EXAMPLE 1: The manufacturer publishes a completed version of table B.1 alongside the product description on their website.

EXAMPLE 2: The manufacturer completes table B.1 for internal record keeping. Sometime later, an external assurance organization assesses a product against the present document and requests information relating to the product's security design. The manufacturer can easily provide this information as it is contained within table B.1.

Cases where a provision is not applicable or not fulfilled by the consumer IoT device can include:

- when a device is a constrained device in such a way that implementation of certain security measures is not possible or not appropriate to the identified risk (security or privacy);
- where the functionality described in the provision is not included (e.g. a device that only presents data without requiring authentication).

EXAMPLE 3: A window sensor with a limited battery life sends alerts via a remote associated service when triggered and is controlled via a hub. Due to its limited battery life and processing power compared to other consumer IoT devices, it is a constrained device. In addition, because the user controls the device via a hub, the user does not need to use passwords, or other authentication mechanisms, to directly authenticate to the device.

5 Cyber security provisions for consumer IoT

5.1 No universal default passwords

Provision 5.1-1 Where passwords are used and in any state other than the factory default, all consumer IoT device passwords shall be unique per device or defined by the user.

NOTE: There are many mechanisms used for performing authentication, and passwords are not the only mechanism for authenticating a user to a device. However if they are used, following best practice on passwords is encouraged according to NIST Special Publication 800-63B [i.3]. Using passwords for machine to machine authentication is generally not appropriate.

Many consumer IoT devices are sold with universal default usernames and passwords (such as "admin, admin") for user interfaces through to network protocols. Continued usage of universal default values has been the source of many security issues in IoT [i.17] and the practice needs to be discontinued. The above provision can be achieved by the use of pre-installed passwords that are unique per device and/or by requiring the user to choose a password that follows best practice as part of initialization, or by some other method that does not use passwords.

EXAMPLE 1: During initialization a device generates certificates that are used to authenticate a user to the device via an associated service like a mobile application.

To increase security, multi-factor authentication, such as use of a password plus OTP procedure, can be used to better protect the device or an associated service. Device security can further be strengthened by having unique and immutable identities.

Provision 5.1-2 Where pre-installed unique per device passwords are used, these shall be generated with a mechanism that reduces the risk of automated attacks against a class or type of device.

EXAMPLE 2: Pre-installed passwords are sufficiently randomized.

As a counter-example, passwords with incremental counters (such as "password1", "password2" and so on) are easily guessable. Further, using a password that is related in an obvious way to public information (sent over the air or within a network), such as MAC address or Wi-Fi® SSID, can allow for password retrieval using automated means.

Provision 5.1-3 Authentication mechanisms used to authenticate users against a device shall use best practice cryptography, appropriate to the properties of the technology, risk and usage.

Provision 5.1-4 Where a user can authenticate against a device, the device shall provide to the user or an administrator a simple mechanism to change the authentication value used.

EXAMPLE 3: For biometric authentication values the device manufacturer allows this change in authentication value through retraining against a new biometric.

EXAMPLE 4: A parent in a household creates an account on the device for their child and selects and manages the PIN or password that the child uses. The parent is an administrator on the device and can restrict the child from changing the PIN or password.

EXAMPLE 5: To make it simple for the user to change a password, the manufacturer designs the password change process in a way that it requires a minimal number of steps. The manufacturer explains the process in a user manual and in a video tutorial.

An authentication mechanism used for authenticating users, whether it be a fingerprint, password or other token, needs to have its value changeable. This is easier when this mechanism is part of the normal usage flow of the device.

Provision 5.1-5 When the device is not a constrained device, it shall have a mechanism available which makes brute-force attacks on authentication mechanisms via network interfaces impracticable.

EXAMPLE 6: A device has a limitation on the number of authentication attempts within a certain time interval. It also uses increasing time intervals between attempts.

EXAMPLE 7: The client application is able to lock an account or to delay additional authentication attempts after a limited number of failed authentication attempts.

This provision addresses attacks that perform "credential stuffing" or exhaust an entire key-space. It is important that these types of attacks are detected by the consumer IoT device and defended against, whilst guarding against a related threat of "resource exhaustion" and denial of service attacks.

5.2 Implement a means to manage reports of vulnerabilities

Provision 5.2-1 The manufacturer shall make a vulnerability disclosure policy publicly available. This policy shall include, at a minimum:

- contact information for the reporting of issues; and
- information on timelines for:
    1) initial acknowledgement of receipt; and
    2) status updates until the resolution of the reported issues.

