SCTE® · ISBE®
STANDARDS

Data Standards Subcommittee

SCTE STANDARD

SCTE 256 2019

IoT Security Considerations and Recommendations for Operators

NOTICE

Table of Contents

List of Tables

1. Introduction

1.1. Executive Summary

This document contains considerations and recommendations for operators with regards to the security aspects of deploying hardware/software/services in the IoT space. It is the result of the SCTE DSS IoT Working Group – Security Subgroup.

These are recommendations and should not be considered to be requirements. The final design is the responsibility of the implementer.

1.2. Scope

The following areas are covered within this document:

1. Cybersecurity Responsibilities
    a. Vendor
    b. Service provider
    c. Customer
2. Product selection (for Service Provider equipment)
    a. Hardware security recommendations
    b. Firmware security recommendations
    c. Suggested vendor responsibilities for the life of the product
        i. Vulnerability notification
        ii. Attack notification
        iii. Mitigation
        iv. Firmware updates for security vulnerabilities
    d. Branding
    e. Qualification
3. Retail Products [customer owned and managed (COAM)]
    a. Qualification process
4. Provisioning
    a. Security onboarding
    b. Secure delivery of provisioning data
    c. Secure provisioning database
    d. Confirmation of provisioning
    e. Secure confirmation management and storage
    f. Blacklisting
5. Maintenance
    a. Firmware upgrade
    b. Device and platform auditing
6. Diagnostics
    a. Logs
    b. Alerts
7. Attack management planning
    a. Vendor
    b. Service provider internal
    c. Customer notification

1.3. Benefits

This document will be used to provide the following benefits:

- Educate the service providers on the infrastructure required to support IoT security
- Develop a common baseline architecture for IoT security such that different operators can use the same or similar tools and platforms

1.4. Intended Audience

The intended audience are the engineering and security teams for service providers.

1.5. Areas for Further Investigation or to be Added in Future Versions

As technology in the IoT space evolves, this document should be reviewed and updated and/or revised where necessary per SCTE policy.

2. Normative References

The following documents contain provisions, which, through reference in this text, constitute provisions of this document. At the time of Subcommittee approval, the editions indicated were valid. All documents are subject to revision; and while parties to any agreement based on this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below, they are reminded that newer editions of those documents might not be compatible with the referenced version.

2.1. SCTE References

- No normative references are applicable.

2.2. Standards from Other Organizations

- No normative references are applicable.

2.3. Published Materials

- No normative references are applicable.

3. Informative References

The following documents might provide valuable information to the reader but are not required when complying with this document.

3.1. SCTE References

- No informative references are applicable.

3.2. Standards from Other Organizations

- OCF Core Framework (https://openconnectivity.org/specs/OCF_Core_Specification_v2.0.2.pdf)
- OCF Security (https://openconnectivity.org/specs/OCF_Security_Specification_v2.0.2.pdf)

- OCF Bridging (https://openconnectivity.org/specs/OCF_Bridging_Specification_v2.0.2.pdf)
- OCF Smart Home Device (https://openconnectivity.org/specs/OCF_Smart_Home_Device_Specification-v2.0.2.pdf)
- OCF Wi-Fi Easy Setup (https://openconnectivity.org/specs/OCF_Wi-Fi_Easy_Setup_Specification_v2.0.2.pdf)
- OCF Cloud (https://openconnectivity.org/specs/OCF_Cloud_Specification_v2.0.2.pdf)
- RDK Security Guidelines (https://wiki.rdkcentral.com/display/RDK/Security+Guidelines?preview=/8388899/8390789/Secure%20Coding%20for%20RDKV2.pdf)
- The Update Framework (https://theupdateframework.github.io)

3.3. Published Materials

- National Institute of Standards Security Framework https://www.nist.gov/cyberframework

4. Compliance Notation

shall	This word or the adjective "***required***" means that the item is an absolute requirement of this document.
shall not	This phrase means that the item is an absolute prohibition of this document.
forbidden	This word means the value specified shall never be used.
should	This word or the adjective "*recommended*" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighted before choosing a different course.
should not	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
may	This word or the adjective "*optional*" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.
deprecated	Use is permissible for legacy purposes only. Deprecated features may be removed from future versions of this document. Implementations should avoid use of deprecated features.

5. Abbreviations and Definitions

5.1. Abbreviations

Acronym	Definition
AAA	authentication, authorization, and administration
AC	alternating current
bps	bits per second
CA	certificate authority
COAM	customer owned and managed

Acronym	Definition
CPE	customer premise equipment
CPU	central processing unit
CRC	cyclic redundancy code
DC	direct current
DHCP	Dynamic Host Configuration Protocol
DNS	domain name system
DNS RPZ	domain name service response policy zones
DNSSEC	domain name system security extensions
DOCSIS	Data-Over-Cable Service Interface Specifications
DRAM	dynamic random access memory
DSM-CC	digital storage media – command & control
DSS	Data Standards Subcommittee (of the SCTE)
EEROM	electrically erasable read only memory
FIPS	Federal Information Processing Standard
FLUTE	file delivery over unidirectional transport
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HDD	hard disk drive
HFC	hybrid fiber/coax
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IoT	Internet of Things
IP	Internet Protocol
IV	initialization vector
JTAG	Joint Test Action Group (IEEE Standard 1149.1-1990)
LED	light emitting diode
MAC	media access control
OCF	open connectivity foundation
OS	operating system
PCB	printed circuit board
POC	point of contact
QA	quality assurance
RDK	reference design kit
ROM	read only memory
SCTE	Society of Cable Telecommunications Engineers
SD	secure digital
SNMP	Simple Network Management Protocol
SoC	system on chip
SRAM	static random access memory
SSH	secure shell
SSM	source-specific multicast
TFTP	trivial file transfer protocol
TR	technical report
UI	user interface
URL	uniform resource locator
USB	universal serial bus

5.2. Definitions

The following are the definitions used within this document.

Term	Definition
asymmetric encryption	Also called public-private key cryptography. Utilizes one key to encrypt and a different key to decrypt.
attack	Any cybersecurity method which is actively being used to gain access to or control of an IoT device or the data associated with an IoT device.
authentication	Methods used to ensure that data comes from a specific user. Assumes that a trust has been established between the service provider and the vendor.
bad actor	A cybersecurity adversary that is interested in attacking information technology systems.
brownout	A reduction in the electrical voltage, either intentional or unintentional, over any period of time from 1 msec to days. Brownouts can cause unintentional behavior in digital circuits.
cybersecurity	The technologies and processes designed to protect computers, networks and data from unauthorized access, vulnerabilities and attacks.
downstream	Information flowing from the hub to the user
dropout	A loss of electrical voltage, either intentional or unintentional, over any period of time from 1 msec to days.
encryption	Encoding a message in a way that only authorized users can access it, usually requiring a key. Encryption is a higher order of obfuscation.
IoT	Internet of Things (IoT) devices have the ability to connect to the Internet either directly or indirectly (via a bridge) to provide greater functionality.
obfuscation	Method used to mask data except for authorized users. It can be simple, e.g., reversing the order of bits, exclusive or with a known value, etc., or complicated such as with encryption.
persistent memory	Any memory device which retains the data when there is no power.
symmetric encryption	The same key can be used for the encryption and decryption of data.
upstream	Information flowing from the user to the hub
vulnerability	Any cybersecurity method which could be used to gain access to data or control of an IoT device or the data associated with an IoT device.

