INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2017-2020

SG17-TD1547

STUDY GROUP 17

Original: English

Question(s): 6/17 Geneva, 29 August – 7 September 2018

TD

Source: Editors

Title: New revised baseline text of draft Recommendation X.secup-iot (Secure software update for IoT devices)

Purpose: Information

Contact: Takeshi Takahashi
NICT
Japan
E-mail: takeshi_takahashi@nict.go.jp

Contact: Koji Nakao
NICT
Japan
E-mail: ko-nakao@nict.go.jp

Keywords: Security, Update, IoT

Abstract: This document presents the new baseline text of X.secup-iot, Secure software update for IoT devices. This document is based on C357, and additional changes (suggested during the Q6/17 meeting) are recorded with change records.

Draft Recommendation ITU-T X.secup-iot

Secure software update for IoT devices

Summary

This Recommendation provides 1) basic models and procedures for securely updating IoT software/firmware and 2) requirements and capabilities for updating IoT firmware.

A common secure update procedure is defined with general requirements. With these, IoT software/firmware updates can be securely implemented in common among stakeholders in IoT environment such as IoT device developer, IoT system/service providers.

Keywords

Security, Update, IoT.

Table of Contents

Introduction

Recently, cyber-attacks against IoT devices/systems are increasingly sophisticated, intelligent and varied. Previously, the functions of most of IoT devices were recognized to be fixed by the IoT device venders in their initial release phase. However, recently, the devices are basically connected to the Internet so as to enhance a set of provided services in IoT. Therefore, we should seriously recognize the fact that the IoT devices in use are facing cyber-threats/attacks and software/firmware implemented in IoT need to be securely updated to fix its found vulnerabilities and weakness. Some device venders have already started providing their firmware update service, but such vendors provide the update service by means of their own scheme.

This Recommendation provides basic models and procedures for securely updating IoT software/firmware and pertaining requirements and capabilities. With the basic models and the common update procedure, IoT software/firmware can be securely exchanged among stakeholders in IoT environment and obsolete IoT software/firmware will be encouraged to be updated.

Draft Recommendation ITU-T X.secup-iot

IoT Software Update Procedure

1. Scope

This Recommendation aims to provide basic models and procedures for securely updating IoT software/firmware. It also describes the requirements and capabilities for IoT software/firmware updates.

2. References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

3. Definitions

<Check in the ITU-T Terms and definitions database on the public website whether the term is already defined in another Recommendation. It may be more consistent to refer to such a definition rather than redefine it>

<TBD>

3.1 Terms defined elsewhere

<Normally terms defined elsewhere will simply refer to the defining document. In certain cases, it may be desirable to quote the definition to allow for a stand-alone document>

This Recommendation uses the following terms defined elsewhere:

3.1.1 <Term 1> [Reference]: <optional quoted definition>

3.1.2 <Term 2> [Reference]: <optional quoted definition>

<TBD>

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 **Author:** An entity that produces software for IoT devices. It could be a company, an organization, a group, or an individual

3.2.2 **Firmware Consumer:** An entity that stores and executes firmware on an IoT device

3.2.3 **Status Tracker:** An entity that checks and keeps tabs on the status of IoT devices under its administration and initiate firmware updates

3.2.4 **Firmware Server:** An entity that distributes firmware packages

3.2.5 **Manifest:** A record that contains metadata of a firmware image

4. Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

FW-Consumer: Firmware Consumer

FW-Server: Firmware Server

5. Conventions

<Describe any particular notation, style, presentation, etc. used within the Recommendation, if any>

<*TBD*>

6. Basic Model

The network architecture of IoT devices may differ, but four functional entities are required in all the cases, i.e., FW-Consumer, Status Tracker, Author, and FW-Server. In the basic model, they play indispensable roles to achieve IoT software/firmware update. The basic scenario in this model is simple; *a Status tracker that recognized the need for IoT software/firmware update initiates the software/firmware update procedure that allows FW-Consumer to receive a software/firmware image from an Author through FW-server*. More detailed behaviors of these four entity models are elaborated below.

