SUIT
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

J. Zhu
Huawei

March 5, 2018

A Secure and Automatic Firmware Update Architecture for IoT Devices
draft-zhu-suit-automatic-fu-arch-00

Abstract

Firmware update is one of the key features for all Internet of Things(IoT) Devices. Incentives of firmware update is to fix bugs, upgrade configurations, add new services from constained devices to connected vehicles. The IETF SUIT working group is focusing on defining a firmware update solution that will be usable on Class 1 devices as well as more capable devices with existing mechanisms. This document provides an firmware update architechture, that aims at helping build a secure, automatic and interoperable IoT firmware update mechanism.

Table of Contents

1. Introduction 2
2. Terminology 3
3. Architecture 3
3.1. Client-Initiated Update 4
3.2. Server-Initiated Update 5
3.3. Negotiated Update 5
4. Requirements 6
5. Conclusion 6
6. Security Considerations 6
7. IANA Considerations 6
8. Acknowledgements 6
9. References 7
9.1. Normative References 7
9.2. Informative References 7
Author's Address 7

1. Introduction

Firmware update is one of the key features for all Internet of Things(IoT) Devices. Incentives of firmware update is to fix bugs, upgrade configurations, add new services from constained devices to connected vehicles. The IETF SUIT working group is focusing on defining a firmware update solution that will be usable on Class 1 devices as well as more capable devices with existing mechanisms. This document provides an firmware update architechture, that aims at helping build a secure, automatic and interoperable IoT firmware update mechanism.

Considering the security aspects of IoT device firmware update, it is not only the cryptographic mechanisms that used to protect the firmware images and the manifest, or the secure transmission protocols, but also the timeliness. An old-fashioned firmware with vulnerabilities is very harmful for an IoT device, which leaves potential target for a cyber attack.

The 2016 Dyn cyberattack is the most impressive example that why all IoT devices SHOULD implement the firmware update. In October 2016, the Dyn cyberattack took place and this huge DDoS attack affected a large number of the US and EU websites. The direct reason of this attack is that many IoT devices such as printers, IP cameras, residential gateways and baby monitors are infected by Mirai malware,

and the attacker used a list of default username/password to access and control the devices. However, one of the actual reasons is that those infected IoT devices are not updated in time. The new version firmware has a mandatory feature to ask the users to change the default username/password pair when they log in.

In order to provide a firmware update in time, the mechanism SHOULD be automatic. The rest of this document describes an automatic solution for the IoT device firmware update. Section 3 provides the architecture. Section 4 provides some of the requirements.

2. Terminology

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

3. Architecture

In the existing IoT firmware update standards such as OMA DM FUMO and LwM2M Firmware Update, the devices usually play the core role during the update procedure. They download the firmware images from a storage location under the guidance of the device management(DM) server, and then setup a status machine and start processing the firmware when they become "Idle".

However, in most IoT use cases, the "Idle" status itself is really confused and hard to reach. For example, a smart electricity meter works 7*24 after it is plugged into the grid. It always runs under "Working" status, then how to execute a firmware update on this kind of devices when needed? Another example is how to update the ECUs in a connected vehicle. The server can only access the ECUs when the vehicle is started, but the ECUs also start to work. Even if the in-car gateway can download the firmware image, it is hardly for a car to update the firmware immediately. When the car gets into "Idle" status, it is usually shut down as well, no ECUs can do further update process.

Moreover, the DM server usually manages multiple different devices at a large amount. The existing standard does not have the scheduling of manage and update firmwares. The DM server MAY store multiple different firmware images at the same time. The firmware images have different functionalities, thus they SHOULD have different urgency levels when updating. For example, a bug fix update firmware has a higher urgency level than a configuration update firmware.

Therefore, there is a need for the secure automatic firmware update mechanism to address the issues mention above. A brief message flow of secure automatic firmware update is shown in Figure 1. There are generally 3 steps. Most of the cryptographic related steps are shown in Figure 4/[I-D.draft-moran-suit-architecture] and not described again in this document.

