< draft-ietf-suit-architecture-05.txt

SUIT B. Moran
Internet-Draft Arm Limited
Intended status: Informational M. Meriac
Expires: October 11, 2019 Consultant
H. Tschofenig
Arm Limited
D. Brown
Linaro
April 09, 2019

A Firmware Update Architecture for Internet of Things Devices
draft-ietf-suit-architecture-05

Abstract

Vulnerabilities with Internet of Things (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 lists requirements and describes an architecture for a

skipping to change at page 1, line 41

symmetric key approach for very constrained devices.

Status of This Memo

Copyright Notice

skipping to change at page 2, line 36

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 where manual intervention is cost prohibitive or otherwise difficult. Updates to the firmware of an IoT device are done to fix bugs in software, to add new

skipping to change at page 5, line 10

- Homogeneous Storage Architecture (HoSA): A device that stores all firmware components in the same way, for example in a file system or in flash memory.

- Heterogeneous Storage Architecture (HeSA): A device that stores at least one firmware component differently from the rest, for example a device with an external, updatable radio, or a device with internal and external flash memory.

draft-ietf-suit-architecture-06.txt >

SUIT B. Moran
Internet-Draft Arm Limited
Intended status: Informational M. Meriac
Expires: February 14, 2020 Consultant
H. Tschofenig
Arm Limited
D. Brown
Linaro
August 13, 2019

A Firmware Update Architecture for Internet of Things Devices
draft-ietf-suit-architecture-06

Abstract

Vulnerabilities with Internet of Things (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 lists requirements and describes an architecture for a

skipping to change at page 1, line 41

symmetric key approach for very constrained devices.

Status of This Memo

Copyright Notice

skipping to change at page 2, line 36

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 where manual intervention is cost prohibitive or otherwise difficult. Updates to the firmware of an IoT device are done to fix bugs in software, to add new

skipping to change at page 5, line 10

- Homogeneous Storage Architecture (HoSA): A device that stores all firmware components in the same way, for example in a file system or in flash memory.

- Heterogeneous Storage Architecture (HeSA): A device that stores at least one firmware component differently from the rest, for example a device with an external, updatable radio, or a device with internal and external flash memory.

- Trusted Execution Environments (TEEs): An execution environment that runs alongside of, but is isolated from, an REE.

- Rich Execution Environment (REE): An environment that is provided and governed by a typical OS (e.g., Linux, Windows, Android, iOS), potentially in conjunction with other supporting operating systems