6. Product Selection

6.1. Cybersecurity Responsibilities

6.1.1. Vendor

The vendor is the responsible party for implementing and maintaining the cybersecurity practices in the development and support of their products and/or services. It is expected that they will meet or exceed all guidelines mentioned in this document.

Vendors are expected to have a third party audit and review their cybersecurity practices including all hardware and software development, and support practices for products and/or services.

6.1.2. Service Provider Internal

The service provider is the responsible party for implementing and maintaining the cybersecurity practices in the deployment and support of their products and/or services in conjunction with the vendors. This includes integration of all products required for a service. It is recommended that they will meet or exceed all guidelines mentioned in this document.

Service providers should have a third party audit and review their cybersecurity practices including all product and system development, and support practices for their products and/or services.

6.1.3. Customer

It is known that there are limitations on what the vendor and service provider can provide to the customer with regards to cybersecurity. There needs to be a balance between customer privacy and customer cybersecurity. Therefore, cybersecurity is the responsibility of the vendor and service provider to determine that balance and is outside the scope of this document.

6.2. Hardware Security Recommendations

No single level of security fits all devices and deployments. The service provider and vendor are expected to work together to select a security/cost relationship that is acceptable for the deployment based upon risk and the impact of a potential exploit.

6.2.1. Persistent Memory

Persistent memory is any memory that retains the data when power is not applied. Examples include flash memory, battery backed up memory, serial electrically erasable read only memory (EEROM), and hard disk drives (HDD). It can include firmware and configuration and general data. Configuration data includes critical data such as the device ID (e.g., MAC address) and encryption keys (e.g., root of trust, certificates), and non-critical data such as customer settings and operation rules.

Persistent memory probes are a common method to locate vulnerabilities. This allows people to read the firmware programs to probe to look for vulnerabilities.

While encryption of the firmware would be ideal, this is generally not practical for some IoT devices. However, obfuscation can add enough complexity to thwart most probes and is relatively simple to implement. Some examples are to swap address lines, swap data lines, and/or logically “exclusive or” all data with known values.

Under all conditions, data integrity needs to be insured by using error detection mechanisms such as hashes or cyclic redundancy codes (CRC) as well as redundancy for critical configuration data, such as the media access control (MAC) address or encryption keys.

If there is an error detected in the firmware, then the firmware should have a mechanism that puts the device in a “safe” mode that minimizes network interaction other than to download new firmware, or if not, cease all operation and be considered as a failed unit. If there are any methods to display status to the user in the device, (e.g., light emitting diodes, or LEDs), then discovering types of problems could potentially be made easier if there is a specific pattern displayed on the LED to indicate type of failure.

If there is an error detected in the critical configuration data, then redundant entries should be validated, and if validated, utilize this value. If there is no validated configuration, then the device should cease all functionality and be considered as a failed unit. If there are any user indicators (e.g., LEDs) in the device, then discovering this type of problem could potentially be made easier if there is a specific pattern displayed on the user indicator to indicate this type of failure.

If there is an error detected in non-critical configuration or general-purpose data in the persistent memory, then the firmware should have a mechanism to gracefully handle this without exposing any security vulnerabilities. One example would be to have the non-critical data stored in a cloud server with a mechanism for the device to request it be reloaded to the device due to memory corruption.

Careful consideration needs to be given in the determination of critical vs. non-critical data. Incorrectly marking certain data as non-critical can allow exploits that, for example, enable device behavior or features that the service provider intentionally disabled.

Table 1 - Persistent Memory Recommendations

6.31.1	It is recommended that all persistent memory be encrypted or at least obfuscated to minimize probes into the firmware.
6.3.1.2	It is strongly recommended that all data in persistent memory contain some method to detect error conditions.
6.3.1.3	When an error condition is detected in firmware stored in persistent memory, the firmware should have a mechanism to put the device in a “safe” mode that minimizes network interaction other than to download new firmware. If there is no ability to download new firmware, then the unit should cease all operation and be considered a failed unit.
6.3.1.4	When an error condition is detected in persistent memory, if the data is critical then the device should cease to function and be considered a failed unit.
6.3.1.5	When an error condition is detected in persistent memory, if the data is not critical then the device should use “safe” values and preferably have the data stored in the cloud and reload those values.
6.3.1.6	Whenever an error occurs with persistent memory, then if there are any user indicators (e.g., LEDs) on the device, then there should be a specific pattern for that type of error to assist with debugging these issues.

6.2.2. Non-Persistent Memory

Non-persistent memory is any memory that does not guarantee the retention of data when power is removed. Note that some memory might retain or partially retain the data depending on the length of power removal or if the device is rebooted. This memory can also be probed even though it is more difficult to probe than persistent memory. This type of memory includes dynamic random access memory (DRAM) , static random access memory (SRAM), internal registers, etc.

Since this memory can contain firmware, working data, configuration data, buffers, etc., it is important that any memory which is externally available be obfuscated, and if the data is critical, preferably by encryption. Note that memory can be internal to the system on chip (SoC) but still be available externally, either through buses or via diagnostic ports, such as Joint Test Action Group (JTAG). It can be very difficult to obfuscate internal memory, and the better mechanism is to not allow access to any internal memory externally.

Table 2 - Non-Persistent Memory Recommendations

6.3.2.1	It is recommended that non-persistent memory that is externally available be obfuscated to minimize probes for vulnerabilities.
6.3.2.2	It is recommended that external non-persistent memory be encrypted.
6.3.2.3	It is recommended that internal non-persistent memory not be available externally (e.g. via data buses).
6.3.2.4	It is recommended that JTAG ports be disabled after manufacture for production units.
6.3.2.5	It is recommended that any other diagnostic or developer ports be disabled after manufacture for production units.

6.2.3. Secure Cryptographic Operations

Many of the new SoCs include a separate secure processor. Where possible, it is strongly recommended that these be utilized for every cryptographic operation. Generally, this also includes secure storage of sensitive data.

In the absence of enough secure storage for sensitive data, it is recommended that this data be stored encrypted in persistent memory with the encryption/decryption being managed by the secure processor along data authentication. Additionally, it is recommended that any sensitive data be stored redundantly (see recommendations in section 6.2.1).

Table 3 - Secure Storage/Processing Recommendations

6.3.3.1	If the SoC provides secure cryptographic operations, it is strongly recommended that this be utilized for all cryptographic operations.
6.3.3.2	If there is insufficient secure memory for all the sensitive data, then it is recommended that the data be stored encrypted in persistent memory with authentication when read (e.g. hash), and that the encryption/decryption be managed by the secure processor along with data authentication.

6.2.4. Communication Ports

All communication ports (e.g., serial, SPI, I^{2}C, USB) that are not utilized in the normal functioning of the device need to be disabled.

Any communication ports that are utilized in the normal functioning of the device need to ensure that the drivers are well designed and do not present any security vulnerabilities.

