Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 19, 2018

B. Moran
M. Meriac
H. Tschofenig
ARM Limited

July 18, 2017

A Firmware Update Architecture for Internet of Things Devices
draft-moran-fud-architecture-00

Abstract

Vulnerabilities with IoT devices have raised the need for a solid and secure firmware update mechanism that is also suitable for constrained devices. Incorporating such update mechanism to fix vulnerabilities, to update configuration settings as well as adding new functionality is recommended by security experts.

This document specifies requires and an architecture for a firmware update mechanism aimed for Internet of Things (IoT) devices. The architecture is agnostic to the transport of the firmware images and associated meta-data.

This version of the document assumes asymmetric cryptography and a public key infrastructure. Future versions may also describe a symmetric key approach for very constrained devices.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 19, 2018.

Table of Contents

1. Introduction

When developing IoT devices, one of the most difficult problems to solve is how to update the firmware on the device. Once the device is deployed, firmware updates play a critical part in its lifetime, particularly when devices have a long lifetime, are deployed in remote or inaccessible areas or where manual intervention is cost prohibitive or otherwise difficult: - Fixes to bugs in software can be applied to the device with a firmware update. - New functionality can be added to the device with a firmware update.

The firmware update process has to ensure that - The firmware is authenticated (attempts to flash a malicious firmware are prevented). - The firmware can be confidentiality protected (attempts by an adversary to recover the plaintext binary can be prevented).

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document uses the following entities:

- Author: The author is the entity that creates the firmware image, signs and/or encrypts it and attaches a manifest to it. The author is most likely a developer using a set of tools. In some deployments the author only triggers the signing/encryption but the actual cryptographic operation are executed by a service or by a party on behalf of the developer.

- Device: The device is the recipient of the firmware image and the manifest. The goal is to update the firmware of the device.

- Untrusted Storage: Firmware images and manifests are stored on untrusted fileservers or cloud storage infrastructure. Some deployments may require storage of the firmware images/manifests to be stored on various entities before they reach the device.

Additionally, the following terms are defined:

- Manifest: The manifest contains meta-data about the firmware image and is protected against modification.

- Firmware Image: The firmware image is a binary that may contain the complete software of a device or a subset of it. The firmware image may consist of multiple images, if the device contains more than one microcontroller. The image may consist of a differential update for performance reasons.

3. Requirements

The firmware update mechanism described in this specification was designed with the following requirements in mind:

- Agnostic to how firmware images are distributed

- Friendly to broadcast delivery

- Uses state-of-the-art security mechanisms

- Operates with a small bootloader

- Minimal impact on existing firmware formats

- Robust permissions

3.1. Agnostic to how firmware images are distributed

Firmware images can be conveyed to devices in a variety of ways, including USB, UART, WiFi, BLE, low-power WAN technologies, etc and use different protocol mechanisms (e.g., CoAP, HTTP). The specified mechanism needs to be agnostic to the distribution of the firmware images/manifests.

3.2. Friendly to broadcast delivery

For an update to be broadcast friendly, it must not rely on any transport security. In addition, the same message must be deliverable to many devices; both those to which it applies and those to which it does not without a chance that the wrong device will accept the update. Considerations that apply to network broadcasts apply equally to the use of third-party content distribution networks for payload distribution.

3.3. Uses state-of-the-art security mechanisms

End-to-end security between the author and the device, as shown in Section 4, is used to ensure that the device can verify firmware images and manifests produced by authorized authors.

If the update payload is to be encrypted, it must be done in such a way that every intended recipient can decrypt it. The information that is encrypted individually for each device must be an absolute minimum.

Rollback attacks must be prevented.

All information necessary for a device to make a decision about the installation of an update must fit into the available RAM of a constrained IoT device. This prevents flash write exhaustion.

Since parsers are known sources of bugs it must be easy to parse only those fields which are required to validate at least one signature with minimal exposure.

