Note: This document is still in draft form, as part of the ongoing work from the Technical Capabilities Working Group, part of NTIA's Multistakeholder Process on IoT Security Updatability

Securing the IoT Update Process

Contents

Voluntary Framework for Enhancing Update Process Security

The proliferation of devices and growth in the Internet of Things (IoT) provides the opportunity for technical advances that could dramatically improve people's lives. As devices become increasingly integrated into society, security and safety risks to individuals, businesses, and society may increase without appropriate, risk-based security measures. One important tool to help mitigate risks is the ability to reliably update software within devices over-the-air.

The level of update process security appropriate for a particular device will vary depending upon the manufacturer's unique business needs, resource availability, and risk tolerance. Executing a basic, over-the-air update can be done without including any enhanced update process security features, however such an update may be vulnerable to blocking, spoofing, or other malicious attack. Absent any security precautions, updates can, in fact, dramatically reduce the security of a device. Decisions concerning whether to employ additional features to enhance the security of the update process should be risk-based, achieving security goals in a cost-effective and prioritized manner.

The term "updatable" does not mean the same thing to everyone. Different people and different organizations may have their own ideas about what the term does (and should) mean. To better address update process security risks, it is important to have a common understanding of updatability to support manufacturers, purchasers, and other IoT stakeholders as they make risk-based decisions to enhance update process security.

This document is intended to provide IoT device and software manufacturers with a common lexicon or language to discuss risk mitigation in the over-the-air update process. A clear framework for updatability, with defined steps, will allow discerning risk-aware decision makers to understand the value of particular security features. Manufacturers[1] of devices and the components that go into devices have many reasons to be interested in their devices' update process. Manufacturers need a way to provide a secure updating mechanism for updatable devices and may want to convey this information to their customers as a competitive advantage of their products and services. Consumers may also find this information useful; it can complement the document by the Working Group on Communicating Upgradability. Further value to the consumer results from a shared model of update processes across the diverse IoT product space. Even if the devices are quite different, the steps of the update and the potential security features may be similar.

Manufacturers can ensure that the devices they design, produce, and sell perform properly and address security risks. To achieve these goals, manufacturers need detailed hardware and software design criteria that they can integrate throughout their product development, production, sales, and support processes.

Accordingly, this document is designed to support manufacturers in identifying and selecting appropriate, risk-based security features to mitigate vulnerabilities in the update process. Part I of this

[1] The term "Manufacturer" used throughout this document is understood to also represent an "assigned agent"; "service provider"; or "vendor" as authorized and enabled by the Manufacturer.

document provides an overview of basic steps in an illustrative update process. Part II provides a menu of voluntary processes that manufacturers may choose to adopt to enhance the level of security in the update process depending upon individual business needs and risk tolerance.

Of note, update process security is only one aspect of many in enhancing the overall security of an IoT device. Devices may have vulnerabilities never addressed by an update. There are also physical and human security vulnerabilities that may not be addressable by software updates. This larger challenge of mitigating IoT device vulnerabilities can only be addressed through a set of coordinated actions, including industry best practices, device management and cyber-hygiene solutions, and greater awareness of context-specific risks. Adoption of the highest level of over-the-air update process security enhancements does not guarantee the security of the device itself. This voluntary guidance, therefore, is intended only for the limited purpose of developing a common lexicon to support manufacturers in enhancing security in effecting over-the-air updates.

Part I: Basic Steps in an Illustrative Over-the-Air Update Process

The update process is broken up into a linear sequence of **steps** that broadly describes the sequence to be followed.

For each step, there are **security features** which can vary greatly depending upon the intended final security posture desired by the manufacturer.

While the sequence of steps is common to all risk models, the security features implemented in each step ultimately determines the level of security of the update process.

The following normative sequence of update steps are considered the basic elements in the update process.

0. Create: Manufacturer creates image
    a. This step is assumed to be out of scope for this guideline, but is represented here as it is seminal to the initialization of this process.
    b. The update image or multiple update images are packaged into a deliverable structure.[2]
1. Sign: Ensure integrity of update
    a. Manufacturer includes a signature or signatures in the update deliverable, to be used to vet the integrity of the update deliverable contents.
2. Protect: Prevent exposure of update deliverable
    a. Manufacturer subjects the update deliverable to a translation (including encryption or obfuscation) to prevent exposure of software image
3. Send: Data in motion