A vulnerability disclosure policy clearly specifies the process through which security researchers and others are able to report issues. Such policy can be updated as necessary to further ensure transparency and clarity in the dealings of the manufacturer with security researchers, and vice versa.

Coordinated Vulnerability Disclosure (CVD) is a set of processes for dealing with disclosures about potential security vulnerabilities and to support the remediation of these vulnerabilities. CVD is standardized by the International Organization for Standardization (ISO) in the ISO/IEC 29147 [i.4] on vulnerability disclosure and has been proven to be successful in some large software companies around the world.

In the IoT industry, CVD is currently not well-established [i.16] as some companies are reticent about dealing with security researchers. Here, CVD provides companies a framework to manage this process. This gives security researchers an avenue to inform companies of security issues, puts companies ahead of the threat of malicious exploitation and gives companies an opportunity to respond to and resolve vulnerabilities in advance of a public disclosure.

Provision 5.2-2 Disclosed vulnerabilities should be acted on in a timely manner.

A "timely manner" for acting on vulnerabilities varies considerably and is incident-specific; however, conventionally, the vulnerability process is completed within 90 days for a software solution, including availability of patches and notification of the issue. A hardware fix can take considerably longer to address than a software fix. Additionally, a fix that has to be deployed to devices can take time to roll out compared with a server software fix.

Provision 5.2-3 Manufacturers should continually monitor for, identify and rectify security vulnerabilities within products and services they sell, produce, have produced and services they operate during the defined support period.

NOTE 1: Manufacturers are expected to exercise due care for all software and hardware components used in the product, this includes due care related to the selected third parties that provide associated services to support the functions of the product.

Software solutions often contain open source and third party software components. Creating and maintaining list of all software components and their sub-components is a pre-requisite to be able to monitor for product vulnerabilities. Various tools exist to scan source code and binaries and build a so-called Software Bill of Materials (SBOM), which identifies third party components and the versions used in the product. This information is then used to monitor for the associated security and licensing risks of each identified software component.

Vulnerabilities are expected to be reported directly to the affected stakeholders in the first instance. If that is not possible, vulnerabilities can be reported to national authorities. Manufacturers are also encouraged to share information with competent industry bodies, such as the GSMA [i.21] and the IoT Security Foundation. Guidance on Coordinated Vulnerability Disclosure is available from the IoT Security Foundation [i.22] which references ISO/IEC 29147 [i.4].

This is expected to be performed for devices within their defined support period. However, manufacturers can continue this outside that period and release security updates to rectify vulnerabilities.

Manufacturers that provide IoT products have a duty of care to consumers and third parties who can be harmed by their failure to have a CVD programme in place. Additionally, companies that share this information through industry bodies can assist others who can be suffering from the same problem.

Disclosures can comprise different approaches depending on the circumstances:

- Vulnerabilities related to single products or services: the problem is expected to be reported directly to the affected stakeholder (usually the device manufacturer, IoT service provider or mobile application developer). The source of these reports can be security researchers or industry peers.
- Systemic vulnerabilities: a stakeholder, such as a device manufacturer, can discover a problem that is potentially systemic. Whilst fixing it in the device manufacturer's own product is crucial, there is significant benefit to industry and consumers from sharing this information. Similarly, security researchers can also seek to report such systemic vulnerabilities. For systemic vulnerabilities, a relevant competent industry body can coordinate a wider scale response.

NOTE 2: The Common Vulnerability Reporting Framework (CVRF) [i.5] can also be useful to exchange information on security vulnerabilities.

Cyber security threat information sharing can support organizations in developing and producing secure products according to ETSI TR 103 331 [i.6].

5.3 Keep software updated

Developing and deploying security updates in a timely manner is one of the most important actions a manufacturer can take to protect its customers and the wider technical ecosystem. It is good practice that all software is kept updated and well maintained.

Each provision from 5.3-3 to 5.3-12 is dependent upon an update mechanism being implemented, as per provision 5.3-1 or 5.3-2.

Provision 5.3-1 All software components in consumer IoT devices should be securely updateable.

NOTE 1: Managing software updates successfully generally relies on communication of version information for software components between the device and the manufacturer.

Not all software on a device will be updateable.

EXAMPLE 1: The first stage boot loader on a device is written once to device storage and from then on is immutable.

EXAMPLE 2: On devices with several microcontrollers (e.g. one for communication and one for the application) some of them might not be updateable.

Provision 5.3-2 When the device is not a constrained device, it shall have an update mechanism for the secure installation of updates.

NOTE 2: There are cases where provision 5.3-1 applies even where 5.3-2 does not.