draft-ietf-suit-architecture-05.txt	draft-ietf-suit-architecture-06.txt
	and hypervisors; it is outside of the TEE. This environment and applications running on it are considered un-trusted. - Trusted applications (TAs): An application component that runs in a TEE. For more information about TEEs see [I-D.ietf-teep-architecture].
The following entities are used:	The following entities are used:
- Author: The author is the entity that creates the firmware image. There may be multiple authors in a system either when a device consists of multiple micro-controllers or when the the final firmware image consists of software components from multiple companies.	- Author: The author is the entity that creates the firmware image. There may be multiple authors in a system either when a device consists of multiple micro-controllers or when the the final firmware image consists of software components from multiple companies.
- Firmware Consumer: The firmware consumer is the recipient of the firmware image and the manifest.	- Firmware Consumer: The firmware consumer is the recipient of the firmware image and the manifest. It is responsible for parsing and verifying the received manifest and for storing the obtained firmware image. The firmware consumer plays the role of the update component on the IoT device typically running in the application firmware. It interacts with the firmware server and with the status tracker, if present.
- Device: A device refers to the entire IoT product, which consists of one or many MCUs, sensors and/or actuators. Many IoT devices sold today contain multiple MCUs and therefore a single device may need to obtain more than one firmware image and manifest to succesfully perform an update. The terms device and firmware consumer are used interchangably since the firmware consumer is one software component running on an MCU on the device.	- (IoT) Device: A device refers to the entire IoT product, which consists of one or many MCUs, sensors and/or actuators. Many IoT devices sold today contain multiple MCUs and therefore a single device may need to obtain more than one firmware image and manifest to succesfully perform an update. The terms device and firmware consumer are used interchangably since the firmware consumer is one software component running on an MCU on the device.
- Status Tracker: The status tracker offers device management functionality to monitor the firmware update process. A status tracker may, for example, want to know what state of the firmware update cycle the device is currently in.	- Status Tracker: The status tracker offers device management functionality to retrieve information about the installed firmware on a device and other device characteristics (including free memory and hardware components), to obtain the state of the firmware update cycle the device is currently in, and to trigger the update process. The deployment of status trackers is flexible and they may be used as cloud-based servers, on-premise servers, embedded in edge computing device (such as Internet access gateways or protocol translation gateways), or even in smart phones and tablets. While the IoT device itself runs the client-side of the status tracker it will most likely not run a status tracker itself unless it acts as a proxy for other IoT devices in a protocol translation or edge computing device node. How much functionality a status tracker includes depends on the selected configuration of the device management functionality and the communication environment it is used in. In a generic networking environment the protocol used between the client and the server-side of the status tracker need to deal with Internet communication challenges involving firewall and NAT traversal. In other cases, the communication interaction may be rather simple. This architecture document does not impose requirements on the status tracker.
- Firmware Server: The firmware server stores firmware images and manifests and distributes them to IoT devices. Some deployments may require a store-and-forward concept, which requires storing the firmware images/manifests on more than one entity before they reach the device.	- Firmware Server: The firmware server stores firmware images and manifests and distributes them to IoT devices. Some deployments may require a store-and-forward concept, which requires storing the firmware images/manifests on more than one entity before they reach the device. There is typically some interaction between the firmware server and the status tracker but those entities are often physically separated on different devices for scalability reasons.
- Device Operator: The actor responsible for the day-to-day operation of a fleet of IoT devices.	- Device Operator: The actor responsible for the day-to-day operation of a fleet of IoT devices.
- Network Operator: The actor responsible for the operation of a network to which IoT devices connect.	- Network Operator: The actor responsible for the operation of a network to which IoT devices connect.
In addition to the entities in the list above there is an orthogonal infrastructure with a Trust Provisioning Authority (TPA) distributing trust anchors and authorization permissions to various entities in	In addition to the entities in the list above there is an orthogonal infrastructure with a Trust Provisioning Authority (TPA) distributing trust anchors and authorization permissions to various entities in
skipping to change at *page 7, line 5*	**skipping to change at *page 7, line 44***
- Operate with a small bootloader - Small Parsers - Minimal impact on existing firmware formats - Robust permissions - Diverse modes of operation	- Operate with a small bootloader - Small Parsers - Minimal impact on existing firmware formats - Robust permissions - Diverse modes of operation
	- Suitability to software and personalization data
3.1. Agnostic to how firmware images are distributed	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 protocols (e.g., CoAP, HTTP). The specified mechanism needs to be agnostic to the distribution of the firmware images and manifests.	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 protocols (e.g., CoAP, HTTP). The specified mechanism needs to be agnostic to the distribution of the firmware images and manifests.
3.2. Friendly to broadcast delivery	3.2. Friendly to broadcast delivery
skipping to change at *page 8, line 28*	**skipping to change at *page 9, line 21***
location for firmware. An alternative approach is to use a 2nd stage bootloader with build-in full featured firmware update functionality such that it is possible to return to the update process after power down.	location for firmware. An alternative approach is to use a 2nd stage bootloader with build-in full featured firmware update functionality such that it is possible to return to the update process after power down.
Note: This is an implementation requirement rather than a requirement on the manifest format.	Note: This is an implementation requirement rather than a requirement on the manifest format.
3.6. Operate with a small bootloader	3.6. Operate with a small bootloader
The bootloader must be minimal, containing only flash support, cryptographic primitives and optionally a recovery mechanism. The recovery mechanism is used in case the update process failed and may include support for firmware updates over serial, USB or even a limited version of wireless connectivity standard like a limited Bluetooth Smart. Such a recovery mechanism must provide security at least at the same level as the full featured firmware update functionalities.	Throughout this document we assume that the bootloader itself is distinct from the role of the fw consumer and therefore does not manage the firmware update process. This may give the impression that the bootloader itself is a completely separate component, which is mainly responsible for selecting a firmware image to boot.
The bootloader needs to verify the received manifest and to install the bootable firmware image. 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.	The overlap between the firmware update process and the bootloader functionality comes in two forms, namely - First, a bootloader must verify the firmware image it boots as part of the secure boot process. Doing so requires meta-data to be stored alongside the firmware image so that the bootloader can cryptographically verify the firmware image before booting it to ensure it has not been tampered with or replaced. This meta-data used by the bootloader may well be the same manifest obtained with the firmware image during the update process (with the severable fields stripped off). - Second, an IoT device needs a recovery strategy in case the firmware update / boot process fails. The recovery strategy may