3.4. High reliability

A power failure at any time must not cause a failure of the device. A failure to validate any part of an update must not cause a failure of the device. To achieve this, the device is required to provide a minimum of two storage locations for firmware and one bootable location for firmware. Note: This is an implementation requirement rather than a requirement on the manifest format.

3.5. Minimal bootloader

The bootloader must be minimal, containing only the flash and cryptographic primitives necessary to read the stored firmware, validate the received firmware, and write the bootable firmware. The bootloader should not require updating, since a failed update poses a risk in reliability. If more functionality is required in the bootloader, it must use a two-stage bootloader, with the first stage comprising the functionality defined above.

3.6. Minimal impact on existing firmware formats

The firmware update must not require changes to existing firmware formats.

3.7. Robust permissions

A device may have many modules that require updating individually. It may also need to trust several different actors in order to authorize an update. For example, a firmware author may not have the authority to install firmware on a device in critical infrastructure without the authorization of a device operator. In this case, the device should reject firmware updates unless they are signed both by the firmware author and by the device operator.

To facilitate complex use-cases such as this, updates require several permissions:

- Author
- Store
- Apply
- Approve
- Qualify

4. Architecture

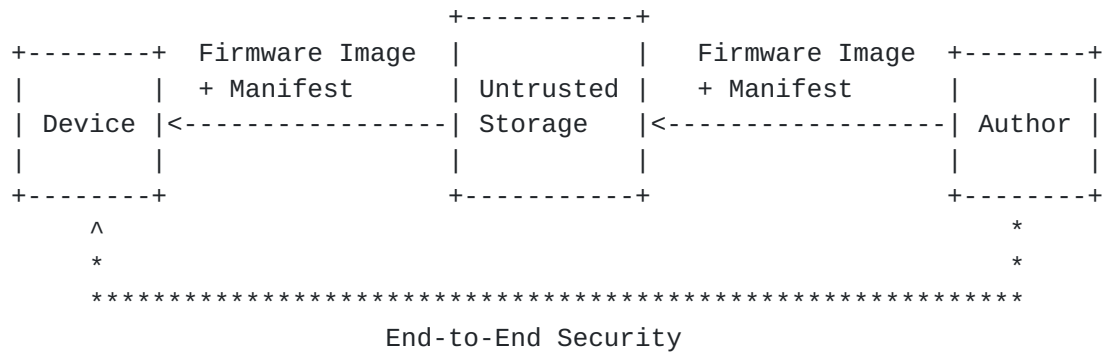
The architecture graphically shown below illustrates that an author creates a firmware image and a manifest. The firmware image may be encrypted and will be integrity protected. The meta-data is integrity protected. When the author is ready to distribute the firmware image it conveys it using his or her favorite communication channel to the device, which will typically involve the use of untrusted storage, like a file server. Whether the firmware image and the manifest is pushed to the device or fetched by the device is outside the scope of this work and existing device management protocols can be used for efficiently distributing this information.

The following assumptions are made to allow the device to verify the received firmware image and manifest before updating software:

- To accept an update, a device needs to decide whether the author signing the firmware image and the manifest is authorized to make the updates. We use public key cryptography to accomplish this. The device verifies the signature covering the manifest using a digital signature algorithm. The device is provisioned with a trust anchor that is used to validate the digital signature produced by the author. This trust anchor is potentially different from the trust anchor used to validate the digital signature produced for other protocols (such as device management

protocols). This trust anchor may be provisioned to the device during manufacturing.

- For confidentiality protection of firmware imagines the author needs to be in possession of the certificate/public key of a device.



5. Manifest

In order for a device to apply an update, it has to make several decisions about the update:

- Does it trust the author of the update?
- Has the firmware been corrupted?
- Does the firmware update apply to this device?
- Is the update older than the active firmware?
- When should the device apply the update?
- How should the device apply the update?
- What kind of firmware binary is it?
- Where should the update be obtained?
- Where should the firmware be stored?

The manifest format encodes the information that devices need in order to make these decisions.