"Securely updateable" and "secure installation" means that there are adequate measures to prevent an attacker misusing the update mechanism.

EXAMPLE 3: Measures can include the use of authentic software update servers, integrity protected communications channels, verifying the authenticity and integrity of software updates. It is recognized that there are great variances in software update mechanisms and what constitutes "installation".

EXAMPLE 4: An anti-rollback policy based on version checking can be used to prevent downgrade attacks.

Update mechanisms can range from the device downloading the update directly from a remote server, transmitted from a mobile application or transferred over a USB or other physical interface. If an attacker compromises this mechanism, it allows for a malicious version of the software to be installed on the device.

Provision 5.3-3 An update shall be simple for the user to apply.

The degree of simplicity depends on the design and intended usage of the device. An update that is simple to apply will be automatically applied, initiated using an associated service (such as a mobile application), or via a web interface on the device. If an update is difficult to apply, then that increases the chance that a user will repeatedly defer updating the device, leaving it in a vulnerable state.

Provision 5.3-4 Automatic mechanisms should be used for software updates.

If an automatic update fails, then a user can, in some circumstances, no longer be able to use a device. Detection mechanisms such as watchdogs and the use of dual-bank flash or recovery partitions can ensure that the device returns to either a known good version or the factory state.

Security updates can be provided for devices in a preventative manner, as part of automatic updates, which can remove security vulnerabilities before they are exploited. Managing this can be complex, especially if there are parallel associated service updates, device updates and other service updates to deal with. Therefore, a clear management and deployment plan is beneficial to the manufacturer, as is transparency to consumers about the current state of update support.

In many cases, publishing software updates involves multiple dependencies on other organizations such as manufacturers that produce sub-components; however, this is not a reason to withhold updates. It can be useful for the manufacturer to consider the entire software supply chain in the development and deployment of security updates.

It is often advisable not to bundle security updates with more complex software updates, such as feature updates. A feature update that introduces new functionality can trigger additional requirements and delay delivery of the update to devices.

EXAMPLE 5: Under the EU Product Legislation, a feature update could change the intended use of a device and thus turn it into a new product, requiring a new conformity assessment to be conducted. However, a software update with limited impact could be considered a maintenance update which would not require a new conformity assessment. More information on the impact of software updates in the context of the EU Product Legislation can be found in the Blue Guide [i.13].

Provision 5.3-5 The device should check after initialization, and then periodically, whether security updates are available.

EXAMPLE 6: The user could be shown the existence of updates via the interface with which the device is initialized.

EXAMPLE 7: A device checks for available updates daily at a randomized time.

For some products, it can be more appropriate for the associated service, rather than the device, to perform such checks.

Provision 5.3-6 If the device supports automatic updates and/or update notifications, these should be enabled in the initialized state and configurable so that the user can enable, disable, or postpone installation of security updates and/or update notifications.

It is important from a consumer rights and ownership perspective that the user is in control of whether or not they receive updates. There are good reasons why a user may choose not to update, including security. In addition, if an update is deployed and subsequently found to cause issues, manufacturers can ask users to not upgrade their software in order that those devices are not affected.

Provision 5.3-7 The device shall use best practice cryptography to facilitate secure update mechanisms.

Provision 5.3-8 Security updates shall be timely.

"Timely" in the context of security updates can vary, depending on the particular issue and fix, as well as other factors such as the ability to reach a device or constrained device considerations. It is important that a security update that fixes a critical vulnerability (i.e. one with potentially adverse effects of a large scale) is handled with appropriate priority by the manufacturer. Due to the complex structure of modern software and the ubiquity of communication platforms, multiple stakeholders can be involved in a security update.

EXAMPLE 8: A particular software update involves a third party vendor of software libraries, an IoT device manufacturer, and an IoT service platform operator. Collaboration between these stakeholders ensures appropriate timeliness of the software update.

Provision 5.3-9 The device should verify the authenticity and integrity of software updates.

A common approach for confirming that an update is valid is to verify its integrity and authenticity. This can be done on the device; however, constrained devices can have power limitations that make performing cryptographic operations costly. In such cases, verification can be performed by another device that is trusted to perform this verification. The verified update would then be sent over a secure channel to the device. Performing verification of updates at a hub and then on the device, can reduce the risk of compromise.

It is good practice for a device to act upon the detection of an invalid and potentially malicious update. Beyond rejecting the update, and without limitation, it can report the incident to an appropriate service and/or inform the user. In addition, mitigating controls can be put in place to prevent an attacker from bypassing or misusing an update mechanism. Giving the attacker as little information as possible as part of the update mechanism reduces their ability to exploit it.