Table 4 - Communication Ports Recommendations

6.3.4.1	All communication ports not utilized should be disabled.
6.3.4.2	Any communication ports that are utilized need to ensure that the drivers are well designed and do not allow any security vulnerabilities.

6.2.5. Physical Security

Physical security can do a lot to limit the opportunity for proving of vulnerabilities, especially if specialized tools are required.

6.2.5.1. Potted Electronics

This consists of placing the electronics within some material, such as epoxy, which will limit access to the physical connections of the device. Note that for devices that are not repairable, the additional cost is very small. For devices that are repairable, the operational costs can be high for repair. Depending on the design, an economical partial potting provides increased security without additional costs can be possible.

Note that some epoxy materials are piezoelectric and can cause electrical surges when mechanically shocked, so care needs to be taken in the selection of the material. Note that the thermal properties of the potting material in conjunction with the electronics thermal requirements needs to be taken into account.

6.2.5.2. PCB Traces

If the device utilizes multi-layer printed circuit boards (PCBs), where possible any signal that can be a security risk should only reside within the inside layers where they are not easily accessible.

6.2.5.3. Enclosure

The mechanical enclosure of the device can assist in the physical security. Devices which are not repairable can utilize mechanisms which are difficult to gain access (e.g., rivets). Devices which are repairable can utilize mechanisms which require additional effort without the correct tools to gain access, such as non-standard screw heads.

Table 5 - Physical Security Recommendations

6.3.5.1	If economically possible, the electronics should be potted.
6.3.5.2	PCB traces for sensitive signals should be only on the inside layers (if a multi-layer PCB is utilized).
6.3.5.3	Non-repairable device enclosures should utilize mechanisms which raise the difficulty of obtaining access.
6.3.5.4	Repairable device enclosures should utilize non-standard mechanisms to gain access.

6.2.6. Power Supply

6.2.6.1. AC Power Supplies

While usually not thought of as important to cyber security, a poor power supply design has been used to gain access into the processor. This can be accomplished by either brownout conditions, short event power dropouts, surges on the power line, or a combination of the above. Without a good protection, this can sometimes crash the processor and cause it to start executing code that allows for access.

Good design practice should be implemented to protect against any surges by initiating a processor reset when one is detected.

The brownout condition requires a good analysis of the demands of the circuit compared to the capabilities of the power supply. The processor reset should become active well before a brownout condition causes out of spec voltages anywhere in the circuit.

For power supply dropouts, similar circuitry to the brownout condition can be utilized. This requires knowing if the capacitors can adequately short dropouts without causing out-of-spec voltages anywhere in the circuit. As an example, in one piece of equipment the processor would lock up (crash) if there was a 5.5 cycle dropout. The processor performed fine for five cycles and reset correctly for six cycles but would fail in this case. This example coincided with the weekly backup generator test which is why it was found.

The power supply and reset circuit design should start at the beginning of the design rather than testing later to see if it is adequate. However, testing is critical not just to cybersecurity but also to the general robustness of the device.

6.2.6.1. Battery Power Supplies

Battery supplies are generally thought to be relatively simple. For non-rechargeable and rechargeable batteries, the same issues exist as the AC power supplies when an attacker is trying to gain access. Therefore, the same design practices as the AC power supplies need to be followed (including the power supply dropouts).

Table 6 - Power Supply Recommendations

6.3.6.1	The power supply should be designed such that when any surge occurs which would cause improper voltages in the device, that all processors will be reset.
6.3.6.2	The power supply should be designed such that when any input voltage is lower than such that the power supply can supply proper voltages in the device, that all processors will be reset.
6.3.6.3	The power supply should be designed such that any interruption to the input voltage that is longer than what the internal capacitors can provide proper voltages, that all processors will be reset.

6.2.7. Development vs Production

As part of the normal development cycle, there is a need to gain access to the various ports and test points within the product. It is a good practice to have two different hardware designs rather than to try to use the same design for development as is used for production. Additionally, the development units should be of limited quantity and tracked to ensure that an attacker cannot gain access.

Table 7 - Development vs Production Recommendations

6.3.7.1	There should be two separate hardware designs, one used only for development with access to the communication ports, and one for production, which does not have any access for communication ports not utilized in normal operation.
6.3.7.2	Any development units should be tracked and accounted for to prevent access to attackers.

6.3. Firmware Security Recommendations

Even with secure hardware, there is always the possibility that there are security holes in the firmware within the device. These should be carefully designed and tested to ensure that there are no known vulnerabilities. Note that there is always the possibility that a vulnerability exists even with good firmware development practices. Therefore, a good and secure methodology to update firmware is a need for any network product.

6.3.1. Coding Security Practices

All coding should follow modern secure coding practices for the coding language utilized.

A good example is the "Introduction to Secure Software Coding Practices for reference design kit (RDK) & CPE" at the RDK Wiki Website for an excellent compendium of good practices (see section 3.2 for the uniform resource locator (URL)). These are recommendations only and should be adapted to the individual device.

Table 8 - Development vs Production Recommendations

6.4.1.1	All coding should follow modern secure coding practices for the programming language utilized.

6.3.2. Bootloader

One of the first lines of cybersecurity defense for any device's firmware is the bootloader. The bootloaders primary function is to validate the firmware stored in persistent memory (usually flash memory) before beginning its execution. If the firmware does not validate, then a secondary function should be to securely load new OS and/or firmware.

Note that there is an assumption that the hardware security recommendations have been followed and that reprogrammable memory is used for firmware storage.

6.3.2.1. Bootloader Overview

Within a device, either a flat memory or segmented firmware memory model can be utilized. A flat memory model is one where the drivers, operating system (OS), middleware (if present), and application are stored in memory as a monolithic package (which may or may not be file based). A segmented model is one in which these different firmware entities are stored is different packages (usually a file system is used). There are advantages and disadvantages to both models which will not be covered in this paper.

Using either model, a bootloader is utilized when the central processing unit (CPU) comes out of reset to initialize any hardware registers required for basic operation, validate the firmware, and start execution of the firmware.

There are generally two different mechanisms which are used to ensure that if the firmware is ever corrupted, valid firmware can still be loaded. One method is to keep two copies of the firmware, generally, the latest version and the previous version. If the latest version fails to validate, then the previous version will be validated and then executed. This allows for a smaller and simpler bootloader at the expense of more persistent memory. The other method is to have the bootloader incorporate all the functionality required to download new firmware, which is then validated and executed. This method allows for less memory at the expense of a larger and more complicated bootloader, which if stored in non-reprogrammable memory, should be extensively tested before production.

6.3.2.2. Bootloader Storage Type

It is recommended that all or the initial bootloader memory reside in non-reprogrammable memory (e.g., read only memory (ROM)) so that attackers cannot change the bootloader. If the bootloader has multiple stages where the next stage is in a different memory, then each stage of the bootloader needs to validate the next stage prior to execution.

If stored in flash memory, then the bootloader should be segmented from all other memory to ensure that mechanisms which wear out flash memory cells will not affect the bootloader. Many flash memory designs have a segment which can only be programmed once and then a fuse is blown (or the equivalent) to prevent writing to that segment again.