A **FW-Consumer** stores and runs firmware on an IoT device. An IoT device has one or more than one FW-Consumers.

A **Status Tracker** checks and keeps tabs on the status of the firmware inside FW-Consumers. It may take care of one or more than one FW-Consumers. For instance, it monitors the status of firmware images used by multiple FW-Consumers inside an IoT device. It may have the list of FW-Consumers that have already completed the update and initiate firmware update procedure upon needed. It may reside inside an IoT device, or inside an network device in an internal network, or inside an network device on the Internet. Multiple Status Trackers may be cascaded so that upstream Status Tracker may judge the need for firmware update and initiate the firmware update procedure through the downstream Status Trackers.

An **Author** produces firmware image of IoT devices and uploads it to a FW-server. It may upload the image to multiple FW-servers. An Author could be an individual or a group, such as a company or any other types of organizations. An Author should not trust a FW-server without careful considerations and should implement measure to secure the firmware image it uploads to FW-servers.

A **FW-server** distributes firmware packages. It could accept firmware images from multiple Authors. A FW-server could be a repository for a particular vendor or a repository that accepts various vendors. Ideally, a FW-server is trustful, but it could be untrusted; it could view or modify the firmware packages received from Authors.

Note that multiple of these functional entities may reside inside one node. For instance, a webcam device contains multiple FW-Consumers and a Status Tracker, while a web server contains Status tracker and FW-Server. Multiple FW-Consumers may reside inside one network and are monitored by a Status Tracker implemented inside the gateway. Depending on the degree of constraints of the IoT devices, such design may differ. Typical deployment models are described in Section 8.

7. Update procedures

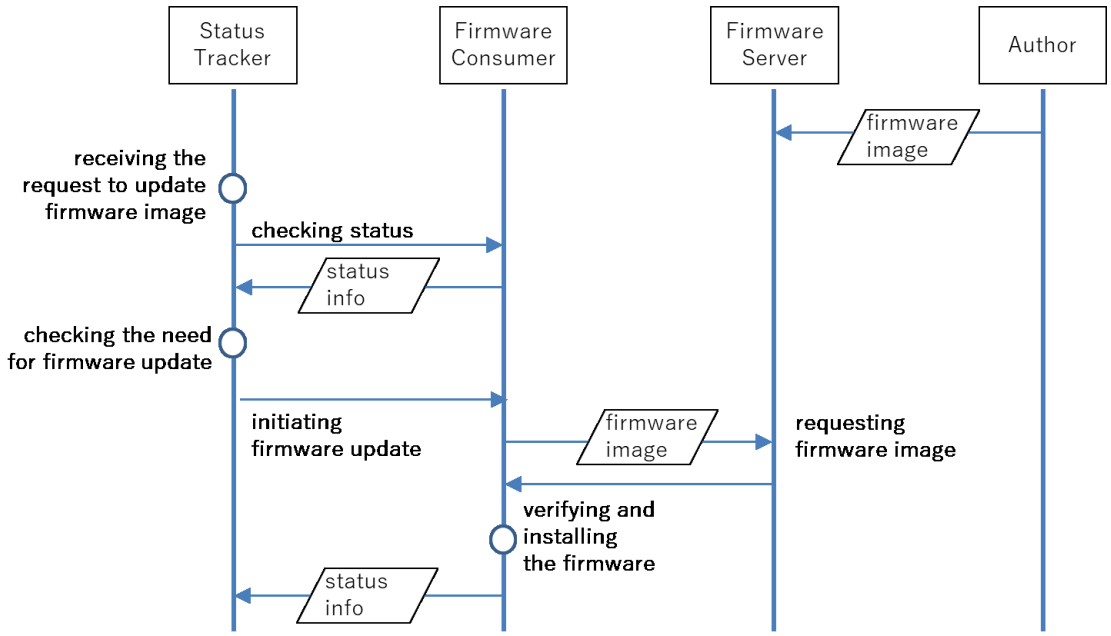

Figure 1 Protocol Procedure

Figure 1 describes the basic procedure for updating the firmware. Prior to initiating firmware update procedure, an Author needs to upload a new firmware image to a Firmware Server. It is desirable that the image is accompanied with a digital signature and is encrypted by the Author.