include storing two or more firmware images on the device or offering the ability to have a second stage bootloader perform the firmware update process again using firmware updates over serial, USB or even wireless connectivity like a limited version of Bluetooth Smart. In the latter case the fw consumer functionality is contained in the second stage bootloader and requires the necessary functionality for executing the firmware update process, including manifest parsing.

In general, it is assumed that the bootloader itself, or a minimal part of it, will not be updated since a failed update of the bootloader poses a risk in reliability.

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

All information necessary for a device to make a decision about the installation of a firmware update must fit into the available RAM of a constrained IoT device. This prevents flash write exhaustion. This is typically not a difficult requirement to accomplish because there are not other task/processing running while the bootloader is active (unlike it may be the case when running the application firmware).

Note: This is an implementation requirement.

Note: This is an implementation requirement.

3.7. Small Parsers

3.7. Small Parsers

Since parsers are known sources of bugs they must be minimal. Additionally, it must be easy to parse only those fields that are required to validate at least one signature or MAC with minimal exposure.

Since parsers are known sources of bugs they must be minimal. Additionally, it must be easy to parse only those fields that are required to validate at least one signature or MAC with minimal exposure.

skipping to change at *page 11, line 24* skipping to change at *page 12, line 34*

process to determine the appropriate timing for an update (such as delaying the update process to a later time when end users are less impacted by the update process).

Installation is the act of processing the payload into a format that the IoT device can recognise and the bootloader is responsible for then booting from the newly installed firmware image.

Each of these steps may require different permissions.

process to determine the appropriate timing for an update (such as delaying the update process to a later time when end users are less impacted by the update process).

Installation is the act of processing the payload into a format that the IoT device can recognise and the bootloader is responsible for then booting from the newly installed firmware image.

Each of these steps may require different permissions.

3.11. Suitability to software and personalization data

The work on a standardized manifest format initially focused on the most constrained IoT devices and those devices contain code put together by a single author (although that author may obtain code from other developers, some of it only in binary form).

Later it turns out that other use cases may benefit from a standardized manifest format also for conveying software and even personalization data alongside software. Trusted Execution Environments (TEEs), for example, greatly benefit from a protocol for managing the lifecycle of trusted applications (TAs) running inside a TEE. TEEs may obtain TAs from different authors and those TAs may require personalization data, such as payment information, to be securely be conveyed to the TEE.

To support this wider range of use cases the manifest format should therefore be extensible to convey other forms of payloads as well.

4. Claims

Claims in the manifest offer a way to convey instructions to a device that impact the firmware update process. To have any value the manifest containing those claims must be authenticated and integrity protected. The credential used to must be directly or indirectly related to the trust anchor installed at the device by the Trust Provisioning Authority.

The baseline claims for all manifests are described in

4. Claims

Claims in the manifest offer a way to convey instructions to a device that impact the firmware update process. To have any value the manifest containing those claims must be authenticated and integrity protected. The credential used to must be directly or indirectly related to the trust anchor installed at the device by the Trust Provisioning Authority.

The baseline claims for all manifests are described in

skipping to change at *page 15, line 25* skipping to change at *page 16, line 25*

```
 |        |                  |           |                |        |
 +--------+                  +-----------+                +--------+
```

Figure 3: Manifest with attached firmware.