Table 9 - Bootloader Memory Storage Recommendations

6.4.2.1	Initial bootloader memory should reside in non-reprogrammable memory within the microcomputer or SoC.
6.4.2.2	If the bootloader has multiple stages, then each stage of the bootloader should be validated before the it is executed.
6.4.2.3	If flash memory is used, then the bootloader program should be in a segment of memory all to itself and not shared with any other files or programs.

6.3.2.1. Bootloader Validation Types

The validation performs two operations: ensures that the bootloader memory is intact and has not had any failures and that an attacker has not modified the bootloader memory. The different type of validation methods generally utilized included cyclic redundancy code (CRC), hash algorithms, and full cryptographic authentication (NOTE: CRC is not recommended as it can be cryptographically insecure). CRC is generally the fastest and smallest method, hash algorithms are not as fast but relatively small, and full cryptographic authentications are the slowest and take up the most memory for the cryptographic data. While recognizing there are different types of cryptographic authentications, for this document they will be considered as just one method.

6.3.2.2. Embedded vs Non-embedded Memory Bootloader Types

The ideal design is all memory is embedded within the silicon that the processor resides within. The second-best design is that the bootloader memory is embedded within the silicon while the application firmware is in external memory (either flash memory or DRAM). The most vulnerable design is where the bootloader and application firmware all reside in external memory. All three cases will be covered.

It is preferred that the base bootloader functionality be in non-reprogrammable memory so that attackers cannot change the bootloader. The bootloader should first be authenticated to verify its integrity before validating the later bootloader stages, if present, and the application memory. If the bootloader is in reprogrammable memory, then it needs to validate its integrity.

For devices with embedded program memory (i.e., within the SoC), there is very little chance of external vulnerabilities. Therefore, the bootloader should only validate the integrity of itself and the firmware memory. Cryptographic tests will still need to be performed on any firmware memory that is upgradeable.

For devices with the bootloader memory embedded within the silicon but with external memory for the later bootloader stages and the application firmware memory, it is preferred that a cryptographic authentication be performed on any external memory prior to execution to ensure that it has not been modified. If a full cryptographic authentication is not possible for whatever reason, then a hash method is always preferred over CRC for validation.

For devices where the program memory, including the bootloader and application firmware, are in external memory, then the bootloader should utilize cryptographic authentication of all firmware memory.

Table 10 - Bootloader and Firmware Validation Recommendations

6.4.2.4	The base bootloader firmware should be in non-reprogrammable memory.
6.4.2.5	The bootloader should always validate itself, especially if it is in reprogrammable memory.
6.4.2.6	If the bootloader, drivers, operating system (OS), middleware, and application firmware are stored in the same silicon as the CPU (internal memory), then CRC or equivalent validation can be utilized, although cryptographic validation is preferred.
6.4.2.7	If the drivers, OS, middleware, and application firmware are in external memory, then cryptographic validation should be utilized for the drivers, OS, middleware, and application firmware.
6.4.2.8	If the bootloader, OS, middleware, and application firmware are all in external memory, then cryptographic validation should be utilized for validation of all firmware memory.

6.3.2.3. Bootloader Secondary Functions

In the rare event that memory does not validate, the bootloader has two possible outcomes. It can either cease execution (except for a mechanism to indicate that power is applied but that the memory is corrupt), or it can download new memory. If the latter is utilized, then care needs to be taken to ensure that the firmware download mechanism is secure. See section 6.3.3 for further details on firmware download security practices.

Table 11 - Bootloader Secondary Functions Recommendations

6.4.2.9	The device should have the ability to download new firmware in the event that the current firmware does not validate.
6.4.2.10	If the device detects that bootloader or firmware does not validate, and it cannot download a new bootloader or firmware, then an indication should be given to the reason and execution should cease.

6.3.3. Firmware Download

From a security viewpoint, every IoT device should have the ability to download new firmware to counteract any vulnerability discovered. However, care should be taken to ensure that the firmware download mechanism itself does not introduce any new vulnerabilities.

One methodology is defined by the update framework (TUF), which provides a framework for securing software update system. The URL can be found in section 3.2.

Generally, there are two mechanisms to deliver the firmware: either through the network or via some hard connection (such as a universal serial bus drive (USB), SD Card, etc.).

There are two stages for network-based firmware download:

1. An event triggers the need to download new firmware.
    a. A firmware management system triggers the download (directed download).
        i. In a push environment, the firmware management system sends the trigger without any knowledge of the management system to the device (an example is the Simple Network Management Protocol (SNMP) trigger used in DOCSIS). Generally, a file name and location will be passed to the device.
        ii. In a pull environment, the device will discover the management system and communicate with it to determine if a download is required (an example is TR-069). The discovery mechanism is not covered in this document. The current version is advertised to the device and the management system will provide the associated file name and location.
        iii. In a hybrid environment, the device discovers the management system, determines if a new version is available (pull) but retains a connection such that the management system can initiate a firmware download (push) at times other than the initial connection (an example is TR-069).
    b. The bootloader detects corrupted program (self-triggered download)
        i. The device will need to access the firmware management system by some means to obtain the correct file name and location. Discovery is essential in this case, and for robustness needs to be straightforward with minimal variables. An example would be a hard-coded FQDN (fully qualified domain name), or via DHCP (Dynamic Host Configuration Protocol) option.
        ii. This will be performed by the bootloader and should be a relatively simple, reliable, and robust implementation.
    c. In all cases, the trigger to download new firmware as well as the file name and location should be cryptographically authenticated.
2. The new firmware is delivered to or retrieved from the device. The mechanism for this is outside the scope of this document. The firmware should be encrypted to prevent reverse engineering.
    a. The device should ensure that the entire firmware contents have been received prior to programming into the persistent memory.
    b. The device should be cryptographically authenticated prior to programming into persistent memory.
    c. It is recommended that after programming into persistent memory, that the persistent memory be validated prior to executing the new firmware, especially if the device is not rebooted after the download.
    d. It is recommended that the new firmware be encrypted to prevent reverse engineering of the firmware in the case of the data being intercepted.

e. For the segmented memory model, it is recommended that some mechanism for version control be implemented to ensure that different versions of the firmware segments have been qualified to work together.
f. The flat memory model is preferred.

Examples of firmware delivery include bidirectional mechanisms such as File Transfer Protocol (FTP) , HTTP, HTTPS, trivial file transfer protocol (TFTP), etc., and unidirectional mechanisms such as digital storage media – command & control (DSM-CC) data carousels, file delivery over unidirectional transport (FLUTE), etc. Each mechanism has its advantages and disadvantages but if the firmware package is not already encrypted, then HTTPS would be the preferred mechanism.

Proprietary mechanisms of delivery have the advantage of being undocumented and difficult to parse by outsiders but have the disadvantage in that they can be more open to vulnerabilities due to the lack of peer review.

A hardware-based firmware download (such as a USB drive) should follow at least the same authentication procedures as the network-based download but could use a higher level of security. In addition, from a security viewpoint it is recommended that each firmware download file authentication be unique to each device so that multiple copies cannot be programmed into all devices.