[2] This step still has security concerns. Recent attacks on the update process have demonstrated the importance of the integrity and security of the update process. One stakeholder recommended the trust in the update process not rest on a single key or servers, and that at least one key required for an update to be trusted should be kept on a non-Internet connected device.

    a. The update deliverable is communicated to the target system / device.
4. Receive: Receive update deliverable
    a. The target system / device receives update deliverable
5. Check: Process update deliverable
    a. Target system / device validates integrity of encrypted update deliverable
    b. Target system / device decrypts update deliverable
    c. Target system performs any special handling of update images as indicated
6. Announce: User awareness of update on device [3]
    a. End user notification and/or approval of update installation
7. Distribute: Distribution
    a. Update deliverable is parsed and distributed to intended target devices
    b. Distribution may be of a recursive nature.
8. Process: Process update image
    a. Each target (CPU, MCU, FPGA, etc.) receives its update image
    b. Each target decrypts the update image
    c. Each target validates the integrity of the plain text update image
9. Stage: System pre-update state
    a. Any activities that need to be performed before the update occurs
10. Apply: Trigger update process
    a. Perform the actual update process of installing the update image
11. Re-verify: Post-update verification
    a. Each target validates the integrity of the installed update
    b. Communicate results of verification to relevant targets
12. Activate: Activate / enable updated code
    a. New updated code actually begins to be executed on the target (assuming successful verification)
13. Clean-up: Post-update activities
    a. Verify that system is functioning appropriately
    b. Post-processing messaging (internal & external) and cleanup from update
    c. This could include a negative outcome.

[3] If the update is automatic, without requiring user involvement, then this step may happen later, if just to allow the user to understand that the current SW/FW is up-to-date.

Part II: Security Features to Enhance Over-the-Air Update Process Security

The steps listed above are necessary to ensure the integrity and the reliability of an update and the update process. However, without the addition of security specific features at each step, the steps themselves are vulnerable to attack by malicious actors and may in fact make the target device more vulnerable during the upgrade process than it was before.[4] The security needs of each step vary based on context, threat, etc.

Below, we present a framework to understand security features that could be implemented at each step to improve the security of an update process. The features themselves map to the steps of an update. Because the security decisions should be based on needs, context, technical capabilities, and risk evaluation, the security features are presented as a 'menu,' from basic security features, all the way to features designed to resist quantum computing-assisted attacks in line with current state-of-the-art encryption guidance.

[4] Such attacks include (but are not limited to): Man in the Middle, Spoofing, 'Bricking' the device, Denial of Service (blocking the upgrade), Version Downgrade, and Key theft.

Basic Update Process Security Features

1. Sign: Ensure integrity of update
    a. A 128 bit cryptographic hash signature
2. Protect: Prevent exposure of update deliverable
    a. Ephemeral and unique cryptographic keys are created / exchanged and stored in the system/device
    b. AES-128 encryption
3. Send: Data in motion
    a. No special processing is assumed
4. Receive: Receive update deliverable
    a. No special processing is assumed
5. Check: Process update deliverable
    a. A 128 bit cryptographic hash signature
    b. Unique encryption key for each system / device
    c. AES-128 decryption
6. Announce: User awareness of update on device
    a. Optional end user approval of update, as indicated in the deliverable
7. Distribute: Distribution to devices
    a. No presumption as to the depth of layers of system / devices / CPUs to be targeted.
8. Process: Process update image
    a. Each target decrypts their specific update image using AES-128 cryptography
    b. Each target vets the integrity of the plain text update image using a 128 bit cryptographic hash signature
9. Stage: System pre-update state
    a. None assumed; Manufacturer defined
10. Apply: Trigger update process
    a. No special processing is assumed
11. Re-verify: Post-update verification
    a. Each target vets the integrity of the installed update using a 128 bit cryptographic hash signature
12. Activate: Activate / enable updated code
    a. No special processing is assumed
13. Clean-up: Post-update activities
    a. No special processing is assumed

Upgraded Update Process Security Features (+1)

The following assumes all cryptographic operations have a minimum key width of 256 bits. Refer to NIST SP 800-57 Part 1 Revision 4 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf) for appropriate guidance on key lengths and mechanisms.

