Voluntary Framework for Enhancing Update Process Security

Technical Capabilities and Patching Expectations Working Group
Working Group Membership: …
Draft: September 12, 2017

Motivation

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 remotely, through a network (often referred to as "over-the-air", or OTA).

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.

Document Overview

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 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.

Audience

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. Further value 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. Note that the information in this document is not explicitly targeted to end-consumers. Manufacturers may wish to consult "Communicating IoT Device Security Update Capability to Improve Transparency for Consumers" by the Working Group on Communicating Upgradability for guidance on how to communicate updatability to end-consumers. This document also does not address security risks from legacy devices or orphan devices that are no longer maintained.

The information in this document may also be useful to enterprise-level procurement processes. Having knowledge of the importance of security updates and the risks associated with certain technologies can better guide decision-making.

[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.

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
    1. 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.
    2. The update image or multiple update images are packaged into a deliverable structure.[2]
1. Sign: Ensure integrity of update
    1. 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
    1. Manufacturer subjects the update deliverable to a translation (including encryption or obfuscation) to prevent exposure of software image
3. Send: Data in motion
    1. The update deliverable is communicated to the target system / device.[3]
4. Receive: Receive update deliverable
    1. The target system / device receives update deliverable
5. Check: Process update deliverable
    1. Target system / device validates integrity of (potentially encrypted) update deliverable
    2. Target system / device decrypts update deliverable (if encrypted)
    3. Target system performs any special handling of update images as indicated
6. Announce: User awareness of update on device [4]

[2] This step still has important 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.
[3] Steps should be taken to ensure that the appropriate code is sent to the appropriate device. This is outside the scope of this document.

    1. End user notification and/or approval of update installation
7. Distribute: Distribution
    1. Update deliverable is parsed and distributed to intended target devices
    2. Distribution may be of a recursive nature.
8. Process: Process update image
    1. Each target (CPU, MCU, FPGA, etc.) receives its update image
    2. Each target decrypts the update image (if encrypted)
    3. Each target validates the integrity of the plain text update image
9. Stage: System pre-update state
    1. Any activities that need to be performed before the update occurs
    2. Potentially initiate a backout state for recovery, in case the update fails
10. Apply: Trigger update process
    1. Perform the actual update process of installing the update image
11. Re-verify: Post-update verification
    1. Each target validates the integrity of the installed update
    2. Communicate results of verification to relevant targets
12. Activate: Activate / enable updated code
    1. New updated code actually begins to be executed on the target (assuming successful verification)
13. Clean-up: Post-update activities
    1. Verify that system is functioning appropriately
    2. Post-processing messaging (internal & external) and cleanup from update
    3. This could include a negative outcome.

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 with an update process than without one.[5] 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

[4] 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 software and firmware is up-to-date. Alternatively, it may occur earlier to give the user notice and choice about downloading the update code.
[5] Such attacks include (but are not limited to): Man in the Middle, Spoofing, 'Bricking' the device, Denial of Service (blocking the update), Version Downgrade, and Key theft./

features designed to resist quantum computing-assisted attacks in line with current state-of-the-art encryption guidance.

Sign: Ensure authenticity and integrity of update

Risks: If there is no authenticity and integrity check, there is no way of verifying what you got is what you were supposed to get, or that it originated from the expected source. Perhaps the largest risk of an update process is the potential for a third party to push malicious code onto the device. Changing the code on the device could lead to any number of deviations from the intended functionality including preventing expected operations, adding new, undesired functions, changing how data flows from the machine, or weaponizing the device to attack other targets.

Mitigations: Signing the update payload cryptographically protects the integrity of the payload, including from undetected intentional modification by a bad actor. It also provides authenticity in the provenance of the payload. This is different from a more traditional approach of using non-cryptographic hash such as a cyclic redundancy check (CRC) or a checksum. These non-cryptographic hashes can validate the integrity against naturally occurring corruption of the payload, but can be easily subverted by bad actors. Similarly, failure to use a strong enough cryptographic signature or hash function also fails to completely mitigate these risks. For older, weaker hash functions, an attacker with sufficient motivation and resources could generate a malicious update that generated the same hash as the legitimate update.

Basic Implementation: Cryptographic signatures are used to detect modification and establish provenance of the update payload. *Per NIST SP 800-131A, acceptable key lengths for signing are 2048-bit for RSA and 224 for ECDSA, and acceptable hash functions are the SHA-2 family.*[6] *Other, less computationally intensive, algorithms exist (e.g. hashing functions like MD5, SHA-1, etc.) but have been found to be susceptible to various forms of attack.* NIST SP 800-89 (*Recommendation for Obtaining Assurances for Digital Signature Applications*) provides recommendations on digital signature key management.