EXAMPLE 9: When a device detects that an update could not be delivered or applied successfully (by failing integrity or authentication checks), the device can mitigate information leakage by not providing any information about the failure to the initiator of the update process. However, the device can generate a log entry and deliver notification of the log entry to a trusted peer (e.g. a device administrator) over a secure channel, so that the occurrence of the incident is known and the owner or administrator of the device can make an appropriate response.

Provision 5.3-10 Where updates are delivered over a network interface, the device shall verify the authenticity and integrity of each update via a trust relationship.

NOTE 3: Valid trust relationships include: authenticated communication channels, presence on a network that requires the device to possess a critical security parameter or password to join, digital signature based verification of the update, or confirmation by the user.

NOTE 4: The validation of the trust relationship is essential to ensure that a non-authorized entity (e.g. device management platform or device) cannot install malicious code.

Provision 5.3-11 The manufacturer should inform the user in a recognizable and apparent manner that a security update is required together with information on the risks mitigated by that update.

EXAMPLE 10: The manufacturer informs the user that an update is required via a notification on the user interface or via an email.

Provision 5.3-12 The device should notify the user when the application of a software update will disrupt the basic functioning of the device.

NOTE 5: This is not necessary if a notification is made by an associated service.

This notification can include extra detail, such as the approximate expected duration for which the device will be offline.

EXAMPLE 11: A notification includes information about the urgency and approximate expected duration of downtime.

It can be critical for users that a device continues to operate during an update. This is why the provision above recommends to notify the user when an update will disrupt functionality where possible. Particularly, devices that fulfil a safety-relevant function are expected not to turn completely off in the case of an update; some minimal system functional capability is expected. Disruption to functionality can become a critical safety issue for some types of devices and systems if not considered or managed correctly.

EXAMPLE 12: During an update, a watch will continue to display the time, a home thermostat will continue to maintain a reasonable temperature and a Smart Lock will continue to lock and unlock a door.

Provision 5.3-13 The manufacturer shall publish, in an accessible way that is clear and transparent to the user, the defined support period.

When purchasing a product, the consumer expects this period of software update support to be made clear.

Provision 5.3-14 For constrained devices that cannot have their software updated, the rationale for the absence of software updates, the period and method of hardware replacement support and a defined support period should be published by the manufacturer in an accessible way that is clear and transparent to the user.

Provision 5.3-15 For constrained devices that cannot have their software updated, the product should be isolable and the hardware replaceable.

There are some situations where devices cannot be patched. For constrained devices a replacement plan needs to be in place and be clearly communicated to the consumer. This plan would typically detail a schedule for when technologies will need to be replaced and, where applicable, when support for hardware and software ends.

Provision 5.3-16 The model designation of the consumer IoT device shall be clearly recognizable, either by labelling on the device or via a physical interface.

This is often performed by communicating with a device over a logical interface, however it can also be part of a UI.

EXAMPLE 13: A device has a HTTP (or HTTPS when appropriate) API that reports the model designation (after user authentication).

Knowledge of the specific designation of the device is often required to check the defined support period of software updates or the availability of software updates.

5.4 Securely store sensitive security parameters

Provision 5.4-1 Sensitive security parameters in persistent storage shall be stored securely by the device.

Secure storage mechanisms can be used to secure sensitive security parameters. Appropriate mechanisms include those provided by a Trusted Execution Environment (TEE), encrypted storage associated with the hardware, Secure Elements (SE) or Dedicated Security Components (DSC), and processing capabilities of software running on a UICC, according to ETSI TR 121 905 [i.29], ETSI TS 102 221 [i.25]/embedded UICC according to GSMA SGP.22 Technical Specification v2.2.1 [i.26].

NOTE: This provision applies to persistent storage, but manufacturers can also implement similar approaches for sensitive security parameters in memory.

EXAMPLE 1: The root keys involved in authorization and access to licensed radio frequencies (e.g. LTE-m cellular access) are stored in a UICC.

EXAMPLE 2: A remote controlled door-lock using a Trusted Execution Environment (TEE) to store and access the sensitive security parameters.

EXAMPLE 3: A wireless thermostat stores the credentials for the wireless network in a tamper protected microcontroller rather than in external flash storage.

Provision 5.4-2 Where a hard-coded unique per device identity is used in a device for security purposes, it shall be implemented in such a way that it resists tampering by means such as physical, electrical or software.