When a Status Tracker receives the request to update firmware image with its location (e.g., URL), it verify the request and, if the request is valid, then it checks the status of the firmware by communicating with FW-Consumer in order to confirm the need for firmware update. Note that some of the typical ways of sending such requests are listed in Section 9.

If the need for updating the firmware is verified, the FW-Consumer initiates firmware update by informing the location of the available firmware. The FW-Consumer then requests the updated firmware image to the Firmware server. The FW-Server provides the firmware image to the FW-Consumer provided the FW-Consumer has the legitimate right to receive the update. Otherwise, the server sends an update message with error code.

Upon receiving the update message, the FW-Consumer verifies the image. If no error is found, the FW-Consumer installs the firmware and sends status information to the status tracker. Note that the cardinalities of the above four functional entities are many to many, i.e., multiple Status Trackers may communicate with multiple FW-Consumers, which may communicate with multiple FW-Servers, which may communicate with multiple Authors.

8. Deployment Scenarios

As mentioned above, multiple of the functional entities may reside inside one node, and multiple entities may serve as a functional entity; the deployment scenarios may differ depending on cases. In this section, several deployment scenarios are illustrated.

8.1 Functional entities inside IoT devices

Figure 2 shows four different types of IoT devices. An IoT device must contain at least one FW-Consumers because it is natural that an IoT device contains multiple firmware images.

An IoT device must contain at least one Status Tracker. It could contain multiple Status Trackers to handle multiple FW-Consumers, but having a single Status Tracker that handles all of the FW-Consumers works fine as well.

A resource-constrained IoT device may wish to minimize the functionality of Status Tracker. In this case, some functionality of Status Tracker is outsourced to another Status Tracker (i.e., upstream Status Tracker) deployed outside the IoT device. However, the IoT device still need a Status Tracker to receive the trigger of initiating the firmware update procedure, to verify the trigger, and to check the status of the FW-Consumer(s) inside the IoT device.

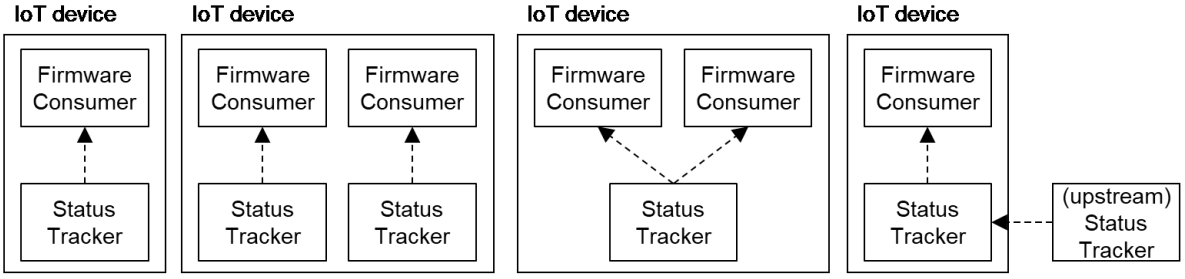

Figure 2 Different types of IoT devices

8.2 Cascading Status Trackers

Status Trackers could be cascaded. Below illustrate the cases where (1) a Status Tracker inside an IoT device directly communicates with a FW-Server, (2)a Status Tracker insdie an IoT device communicates with a FW-Server via another Status Tracker residing inside the Intranet, and (3) a Status Tracker inside an IoT device communicates with a FW-Server via multiple Status Trackers.

