UL MCV 1376

Methodology for Marketing Claim Verification: Security Capabilities to Levels Bronze/Silver/Gold/Platinum/Diamond

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

2021-09-23

UL MCV 1376

Methodology for Marketing Claim Verification: Security Capabilities to Levels Bronze/Silver/Gold/Platinum/Diamond

Second Edition

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

1 Introduction

As the world is becoming more connected, we are seeing an enormous number of network-connected products hitting the market. However, it has been found that a significant amount of these products lack even basic and fundamental security capabilities. For this reason, governments have started introducing regulations to require connected products to be more secure, and accompanying standards and guidelines are being introduced by the industry to help manufacturers address these new challenges.

UL MCV 1376 is a baseline-driven security verification framework that groups sets of industry-referenced security best practices into 5 different tiers (“levels”) based on their necessity for implementation. Level 1 references best practices that are considered an absolute minimum (a “baseline”) for any connected device, followed by 4 more levels of increasingly expanding sets of industry-acknowledged security capabilities that become more advanced and comprehensive in nature. UL MCV 1376 references and/or maps to various industry-leading security frameworks, such as EN 303 645, NISTIR 8259A, the CSDE C2 Consensus Report.

UL MCV 1376 provides manufacturers/developers of IoT products a means to measure the security maturity of their product, prioritize the implementation of device security capabilities, and plan for product security maturity growth. Additionally, it provides buyers/end users of IoT solutions the opportunity to formulate concise security requirements in line with industry best practices.

The UL MCV 1376 framework serves as the inspiration and backbone of several industry security verification solutions, such as UL's IoT Security Rating, retailers' connected devices security test protocols, and it is accepted by the Design Lights Consortium (DLC) as a security assessment framework for networked lighting controls.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

2 Scope of Applicability

This framework focuses primarily on device-centric security capabilities and can be used by all kinds of IoT implementations, but particularly those targeting consumer use and/or basic commercial and/or basic industrial use. Applicable product categories include, but are not limited to, the following:

- Commercial HVAC
- Commercial/industrial lighting systems
- Connected baby monitors
- Connected children's toys
- Gateways and hub devices
- Home automation devices
- Home entertainment devices (e.g. smart TV, smart speaker)
- (Home) Security devices (e.g. smart door locks, security cameras)
- Smart home appliances
- Wearables

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

3 Table of Contents

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

4 Acronyms

0RTT	Zero Round-Trip Time
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CCTV	Closed-circuit Television
CRP	Code Readout Protection
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
CSRNG	Cryptographically Secure Random Number Generator
CSV	Comma Separated Values
CTR	Counter Mode
CVSS	Common Vulnerability Scoring System
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSA	Digital Signature Algorithm
EAX	Encrypt-then-Authenticate-then-Translate
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HMAC	Hash-based Message Authentication Code
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

IoT	Internet of Things
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
MAC	Message Authentication Code
MCV	Methodology of Marketing Claim Verification
MIC	Message Integrity Code
MLO	Minimal Loader
MQTT	Message Queueing Telemetry Transport
NTP	Network Time Protocol
OEM	Original Equipment Manufacturer
PBKDF2	Password-Based Key Derivation Function 2
PFS	Perfect Forward Secrecy
PII	Personally Identifiable Information
PKI	Public Key Infrastructure
RAM	Random Access Memory
RSA	Rivest-Shamir-Adleman
RTSP	Real-time Streaming Protocol
RTSPS	Real-time Streaming Protocol Secure
SHA	Secure Hash Algorithm
SQL	Structured Query Language
TCP	Transmission Control Protocol
TDES	Triple DES
TLS	Transport Layer Security
TPM	Trusted Platform Module
USB	Universal Serial Bus
WEP	Wired Equivalent Privacy
WPA2	Wi-Fi Protected Access 2
WPS	Wi-Fi Protected Setup
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

5 Summary of Requirements

The following table gives a summary of the UL MCV 1376 requirements broken down into their respective categories. For clarification, the following symbols will be used:

Symbol	Meaning
✓	Implementation of base requirement mandatory
✓+	Implementation of base requirement and one requirement enhancement mandatory
✓++	Implementation of base requirement and two requirement enhancements mandatory
✓+++	Implementation of base requirement and three requirement enhancements mandatory
✓++++	Implementation of base requirement and four requirement enhancements mandatory

Topic	Requirement	L1	L2	L3	L4	L5
Software Update	SWU-SUPP: Remote software updates supported	✓	✓+	✓+	✓++	✓++
	SWU-AUTO: Automatic Software Update Tracking	✓	✓	✓	✓+	✓+
	SWU-AUTH: Software Update Authentication	✓	✓+	✓+	✓++	✓++
	SWU-ROT: Hardware root of trust					✓
Data & Cryptography	DC-NDK: No default credentials or secret keys	✓	✓	✓	✓	✓
	DC-PROT: Protect sensitive data	✓	✓	✓+	✓++	✓++
	DC-PWD: Passphrase complexity enforcement		✓	✓	✓+	✓+
	DC-RECVR: Credential recovery		✓	✓	✓+	✓+
	DC-TRNG: Cryptographically strong random number generation			✓	✓	✓

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

Topic	Requirement	L1	L2	L3	L4	L5
Logical Security	LS-DBG: Disable debug interfaces			✓	✓+	✓+
	LS-SECDEF: Systems configured to secure defaults			✓	✓	✓
	LS-FDIS: Unwanted functionality can be disabled			✓	✓	✓
	LS-LPP: Least Privilege Principle			✓	✓	✓
	LS-MPF: Memory Protection Features				✓	✓
	LS-KVUL: Software free from known vulnerabilities				✓	✓
	LS-LOGS: Logs or errors do not expose sensitive data				✓	✓
	LS-UVUL: Software tested for unknown vulnerabilities					✓
System Management	SM-AUTH: Sensitive services require authentication	✓	✓	✓	✓+	✓+
	SM-ERASE: Permanent erasure of sensitive data	✓	✓	✓	✓	✓
	SM-SFTY: Manual override for safety-critical operations		✓	✓	✓	✓
	SM-INPT: Input validation and sanitization			✓	✓	✓
	SM-SESS: Sensitive services implement session management				✓	✓
Communication Security	CS-XMIT: Cryptographically Secure Data Transmission	✓	✓+	✓++	✓++	✓++
	CS-RFXMIT: Wireless Communication Security	✓	✓+	✓++	✓+++	✓+++

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

Topic	Requirement	L1	L2	L3	L4	L5
Process & Documentation	PD-DEVID: Product Identification	✓	✓	✓	✓	✓
	PD-COLL: Data Collection and Handling	✓	✓+	✓++	✓+++	✓++++
	PD-PROCS: Documented patch/update process	✓	✓	✓	✓+	✓+
	PD-EOL: End-of-life policy	✓	✓	✓	✓	✓
	PD-VMGMT: Vulnerability management program	✓	✓	✓+	✓+	✓++
	PD-CLMON: Regularly monitored cloud/app environment(s)			✓	✓	✓
	PD-SBOM: Software Bill-of-Materials				✓	✓
	PD-PHYIF: Physical Interface Documentation					✓
	PD-SDOC: All services documented					✓

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6 List of Requirements

6.1 Software Update

6.1.1 SWU-SUPP: Remote software updates supported

Software updates must be supported, using network or wireless interfaces where available

Base requirement: L1–L5

No matter how well the software is designed or tested, there will always be bugs and vulnerabilities that are missed. This is just a fact of software development and the sheer complexity of any body of code. So, the update of the software must be allowed in any device to ensure that it can be patched when any such bugs are found. It is an additional requirement that the software update must be able to be performed across a wireless or network interface, should the device provide such an interface. Any patches need to be either automatically remotely triggered (e.g., via a cloud backend connection) or the user itself must be able to update the firmware of the device. An update mechanism that can only be performed by the device vendor or by using specialized equipment is not considered fulfilling this requirement.

This way, it is ensured that security updates are possible and user-friendly, minimizing the risk that devices become permanently vulnerable through new attack methods that are discovered after the initial evaluation/shipping of the device.

For devices that fulfill level 1 at least, the applications must be updatable.