EXAMPLE 4: A master key used for network access that is unique to the device is stored in UICC which is compliant to relevant ETSI standards (see, for example ETSI TS 102 221 [i.25]).

Provision 5.4-3 Hard-coded critical security parameters in device software source code shall not be used.

Reverse engineering of devices and applications can easily discover credentials such as hard-coded usernames and passwords in software. These credentials can also be API keys that allow usage of security-sensitive functions in a remote service, or private keys used in the security of protocols that the device uses to communicate. Such credentials will often be found within source-code, which is well-known bad practice. Simple obfuscation methods also used to obscure or encrypt this hard-coded information can be trivially broken.

Provision 5.4-4 Any critical security parameters used for integrity and authenticity checks of software updates and for protection of communication with associated services in device software shall be unique per device and shall be produced with a mechanism that reduces the risk of automated attacks against classes of devices.

EXAMPLE 5: A different symmetric key is deployed on every device of the same product class for generating and verifying message authentication codes for software updates.

EXAMPLE 6: The device uses the manufacturer's public key to verify a software update. This is not a critical security parameter and does not need to be unique per device.

Provisioning a device with unique critical security parameters helps to protect the integrity and authenticity of software updates as well as the communication of the device with associated services. If global critical security parameters are used, their disclosure can enable wide-scale attacks on other IoT devices such as to enable the creation of botnets.

5.5 Communicate securely

Provision 5.5-1 The consumer IoT device shall use best practice cryptography to communicate securely.

Appropriateness of security controls and the use of best practice cryptography is dependent on many factors including the usage context. As security is ever-evolving it is difficult to give prescriptive advice about cryptography or other security measures without the risk of such advice quickly becoming obsolete.

Provision 5.5-2 The consumer IoT device should use reviewed or evaluated implementations to deliver network and security functionalities, particularly in the field of cryptography.

Reviews and evaluations can involve an independent internal or external entity.

EXAMPLE 1: Distributed software libraries within the development and test community, certified software modules, and hardware equipment crypto-service providers (such as the Secure Element and Trust Execution Environment) are all reviewed or evaluated.

Provision 5.5-3 Cryptographic algorithms and primitives should be updateable.

NOTE 1: This is also known as "cryptoagility".

For devices that cannot be updated, it is important that the intended lifetime of the device does not exceed the recommended usage lifetime of cryptographic algorithms used by the device (including key sizes).

Provision 5.5-4 Access to device functionality via a network interface in the initialized state should only be possible after authentication on that interface.

NOTE 2: Functionality can vary significantly on the use case and can encompass a range of things, including access to personal data and device actuators.

There are devices that provide public, open data for example in the Web of Things [i.18]. These devices are accessible without authentication to provide open access to all.

The device can be compromised via vulnerabilities in network services. A suitable authentication mechanism can protect against unauthorized access and can contribute to defence-in-depth in the device.

Provision 5.5-5 Device functionality that allows security-relevant changes in configuration via a network interface shall only be accessible after authentication. The exception is for network service protocols that are relied upon by the device and where the manufacturer cannot guarantee what configuration will be required for the device to operate.

NOTE 3: Protocols that are an exception include ARP, DHCP, DNS, ICMP, and NTP.

EXAMPLE 2: Security-relevant changes include permission management, configuration of network keys and password changes.

Provision 5.5-6 Critical security parameters should be encrypted in transit, with such encryption appropriate to the properties of the technology, risk and usage.

Provision 5.5-7 The consumer IoT device shall protect the confidentiality of critical security parameters that are communicated via remotely accessible network interfaces.

Many different methods exist for enrolment and authentication. Some authentication values are provided by out-of-band authentication mechanisms, such as a QR code, and some are human-readable, such as a password.

Where an authentication mechanism uses unique values per authentication attempt (e.g. in a challenge-response mechanism or when using one time passwords as a second factor), the response is not the authentication value itself. However, it is still good practice to apply confidentiality protection to those values.

Confidentiality protection can be achieved using an encrypted communication channel or payload encryption. This is often done using protocols or algorithms at least as strong as the key material transmitted, however other mitigations, such as the need for close proximity, are available.

Provision 5.5-8 The manufacturer shall follow secure management processes for critical security parameters that relate to the device.

The use of open, peer-reviewed standards for critical security parameters (commonly referred to as "key management") is strongly encouraged.

5.6 Minimize exposed attack surfaces

The "principle of least privilege" is a foundation stone of good security engineering, applicable to IoT as much as in any other field of application.

Provision 5.6-1 All unused network and logical interfaces shall be disabled.