(1) A Status Tracker inside an IoT device directly communicates with a FW-server

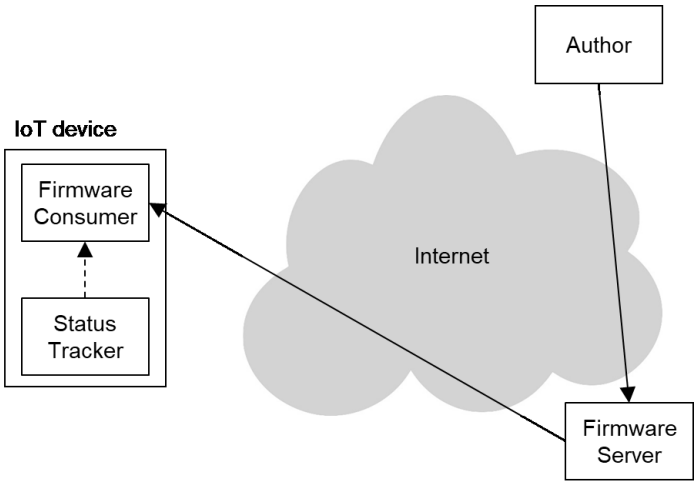

Figure 3 IoT device is directly connected to the Internet

Figure 2 describes a scenario, where a FW-Consumer and a Status Tracker resides inside an IoT device that is directly connected to the Internet. When the Status Tracker realizes the need for the firmware update, it asks the FW-Consumer to receive firmware images from the FW-Server.

(2) A Status Tracker inside an IoT device and the one on the Intranet works together

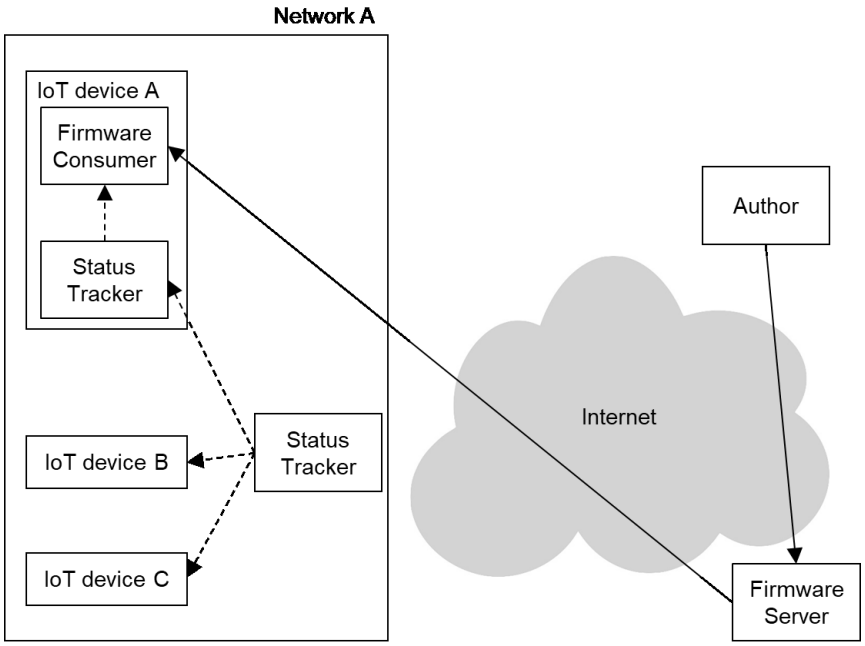

Figure 4 There is a Status Tracker taking care of multiple IoT devices

Figure 3 describes a scenario, where an Status Tracker inside a network monitors several IoT devices. The Status Tracker inside IoT devices simply verify the message from upstream Status Tracker and act accordingly. It could be seen that a functional entity of Status Tracker is realized through the cooperation between upstream Status Tracker and downstream Status Tracker. The upstream Status Tracker initiates the firmware update procedure.