It is recommended that any firmware download mechanism be audited by security experts prior to deployment.

Table 12 - Firmware Download Security Recommendations

6.4.3.1	Every IoT device should have the ability to download new firmware.
6.4.3.2	Every mechanism used to trigger a firmware download and the associated information to perform the download should be cryptographically authenticated.
6.4.3.3	The firmware package should be encrypted to prevent reverse engineering.
6.4.3.4	The firmware package should be authenticated prior to programming into persistent memory.
6.4.3.5	Any hardware connected method to download new firmware (such as USB drives) needs to utilize the same security as the network download mechanism or a higher level of security.
6.4.3.6	Any hardware connected method should be unique to an individual device in order to prevent programming into multiple devices, especially if the code can be rolled back.
6.4.3.7	The firmware download mechanism should be audited by security experts prior to deployment.

6.3.4. Network Protocols

The lack of security conformance standards for the IoT space is its greatest security vulnerability. There are several practices that can be implemented to minimize this vulnerability. First, there are some services, ports, and protocols which have known vulnerabilities. If there is a need to utilize a service, port, or protocol with a known vulnerability (for example, legacy control), then care should be taken to ensure that countermeasures for the vulnerabilities are implemented. These should not be available in the IoT device, which is not always as easy as it sounds since many operating systems utilize a wide range of protocols besides the ones used by the application. An examination of all open ports can assist in determining which processes are servicing requests and any that are not required need to be removed. Full removal should consist of removing all binaries, libraries, and associated files (e.g. configuration) from the firmware as opposed to simply disabling them.

Note that there are vulnerabilities even for the low-level protocols, such as ARP poisoning, TCP spoofing, and domain name system (DNS) attacks. Therefore, it is not expected that the IoT devices will use only invulnerable protocols. It is expected that best industry practices will be utilized to ensure that known vulnerabilities cannot be exploited by an attacker.

In general, any higher-level protocol which does not include some method of authentication and encryption should not be used. For example, the device should not utilize telnet, rlogin, rsh, and rsexec as these are less secure than secure shell (SSH). SNMPv3 should be utilized instead of SNMPv1 and SNMPv2. HTTPS should be used instead of HTTP and DNSSEC should be used instead of DNS.

It is strongly recommended not to use self-signed certificates but to utilize certificates from a trusted certificate authority (CA).

Care needs to be taken to ensure that legacy support does not create vulnerabilities. Any service, port, or protocol which is not required for the current implementation should be removed, even if it was in a previous version. If a service, port, or protocol is being utilized with a known vulnerability, then both the control system and the device need to be updated to utilize a secure option.

If multicast is utilized, then source specific multicast (SSM) should be utilized.

In general, use good security practices when selecting and implementing network protocols with the assumption that any vulnerability will lead to an attack, and architect the system appropriately.

In addition, care should be taken to be sure that all data is kept secure not only during transmissions but even when stored in cloud servers.

Table 13 - Network Protocol Security Recommendations

6.4.4.1	Communication protocols with known vulnerabilities should not be utilized unless additional security is added.
6.4.4.2	An examination to confirm that the operating system does not utilize any vulnerable communication service, ports, or protocols by default should be performed.
6.4.4.3	For Internet Protocol (IP) based devices, a test to determine which IP ports are open should be performed to guarantee that there is no unexpected access to the device.
6.4.4.4	Industry best practices should be utilized to ensure that all known vulnerabilities cannot be used to cause security attacks to the IoT device or its associated cloud network servers.
6.4.4.5	High level protocols which do not include some method of authentication and encryption should not be used.
6.4.4.6	Self-signed certificates are not recommended without careful consideration by the Service Provider.
6.4.4.7	Certificates from a trusted certificate authority should be used.
6.4.4.8	Legacy protocol support should be examined and if known vulnerabilities cannot be removed, then they should not be supported. As soon as a vulnerability is known, it should be removed.
6.4.4.9	If multicast is utilized, then source specific multicast (SSM) should be utilized.

6.3.5. Passwords

Use of hardcoded passwords and "password of the day" should be avoided for any communication sessions, especially diagnostic sessions. Instead, either utilize authentication, authorization, and administration (AAA) protocols or a multistage password authorization method where a separate encrypted, authenticated message supplies the needed password for access.

It is recognized that a default password is required, however, these should be unique to each device and no model-wide password be utilized.

Table 14 - Password Security Recommendations

6.4.5.1	Hardcoded passwords should be avoided for any communication sessions.
6.4.5.2	Password of the day should be avoided for any communication sessions.
6.4.5.3	AAA protocols should be used.
6.4.5.4	Multistage password authorization methods may be used where a separate encrypted, authenticated message supplies the needed password for access.
6.4.5.5	Any default password should be unique to each device.

6.3.6. User Interface (UI) Security

Some attacks have occurred via a user interface (UI) portal on the device. This can be addressed by the following:

- Require that any UI requires a password prior to making any configuration changes (status is generally okay to display without a password)
- Do not use hardcoded default passwords to gain access to the UI (this has become a common mechanism to attack). See section 6.3.5.
- Do not use MAC address-based passwords. The MAC address is easily accessed.
- It is acceptable to print the password on the device, preferably in an area that is not easily accessible (e.g., on the bottom rather than on the top)
- If possible, utilize HTTPS

Table 15 - User Interface Security Recommendations

6.4.6.1	Any user interface which can change the device configuration should require a password entry.
6.4.6.2	User interface status displays may be displayed if no sensitive information is displayed.
6.4.6.3	Hardcoded default passwords should not be used to gain access to the UI.
6.4.6.4	MAC address-based password should not be utilized.
6.4.6.5	The default password may be printed on the device, preferably in an area that is not easily accessible.
6.4.6.6	HTTPS should be utilized when possible.

6.3.7. Certificate and Key Management

The actual format and requirements of certificates is outside the scope of this document. This document will cover the best practices for storage and access of any certificates within the IoT device.

Certificates should be considered as secure information. While they do not need to be encrypted, they should be managed such that no unallowed substitution or addition can be made.

Cryptographic data other than standardized certificates (e.g., non-standard certificates, keys) should be stored securely. In order of preference, the following mechanisms should be utilized for their storage:

1. Internal in the SoC with no outside access (clear or obfuscated storage)
2. Internal in the SoC with outside access (obfuscated storage)
3. External to the SoC (obfuscated storage)
4. Internal in the SoC with outside access (clear storage)
5. External to the SoC (clear storage)

Items 5 in the previous list should be discouraged.

Table 16 - Certificate and Key Management Security Recommendations

6.4.7.1	Security certificates should be stored in a manner which prevents non-allowed substitution or additions.
6.4.7.2	Cryptographic data should be stored within the SoC with no outside access.
6.4.7.3	Cryptographic data that are stored within the SoC with outside access should be obfuscated.
6.4.7.4	Cryptographic data that are stored external to the SoC should be obfuscated.

6.3.8. Miscellaneous

The following items do not fall under any particular subject but should be considered.

6.3.8.1. Factory Reset

Several attacks have utilized performing a factory reset on a device as there can be a default configuration value which has a vulnerability. Therefore, an examination of all factory default configuration values should be undertaken to ensure that no value enables a known vulnerability.