EXAMPLE 1: An administrative UI that is supposed to be accessed from the LAN is not accessible from the WAN by default.

EXAMPLE 2: A Direct Firmware Update (DFU) service exposed over Bluetooth® Low Energy is used for development but not expected to be used in production. It is disabled in the final product.

Provision 5.6-2 In the initialized state, the network interfaces of the device shall minimize the unauthenticated disclosure of security-relevant information.

Security-relevant information can be exposed over a network interface as part of the initialization process. When security-relevant information is shared by a device when establishing a connection, it can be used by attackers to identify vulnerable devices.

EXAMPLE 3: When finding vulnerable devices throughout the whole IP address space, security-relevant information could be information about the device configuration, kernel version or software version.

Provision 5.6-3 Device hardware should not unnecessarily expose physical interfaces to attack.

Physical interfaces can be used by an attacker to compromise firmware or memory on a device. "Unnecessarily" refers to the manufacturer's assessment of the benefits of an open interface, used for user functionality or for debugging purposes.

EXAMPLE 4: A micro-USB port meant to be used to power the device only is physically configured so as not to also allow command or debug operations.

Provision 5.6-4 Where a debug interface is physically accessible, it shall be disabled in software.

EXAMPLE 5: A UART serial interface is disabled through the bootloader software on the device. No logon prompt and no interactive menu is available due to this disabling.

Provision 5.6-5 The manufacturer should only enable software services that are used or required for the intended use or operation of the device.

EXAMPLE 6: The manufacturer does not provision the device with any background processes, kernel extensions, commands, programs or tools that are not required for the intended use.

Provision 5.6-6 Code should be minimized to the functionality necessary for the service/device to operate.

EXAMPLE 7: "Dead" or unused code is removed and not considered to be benign.

Provision 5.6-7 Software should run with least necessary privileges, taking account of both security and functionality.

EXAMPLE 8: Minimal daemons/processes run with "root" privileges. In particular the processes that use network interfaces require unprivileged users rather than requiring a "root" user.

EXAMPLE 9: Applications running on a device that includes a multi-user operating system (e.g. Linux®) use different users for each component or service.

Software attacks on devices that aim to corrupt memory can be mitigated through mechanisms such as stack canaries, Address Space Layout Randomization (ASLR). The manufacturer can use platform security features where they are available to help further reduce the risk. Reducing privileges that they run at and minimizing code also helps to mitigate this risk.

Provision 5.6-8 The device should include a hardware-level access control mechanism for memory.

Software exploits often use the lack of access control in memory to execute malicious code. Access control mechanisms limit whether data in memory on the device can be executed. Suitable mechanisms include technologies such as MMUs or MPUs, executable space protection (e.g. NX bits), memory tagging, and trusted execution environments.

Provision 5.6-9 The manufacturer should follow secure development processes for software deployed on the device.

Secure development processes, including using version control, or enabling security-related compiler options (e.g. stack protection) can help ensure software artefacts are more secure. Manufacturers can use these options when using toolchains that support them.

5.7 Ensure software integrity

Provision 5.7-1 The consumer IoT device should verify its software using secure boot mechanisms.

A hardware root of trust is one way to provide strong attestation as part of a secure boot mechanism. A hardware root of trust is a component of a system from which all other components derive their "trust" - i.e. the source of cryptographic trust within that system. To fulfil its function, the hardware root of trust is reliable and resistant to both physical and logical tampering, as there is no mechanism to determine that the component has failed or been altered. By utilizing a hardware root of trust, a device can have confidence in results of cryptographic functions, such as those utilized for secure boot. A hardware root of trust can be either backed by mechanisms used for secure storage of credentials or other alternatives providing baseline levels of security assurance proportionate to the required level of security for a given device.

Provision 5.7-2 If an unauthorized change is detected to the software, the device should alert the user and/or administrator to the issue and should not connect to wider networks than those necessary to perform the alerting function.

The ability to recover remotely from unauthorized changes can rely on a known good state, such as locally storing a known good version to enable safe recovery and updating of the device. This will avoid denial of service and costly recalls or maintenance visits, whilst managing the risk of potential takeover of the device by an attacker subverting update or other network communications mechanisms.

If a consumer IoT device detects an unauthorized change to its software, it will be able to inform the right stakeholder. In some cases, devices can have the ability to be in administration mode.

EXAMPLE: A thermostat in a room can have a user mode; this mode prevents changing of other settings. If an unauthorized change to software is detected, an alert to the administrator is appropriate, as the administrator has the ability to act on the alert (whereas a user does not).