Protect: Prevent exposure of update deliverable

Risks: This step refers to improving the protection of an update's confidentiality or intellectual property by encrypting it, rather than sending it 'in the clear.' (Note that protecting confidentiality through encryption is not the same thing as protecting integrity through cryptographic hashes and signatures.) An update sent without encryption risks exposure of code contained in the device or update payload. This could allow an attacker to subvert other protections in the device through reverse engineering thus allowing all similar devices in the field to become exploited for botnets. In a business risk to the manufacturer, a competitor could extract valuable algorithms or techniques in the software.

[6] http://dx.doi.org/10.6028/NIST.SP.800-131Ar1

Mitigation: Encryption of the update before transmission and decryption of the update on the device can reduce the risk of exposure during transmission regardless of the communications path(s) of the update deliverable.

Basic Implementation: Application layer encryption is used to protect the confidentiality of the update from time/place of creation to time/place of use. [TODO: cite standards]

Further security concerns:
Once the update deliverable has been received by the device, device design mitigations should be considered to avoid exposing the decrypted update deliverable in the event of a physical attack on the device during the update process.

Optionally, additional encryption provided by the communication path(s) can also be implemented as a secondary level of encryption to further mitigate the risk of exposure while the encrypted update deliverable is being distributed to the device. Note that communications path encryption only prevents exposure over the communications path. Any intermediate holding locations may result in exposure of the information (See data in motion section).

Send: Data in motion

While an update payload is being transmitted to a device, the path of communications may take several radically different types of communications (i.e. ethernet; cellular baseband; Wi-Fi; Bluetooth; etc.), many with no inherent security or integrity checking of the update payload. Identifying communicating parties is often part of establishing a secure communication channel so that senders and receivers know who with whom they are communicating.

Risks: The risks at this step are captured in the two preceding steps: compromise of the integrity of the update and the confidentiality of the update. Transport layer authentication and encryption allow an extra layer of defense. [authentication]

Basic implementation: Transport-layer encryption, such as TLS or BLE 4.2+, can provide widely-accepted levels of security between the endpoints. Using features such as pinning of certificates in TLS or user-pairing of devices in BLE can authenticate endpoints. VPNs also offer confidentiality and integrity of data in motion.

Further security concerns: “Defense in Depth”, where multiple security mitigations are overlaid in a redundant manner may be desirable. Validation of the Endpoint by cryptographically confirming that the end system/device is the correct target before delivering the update deliverable such as through using Challenge/response mechanisms or pre-shared secrets.

Receive: Receive update deliverable

The device receives the update.

No design risks are specifically associated with the required step, however normal good security hygiene practices should be followed, such as mitigations against buffer overflow

Check: Process update deliverable

For this step, the device confirms that the update is valid. This step validates the security of the previous steps, which may include checking the identity of the device, and confirming that the update is appropriate for this device. The target system also performs any necessary special handling of update images. This is a critical step in the protection of the device itself, where ultimately each device needs to protect itself from a potentially malicious update payload and all previous mitigations are enforced during this step.

Risks: In addition to the concerns above about authenticity and integrity (signature) and confidentiality (encryption), there are a number of security and performance concerns. In a 'downgrade attack', an older authentic update payload is substituted by a bad actor to re-introduce known vulnerabilities in the target to be exploited in a secondary attack. The update itself could be mis-configured so as to harm the device or its functionality.

Basic implementation: In addition to the signature and encryption features above, a monotonic versioning system can prevent a downgrade attack.

Further concerns: A system capable of disallowing previous versions requires an additional step for a manufacturer-driven rollback update, and can make user-driven rollbacks more complex. Alternatively, the device can securely validate the path and source of the update to ensure that the older version is not coming from an untrustworthy source.

Announce: User awareness of update on device

The manufacturer may wish user engagement in the update process. An update process may temporarily impede the functionality of a device, or the user may have some other interest in approving an update beyond an automated process. In these instances, the user should be made aware of the presence of an update on the device to take further action.

Risks: Strictly speaking this is not a security risk, but a functional risk as the process of updating the device may interrupt intended functionality or timing of functionality resulting in a period of denial of service.

Risks of failing to properly address the concerns of this step include:

1. Non-performance of device intended functionality during an update process.
2. If the device is under the control or partial control of a bad actor (e.g botnet), they may wish to block a security update.
3. A legitimate user may wish to avoid an update that implements unwanted features.
4. The device may not be able to communicate to the appropriate user that an update is available, leading to out-of-date, vulnerable devices.