(3) Multiple Status Tracker s work together

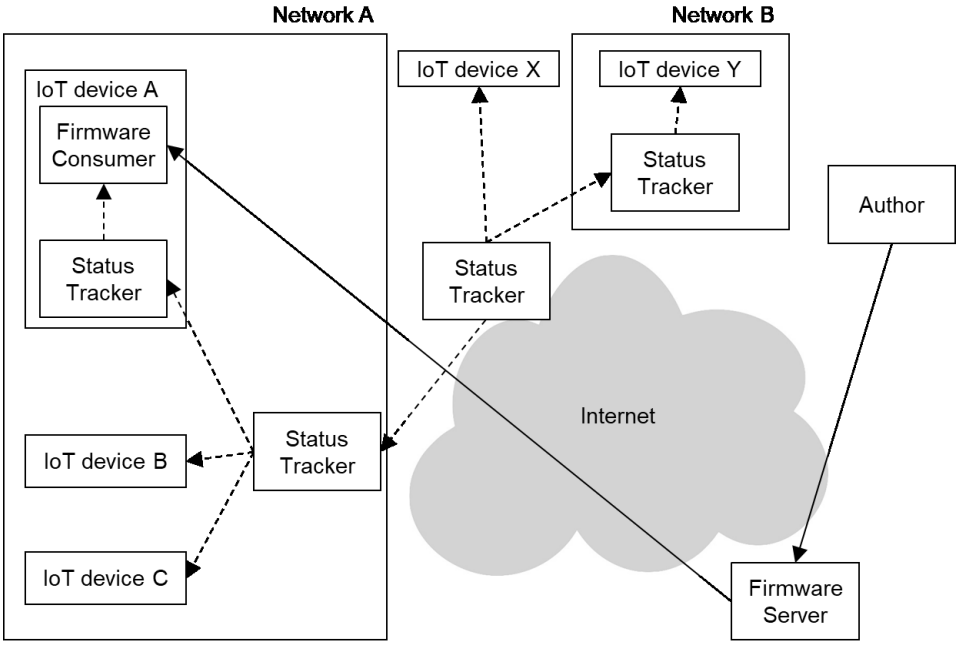

Figure 5 There is a Status Tracker taking care of multiple IoT devices inside multiple networks

Figure 4 describes a scenario, where a Status Tracker monitors IoT devices inside multiple networks. Multiple Status Tracker exist, and a functional entity of Status Tracker is realized through the cooperation between upstream Status Trackers and downstream Status Trackers. The most upstream Status Tracker initiates the firmware update procedure.

9. Discovery of available new firmware images and initiation of the procedure

The whole process is initiated by Status Tracker. It initiates the procedure when it receives a request to update the firmware image. This type of request may take various forms, including the followings:

1. An Author who publishes new version of firmware images may send the request
2. A FW-Server who receives new version of firmware image may send the request
3. An administrator of the IoT device recognizes the release of the new version of the firmware images and may send the request.
4. A Status Tracker or one of its upstream Status Trackers discover the new version of the firmware images by periodically polling the FW-Server
5. A Status Tracker or one of its upstream Status Trackers recognizes the existence of new version of the firmware image by observing a firmware update procedure of another IoT device that it takes care of

Various other events may issue the requests, but those requests need to deliver information on the location (URL) of the firmware images and their versions to the Status Tracker. If the Status Tracker judges that the information is reliable and trustful, it can initiate the procedure mentioned in Section 7.

Note that the IETF SUIT WG is currently working on the format of Manifest and firmware update procedure using the Manifest.

10. Requirements

In this section, functional requirements pertaining to IoT software update are listed. Due to the resource constraints, not all the software update procedures available in non-constrained environment are applicable. It is often the case that no human user or operator is near the IoT devices. Therefore, when designing the concrete security update procedure, these differences need to be taken into

accounts. Note that confidentiality, integrity, and availability of the four functional entities must be preserved, and these are prerequisite for the software update thus are omitted in the following sections.

1.1.