1. Sign: Ensure integrity of update
    a. NIST Key management for hashing (See NIST SP 800-147 and SP 800-57 for handling)
2. Protect: Prevent exposure of update deliverable
    a. NIST Key management for decryption (See NIST SP 800-147 and SP 800-57 for handling)
3. Send: Data in motion
    a. Validation of the Endpoint by cryptographically confirming that the end system/device is the correct target before delivering the update deliverable.
    b. Challenge/response mechanisms
4. Receive: Receive update deliverable
    a. The communication medium utilized to deliver the upgrade deliverable needs to be secured to current best practices level.
    b. TLS v1.2
    c. Pinning certificates
5. Check: Process update deliverable
6. Announce: User awareness of update on device
    a. No special processing, beyond basic level is assumed
7. Distribute: Distribution to devices
    a. Update image remains encrypted while in motion if traveling across exposed buses.
    b. Support for multiple of layers of system / devices / CPUs to be targeted.
8. Process: Process update image
    a. Each target decrypts the update image using AES-256 cryptography**
    b. Each target validates the integrity of the plain text update image using a 256 bit cryptographic hash signature**
9. Stage: System pre-update state
    a. No special processing, beyond minimum level is assumed
10. Apply: Trigger update process
    a. Coordination between updates for synchronized updated is supported
    b. Coordination with end user supported
    c. Persistent data conversion on each target is supported
11. Re-verify: Post-update verification
    a. Each target validates the integrity of the installed update using at a minimum a CRC-16 and a cryptographically significant hash, with a width of no less than 256 bits
12. Activate: Activate / enable updated code
    a. No special processing, beyond minimum level is assumed
13. Clean-up: Post-update activities
    a. Local to the target system notification of successful update by each target
    b. External to the target system, communications to external database of successful upgrade to system, including identification and versioning information (i.e. “non-repudiation”)

Enhanced Update Process Security Features (+2)

This enhanced level of security features addresses foreseeable exploits. A hardware root of trust may be used as a backstop for all activities, including a signed chain of software execution, starting with a hardware secure boot operation.

1. Sign: Ensure integrity of update
    a. NIST Key management for hashing (See NIST SP 800-147 and SP 800-57 for handling)
    b. Secure memory in which to perform hashing
    c. PKI key derivation
2. Protect: Prevent exposure of update deliverable
    a. NIST Key management for decryption (See NIST SP 800-147 and SP 800-57 for handling)
    b. Secure memory in which to decrypt & store update
    c. PKI key derivation
3. Send: Data in motion
    a. Validation of the Endpoint by cryptographically confirming that the end system/device is the correct target before delivering the update deliverable.
    b. Challenge/response mechanisms
    c. PKI.
4. Receive: Receive update deliverable
    a. The communication medium utilized to deliver the upgrade deliverable needs to be secured to current best practices level.
    b. TLS v1.3 or greater
    c. Pinning certificates
5. Check: Process update deliverable
6. Announce: User awareness of update on device
    a. No special processing, beyond minimum level is assumed
7. Distribute: Distribution to devices
    a. Update image remains encrypted while in motion**
    b. Support for multiple of layers of system / devices / CPUs to be targeted.
8. Process: Process update image
    a. Each target decrypts their specific update image using AES-256 cryptography**
    b. Each target validates the integrity of the plain text update image using a 256 bit cryptographic hash signature**
9. Stage: System pre-update state
    a. No special processing, beyond minimum level is assumed
10. Apply: Trigger update process
    a. Coordination between updates for synchronized updated is supported
    b. Coordination with end user supported
    c. Persistent data conversion on each target is supported
11. Re-verify: Post-update verification
    a. Each target validates the integrity of the installed update using at a minimum a CRC-16 and a cryptographically significant hash, with a width of no less than 256 bits**
12. Activate: Activate / enable updated code
    a. No special processing, beyond minimum level is assumed
13. Clean-up: Post-update activities
    a. Local to the target system notification of successful update by each target
    b. External to the target system, communications to external database of successful

upgrade to system, including identification and versioning information (i.e. “non-repudiation”)

Quantum Resistant Update Process Security Features (+3)

In the future, computationally intensive cryptographic operations may become insecure through the use of quantum computers.[5] Additional precautions may be taken beyond any of the previous detailed security features. The following list assumes these are features being added to a system which has already implemented the Enhanced level of security.

