Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

S. Nandakumar
C. Jennings
S. Cooley
Cisco

October 30, 2017

Solution Architecture - Secure Firmware Upgrade (SecFU)
draft-nandakumar-suit-secfu-solution-arch-00

Abstract

This specification defines a solution architecture for performing secure firmware upgrade for Internet of Things (IoT). The ulterior motive is to have a framework that is simple, secure, and that uses most common formats and standards in the industry and that works over Internet.

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 May 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

Internet of Things (IOT) represents a plethora of devices that come in varying flavors of constrained sizes, computing power, and operating considerations. These devices usually need minimal or no management for their operation.

Vulnerabilities within IOT devices have raised serious concerns. There needs to be a way to install or update the firmware on these devices in an automated and secure fashion. A common challenge with the existing firmware update mechanism is they do not work in an automated manner in many environments where IoT devices are deployed. Hence, there is a need to define a firmware update solution that is light weight, secure, can operate in variety of deployment environments, and is built on well established standards.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Device Considerations

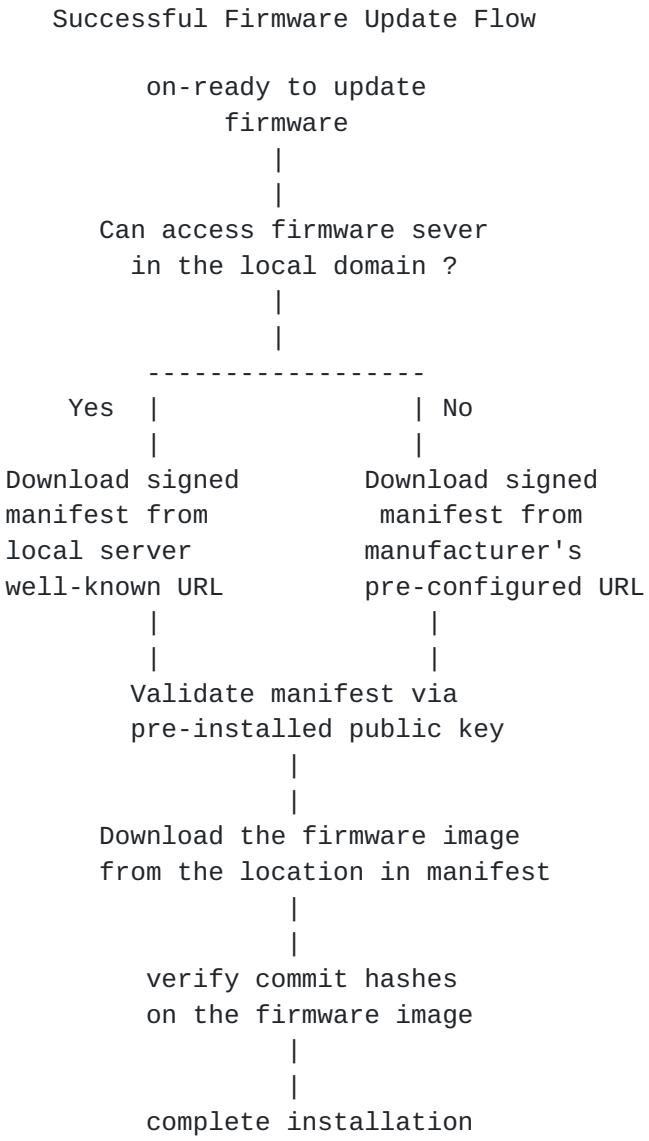
This draft targets devices that have a boot loader that run in less than 100K bytes of flash and less than 32K bytes of RAM.

There are certain types of devices that delete the firmware image, except for the boot-loader, before proceeding with the upgrade. Alternatively, many devices have sufficient storage to completely download a new firmware image before updating. This solution should be naturally applicable to both.

4. Solution Overview

draft-nandakumar-suit-secfu-requirements captures various requirements that drives the solution defined in this specification.

Below is a high-level solution flow for a successful firmware update on a IoT device.



5. Solution Components

Following several sub-sections define various components that makes up the proposed solution architecture

5.1. Manifest

A firmware manifest serves as information representation for metadata about the firmware. A manifest file identifies information about the actual firmware image, its location, applicable device(s), and so on. It is cryptographically signed by the provider (usually the manufacturer) of the firmware.