Requirement enhancement 1: L2–L5

For level 2 and up, additionally, the operating system and firmware must be updatable.

Requirement enhancement 2: L4, L5

For level 4 and up, also second stage bootloader components (i.e., those that are software-updatable as e.g., U-Boot) must be updatable. Bootloaders/MLOs that are in hardware are excluded.

6.1.2 SWU-AUTO: Automatic Software Update Tracking

Automatic querying of devices for available software updates must be enabled by default

Base requirement: L1–L5

Although software must be maintained to ensure ongoing patching of security vulnerabilities, it is reasonable to expect that customers may not always know about the latest vulnerability in a device. Therefore, users must be updated on such occurrences to be able to make an informed decision so that they do not become a blocking factor in patching a high-risk vulnerability because they are unaware of it. In any case, even if the user declines to install a software update, there must be a mechanism for the user to change their mind later on and trigger the installation of the available update. Subsequently published updates should lead to new notifications on the user side. It is expected that device vendors will need to consider the business/operational logic of the device when implementing such updates so that a system does not reset during operation, but such business logic is considered outside of the scope of this assessment. Also note that while an individual package might have a vulnerability with a high CVSS score, the relevant metric for the following cases is the CVSS score as it applies to the overall system. For example, a vulnerability that is critical but does not impact the system at all can be safely disregarded. Such happens frequently in practice when a vulnerability is discovered that might only apply to specific architectures or constellations that the affected component is used in.

In all cases, for discovered vulnerabilities rated to CVSS 7 or higher a patch must be made available within 30 days, for those rated between CVSS 4 to 7 a patch must be made available within 90 days, and vulnerabilities rated to less than CVSS 4 may be left unpatched.

When the device under test has some form of HMI (which could also be in form of a smartphone app that controls the device or in form of a management web interface), this requirement is considered fulfilled if the device and/or paired HMI component automatically checks for available software updates, informs the user of such availability of new software, and prompts the user to decide if they would like that software update to be installed.

These timelines refer to the case in which a zero-day vulnerability is present; they may be longer up to a period of 90 days from the day of reporting when the vendor is informed via a responsible disclosure process.

If the vendor elects to always force-push the latest firmware version onto all devices remotely, this requirement is also considered fulfilled.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

Requirement enhancement 1: L4, L5

For devices that fulfill security levels 4 and 5, additionally, a method shall be available for the vendor to force a software update regardless of user consent. This is intended to be a last resort to force wide-scale updates of vulnerable devices in the field if a critical vulnerability is found and exploited in the wild and would typically be used exclusively for such high-risk vulnerabilities, with an informed consent process used for all regular software updates.

6.1.3 SWU-AUTH: Software Update Authentication

Software updates must be cryptographically authenticated, and provide anti-rollback features

Base requirement: L1–L5

Although it is important to support software updates to ensure that devices can be patched and maintained in the field, such features can lead to additional vulnerabilities – where a “bad actor” can install their own software into the device to take control of the device or to prevent its normal operation.

To prevent this, any software update must be cryptographically authenticated. Often this will be implemented by using a digital signature across the software image, which can be checked by the original software (or bootloader of the device) either before installation or at boot-up. Using a digital signature based on a public key algorithm (such as EdDSA, ECDSA, or RSA) ensures that the devices themselves do not need the private key that is used to generate the authentication data.

It is additionally required that the update implements “anti-rollback” features – such as a “monotonic” version number included in each release (that is a version number that only increases with each version) which is also checked during installation to ensure that a bad actor cannot just install a previous version of the software; to “reinstate” any otherwise patched vulnerabilities. This anti-rollback functionality may be waived for patches that only offer different functionalities, but do not patch vulnerabilities. For example, the switch back and forth between two different software flavors is allowed, but as soon as an update incorporates a fix for a security vulnerability, going back to the vulnerable version must be disallowed.