Basic Implementation: Manufacturer should consider the use and installation of a device to determine the optimal approach to automatic updates, user control, and uptime criticality.

Optional end user approval of update, as indicated in "Communicating IoT Device Security Update Capability to Improve Transparency for Consumers" by the Working Group on Communicating Upgradability.

Further security concerns: If the user does not take action to update the device, the manufacturer or device administrator may wish to take further actions. How to address a non-updated device is outside the scope of this document.

Distribute: Distribution to devices

This guideline allows for multiple update targets in a given device or system, as such the potential for a hierarchical relationship between update targets is supported.

Risks: The update payload may be decrypted by the initial target and then distributed to sub-systems without further integrity or content protection (see above). This could expose intellectual property, more easily enabling reverse engineering, or potentially allow the code to be modified as it moves through the systems.

Basic implementation: Update image remains encrypted while in motion if traveling across exposed transport media; Support for multiple of layers of system / devices / CPUs to be targeted.

Further security concerns: The adversary might still be able to try to compromise non-exposed internal communication channels. To address this residual risk, the update image should remain encrypted while in motion.

Process: Process update image

Similar to step #5 “Check”, however this may be performed on a target in a lower “child” relationship, if a hierarchical relationship between update targets is supported.

Basic Implementation: Each target validates the integrity of the plain text update image using a cryptographic hash signature. Each target decrypts their specific update image, if encrypted.

Stage: System pre-update state

During this step any needed activities necessary to performing the update on the device can occur, this is inclusive of functions such as erasing flash memory; placing the device in a ‘safe mode’ of operation; etc.

Risks: Decrypted update image is present in device’s memory for an extended period of time while waiting for this step to complete.

Basic Security Features: None assumed; Manufacturer defined.

[potentially MMU-protected memory regions, seperate memory space on a different bus]

Apply: Trigger update process

The actual update process occurs. This includes writing to a file structure; updating the binary program space in flash memory; etc.

Risks: Decrypted update image may be communicated via internal communication channels or busses and stand the risk of being intercepted and exposed

Basic Security Implementation: No special processing is assumed

Further security concerns: Coordination between updates for synchronized updated is supported; Coordination with end user supported; Persistent data conversion on each target is supported. Update is placed into a separate flash region from existing image for reliability purposes (in case of failed update).

Re-verify: Post-update verification

As the importance of utilizing the intended, correct update image is paramount to the update process, one last test of the update functionality is performed before this new update software is executed.

Risks: Something went wrong. (More robustness than security)

1. This redundant test of update integrity confirms that the process of writing the update image to the intended target was performed correctly and that no malicious actor or device / memory failure altered the intended update image.

Basic Security Implementation: Each target vets the integrity of the installed update.

Further security concerns: Potentially use cryptographic hashing: 128 bit, or higher.

Activate: Activate / enable updated code

No additional risks are incurred during this step.

Basic Security Implementation: No special processing is assumed

Further Security Concerns: If multiple flash images are stored (redundant duplicate copies, or previous and current), then activation may entail pointing to the new image for subsequent boot cycles.

Clean-up: Post-update activities

Following an update operation, one final step is performed to clean up any loose ends, such as buffered data or encryption keys and intermediate values in memory. Post processing logging and messaging may be performed to inform other device targets of the status of the update on this target as well as communicating back to an external server the status of the update process. This status should identify the target in question; the particular device in question; and the status may not be a successful outcome.

Risks:

1. Other targets may not receive the status update and not act in concert with all the other targets
2. The external server may not receive the status update and believe the target or device has not been updated or be unaware that the target or device hasn't been updated, if the update operation did not succeed.

Basic Security Features: No special processing is assumed
Upgraded Security Features: Local to the target system notification of successful update by each target; External to the target system, communications to external database of successful upgrade to system,

including identification and versioning information (i.e. “non-repudiation”)
Enhanced Security Features: n/a

Potential Appendix: Areas of concern that are common to multiple steps in the update system

Key Management

This requires Ephemeral and unique cryptographic keys are created / exchanged and stored in the system/device.

A determined attacker could still subvert this protection by accessing the keys stored on the device, and then decrypting the update. To mitigate this risk, proper key management techniques should be followed. (See NIST SP 800-147 and SP 800-57 for handling)

The attacker may try to read the keys from the device’s memory during the decryption phase. Against this level of attacker, the device would need secure memory in which to decrypt and store the update.

Also language about protected memory?