

# Recommendations for IoT Firmware Update Processes

PREPARED BY THE CLOUD SECURITY ALLIANCE  
INTERNET OF THINGS (IOT) WORKING GROUP



© 2018 Cloud Security Alliance – *All Rights Reserved*

All rights reserved. You may download, store, display on your computer, view, print, and link to the Cloud Security Alliance at <https://cloudsecurityalliance.org> subject to the following: (a) the draft may be used solely for your personal, informational, non-commercial use; (b) the draft may not be modified or altered in any way; (c) the draft may not be redistributed; and (d) the trademark, copyright or other notices may not be removed. You may quote portions of the draft as permitted by the Fair Use provisions of the United States Copyright Act, provided that you attribute the portions to the Cloud Security Alliance.

# ACKNOWLEDGEMENTS

## Co-Chair

Brian Russell

## Lead Author

Sabri Khemissa

## Contributors

Aaron Guzman

Paul Lanois

Ashish Mehta

Todd Nelson

Michael Roza

Brian Russell

Carlos Samaniego

## CSA Research

Hillary Baron

Stephan Lumpe (Design)

## Special Thanks

Daniel Hiestand

# TABLE OF CONTENTS

## Acknowledgements

## Table of Contents

## Establishing a secure, scalable firmware update process for Internet of Things (IoT) devices

1. Backup the current working configuration of IoT device before applying an update.
2. Rollbacks should be supported; however, older images should not be reloaded without vendor authorization.
3. System design should allow administrators to schedule updates to their devices to avoid network saturation and limit unintended downtime.
4. Vendors should support configuration options by system administrators in support of automatic updates.
5. One component must manage updates of multiple microcontrollers that compose IoT devices.
6. Update strategy, differential or complete image should be adapted to the bandwidth constraint.
7. Updates should be authenticated and integrity protected from end-to-end.
8. Verification signing keys storage must be secured.
9. Provide a recovery procedure to cover update failure.
10. Ensure a long-term support contract by vendors.