Minimal Manifest in JSON format

```
{
  "manifestVersion" : "1.0",
  "timestamp": "2017-12-10T15:12:15Z",
  "manufacturer": "manufacturer.com",
  "model": "c7960",
  "firmwareVersion": "10.4.12",
  "firmwareLocation": "well-known location",
  "firmwareCryptoInfo": {
    {
      "commitHash": [
        {
          "digestAlgo": "sha256",
          "hash": "..................."
        },
        {
          "digestAlgo": "sha512",
          "hash": "..................."
        }
      ]
    }
  }
  "key-info": <most recent public key info>
}
```

Above shows example of a minimally defined manifest that identifies the mandatory attributes as explained below

- manifestVersion: Version of the manifest
- timestamp: Time when the manifest was created.
- manufacturer: An identifier of the manufacturer providing the firmware image, represented as String
- model: Device Model, a String
- firmwareVersion: Firmware Version in the format "major.minor.revision"
- firmwareLocation: Location of the firmware images. This can be an absolute URI or a relative URI that is relative to where the manifest was downloaded from.
- firmwareCryptoInfo: Commit Hash information

5.2. Manifest Format

JSON representation is recommended as the default format for describing the manifest. Optionally, formats such as CBOR (see example section) can be used for the same. If more than one format is used, the IoT device can pick one based on its implementation. The firmware download protocol identifies the right format supported by the IoT device.

5.3. Manifest Security

The Manifest file MUST be cryptographically signed by the private key of the manufacturer or the provider of the firmware. This is to ensure source authenticity and to protect integrity of the manifest and the firmware itself.

JWS is the format recommended to store the signed manifest.

```
signed_manifest := JWS(manifest.json)
```

If CBOR is used for describing the manifest, COSE is recommended for signing.

Optionally, the proposed solution also recommends hash based signatures (hash-sigs) to sign the manifest.

```
signed_manifest := hash-sigs(manifest.json, private-key)
```

5.4. Manifest Optional Extensions

There may be scenarios where the minimalistic manifest defined above may not capture all the requirements for a given deployment setting. In those circumstances, the manifest can be optionally extended to meet the requirements in a extensional specifications.

5.5. Firmware Server Discovery

When it is time for an IoT device to perform a firmware upgrade, the device performs couple of steps to decide the location to download the needed firmware. A device might need to download the new firmware when it is either booting for the first time after deployment or there is a need to upgrade to a newer firmware.

The server discovery procedure starts with the boot-loader attempting to access a server that is local to the domain in which the device operates. The URL to look for a local server is automatically generated using the DHCP domain name.

For example, if the domain name was example.com, and the device was a Cisco 7960, the HTTP URL might be

http://_firmware.example.com/.wellknown/firmware/cisco.com/c7960/manifest.json

In situations where the IoT device cannot access the Internet (factory/enterprise settings, for example), the aforementioned approach might be the only way for the device to perform any kind of firmware or security updates.

However, if the local server cannot be reached or not deployed (say home environments), the device proceeds to download the manifest and firmware from the firmware server URL pre-configured in the boot-loader by the manufacturer of the device. For example

http://_firmware.cisco.com/.wellknown/firmware/cisco.com/c7960/manifest.json

If either of the procedures doesn't work, the IoT device is either unusable or might end up running an old version of the firmware.

5.6. Firmware Download protocol

One can envision two possibilities while downloading the firmware:

- Scenarios where the IoT device downloads firmware directly. This is done in order to minimize number of connections. In this scenario, the firmware image must have a digital signature included within the downloaded firmware. The exact placement of this digital signature (prepended, appended, etc) is up to the device manufacturer, but it MUST provided source and integrity guarantees on the entirety of the firmware image and must be verified by the device prior to upgrade.

- Scenarios where a manifest is retrieved and followed by downloading the actual firmware image.

5.6.1. Validation Procedures

The downloaded manifest and firmware is validated before being used:

- Manifest file signature is validated for source and integrity verification. If encrypted, the manifest is decrypted before proceeding with the firmware download.

- On successful validation of the manifest, the device verifies the commit hashes for component(s) of the firmware downloaded against the ones provided in the "firmwareCryptoInfo" section of the manifest.

5.6.2. Manifest Download

Firmware download protocol enables choosing the approach appropriate to the IoT device for downloading the manifest file.

For example, on performing the "Firmware Server Discovery", if a local server is chosen, the device forms a query URL by constructing an endpoint at ".well-known/manifest/<manufacturer>/<model-no>/manifest.json"

Then a HTTP GET request is sent to that URL. For example

http://_firmware.example.com/.wellknown/manifest/cisco.com/c7960/manifest.json

The response would be a JSON result of the manifest file. Similarly, the end-point supporting CBOR parsing can request for the CBOR version of the mannifest.

5.6.3. Firmware Download

Once the manifest is downloaded and validated, the device proceeds to download the firmware image from the location identified in the firmware manifest. There might be situations where a firmware image is split into multiple files to imply a functional division of the components. This type of firmware can be used by devices that are memory constrained and thus loading the complete image might not be possible. The manifest file may contain the information to indicate the same.

Above example shows use of HTTP as the communication protocol to talk to the firmware server. If the end-point is capable of doing COAP or other protocols, a similar process as above can be applied to retrieve the manifest and the firmware from a well-known place on the local server.

6. IANA Consideration

TODO

7. Security Considerations

TODO - Talk about roaming IoT Device

8. Acknowledgements

9. 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>.

Authors' Addresses

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com

Cullen Jennings
Cisco

Email: fluffy@iii.ca

Shaun Cooley
Cisco

Email: scooley@cisco.com