1. Malicious software/firmware must not be distributed
   A) Communication among parties must be secured to avoid man-in-the-middle attacks
   B) Malicious images should be identified before being uploaded or exchanged

2. Failure caused during the update procedure must be recoverable
   A) In case a software/firmware update failed, there should be a means to know the situation
   B) There should be a fallback means and/or protection means for the case of update process failure

3. Only intended and needed updating should be conducted
   A) Only the newer versions of IoT software/firmware are allowed to installed
   B) Only trusted IoT software/firmware images are allowed to be installed

4. Resource constraints need to be considered
   A) Update procedure should not occur if there is no need for that to minimize the network resources
   B) Status Tracker functions could be cascaded to minimize the burden of resource-constrained IoT devices

5. Intellectual Property Right of Authors must be preserved
   A) The confidentiality, integrity, and availability of IoT software/firmware images must be preserved

11. Capabilities

Based on the requirements mentioned above, the capabilities of functional entities are listed.

11.1 Capabilities of a FW-Consumer

1. A FW-Consumer shall be able to verify whether the previous run of firmware update was successful or not
2. It shall be able to share the information on its current software/firmware images (e.g., version number) with the parties who inquire this information with legitimate rights
3. It should have a fallback means and/or protection means for the case of update process failure
4. It had better have a “safe mode” that runs the IoT device with minimal functionality and that at least provides a means to manually install/restore/update firmware

5. It should be able to have a means to notify Status Tracker of the need for firmware update
6. It should be able to verify the authenticity and integrity of firmware images (e.g., by verifying their certificates)

11.2 Capabilities of a Status Tracker

1. A status tracker needs to know the lists of FW-Consumers under its administration, those with updated firmware images and those with obsolete ones
2. It should be able to know the status of the FW-Consumers under its administration
   A) There must be a means to confirm that the FW-Consumer (and the IoT device that contains the FW-Consumer) is up or not by communicating with the FW-Consumer and by checking the communication logs with the FW-Consumer.
   B) There should be a means to confirm whether previous run of firmware update at a FW-Consumer was successful or not
   C) There should be a means to know the versions of firmware the FW-Consumers run
3. It should maintain the list of FW-Consumers under the gateway, that of those with updated firmware images, and those with obsolete ones
4. It should be able to identify FW-Consumers with obsolete firmware
5. It should be able to judge whether software/firmware update is necessary or not
6. It must be able to verify the authenticity of upstream Status Tracker.

11.3 Capabilities of a firmware server

1. A firmware server must have a means to accept submissions of IoT software/firmware images from Authors
2. It must have a means to provide software/firmware images it contains to the FW-Consumers
3. It should have a means to identify malicious software/firmware images and take appropriate actions such as removing them from its internal storage and banning the submission from the Authors who submitted the images.
4. It may have a means to manage the list of Authors and FW-Consumers that use it
5. It may maintain the list of FW-Consumers and the software/firmware images they downloaded in the past.
6. It may have a means to manage versions of IoT software/firmware images
7. It may have a means to notify the IoT devices that has downloaded obsolete IoT software/firmware images in the past of the availability of their new versions
8. It may check the geographical or logical location of IoT devices in order to avoid distributing firmware in forbidden location identified by policies or the other means

11.4 Capabilities of an Author

An Author need to maintain the authenticity, confidentiality, and integrity of the firmware images it produced.

1. Firmware must not be replaced or compromised by third parties (authenticity and integrity)
2. The intellectual property of vendors within the firmware must be preserved (confidentiality)

12. Security Profiles

Several security profiles may be prepared. Needed security levels differ depending on the capability and environment of IoT devices. With the profile, appropriate firmware packages can be selected and installed/updated. For instance, boot loader may calculate the firmware size and memory size and choose appropriate firmware packages to install/update.

Bibliography

Appendix 1: Related activities outside ITU-T

The following list is the pointer to the related activities on IoT software update that have been studied outside ITU-T.

1. IOTSU workshop, https://www.iab.org/activities/workshops/iotsu/
2. IETF SUIT working group, https://datatracker.ietf.org/wg/suit/about/
3. oneM2M: Standards for M2M and the Internet of Things, http://www.onem2m.org/technical/published-drafts

etc.

_____