6. Manifest

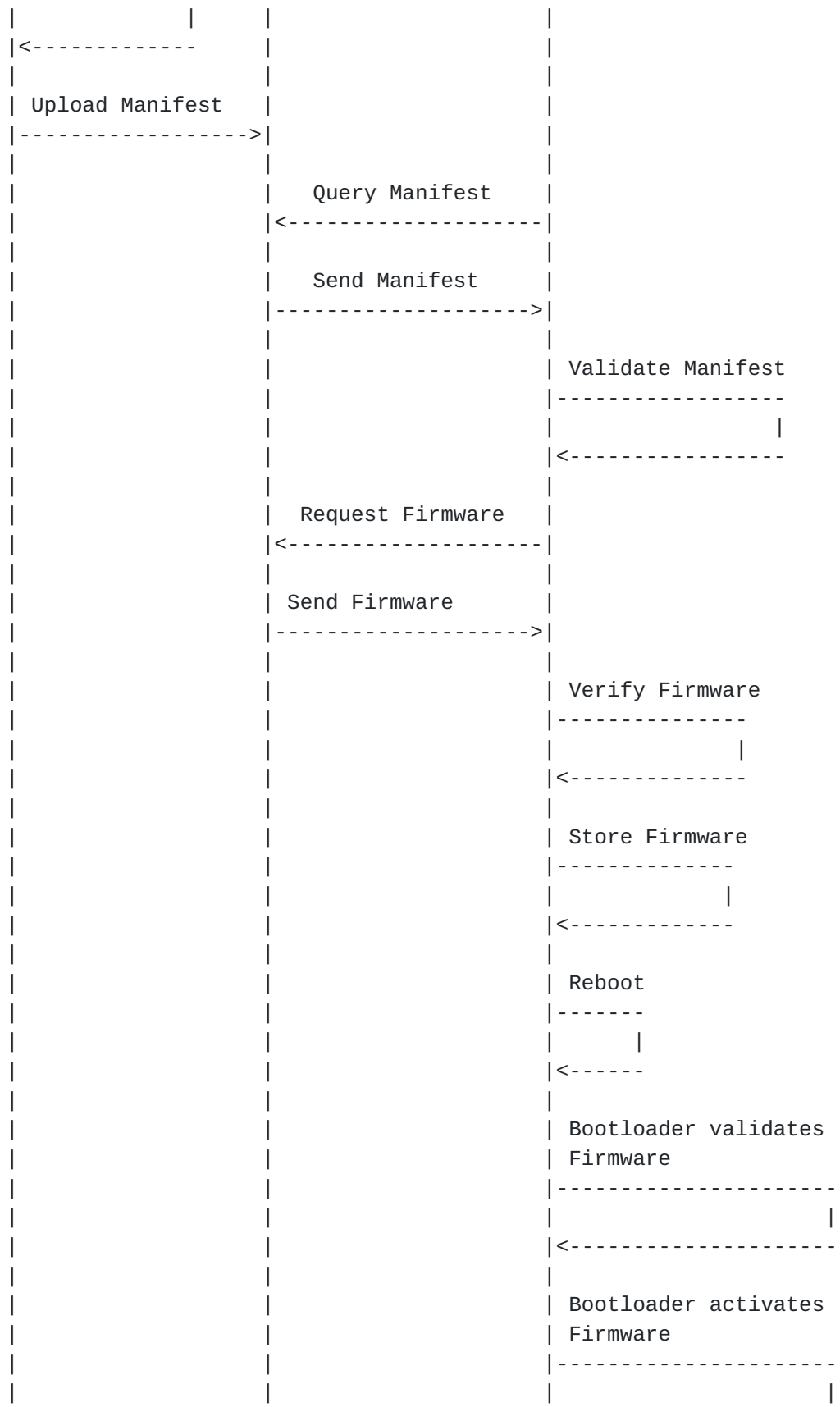
The manifest is a data structure that contains the following information:

