1	00H	Host to Device, Standard
bRequest	1	1	1BH	SET_FW_STATUS
wValue	2	2	00H or 01H	00H = Disallow FW update 01H = Allow FW update 02H-0FFH reserved
wIndex	4	2	00H	Zero
wLength	6	2	0H	Data Length

9.4.15 Get_FW_Status

In response to this read command, depending on the wValue, the device returns the current status of the FW update allowed/disallowed state or the current image hash using SHA256 algorithm. The device is required to add minimal latency overhead with this request.

Field	Byte	Size	Value	Description
bmRequestType	0	1	80H	Device to host, Standard
bRequest	1	1	1AH	GET_FW_STATUS
wValue	2	2	00H	00=Allowed/Disallowed State
wIndex	4	2	00H	Zero
wLength	6	2	01H	Data Length 01H when wValue is 00
Data	8	1	Update_Allowed	01 when Update allowed, 00 when Update disallowed. 02-0FF reserved

Field	Byte	Size	Value	Description
bmRequestType	0	1	80H	Device to host, Standard
bRequest	1	1	1AH	GET_FW_STATUS
wValue	2	2	01H	01=Hash, 02H-0FFH reserved
wIndex	4	2	00H	Zero
wLength	6	2	20H	Data Length 20H when wValue is 01H
Data[31:00]	8	32	Hash	SHA256 Hash

The implementation of this command must be equivalent to a ROM based implementation such that future updates to the device FW should not result into changes in the behavior of this command execution. Such an implementation is necessary to ensure that if the updateable portion of the device FW is updated by malware it can be detected by higher level SW using this command.

In order to minimize the latency associated with actual execution of this command the hash can be pre-computed and stored in a persistent storage that is only accessible to the hardened code. This will require the device to disallow writes to the address space storing the hash prior to the exiting the ROM based code execution.

If SW determines that the received hash does not match the expected hash value, it may use FW update to ensure that the firmware is updated to a gold state. The device may continue to reuse the existing protocol for firmware update.

INSTRUCTIONS: In Table 9-10 a new entry is added at the end:

Parameter	Event							
	Warm Reset	Hot Reset	Set Address 0	Set Address	Set Configuration	Set Interface	Clear feature	Disconnect
Disallow /Allow FW update	X	X						X

INSTRUCTIONS: In Table 9-14 a new entry is added at the end:

11H	FWStatus	Describes the capability to support FWStatus

INSTRUCTIONS: Add a new section 9.6.2.7 with following description

9.6.2.7 FWStatus

This section defines the required device level capabilities descriptor which shall be implemented by all hubs and devices that support the FWStatus capability. This capability descriptor cannot be directly accessed with GetDescriptor() or SetDescriptor() request.

Table 9-21 FWStatus Capability

Offset	**Field**	**Size**	**Value**	**Description**
0	bLength	1	Number	Size of the descriptor
1	bDescriptorType	1	Constant	DEVICE CAPABILITY DESCRIPTOR TYPE
2	bDevCapabilityType	1	Constant	Capability Type: FWStatus
3	bcdDescriptorVersion	1	Number	Shall be set to current definition of the capability as defined. Initial value will be one
4	bmAttributes	4	Bitmap	Bitmap encoding of supported device level features Bit: Encoding 0: Get FW Image Hash support 1: Disallow FW Update support 2-31: Reserved