## Conclusion

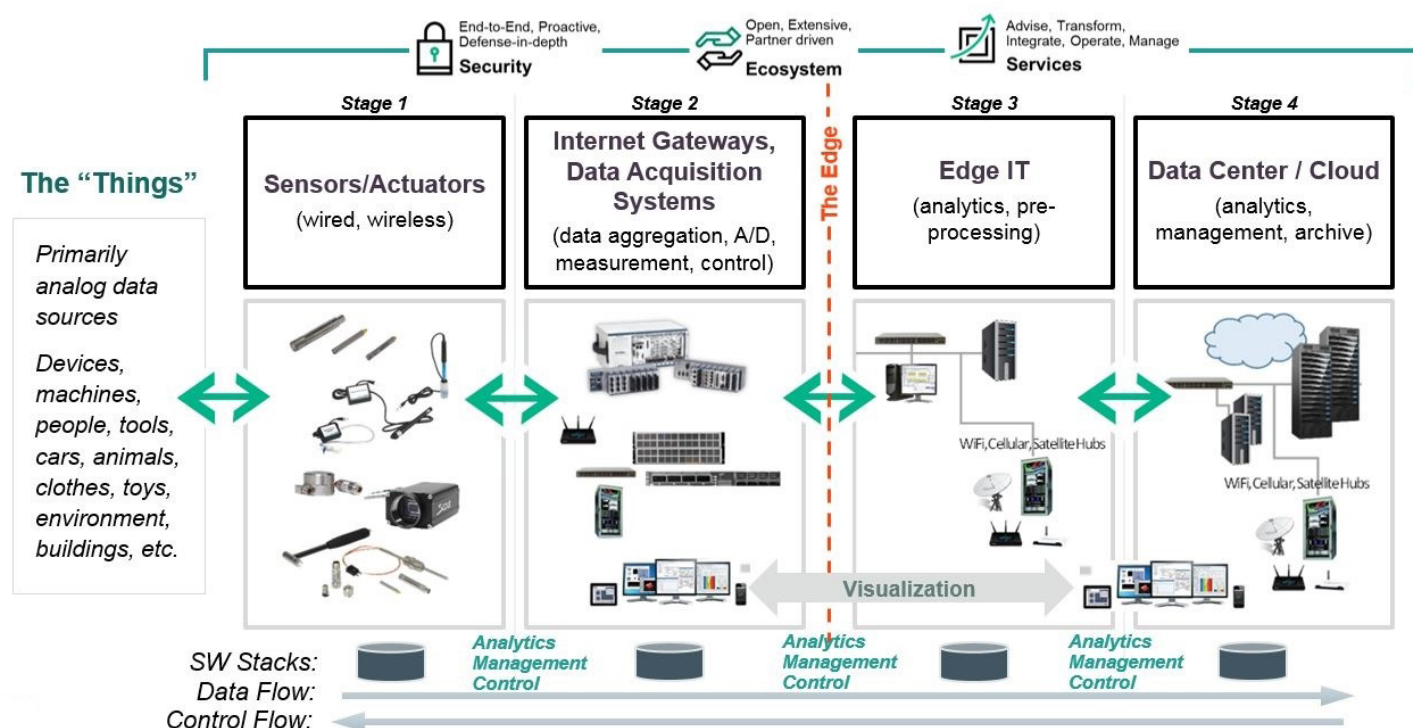
## Other Readings

# ESTABLISHING A SECURE, SCALABLE FIRMWARE UPDATE PROCESS FOR INTERNET OF THINGS (IOT) DEVICES

The traditional approach to updating software for IT assets involves analysis, staging and distribution of the update—a process that usually occurs during off-hours for the business. These updates typically have cryptographic controls (digital signatures) applied to safeguard the integrity and authenticity of the software. However, the Internet of Things (IoT)—with its vast ecosystem of connected devices deployed in many environments—introduces complexities associated with the update process that drives the need for process re-engineering. To answer that call, the *Cloud Security Alliance IoT Working Group* has compiled key recommendations for establishing a secure and scalable IoT update process. These recommendations are the focus of this document, and are directed at the following audiences:

1. **Developers:** Developers must consider how their IoT can be updated securely and acknowledge that their IoT is integrating into a complex system—and coexisting with other products.
2. **Implementers:** Implementers must consider the whole and complex system, including the specific constraints of each IoT component that comprises the system<sup>1</sup>.

## The 4 Stage IoT Solutions Architecture



<sup>1</sup> <https://techbeacon.com/4-stages-iot-architecture>

There are many variations in the IoT systems that require software and firmware updates. For example, some IoT systems are often on the move and require relatively large downloads—such as connected vehicles (CV), or the potential influx of autonomous vehicles. Due to CV mobility (particularly during a download period), processes related to vehicle-to-everything (V2X) communications should consider how to better orchestrate handing off downloads from one or more road side units (RSUs) to another.

Other IoT systems are more static. Smart home and smart building devices, for example, may be attached to a single access point. However, factors associated with network saturation during downloads to hundreds or thousands of devices may need to be considered.

Attention must also be directed to the impact of failed firmware updates on consumers. In 2017, there were two incidents that demonstrate what can occur when an IoT firmware update fails for consumers:

1. A smart lock developer transmitted a firmware update that automatically upgraded firmware across its product line. Unfortunately, a particular firmware update was corrupt and took the updated smart locks offline. These mechanisms were the preferred locks for Airbnb and, as a result of the mishap, many renters were unable to enter their rental properties. Because the locks were brought offline by the corrupt update, they had to be manually updated by the vendor<sup>2</sup>.
2. A firmware update was recommended by the United States Food and Drug Administration (FDA) to mitigate a remote access vulnerability in Abbott pacemakers. The recommended firmware update was expected. However, given that the device was used for medical means, any modification posed a degree of risk. In the case of Abbott, the FDA noted in their advisory there was a minor risk the firmware update itself could cause device failure. Ultimately, the decision to update the device to mitigate vulnerabilities was weighed against the possibility that the update itself could cause the device to fail. Failure in this scenario would have required surgery to replace the device<sup>3</sup>. Additionally, there have been reports of issues related to the heating system<sup>4</sup> upgrades introduced by Tesla's over-the-air 2016 software update<sup>4</sup>.

This document provides guidelines that developers and implementers can fully or partially integrate. Suggestions can be adapted and designed for custom firmware update processes that recognize unique constraints, dependencies, and risks associated with products and complex systems.