NOTE: An attack that forces a device to revert to a known good state can introduce a DoS risk if the device is unable to successfully perform this or if the attacker is able to repeatedly cause this effect.

5.8 Ensure that personal data is secure

Provision 5.8-1 The confidentiality of personal data transiting between a device and a service, especially associated services, should be protected, with best practice cryptography.

Provision 5.8-2 The confidentiality of sensitive personal data communicated between the device and associated services shall be protected, with cryptography appropriate to the properties of the technology and usage.

NOTE 1: In the context of this provision, "sensitive personal data" is data whose disclosure has a high potential to cause harm to the individual. What is to be treated as "sensitive personal data" varies across products and use cases, but examples are: video stream of a home security camera, payment information, content of communication data and timestamped location data. Carrying out security and data protection impact assessments can help the manufacturer make appropriate choices.

NOTE 2: Associated services in this context are typically cloud services. Moreover these services are controlled or can be influenced by the manufacturer. These services typically are not operated by the user.

NOTE 3: Confidentiality protection often includes integrity protection according to best practice cryptography.

Provision 5.8-3 All external sensing capabilities of the device shall be documented in an accessible way that is clear and transparent for the user.

EXAMPLE: An external sensing capability can be an optic or acoustic sensor.

Clause 6 of the present document contains provisions specific to protecting personal data.

5.9 Make systems resilient to outages

The aim of the provisions in the present clause is to ensure that IoT services are kept up and running as the adoption of IoT devices across all aspects of a consumer's life increases, including in functions that are relevant to personal safety. It is important to note that safety-related regulations can apply, but the key is to avoid making outages the cause of impact on the user and to design products and services that provide a level of resilience to these challenges.

Provision 5.9-1 Resilience should be built in to consumer IoT devices and services, taking into account the possibility of outages of data networks and power.

Provision 5.9-2 Consumer IoT devices should remain operating and locally functional in the case of a loss of network access and should recover cleanly in the case of restoration of a loss of power.

NOTE: "Recovering cleanly" normally involves resuming connectivity and functionality in the same or improved state.

Provision 5.9-3 The consumer IoT device should connect to networks in an expected, operational and stable state and in an orderly fashion, taking the capability of the infrastructure into consideration.

EXAMPLE 1: A Smart Home loses connection to the internet following a power outage. When the network connection is restored, the devices in the home reconnect after a randomized delay to minimize network utilization.

EXAMPLE 2: After making an update available, the manufacturer notifies devices in batches to prevent them all simultaneously downloading the update.

IoT systems and devices are relied upon by consumers for increasingly important use cases that can be safety-relevant or life-impacting. Keeping services running locally if there is a loss of network is one of the measures that can be taken to increase resilience. Other measures can include building redundancy into associated services as well as mitigations against Distributed Denial of Service (DDoS) attacks or signalling storms, which can be caused by mass-reconnections of devices following an outage. It is expected that the level of resilience necessary is proportionate and determined by usage, with consideration given to others that rely on the system, service or device given that an outage can have a wider impact than expected.

Orderly reconnection means in a manner that takes explicit steps to avoid simultaneous requests, such as for software updates or reconnections, from a large number of IoT devices. Such explicit steps can include the introduction of a random delay before a reconnection attempt according to an incremental back-off mechanism.

5.10 Examine system telemetry data

Provision 5.10-1 If telemetry data is collected from consumer IoT devices and services, such as usage and measurement data, it should be examined for security anomalies.

EXAMPLE 1: Security anomalies can be represented by a deviation from normal behaviour of the device, as expressed by the monitored indicators, for example an abnormal increase of failed login attempts.

EXAMPLE 2: Telemetry from multiple devices allows a manufacturer to notice that updates are failing due to invalid software update authenticity checks.

Examining telemetry, including log data, is useful for security evaluation and allows for unusual circumstances to be identified early and dealt with, minimizing security risk and allowing quick mitigation of problems.

Clause 6 of the present document contains provisions specific to protecting personal data when telemetry data is collected.

5.11 Make it easy for users to delete user data

Provision 5.11-1 The user shall be provided with functionality such that user data can be erased from the device in a simple manner.

NOTE 1: User data in this context means all individual data which is stored on the IoT device including personal data, user configuration and cryptographic material such as user passwords or keys.

Provision 5.11-2 The consumer should be provided with functionality on the device such that personal data can be removed from associated services in a simple manner.

Such functionality is intended for situations when there is a transfer of ownership, when the consumer wishes to delete personal data, when the consumer wishes to remove a service from the device and/or when the consumer wishes to dispose of the device. It is expected that such functionality is compliant to applicable data protection law, including the GDPR [i.7].