An example is that the usernames and passwords go to the default values. When there are well known default values, then this can allow a vulnerability to open (see section 6.3.5).

6.3.8.2. Open Source Software

There has been a large growth in the use of open source software. This software is often assumed to be well known and free of vulnerabilities. However, there have been several vulnerabilities and attacks against devices that utilized open source software. There should be no assumption that open source software is free from security vulnerabilities. It is the implementors responsibility to ensure that there are no vulnerabilities in the product and they should treat open source as carefully as they would their own written code.

As a responsibility of using open source, the vendor should monitor each component for vulnerabilities. If a vulnerability is discovered, then the vendor should follow the recommendations of section 6.4.3.

6.3.8.3. Cryptographic Implementations

It is strongly recommended to utilize standard cryptographic algorithms over proprietary algorithms, preferably using standard implementations. Standard algorithms, especially those defined by the FIPS Data Encryption Standards (FIPS 46-3) and the Advanced Encryption Standards (FIPS 197) have been thoroughly tested. Proprietary algorithms usually have not been mathematically proven to not have vulnerabilities that can be exploited.

Care needs to be taken to ensure that the cryptographic algorithms are implemented correctly. For example, initialization vectors (IV) need to be carefully selected to avoid any weak implementations.

Key management needs to be carefully implemented. Avoid hard coded keys. Keys need to be generated and protected in physically secure environments. Keys should always be revocable. Key strength needs to be adequate for the implementation. Key distribution channels need to be secure, especially at the factory.

Where possible, plan for future growth of cryptographic requirements and allow for updating of cryptographic functions in the field.

6.3.8.4. Documentation

It is highly recommended that a security requirements document be generated prior to the development of a device. Quality assurance (QA) testing needs to assure that all the different requirements, including security requirements, have been met.

A security architecture document should be developed with the security requirements document. This can be very valuable if an attack occurs to determine how best to mitigate the attack, especially if the original developers are not available.

Implementation decisions should go through a security design review to ensure that there are no fundamental security vulnerabilities.

Additionally, the vendor should generate threat model documentation that shows the possible threats the device will encounter and how they are denied (or not). This is to be made available to the service provider for every release of the product version and firmware version.

Table 17 - Documentation Security Recommendations

6.4.8.1	Each IoT device should have a review to determine that there are no security vulnerabilities with the factory default configurations.
6.4.8.2	When the vendor utilizes open source software, there should be a review to determine that there are no known vulnerabilities in the version of code utilized.
6.4.8.3	Only standard cryptographic algorithms should be utilized.
6.4.8.4	The initialization vectors (IV)utilized for cryptographic algorithms should be analyzed for weak implementations.
6.4.8.5	Avoid hard coded keys.
6.4.8.6	Keys should be generated and protected in a physically secure environment.
6.4.8.7	Keys should always be revocable.
6.4.8.8	Key strength should be adequate for the implementation.
6.4.8.9	Key distribution should be analyzed to determine that it is secure.
6.4.8.10	Cryptographic implementations should be capable of being updated.
6.4.8.11	A security requirements document should be generated prior to development of any device.
6.4.8.12	A security architecture document should be developed with the security requirements document.
6.4.8.13	All implementation decisions should go through a security design review.
6.4.8.14	The vendor should generate threat model documentation delivered to the service provider for each product version and firmware version.

6.4. Recommended Vendor Responsibilities for the Life of the Product

The following section outlines the recommended vendor cybersecurity responsibilities for the life of the product. It is recommended that the service provider take this under consideration for the service level agreement with the vendor.

6.4.1. Point of Contact

For all aspects of cybersecurity notifications, both the vendor and the service provider should have a well-known primary point of contact and a secondary point of contact. These points of contact need to understand the sensitivity of all cybersecurity notifications and will keep the details of these communications on a need to know basis, unless it is jointly decided that the notification will be made public (if it is not already public).

Table 18 - Point of Contact Recommendations

6.5.1.1	The service provider should provide both primary and secondary points of contact for vendors for any cybersecurity notifications.
6.5.1.2	The vendor should provide both primary and secondary points of contacts for service providers for any cybersecurity notifications.
6.5.1.3	All cybersecurity notifications should be considered as proprietary and confidential information and handled on a need to know basis unless both parties agree otherwise.

6.4.2. Public Disclosure

Ultimately, it is the vendor that should determine whether any notification needs to be made public as well as the way is it made public. The vendor should notify the service provider that it will be making a public disclosure with enough time for the service provider to have a public response ready.

Table 19 - Public Disclosure Recommendations

6.5.2.1	Any decision to make any cybersecurity information regarding a certain product should be made by the vendor.
6.5.2.2	Any public notification should be provided to the service provider with enough time for the service provider to have a public response ready.

6.4.3. Vulnerability Notification

When a vendor discovers or is notified of a vulnerability of a product used by the service provider, the vendor should notify the service provider within a reasonable period, for example, five business days after the discovery although one business day is preferred. If the service provider discovers the vulnerability, they should utilize the point of contact (see section 6.4.1) to notify the vendor.

The vendor should provide a vulnerability mitigation plan to the service provider with the vulnerability notification.

Since vulnerabilities are many times discovered by third parties, when a vendor is notified of a vulnerability, these will usually be made public after a certain period of 45-90 days. For these notifications, the vendor should notify the service provider of the vulnerability discoverer and when the vulnerability will be made public.

Table 20 - Vulnerability Notification Recommendations

6.5.3.1	The vendor should be required to notify the service provider of all vulnerabilities within a reasonable period after discovery through the point of contact. An example of a reasonable time is five business days.
6.5.3.2	The vendor should provide a vulnerability mitigation plan to the provider with the vulnerability notification.
6.5.3.3	For third party vulnerability discoveries, the vendor should notify the service provider of the identity of the vulnerability discoverer and when the vulnerability will be made public.

6.4.4. Attack Notification

When a vendor discovers or is notified of an attack of a product used by the service provider, the vendor should notify the service provider immediately. The vendor should provide the service provider with their public disclosures and when they will be disclosed. The service provider should work with the vendor on the service providers disclosures to be sure that there are no discrepancies.

When the vendor notifies the service provider of an attack, the vendor should provide attack mitigation solutions to the service provider as soon as possible.

If the service provider discovers an attack, then the service provider should notify the vendor as soon as possible.

Table 21 - Attack Recommendations

6.5.4.1	The vendor should be required to notify the service provider immediately when any attack is detected.
6.5.4.2	Any vendor attack notification should include any public disclosure releases and the schedule of the releases.
6.5.4.3	The vendor and service provider should work jointly on all disclosures to ensure that there are no discrepancies between them.
6.5.4.4	The vendor should provide attack mitigation solutions to the service provider as soon as possible.
6.5.4.5	If a service provider discovers an attack, then the service provider should notify the vendor as soon as possible, and the vendor needs to assist with mitigating the attack.