## **1 Backup the current working configuration of IoT device before applying an update.**

If an issue occurs during the update process, the IoT device should be able to reload the last known and trusted configuration. This configuration could be stored on an upstream repository hosted by a private or public cloud platform.

The new image must be able to load the previous configuration, even if this configuration makes new enhancements unavailable. This feature ensures the IoT device is kept up, running and connected in order to make the download of a (new) updated image possible.

---

<sup>2</sup> <https://techcrunch.com/2017/08/14/wifi-disabled/>

<sup>3</sup> <https://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm573669.htm>

<sup>4</sup> <https://medium.com/@Tesla.ShortCircuit/tesla-firmware-heating-problem-over-the-air-ota-software-update-failed-2719fa7758bb>

*The partition must be protected to avoid tampering an old configuration (e.g., changing the back-end server destination to force sending data to a rogue server).*

Today, there are many options that support the backup of the current working configuration of your device. For example, Amazon Web Services (AWS) IoT service supports Thing Shadows, while the Microsoft Azure IoT Hub supports device twins. These services store the latest device configurations in a JavaScript Object Notation (JSON) file in the cloud, which is able to synchronize with the device upon reconnection. Product designers are well advised to integrate these or custom services into their product offerings.

2

## **Rollbacks should be supported; however, older images should not be reloaded without vendor authorization.**

An attacker can launch a downgrade assault by reloading an old, unsecure version of firmware to exploit a known vulnerability. An anti-rollback feature must be implemented to prevent an older image from being loaded on a system. For example, all older firmware deemed out-of-date or insecure should require a password provided by the vendor.

### **Best practices:**

- Rollbacks must be integrated in the testbed of the firmware development process.
- Basic IoT properties—such as hostname, networking configuration and basic features configuration—should be stored in a specific file that can be read by any version of the firmware. This practice ensures that any rollbacks keep the IoT device connected and running. An alternative to consider: erase setup and configuration files of the current firmware version and run a new setup process for the IoT device with the old firmware loaded.

3



## **System design should allow administrators to schedule updates to their devices to avoid network saturation and limit unintended downtime.**

Considering that IoT devices currently number in the billions, a smart strategy must be put in place to avoid:



1. Network congestion due to an increase in transmissions between IoT devices and update servers during the deployment of a new firmware.
2. Overstressing update servers, which can cause denial-of-service (DoS) issues.
3. IoT devices resources saturation, due to continuous update requests from IoT devices to update servers during network congestion, and/or update server unavailability.

The two main IoT update models are:

**Push mode:** Update servers send or “push” the new version of the firmware to IoT devices.

PROS 	CONS 
<ul style="list-style-type: none"><li>• Efficient in the case of small-sized updates.</li><li>• All connected IoT devices could be updated in a controlled timeframe.</li><li>• Rule-driven push mode enables multiple-use cases, such as:<ul style="list-style-type: none"><li>- Auto push of firmware updates upon attacks (if any) on a device model.</li><li>- Auto rollback of a previous version upon firmware issues.</li></ul></li></ul>	<ul style="list-style-type: none"><li>• IoT devices must be registered into update servers and uniquely identified.</li><li>• Push needs to be carefully scheduled so that there is less load on the network/update server.</li></ul>

**Pull mode:** Periodically, IoT devices “pull” update servers to get a new version of the firmware.

PROS 	CONS 
<ul style="list-style-type: none"><li>• Enterprises (industrial IOT) or users (consumer IOT) can plan downtime to enable firmware upgrades.</li></ul>	<ul style="list-style-type: none"><li>• The pull is not conducted in “real time.” This could be an issue for deploying emergency updates because IoT device are vulnerable between pulls—especially if the interval of time between upgrades is lengthy (e.g., 24 hours).</li><li>• If the scale of pull is significant, the mode could consume a lot of resources on the update server.</li></ul>

*To avoid designing different communication streams that may use different channels and/or technical components, both “push” or “pull” update strategies must be consistent with data collection strategy (DCS). The adoption of the same strategy for both cases to optimize IoT resource usage and bandwidth allocation is strongly advised.*

### **Best practices:**

- An update must be gradually deployed. First, plan to update a small number of IoT devices for testing purposes, then gradually deploy the new firmware at a larger scale.
- The number of unsuccessful requests to/from IoT devices must be limited. Time shifting must also be implemented (e.g., after five unsuccessful requests, wait 24 hours before starting the next update process).