Removing personal data "easily" means that minimal steps are required to complete that action that each involve minimal complexity.

Such functionality can potentially present an attack vector.

Provision 5.11-3 Users should be given clear instructions on how to delete their personal data.

Provision 5.11-4 Users should be provided with clear confirmation that personal data has been deleted from services, devices and applications.

Consumer IoT devices often change ownership and will eventually be recycled or disposed of. Mechanisms can be provided that allow the consumer to remain in control and remove personal data from services, devices and applications. When a consumer wishes to completely remove their personal data, they also expect retrospective deletion of backup copies.

Deleting personal data from a device or service is often not simply achieved by resetting a device back to its factory default state. There are many use cases where the consumer is not the owner of a device, but wishes to delete their own personal data from the device and all associated services such as cloud services or mobile applications.

EXAMPLE: A user can have temporary usage of consumer IoT products within a rented apartment. Carrying out a factory reset of the product can remove configuration settings or disable the device to the detriment of the apartment owner and a future user. A factory reset, deleting all data from the IoT device, would not be an appropriate way to delete the personal data of a single user in a shared use context such as this.

NOTE 2: Annex A of the present document contains an example model of device states including data storage for each state.

5.12 Make installation and maintenance of devices easy

Provision 5.12-1 Installation and maintenance of consumer IoT should involve minimal decisions by the user and should follow security best practice on usability.

EXAMPLE: The user uses a wizard to setup the device where a subset of configuration options is presented with the common defaults already specified and with appropriate security options already turned on by default.

Provision 5.12-2 The manufacturer should provide users with guidance on how to securely set up their device.

However, the ideal is for a process that involves the minimum of human intervention and which achieves a secure configuration automatically.

Provision 5.12-3 The manufacturer should provide users with guidance on how to check whether their device is securely set up.

Security issues caused by consumer confusion or misconfiguration can be reduced and sometimes eliminated by properly addressing complexity and poor design in user interfaces. Clear guidance to users on how to configure devices securely can also reduce their exposure to threats.

In the general case, the average overhead of securely setting up a device is higher than the average overhead of checking whether a device is securely setup. The check of a secure setup, from a process standpoint, can be undertaken in large part by the manufacturer through an automated process that communicates with the device remotely. Part of such an automated process could include validation of the device's capacity to establish a secure communication channel.

5.13 Validate input data

Provision 5.13-1 The consumer IoT device software shall validate data input via user interfaces or transferred via Application Programming Interfaces (APIs) or between networks in services and devices.

Systems can be subverted by incorrectly formatted data or code transferred across different types of interface. Automated tools such as fuzzers can be used by attackers or testers to exploit potential gaps and weaknesses that emerge as a result of not validating data.

EXAMPLE 1: The device receives data that is not of the expected type, for example executable code rather than user inputted text. The software on the device has been written so that the input is parameterized or "escaped", preventing this code from being run.

EXAMPLE 2: Out of range data is received by a temperature sensor, rather than trying to process this input it identifies that it is outside of the possible bounds and is discarded and the event is captured in telemetry.

6 Data protection provisions for consumer IoT

Many consumer IoT devices process personal data. It is expected that manufacturers provide features within consumer IoT devices that support the protection of such personal data. In addition, there exist laws and regulations that relate to the protection of personal data in consumer IoT devices (for example the GDPR [i.7]). The present document intends to help manufacturers of consumer IoT devices provide a number of features for the protection of personal data from a strictly technical perspective.

Provision 6-1 The manufacturer shall provide consumers with clear and transparent information about what personal data is processed, how it is being used, by whom, and for what purposes, for each device and service. This also applies to third parties that can be involved, including advertisers.

Provision 6-2 Where personal data is processed on the basis of consumers' consent, this consent shall be obtained in a valid way.

Obtaining consent "in a valid way" normally involves giving consumers a free, obvious and explicit opt-in choice of whether their personal data can be used for a specified purpose.

Provision 6-3 Consumers who gave consent for the processing of their personal data shall have the capability to withdraw it at any time.

Consumers expect to be able to preserve their privacy by configuring IoT device and service functionality appropriately.

Provision 6-4 If telemetry data is collected from consumer IoT devices and services, the processing of personal data should be kept to the minimum necessary for the intended functionality.

Provision 6-5 If telemetry data is collected from consumer IoT devices and services, consumers shall be provided with information on what telemetry data is collected, how it is being used, by whom, and for what purposes.