Figure 4 shows an option for remotely updating a device where the device fetches the firmware image from some file server. The manifest itself is delivered independently and provides information about the firmware image(s) to download.

```
                            /------------\
                           /              \
                           |   Manifest   |
                           \              /
 +--------+                 \------------/                +--------+
 |        |<..............................................>|        |
 | Device |                                            -- | Author |
 |        |<-                                       ---   |        |
 +--------+  --                                  ---      +--------+
               --                              ---
                 ---                         ---
                    --     +-----------+    --
                      --   |           |  --
      /------------\    -- | Firmware  |<-   /------------\
     /              \    --| Server    |    /              \
     |   Firmware   |      |           |    |   Firmware   |
     \              /      +-----------+    \              /
      \------------/                         \------------/
```

Figure 4: Independent retrieval of the firmware image.

This architecture does not mandate a specific delivery mode but a solution must support both types.

6. Manifest

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

```
 |        |                  |           |                |        |
 +--------+                  +-----------+                +--------+
```

Figure 3: Manifest with attached firmware.

Figure 4 shows an option for remotely updating a device where the device fetches the firmware image from some file server. The manifest itself is delivered independently and provides information about the firmware image(s) to download.

```
                   /--------\                     /--------\
                  /          \                   /          \
                  | Manifest |                   | Manifest |
                  \          /                   \          /
                   \--------/                     \--------/
                               +-----------+
 +--------+                    |           |                +--------+
 |        |<...................| Status    |...............>|        |
 | Device |                    | Tracker   |             -- | Author |
 |        |<-                  |           |          ---   |        |
 +--------+  --                +-----------+       ---      +--------+
               --                                ---
                 ---                          ---
                    --      +-----------+   --
                      --    |           |  --
      /------------\    --  | Firmware  |<-    /------------\
     /              \    -- | Server    |     /              \
     |   Firmware   |       |           |     |   Firmware   |
     \              /       +-----------+     \              /
      \------------/                           \------------/
```

Figure 4: Independent retrieval of the firmware image.

This architecture does not mandate a specific delivery mode but a solution must support both types.

6. Manifest

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

skipping to change at *page 20, line 9* skipping to change at *page 21, line 9*

measurements).

While the software architecture of the bootloader and its security mechanisms are implementation-specific, the manifest can be used to control the firmware download from the Internet in addition to augmenting secure boot process. These building blocks are highly relevant for the design of the manifest.

9. Example

The following example message flow illustrates a possible interaction for distributing a firmware image to a device starting with an author uploading the new firmware to firmware server and creating a manifest. The firmware and manifest are stored on the same firmware server.

measurements).

While the software architecture of the bootloader and its security mechanisms are implementation-specific, the manifest can be used to control the firmware download from the Internet in addition to augmenting secure boot process. These building blocks are highly relevant for the design of the manifest.

9. Example

Figure 5 illustrates an example message flow for distributing a firmware image to a device starting with an author uploading the new firmware to firmware server and creating a manifest. The firmware and manifest are stored on the same firmware server. This setup does not use a status tracker and the firmware consumer component is therefore responsible for periodically checking whether a new firmware image is available for download.

```
+--------+    +-----------------+      +------------+ +----------+
| Author |    | Firmware Server |      |FW Consumer | |Bootloader|
+--------+    +-----------------+      +------------+ +----------+
  |                   |                     |                +
  | Create Firmware   |                     |                |
  |--------------+    |                     |                |
  |              |    |                     |                |
  |<-------------+    |                     |                |
  |                   |                     |                |
  | Upload Firmware   |                     |                |
  |------------------>|                     |                |
  |                   |                     |                |
  | Create Manifest   |                     |                |
  |---------------+   |                     |                |
  |               |   |                     |                |
  |<--------------+   |                     |                |
  |                   |                     |                |
  | Sign Manifest     |                     |                |
  |-------------+     |                     |                |
  |             |     |                     |                |
  |<------------+     |                     |                |
  |                   |                     |                |
  | Upload Manifest   |                     |                |
  |------------------>|                     |                |
  |                   |                     |                |
  |                   |   Query Manifest    |                |
  |                   |<--------------------|                |
  |                   |                     |                |
  |                   |   Send Manifest     |                |
  |                   |-------------------->|                |
  |                   |                     | Validate       |
```