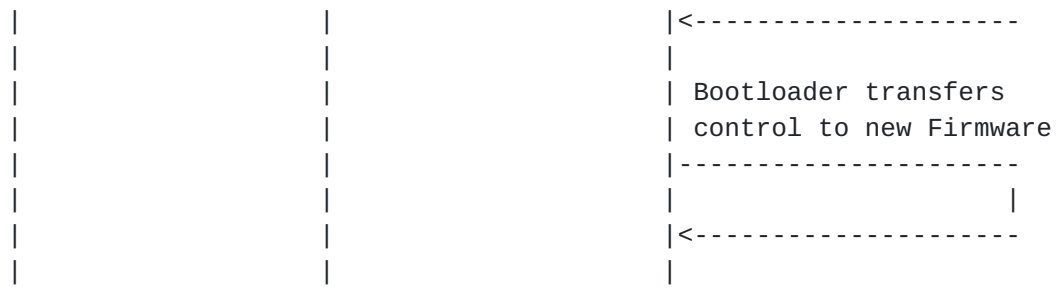
- information about the device(s) the firmware image is intented to be applied to,
- information about when the firmware update has to be applied,
- information about when the manifest was created,
- dependencies to other manifests,
- pointers to the firmware image and information about the format,
- information about where to store the firmware image,
- cryptographic information, such as digital signatures.

The manifest structure is described in a companion document.

7. Example Flow

The following example message flow illustrates the interaction for distributing a firmware image to a device starting with an author uploading the new firmware to untrusted storage and creating a manifest.


Upload Manifest
Query Manifest
Send Manifest
Validate Manifest
Request Firmware
Send Firmware
Verify Firmware
Store Firmware
Reboot
Bootloader validates Firmware
Bootloader activates Firmware



8. IANA Considerations

This document does not require any actions by IANA.

9. Security Considerations

Firmware updates fix security vulnerabilities and are considered to be an important building block in securing IoT devices. Due to the importance of firmware updates for IoT devices the Internet Architecture Board (IAB) organized a 'Workshop on Internet of Things (IoT) Software Update (IOTSU)' which took place at Trinity College Dublin, Ireland on the 13th and 14th of June, 2016 to take a look at the big picture. A report about this workshop can be found at [I-D.iab-iotsu-workshop]. This document (and associated specifications) offer a standardized firmware manifest format and an approach for offering end-to-end security from the author to the device.

There are, however, many other considerations raised during the workshop. Many of them are outside the scope of standardization organizations since they fall into the realm of product engineering, regulatory frameworks, and business models. The following considerations are outside the scope of this document, namely

- installing firmware updates in a robust fashion so that the update does not break the device functionality of the environment this device operates in.

- installing firmware updates in a timely fashion considering the complexity of the decision making process of updating devices, potential re-certification requirements, and the need for user's consent to install updates.

- the distribution of the actual firmware update, potentially in an efficient manner to a large number of devices without human involvement.

- energy efficiency and battery lifetime considerations.

- key management required for verifying the digitial signature protecting the manifest.

- incentives for manufacturers to offer a firmware update mechanism as part of their IoT products.

10. Mailing List Information

The discussion list for this document is located at the e-mail address fud@ietf.org [1]. Information on the group and information on how to subscribe to the list is at https://www1.ietf.org/mailman/listinfo/fud

Archives of the list can be found at: https://www.ietf.org/mail-archive/web/fud/current/index.html

11. Acknowledgements

We would like the following persons for their support in designing this mechanism

- Geraint Luff

- Amyas Phillips

- Dan Ros

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

12.2. Informative References

[I-D.iab-iotsu-workshop] Tschofenig, H. and S. Farrell, "Report from the Internet of Things (IoT) Software Update (IoTSU) Workshop 2016", draft-iab-iotsu-workshop-01 (work in progress), February 2017.

12.3. URIs

[1] mailto:fud@ietf.org

Authors' Addresses

Brendan Moran
ARM Limited

EMail: Brendan.Moran@arm.com

Milosch Meriac
ARM Limited

EMail: Milosch.Meriac@arm.com

Hannes Tschofenig
ARM Limited

EMail: hannes.tschofenig@gmx.net