6.4.5. Firmware Updates for Security Vulnerabilities and Attacks

When a vulnerability is discovered, if new firmware is required to close the vulnerability, then the vendor should provide dates on when the patched firmware will be available to the service provider as well as any known network configurations, side effects, etc. of the vulnerability patch. Often the vulnerability patch will be integrated with a firmware build which includes new features and bug fixes. In some cases, there can be network configurations which will minimize the vulnerability. If network configurations exist which minimize the vulnerability, then the vendor should provide these to the service provider as soon as possible.

When an attack is discovered, if new firmware is required to defeat the attack, then the vendor should utilize the current build that the service provider is utilizing with no new features, bug fixes, etc. This is because the service provider generally has a QA cycle that the firmware must go through, and it will delay the QA cycle if there are new features or bug fixes.

Table 22 - Firmware Update for Security Vulnerabilities and Attacks Recommendations

6.5.5.1	If a vulnerability is discovered, then the vendor should provide data on when patched firmware that closes the vulnerability is available as soon as possible.
6.5.5.2	If a vulnerability is discovered, then the vendor should provide any network configuration workarounds to the service provider as soon as possible.
6.6.5.3	The vendor and service provider may choose to include vulnerability mitigation in a future firmware release depending on the vulnerability risk analysis.
6.5.5.4	If an attack is discovered, then the vendor should utilize the current build that the service provider is utilizing to fix the vulnerability that is being attacked. This new firmware is not to include any new features, bug fixes, or any other change other than closing the vulnerability in order to reduce the service provider's QA cycle.
6.5.5.5	If an attack is discovered, then the vendor should provide mitigated firmware to the service provider as soon as possible.

6.5. Branding

All products that have been certified by a cybersecurity authority, such as Open Connectivity Foundation (OCF), should have the certification brand displayed on the product. This is especially true if the product is available in the retail market (see section 7).

A product that has not been cybersecurity certified should not display a certification brand.

Table 23 - Service Provider Purchased Product Branding Recommendations

6.6.1	All products that have been certified by a cybersecurity authority should have the certification brand displayed.
6.6.2	A product that has not been cybersecurity certified should not display a certification brand.

6.6. Security Qualification

It is expected that the vendor perform penetration testing per industry standards prior to release to the service provider the hardware and all firmware releases.

The service provider should perform their own penetration testing, either internally or via a third party.

All vulnerabilities should be handled as described in section 6.4.3.

Table 24 - Security Qualification Recommendations

6.7.1	The service provider should perform independent penetration testing of all IoT devices.

7. Retail Products

Customers will place unmanaged retail IoT devices on the network, and the service provider has no control over these devices. The only option they have if there is a rogue device is to isolate the customer and inform them that the device is rogue. Each service provider needs to come up with a plan on how to deal with a customer's IoT device going rogue.

When a service provider offers management of retail IoT devices purchased by the customer, they now have some control over which devices are allow or not, and what version of firmware is required. This section covers the minimum cybersecurity requirements that the managed retail IoT devices need to meet. It does not cover unmanaged retail IoT devices.

Additionally, if the service provider sells IoT devices through their retail store, then these devices need to meet the same cybersecurity requirements even if they are not managed via the service provider.

7.1. Branding

Only products which have been certified by a trusted cybersecurity authority, such as OCF, should be allowed to be listed in the service provider's allowed IoT devices.

All products certified by a cybersecurity authority, such as OCF, should have the certification brand displayed on the product and the retail packaging.

Table 25 - Retail IoT Devices Branding Recommendations

7.1.1	All allowed retail IoT devices should be certified by a trusted cybersecurity authority, such as OCF.
7.1.2	All allowed retail IoT devices should have the cybersecurity authority brand displayed on the product and retail packaging.

7.2. Qualification Process

The service provider should utilize a qualification process, either within the service provider or by a trusted third party. All allowed retail IoT devices should go through a qualification process determined by the service provider which includes the same cybersecurity qualifications as service provider provided products.

Table 26 - Retail IoT Devices Qualification Recommendations

7.2.1	All allowed retail IoT devices should go through a qualification process determined by the service provider which includes the same cybersecurity qualifications as leased products.

7.3. Retail Firmware

All allowed retail IoT devices should follow the guidelines defined in section 6.3.

All allowed retail IoT devices should have a mechanism to allow the service provider or device vendor to update the firmware in the devices in the event of a vulnerability and/or attack. The vendor should be required to update and deliver firmware as defined in 6.4.5.

Table 27 - Retail Firmware Recommendations

7.3.1	All allowed retail IoT devices should follow the same firmware requirements as leased products as defined in section 6.3.
7.3.2	All allowed retail IoT devices should have mechanisms to allow the service provider to update the firmware in the devices in the event of a vulnerability and/or attack as defined in section 6.4.5.

8. Provisioning

Provisioning covers the onboarding, configuration, and enablement of features of any managed device onto the network. The following will focus only on the cybersecurity aspects of provisioning, commonly called security onboarding.

8.1. Security Onboarding

When a new product is introduced into the service provider's system, it goes through a provisioning process which will include the security onboarding of all relevant data required to security operate within the system. This can include exchanging and authenticating certificates both from the device to the service provider's system, and from the service provider's system to the device. In some cases, it can include the secure delivery of new certificates from the service provider's system to the device.

It is recommended that standard methods which utilize a well-known CA, such as the method described in the OCF onboarding process.

8.2. Secure Delivery of Provisioning Transactions

All provisioning transactions received by the IoT device should be authenticated. All provisioning authentication should only use the service provider credentials or their proxy. Provisioning transactions should never be authenticated by using vendor credentials unless the vendor is the service provider's proxy. The provisioning transactions are expected to follow the guidelines called out in section 6.3.4.

Access to provisioning interfaces and platforms should be secure and only accessible by authorized individuals. The entire provisioning operations system including platforms should go through a cybersecurity audit.

Table 28 - Secure Delivery of Provisioning Transactions Recommendations

8.2.1	All provisioning transactions received by the IoT device should be authenticated.
8.2.2	All provisioning authentication processes should only use the service provider credentials or their proxy.
8.2.3	Provisioning transactions should never be authenticated by using vendor credentials unless the vendor is the service provider's proxy.
8.2.4	All access to provisioning interfaces and platforms should be secure and only accessible by authorized individuals. Good security practices are expected to be followed.
8.2.5	The entire provisioning operations system including platforms should go through a cybersecurity audit.

8.3. Secure Provisioning Database

The customer provisioning database should be secure and only accessible to authorized individuals, especially if it is provided by a third-party entity.

Table 29 - Secure Provisioning Database Recommendations

8.3.1	The customer provisioning database should be secure and only accessible to authorized individuals.

8.4. Confirmation of Provisioning

Whenever an IoT device receives a provisioning transaction, it should send an authenticated confirmation transaction back to the provisioning system. This allows for a mechanism for discovery of third-party provisioning attacks.

All provisioning operations should be logged and stored securely.

Table 30 - Confirmation of Provisioning Recommendations

8.4.1	When an IoT device receives a provisioning transaction, it should send an authenticated confirmation back to the provisioning system.
8.4.2	All provisioning transactions should be logged with the log being stored securely.