skipping to change at *page 21, line 10*

```
  |                   |                     |         |      |
  |                   |                     |<--------+      |
  |                   |                     |                |
  |                   |  Request Firmware   |                |
  |                   |<--------------------|                |
  |                   |                     |                |
  |                   | Send Firmware       |                |
  |                   |-------------------->|                |
  |                   |                     | Verify         |
  |                   |                     | Firmware       |
  |                   |                     |--------------+ |
  |                   |                     |              | |
  |                   |                     |<-------------+ |
  |                   |                     |                |
  |                   |                     | Store          |
  |                   |                     | Firmware       |
  |                   |                     |-------------+  |
  |                   |                     |             |  |
  |                   |                     |<------------+  |
  |                   |                     |                |
  |                   |                     | Reboot         |
  |                   |                     |--------------->|
  |                   |                     |                |
  |                   |                     | Verify         |
  |                   |                     | Firmware       |
  |                   |                     | --------------|
  |                   |                     | |              |
  |                   |                     | -------------->|
  |                   |                     |                |
  |                   |                     | Activate new   |
  |                   |                     | Firmware       |
  |                   |                     | --------------|
  |                   |                     | |              |
  |                   |                     | -------------->|
  |                   |                     |                |
  |                   |                     | Boot new       |
  |                   |                     | Firmware       |
  |                   |                     | --------------|
  |                   |                     | |              |
  |                   |                     | -------------->|
  |                   |                     |                |
```

Figure 5: Example Flow for a Firmware Upate.

```
+--------+    +-----------------+      +------------+ +----------+
| Author |    | Firmware Server |      |FW Consumer | |Bootloader|
+--------+    +-----------------+      +------------+ +----------+
  |                   |                     |                +
  | Create Firmware   |                     |                |
  |--------------+    |                     |                |
  |              |    |                     |                |
  |<-------------+    |                     |                |
  |                   |                     |                |
  | Upload Firmware   |                     |                |
  |------------------>|                     |                |
  |                   |                     |                |
  | Create Manifest   |                     |                |
  |---------------+   |                     |                |
  |               |   |                     |                |
  |<--------------+   |                     |                |
  |                   |                     |                |
  | Sign Manifest     |                     |                |
  |-------------+     |                     |                |
  |             |     |                     |                |
  |<------------+     |                     |                |
  |                   |                     |                |
  | Upload Manifest   |                     |                |
  |------------------>|                     |                |
  |                   |                     |                |
  |                   |   Query Manifest    |                |
  |                   |<--------------------|                |
  |                   |                     |                |
  |                   |   Send Manifest     |                |
  |                   |-------------------->|                |
  |                   |                     | Validate       |
```

skipping to change at *page 22, line 12*

```
  |                   |                     |         |      |
  |                   |                     |<--------+      |
  |                   |                     |                |
  |                   |  Request Firmware   |                |
  |                   |<--------------------|                |
  |                   |                     |                |
  |                   | Send Firmware       |                |
  |                   |-------------------->|                |
  |                   |                     | Verify         |
  |                   |                     | Firmware       |
  |                   |                     |--------------+ |
  |                   |                     |              | |
  |                   |                     |<-------------+ |
  |                   |                     |                |
  |                   |                     | Store          |
  |                   |                     | Firmware       |
  |                   |                     |-------------+  |
  |                   |                     |             |  |
  |                   |                     |<------------+  |
  |                   |                     |                |
  |                   |                     | Trigger Reboot |
  |                   |                     |--------------->|
  |                   |                     |                |
  |                   |                     |                |
  |                   |                 +---+----------------+--+
  |                   |                S|   |                |  |
  |                   |                E|   | Verify         |  |
  |                   |                C|   | Firmware       |  |
  |                   |                U|   | +--------------|  |
  |                   |                R|   | |              |  |
  |                   |                E|   | +------------->|  |
  |                   |                 |   |                |  |
  |                   |                B|   | Activate new   |  |
  |                   |                O|   | Firmware       |  |
  |                   |                O|   | +--------------|  |
  |                   |                T|   | |              |  |
  |                   |                 |   | +------------->|  |
  |                   |                P|   |                |  |
  |                   |                R|   | Boot new       |  |
  |                   |                O|   | Firmware       |  |
  |                   |                C|   | +--------------|  |
  |                   |                E|   | |              |  |
  |                   |                S|   | +------------->|  |
  |                   |                S|   |                |  |
  |                   |                 +---+----------------+--+
  |                   |                     |                |
```

Figure 5: First Example Flow for a Firmware Upate.