- Provide rollback options for any firmware updates.
- Adopt a progressive deployment approach for the following three cases:
  - Minor update: fixing bugs.
  - Major update: new release of firmware.
  - Emergency update: fixing bugs or security issues.
- Consider criticality-of-use issues when scheduling specific updates (e.g., there should be no updates when a connected car is driven or when medical devices are used).

4

## Vendors should support configuration options by system administrators in support of automatic updates.

The automatic update process includes two actions: downloading a new update and then installing that update. Additionally, two categories of automatic updates must be distinguished: updates for personal IoT devices and managed IoT deployment.

1. In the case of personal IoT devices, updates are downloaded directly from IoT vendor update servers. Automatic updates are enabled in three ways: by default, upon device setup, or by opting-in via a configuration.
2. In the case of managed IoT deployment, IoT devices are centrally overseen by an organization, private or a public company (such as enterprises or smart cities), and updates are downloaded from organization update servers after testing and approval. Automatic updates should be enabled by default in this case.

### **Best practices:**

- Disabling update or user notifications for a new update must be avoided. However, if updates or user notifications are disabled for any reason, they must be limited in a timely fashion (e.g., suspend update checks for 30 days).
- Allow regression validation for a set of devices in the production environment before rolling out a massive batch (Progressive deployment).

5

## One component must manage updates of multiple microcontrollers that compose IoT devices.

An IoT device may have one or more microcontrollers and sensors embedded, and these components may be provided by different manufacturers. These manufacturers continuously provide updates for improving their products, solving bugs and correcting security issues.

IoT device vendors must manage the updates of all components of their devices—even those components provided by third parties. To that end, the vendor should validate and provide the firmware of all components that constitute their products. This will help address two key issues:

1. The prevention of side effects impacting other components that disrupt the operation of the IoT device.
2. The installation of a tampered/malicious firmware that compromises the component, and then the whole IoT device.

**Best practices:**

- A direct update to the firmware of a microcontroller or sensor (which bypasses the full update package of the vendor) must be rejected by the IoT device.

## 6 Update strategy, differential or complete image should be adapted to the bandwidth constraint.

In a low-bandwidth deployment, a full-image download could impact the network traffic that blocks the monitoring and data collection of the IoT device. Whenever possible, package and differential updates are preferred over complete image updates

**Best practices:**

- The firmware image/update must be designed to be deployed over-the-air (e.g., Bluetooth Low Energy, long-range wireless, etc.), in addition to low- and high-speed networks, universal serial bus (USB) or serial interfaces.

## 7 Updates should be authenticated and integrity protected from end-to-end.

The main challenge of a secure update process is ensuring that an update has not been tampered with in-transit to the IoT device and that the update originates from the expected update servers. This process is verified by data origin authentication (DOA). Each firmware update must be digitally signed by the vendor, and the digital signature must be verified by the IoT device before starting the update. Complete, hardware cryptography-enabled microcontrollers should be provided to address security challenges for the IoT. These solutions are more and more being optimized and integrated within main cloud solutions providers.

To aid in the process, the firmware's digital signature could be stored in a blockchain as showcased in CSA's report, "Using Blockchain Technology to Secure the Internet of Things."

*Hostname and internet protocol (IP) addresses must be used to authenticate an IoT device by update servers. This information can be easily spoofed to connect IoT devices to "fake" update servers that may deliver malicious firmware.*

**Best practices:**

- Vendors must better secure their update servers to avoid replacing valid firmware with malicious firmware.
- Vendors must also secure supply chain software development. This will ensure malicious code isn't integrated during the development lifecycle while preventing the exposure of private signing keys.

## 8 Verification signing keys storage must be secured.

Signing keys—and any secret keys—used by an IoT device must be stored in a secured manner. The secure element could be a specialized, dedicated and embedded component of the IoT device that offers secure storage for secret keys.