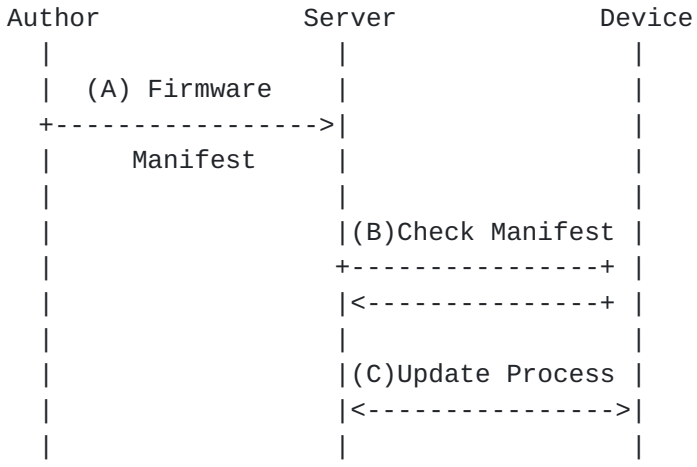

Figure 1: Brief message flow of Firmware Update

(A) The server gets the firmware image and related manifest from the firmware author.

(B) The server checks the manifest to know the firmware image urgency class and determine the update mode of the targeted device.

(C) The device downloads the firmware image and manifest, and then start to process the firmware update.

For a secure automatic firmware update solution, there may be three different update mode to be considered.

1. Client-Initiated Update

2. Server-Initiated Update

3. Negotiated Update

3.1. Client-Initiated Update

The Client-Initiated Update is a mechanism that the client update itself unilaterally.

In this case, the client MAY retrieve the latest firmware image information periodically from the server. Once there is a new

firmware image on the server, the client itself started to download the firmware image and manifest. After successful downloading, the client start to process the firmware update when it goes into idle status.

3.2. Server-Initiated Update

The Server-Initiated Update is a mechanism that the server update the device unilaterally.

This mechanism is suggested especially for the 7*24 working device since it has no idle status. Then even if it has downloaded the firmware images from the server. It is hard for the device itself to determine when to process a firmware update.

The traditional way of updating this kind of device is to ask human intervetion. This is not a good idea for large amount of devices and also for the automatic solutions. This has already been proved in the Mirai Cyberattack example.

A better solution for this case is to let the server take charge of the update process. The server determines when to update the devices according to the service status retrieved from the device or a pre-provisioned update strategy. The server pushes the firmware image when the device is accessible and trigger the update process based on the previous determinations.

The device in this Solution is considered as an executor and SHOULD follow the commands from server strictly.

3.3. Negotiated Update

The Negotiated Update is the most common use mechanism for existing firmware update standards for IoT devices. E.g.OMA DM and LwM2M .

The server notifies the client once it gets a firmware image from the author. This notification MAY only contain the existance information of a new firmware image. It may also contain the firmware manifest.

The client then decided if it is OK with this update automatically. If the status is idle the client may generates a OK response to the server to ack the notification. Then the server SHOULD start pushing the firmware image to the client; It the status is not good enough for update, the client then generates a response with a scheduled update time. The server then waited and trigger the update after the scheduled update time expires.

4. Requirements

The firmware update mechanism SHOULD be automatic in order to enhance the availability.

The firmware update mechanism SHOULD consider not only single image update, but also multiple images update use cases.

The firmware manifest SHOULD contain enough information for the server to determine the urgency class of the related firmware image.

The server SHOULD be pre-provisioned some strategies to determine the urgency class of received firmware image.

If the denial of service is unavoidable during the firmware update process, this situation SHOULD be notified to the device.

The update mode for each device SHOULD be determined during the registration process.

5. Conclusion

Timeliness is a key consideration when discussing and defining the IoT firmware update mechanism. Unlike the traditional cryptographic considerations that focus more on the confidentiality and integrity, this automatic firmware update solution focuses more on the availability aspect of security when updating an IoT device.

The automatic firmware update solution provided in this document makes the IoT device update in time to shorten the possible vulnerability exposure time for the attackers. This solution also consider the impact of the update process itself SHOULD not cause a denial of service for the device for the minimum period of time.

6. Security Considerations

The whole document can be seen as security considerations for IoT device firmware update.

7. IANA Considerations

TBD.

8. Acknowledgements

TBD.

9. References

9.1. 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>.

9.2. Informative References

[I-D.draft-moran-suit-architecture] Moran, B., Meriac, M., and H. Tschofenig, "A Firmware Update Architecture for Internet of Things Devices", draft-moran-suit-architecture-02 (work in progress), March 2018.

Author's Address

Jintao Zhu
Huawei
P.R.China

Email: jintao.zhu@huawei.com