8.5. Whitelisting/Blacklisting

A mechanism needs to exist which will allow for the ability to disallow a known bad actor device from being added or removed from the network. Two common mechanisms are whitelisting and blacklisting. Note that these are from the cybersecurity perspective and not normal operation perspective.

A whitelist requires that every device be inspected prior to being provisioned and through exclusion, disallows certain devices. This has the advantage of requiring that all devices go through some process before being added to the whitelist. The disadvantage is that the process may take a while and be quite expensive and add complexity. Additionally, if a device is removed from the whitelist, some mechanism needs to exist to deprovision the device from the network, so a whitelist should also include the functionality of the blacklist to deprovision a device from the network.

A blacklist allows all devices to be provisioned unless they are on the blacklist. Any device that is placed on the blacklist will also be deprovisioned from the network. A blacklisting system is usually simpler to implement than a whitelist system.

8.5.1. Whitelist System

The whitelist system should have a process to certify products that are placed on the whitelist. The whitelist system should have a secure process to manage additions and deletions to the whitelist. All devices which are provisioned on the network should be on the whitelist. Any entry which is removed from the whitelist should have all instances of the devices on the network deprovisioned.

All devices on the whitelist should include at a minimum a hardware identifier (e.g., model number) and software version. If desired, it can be expanded to include MAC addresses.

Table 31 - Whitelist System Recommendations

8.5.1.1	If a whitelist system is used, then the system should have a process to certify products that are placed on the whitelist
8.5.1.2	A whitelist system should have a secure process to manage additions and deletions to the whitelist.
8.5.1.3	If a whitelist system is used, all devices which are provisioned on the network should be on the whitelist.
8.5.1.4	If a whitelist system is used, any entry which is removed from the whitelist should have all instances of the removed devices on the network deprovisioned.
8.5.1.5	If a whitelist system is used, all devices on the whitelist should include at a minimum a hardware identifier and software version.
8.5.1.6	If a whitelist system is used, devices on the whitelist may include MAC addresses.

8.5.2. Blacklist System

The blacklist system should have a secure process to manage additions and deletions to the blacklist. Any device which is on the blacklist should not be provisioned. Any device added to the blacklist but is already provisioned should be deprovisioned.

At a minimum, devices on the blacklist should include a hardware identifier (e.g., model number) and software version. If desired, it can be expanded to include MAC addresses.

Table 32 - Blacklist System Recommendations

8.5.2.1	If a blacklist system is used, then the system should have a secure process to add or delete devices on the blacklist from a trusted source.
8.5.2.2	If a blacklist system is used, then any device that is on the blacklist should not be provisioned.
8.5.2.3	If a blacklist system is used, then and device that is added to the blacklist but already provisioned should be deprovisioned.
8.5.2.4	If a blacklist system is used, then all devices on the blacklist should include at a minimum a hardware identifier and software version.
8.5.2.5	If a blacklist system is used, then devices on the blacklist may include MAC addresses.

9. Cybersecurity Maintenance

Regular maintenance is required to reduce the chance of an attack. The primary objectives are to keep the firmware upgraded and to perform regular audits.

9.1. Firmware Upgrades

The service provider should have all devices and platforms within the IoT system upgraded to the latest firmware at the vendors recommendation after going through the system provider's quality assurance program. The vendor should provide detailed release notes with each firmware version to the service provider. The vendor should periodically provide a release roadmap to the service provider including all vulnerability fixes. The service provider needs to be sure that their cybersecurity team is kept up to date.

Table 33 - Firmware Upgrade Recommendations

9.1.1	The service provider should have all devices and platforms within the IoT system upgraded to the latest firmware at the vendors recommendation after going through the system provider's quality assurance program.
9.1.2	The vendor should provide detailed release notes with each firmware version to the service provider.
9.1.3	The vendor should periodically provide a release roadmap to the service provider including all vulnerability fixes.

9.2. Device and Platform Auditing

The service provider should have all devices and platforms within the IoT system go through an annual audit to ensure that there are no rogue devices in the network, that all devices are accounted for, and that all devices and platform firmware is up to date.

Table 34 - Device & Platform Auditing Recommendations

9.2.1	The service provider should have all devices and platforms within the IoT system go through an annual audit to ensure that there are no rogue devices in the network, that all devices are accounted for, and that all devices and platform firmware is up to date.

10. Diagnostics

10.1. Logs

All devices and platforms within the IoT system should keep logs of all provisioning transactions, configuration changes, and anything else that might indicate a vulnerability or attack. All log entries are to include a timestamp. The service provider should utilize these logs as part of their cybersecurity auditing process (see section 9.2).

Table 35 - Logs Recommendations

10.1.1	All devices and platforms within the IoT system should keep logs of all provision transactions, configuration changes, and anything else that might indicate a vulnerability or attack, with each entry including a timestamp.
10.1.2	The service provider should utilize diagnostic logs as part of their cybersecurity auditing process.

10.2. Alerts

All devices and platforms within the IoT system should have a mechanism to alert the service provider when a possible vulnerability or attack occurs.

Table 36 - Alerts Recommendations

10.2.1	All devices and platforms within the IoT system should have a mechanism to alert of the service provider when a possible vulnerability or attack occurs.

11. Attack Management Planning

The following is planning for what the steps and responsibilities are in the case that an attack occurs.

11.1. Vendor

The vendor should be required to have an established plan in case of attack (see section 6.4).

Table 37 - Vendor Attack Management Recommendations

11.1.1	The vendor should be required to have an established plan in case of attack (see section 6.5).

11.2. Service Provider Internal

The service provider should have an established plan in case of attack including all responsibilities internal to the company and in coordination with the vendor (see section 6.4).

Table 38 - Service Provider Internal Attack Management Recommendations

11.2.1	The service provider should have an established plan in case of attack including all responsibilities internal to the company and in coordination with the vendor (see section 6.5).

11.3. Customer Notification

If an attack affects customer's privacy and/or security, a prepared statement should be released as soon as possible to customers by all means possible (e-mail, written mail, press releases, etc.) which includes mitigation plans and dates. A template for the prepared statement should be developed and reviewed by the cybersecurity, business, technical, customer relations, and legal departments of the service provider.

The service provider should have their cybersecurity, business, technical, customer relations, and legal department review the prepared statement prior to release with the understanding that each team will respond promptly. A single responsible person should be assigned for each team who is responsible for expediting the approval from their department.

Table 39 - Customer Notification Attack Management Recommendations

11.3.1	If an attack affects customer’s privacy and/or security, a prepared statement should be released as soon as possible to customers by all means possible (e-mail, written mail, press releases, etc.) which includes mitigation plans and dates.
11.3.2	A template for the prepared statement should be developed and reviewed by the cybersecurity, business, technical, customer relations, and legal departments of the service provider.
11.3.3	The service provider should have their cybersecurity, business, technical, customer relations, and legal department review the prepared statement prior to release with the understanding that each team will respond promptly.
11.3.4	A single responsible person should be assigned for each team who is responsible for expediting the approval from their department.

12. Conclusion

By following these recommended practices, and other industry standard cybersecurity practices, a service provider will be able to defer cybersecurity attacks from their network, and provide greater security and privacy to their customers.