These include the avoidance of the following encryption standards:

- ❑ RSA
- ❑ Elliptic Curve Cryptography (“ECC”)
- ❑ ElGamal
- ❑ DSA
- ❑ Diffie-Hellman key exchange

1. Sign: Ensure integrity of update
    a. Secure memory in which to perform hashing
    b. Cryptographic hash shall be SHA3-256 or Lamport Signature 256 bits
2. Protect: Prevent exposure of update deliverable
    a. Encrypted with “Learning with Errors” or “Ring Learning with Errors”
3. Send: Data in motion
    a. Validation of the Endpoint by cryptographically confirming that the end system/device is the correct target before delivering the update deliverable.
4. Receive: Receive update deliverable
    a. The communication medium utilized to deliver the upgrade deliverable needs to be secured to current best practices level.
    b. TLS v1.3 or greater
    c. Pinning certificates
5. Check: Process update deliverable
6. Announce: User awareness of update on device
    a. No special processing, beyond minimum level is assumed
7. Distribute: Distribution to devices
    a. Update image remains encrypted while in motion**
    b. Support for multiple of layers of system / devices / CPUs to be targeted.
8. Process: Process update image
    a. Each target decrypts their specific update image
    b. Each target validates the integrity of the plain text update image using a SHA3-256 or Lamport Signature 256 bits

[5] This broader threat has been documented in several other places, including NIST-IR 8105 “Report on Post-Quantum Cryptography.” (2016) Available at: http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

9. Stage: System pre-update state
    a. No special processing, beyond minimum level is assumed
10. Apply: Trigger update process
    a. Coordination between updates for synchronized updated is supported
    b. Coordination with end user supported
    c. Persistent data conversion on each target is supported
11. Re-verify: Post-update verification
    a. Each target validates the integrity of the installed update using at a minimum a CRC-16 and a (SHA3-256 or Lamport Signature 256 bits)
12. Activate: Activate / enable updated code
    a. No special processing, beyond minimum level is assumed
13. Clean-up: Post-update activities
    a. Local to the target system notification of successful update by each target
    b. External to the target system, communications to external database of successful upgrade to system, including identification and versioning information (i.e. “non-repudiation”

Robustness Enhancing Update Process Features

These improve the overall quality and robustness of the product and the update process, but are not directly related to the security of the process. Of course, robustness and resilience are strongly related to security themselves, but are more ‘second order’ effects.

- System documents persistent, non-volatile data
- Potential fall back to the last known good state
- Post-update testing - and activated, the functionality of the device may be verified. This includes the essential functions of the device and any optional features of the device.
- Update persistent data
- Post-processing messaging and verification (internal, component-to-component)
- Storage of state beyond that which is to be updated.

Step		Description	Basic	+1 ("Upgraded")	+2 ("Enhanced")	+3 ("Quantum")
0	***Create***	*Update creation is important, but not in scope for this guideline. There are still security considerations inherent in this step.*				
1	**Sign**	Update signed.	128-bit hash.	NIST key management for hashing.	Secure memory, PKI.	SHA3-256 or Lamport.
2	**Protect**	Encryption and/or obfuscation	Ephemeral, unique AES-128 keys in device.	NIST key management for decryption.	Secure memory, PKI.	LWE or RLWE key exchange
3	**Send**	Communicated to target device.	*No special assumption.*	Endpoint verification	PKI.	← *See* ***Enhanced****.*
4	**Receive**	Target device receives update.	*No special assumption.*	Best practices (e.g. TLSv1.2, cert. pinning)	TLSv1.3.	
5	**Check**	Target validates, decrypts, and processes as needed.	Validation, decryption w/per-device keys.	← *See* ***Basic****.*		
6	**Announce**	End-user notified about / approves update install.	Optional end-user approval			
7	**Distribute**	Image parsed, distributed to HW targets (e.g. CPU, FPGA).	*No special assumption.*	Encrypted in motion; can target multiple layers.	← *See* ***Upgraded****.*	
8	**Process**	Hardware target receives, validates, and decrypts image.	Target hardware validates, decrypts.	Target validation, decryption w/ AES-256.	Target and image validation w/ AES-256.	SHA3-256 or Lamport.
9	**Stage**	System-specific pre update tasks.	Manufacturer defined.	← *See* ***Basic****.*		
10	**Apply**	Image install process runs.	*No special assumption.*	Opt. update and end-user coordination, data persistence.	← *See* ***Upgraded****.*	
11	**Re-verify**	Install integrity check; optional communication of results.	Install results validated.	Validated with hash, checksum.	Minimum CRC-16 checksum and AES-256 hash.	SHA3-256 or Lamport hash and checksum.
12	**Activate**	New code executes if verified.	*No special assumption.*	← *See* ***Basic****.*		
13	**Clean-up**	System-specific: verification, messaging, and clean-up.	*No special assumption.*	Local notification and external logging of update.	← *See* ***Upgraded****.*	

Table 1: Summary of Security Features and options for progressive enhancement.