Figure 6 shows an example follow with the device using a status tracker. For editorial reasons the author publishing the manifest at the status tracker and the firmware image at the firmware server is not shown. Also omitted is the secure boot process following the successful firmware update process.

The exchange starts with the device interacting with the status tracker; the details of such exchange will vary with the different device management systems being used. In any case, the status tracker learns about the firmware version of the devices it manages. In our example, the device under management is using firmware version A.B.C. At a later point in time the author uploads a new firmware along with the manifest to the firmware server and the status tracker, respectively. While there is no need to store the manifest and the firmware on different servers this example shows a common pattern used in the industry. The status tracker may then automatically, based on human intervention or based on a more complex policy decide to inform the device about the newly available firmware image. In our example, it does so by pushing the manifest to the FW consumer. The firmware consumer downloads the firmware image with the newer version X.Y.Z after successful validation of the manifest. Subsequently, a reboot is initiated and the secure boot process starts.

```
 +---------+   +-----------------+    |-----------------------------.
 | Status  |   | Firmware Server |    | +------------+ +----------+ |
 | Tracker |   |                 |    | |FW Consumer | |Bootloader| |
 +---------+   +-----------------+    | +------------+ +----------+ |
      |                 |             |      |  IoT Device    |     |
      |                 |              `''''''''''''''''''''''''''''
      |                 |                    |                |
      |         Query Firmware Version       |                |
      |------------------------------------->|                |
      |         Firmware Version A.B.C       |                |
      |<-------------------------------------|                |
      |                 |                    |                |
      |          <<some time later>>         |                |
      |                 |                    |                |
    _,...._          _,...._                 |                |
```

10. IANA Considerations

This document does not require any actions by IANA.

11. 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

skipping to change at *page 22, line 50*

protecting the manifest.

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

12. Mailing List Information

The discussion list for this document is located at the e-mail address suit@ietf.org [1]. Information on the group and information on how to subscribe to the list is at https://www1.ietf.org/mailman/listinfo/suit [2]

Archives of the list can be found at: https://www.ietf.org/mail-archive/web/suit/current/index.html [3]

13. Acknowledgements

We would like to thank the following persons for their feedback:

- Geraint Luff
- Amyas Phillips
- Dan Ros

skipping to change at *page 24, line 4*

- Markus Gueller
- Henk Birkholz
- Jintao Zhu
- Takeshi Takahashi
- Jacob Beningo

We would also like to thank the WG chairs, Russ Housley, David Waltermire, Dave Thaler for their support and their reviews.

14. References

14.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

14.2. Informative References

[I-D.ietf-suit-information-model] Moran, B., Tschofenig, H., and H. Birkholz, "Firmware Updates for Internet of Things Devices - An Information Model for Manifests", draft-ietf-suit-information-model-02 (work in progress), January 2019.

[LwM2M] OMA, ., "Lightweight Machine to Machine Technical Specification, Version 1.0.2", February 2018, <http://www.openmobilealliance.org/release/LightweightM2M/V1_0_2-20180209-A/OMA-TS-LightweightM2M-V1_0_2-20180209-A.pdf>.

```
  ,'      `.  ,'      `.                    |                |
 |   New    | |   New    |                  |                |
  \ Manifest /  \ Firmware /                |                |
   `'-_ _,,'     `'-_ _,,'                  |                |
              Push manifest                 |                |
       |----------------+------------------>|                |
       |                |                   |                |
       |                |                   | Validate       |
       |                |                   | Manifest       |
       |                |                   |---------+      |
       |                |                   |         |      |
       |                |                   |<--------+      |
       |                | Request firmware  |                |
       |                | X.Y.Z             |                |
       |                |<------------------|                |
       |                |                   |                |
       |                | Firmware X.Y.Z    |                |
       |                |------------------>|                |
       |                |                   |                |
       |                |                   | Verify         |
       |                |                   | Firmware       |
       |                |                   |--------------+ |
       |                |                   |              | |
       |                |                   |<-------------+ |
       |                |                   |                |
       |                |                   | Store          |
       |                |                   | Firmware       |
       |                |                   |-------------+  |
       |                |                   |             |  |
       |                |                   |<------------+  |
       |                |                   |                |
       |                |                   | Trigger Reboot |
       |                |                   |--------------->|
       |                |                   |                |
       |                |                   | __..-------..._'
       |                |                  ,-'               `-.
       |                |                 |    Secure Boot     |
       |                |                  `-.               _/
       |                |                   |`--..._____,,.,-'
       |                |                   |                |
```

Figure 6: Second Example Flow for a Firmware Upate.