Advanced secure elements offer cryptographic operation because secret keys are never separated from the element. Cryptographic operations—such as signature verification—are processed inside the secure element, and only the results of cryptographic operations are exposed to other components and applications.

### **Best practices:**

- Vendors should consider implementing secure elements, such as a Trusted Platform Module (TPM).
- Vendors should consider new approaches, such as those offered through physically unclonable functions<sup>5</sup> (PUFs). The PUFs generate private keys on-demand based on “digital fingerprints” from IoT device components, such as static random-access memory (SRAM) and microprocessors. The private key is never stored in the IoT.

## 9 Provide a recovery procedure to cover update failure.

If the automatic update fails and the IoT device network is offline, a manual recovery procedure must be provided.

### **Best practices:**

The recovery procedure must:

- Allow for (re)loading of only the last working firmware version.
- Avoid firmware downgrades after applying security fixes and/or major firmware version upgrades.

## 10 Ensure a long-term support contract by vendors.

A long-term service agreement (spanning at least 10 years) must be implemented between the IoT device vendors—or the provider executing the support solution—and microcontroller manufacturers, including original design manufacturer (ODM) and/or original equipment manufacturer (OEM)

### **Best practices:**

The service agreement shall provide the following items (at a minimum):

- Advanced notice of support termination.
- Clear instructions related to product trade-ins, product replacement and/or product disposal.
- Explicit statement of responsibilities and liabilities for all parties.
- Ongoing product usage and end-of-life reporting.

---

<sup>5</sup> [https://en.wikipedia.org/wiki/Physical\\_unclonable\\_function](https://en.wikipedia.org/wiki/Physical_unclonable_function)

# CONCLUSION:

The 10 recommendations outlined in this document were highlighted to help organizations successfully implement a safe and secure IoT firmware update process. These recommendations target developers and implementers, but also vendors who must carefully design their solutions with security in mind (including the update process). By addressing this process, an attack vector that a hacker can exploit is mitigated.

# OTHER READINGS:

Moran, B., Meriac, M., Tschofenig, H. (2017, October 30) *A Firmware Update Architecture for Internet of Things Devices (Draft)*. Internet Engineering Task Force. Available @ <https://tools.ietf.org/html/draft-moran-suit-architecture-00>

Fujitsu Laboratories Ltd. (2018, March 7) *Fujitsu Develops Data Processing Architecture 'Dracena,' Can Reconfigure Content within IoT Data Processing Stream* Available @ <http://www.fujitsu.com/global/about/resources/news/press-releases/2018/0307-02.html>

Interagency International Cybersecurity Standardization Working Group (2018, February) *Interagency Report on Status of International Cybersecurity Standardization for the Internet of Things (IoT) (Draft NISTIR 8200)*. National Institute of Standards and Technology. (Note: of particular interest is Chapter 8 Standards Landscape for IoT Cybersecurity.) Available @ <https://csrc.nist.gov/CSRC/media/Publications/nistir/8200/draft/documents/nistir8200-draft.pdf>

European Union Agency for Network and Information Security (2017, November 20) *Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures* (Note: of particular interest is Chapter 4 Technical Standards, 4.3.6 Secure Software / Firmware updates page 50.) Available @ <https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot>

International Organization for Standardization, ISO/IEC CD 30141 *Internet of Things Reference Architecture (IoT RA)* (Under Development) Available @ <https://www.iso.org/standard/65695.html>

United States Food and Drug Administration, *Postmarket Management of Cybersecurity in Medical Devices—Guidance for Industry and Food and Drug Administration Staff* (2016, December 28) (Note: of particular interest are pages 9, 10, 13, 18, 20, 30) Available @ <https://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm482022.pdf>

United States Food and Drug Administration, *Design Considerations and Premarket Submission Recommendations for Interoperable Medical Devices—Guidance for Industry and Food and Drug Administration Staff* (2017, Sept. 6) (Note: of particular interest are pages 6, 17) Available @ <https://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM482649.pdf>

United States Food and Drug Administration, *The FDA'S Role in Medical Device Cybersecurity FDA Fact Sheet—Dispelling Myths and Understanding Facts* Available @ <https://www.fda.gov/downloads/MedicalDevices/DigitalHealth/UCM544684.pdf>