10. IANA Considerations

This document does not require any actions by IANA.

11. 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

skipping to change at *page 25, line 43*

protecting the manifest.

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

12. Mailing List Information

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

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

13. Acknowledgements

We would like to thank the following persons for their feedback:

- Geraint Luff
- Amyas Phillips
- Dan Ros

skipping to change at *page 26, line 48*

- Markus Gueller
- Henk Birkholz
- Jintao Zhu
- Takeshi Takahashi
- Jacob Beningo

We would also like to thank the WG chairs, Russ Housley, David Waltermire, Dave Thaler for their support and their reviews.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

14.2. Informative References

[I-D.ietf-suit-information-model] Moran, B., Tschofenig, H., and H. Birkholz, "Firmware Updates for Internet of Things Devices - An Information Model for Manifests", draft-ietf-suit-information-model-03 (work in progress), July 2019.

[I-D.ietf-teep-architecture] Pei, M., Tschofenig, H., Wheeler, D., Atyeo, A., and D. Liu, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-03 (work in progress), July 2019.

[LwM2M] OMA, ., "Lightweight Machine to Machine Technical Specification, Version 1.0.2", February 2018, <http://www.openmobilealliance.org/release/LightweightM2M/V1_0_2-20180209-A/OMA-TS-LightweightM2M-V1_0_2-20180209-A.pdf>.

```
   [RFC5649]  Housley, R. and M. Dworkin, "Advanced Encryption Standard
              (AES) Key Wrap with Padding Algorithm", RFC 5649,
              DOI 10.17487/RFC5649, September 2009,
              <https://www.rfc-editor.org/info/rfc5649>.

   [RFC6024]  Reddy, R. and C. Wallace, "Trust Anchor Management
              Requirements", RFC 6024, DOI 10.17487/RFC6024, October
              2010, <https://www.rfc-editor.org/info/rfc6024>.

   [RFC8240]  Tschofenig, H. and S. Farrell, "Report from the Internet
              of Things Software Update (IoTSU) Workshop 2016",
              RFC 8240, DOI 10.17487/RFC8240, September 2017,
              <https://www.rfc-editor.org/info/rfc8240>.

14.3.  URIs

   [1] mailto:suit@ietf.org

   [2] https://www1.ietf.org/mailman/listinfo/suit

   [3] https://www.ietf.org/mail-archive/web/suit/current/index.html
```



```
Authors' Addresses

   Brendan Moran
   Arm Limited

   EMail: Brendan.Moran@arm.com

   Milosch Meriac
   Consultant
```



```
   [RFC5649]  Housley, R. and M. Dworkin, "Advanced Encryption Standard
              (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI
              10.17487/RFC5649, September 2009, <https://www.rfc-
              editor.org/info/rfc5649>.

   [RFC6024]  Reddy, R. and C. Wallace, "Trust Anchor Management
              Requirements", RFC 6024, DOI 10.17487/RFC6024, October
              2010, <https://www.rfc-editor.org/info/rfc6024>.

   [RFC8240]  Tschofenig, H. and S. Farrell, "Report from the Internet
              of Things Software Update (IoTSU) Workshop 2016", RFC
              8240, DOI 10.17487/RFC8240, September 2017,
              <https://www.rfc-editor.org/info/rfc8240>.

14.3.  URIs

   [1] mailto:suit@ietf.org
```



```
Authors' Addresses

   Brendan Moran
   Arm Limited

   EMail: Brendan.Moran@arm.com

   Milosch Meriac
   Consultant
```


End of changes. 44 change blocks.

130 lines changed or deleted	*310 lines changed or added*

This html diff was produced by rfcdiff 1.47. The latest version is available from http://tools.ietf.org/tools/rfcdiff/