If the device does not computationally permit the use of public-key cryptography for securing the integrity of the software, and device-unique symmetric software authentication keys are also not feasible from the perspective of organizational overhead, for level 1 devices shared symmetric keys are permissible under the following constraints:

- The embedded device must utilize readout protection of the program memory, e.g., using SoC-provided fuse bits which can lock the readout of the software. This means that an attacker will at least have to break this security feature to retrieve the symmetric key.
- The software image that is transmitted during an update does not contain the shared symmetric key (e.g., if the bootloader contains said key and the bootloader itself is not part of the software image) or the whole software image is protected by strong encryption. For these purposes, the algorithms listed in Appendix B are considered acceptable. When an AEAD cipher is used, the authentication tag itself can be used instead of a separate MAC computation.
- The encryption and/or MAC keys may never be used for any other purpose but software update.

Requirement enhancement 1: L2–L5

Where a symmetric key system – such as an HMAC – is used for update authentication, the secret key in each device must be unique per device. Otherwise once the software of one device is exposed (e.g., through a physical attack on one device), a valid software signature for all other devices of this type can be created. Therefore, public-key cryptography is recommended to avoid the complexities of managing unique symmetric keys across device portfolios.

Requirement enhancement 2: L4, L5

For target security level 4 and up, the software update image must not only be authenticated but encrypted as well. Since device-individual firmware images are not desirable, it is acceptable for this single key to be symmetric and shared across all devices. However, it may not be exposed in any firmware update (i.e., must always be omitted from the image or only contained in the encrypted portion of the image). The algorithms that are considered acceptable for this purpose are listed in Appendix B.

Note that this is a measure that is controversial as it is following the security by obscurity paradigm. Some vendors might value public auditing of their binary code basis in the expectation of improving security; firmware encryption hampers this goal. Therefore, if an applicant demonstrates they are deliberately opening the firmware to public audit (e.g., by publishing it unencrypted on their website), this requirement enhancement may be waived.

6.1.4 SWU-ROT: Hardware root of trust

The device uses a hardware-based root of trust for securely storing sensitive data

Base requirement: L5

A hardware root of trust (e.g., secure element, TPM) is a dedicated embedded component/memory area that can securely store sensitive data such as cryptographic keys. It is the foundation on which all secure operations depend on and a source that can always be trusted by the system and is, therefore, a crucial component for processes such as the update or boot authentication process, providing a protected environment for encryption and signature verification keys. Having a hardware root of trust increases overall system security as it is exceedingly more difficult to extract or modify its stored data as compared to storing the data in software. Hardware roots of trust provide varying degrees of protection against different attack scenarios. The intent of this specification does not require extensive countermeasures against hardware attacks (such as fault injection or differential power analysis), but it needs to be ensured that the hardware dealing with sensitive cryptographic material is properly isolated from the remaining system. For instance, this can be achieved by a separate microcontroller that exclusively handles cryptographic computation using an API that does not permit reading out of private keys. To fulfill this requirement the device must store sensitive data, such as private keys (e.g., TLS client certificate keys) in such a hardware root of trust.

6.2 Data & Cryptography

6.2.1 DC-NDK: No default credentials or secret keys

System defaults such as password and/or cryptographic keys must be changed on the initial setup

Base requirement: L1–L5

Ideally, system defaults should be avoided – but realistically that is not always possible. It may be necessary for something to be set to a default value to allow for the “boot-strapping” of the system for the first time. However, the risk of using the default should be clearly outlined to the people operating that system for that first time, and this requirement outlines the need to force them to make a change from this default as part of the overall setup.

Where system defaults are automatically changed as part of the personalization/manufacturing process, these values must be set such that they are unique per device and statistically uncorrelated between devices (i.e., assigned randomly).

Ultimately, defaults for passwords and other such items should only be implemented for values that are minimally required to be present for normal operation, but which must be changed by the user before operation. This also covers any debugging/backdoor accounts that may be used during development – such values must never be left in a production system.

Shared public key material such as certificates or raw public keys may be used. For example, it is common to have a globally shared certificate across a family of devices to authenticate their respective software updates.

6.2.2 DC-PROT: Protect sensitive data

Sensitive data must be protected in transit and at rest

Base requirement: L1–L5

Bad actors will often attempt to recover sensitive data, such as passwords, secret cryptographic keys, and customer data, as the start of an attack on a system. This data may be easily accessed if it is not protected, and electronic protection must always involve strong cryptography and key management to ensure that it is providing the controls at a sufficient level. Therefore, any data that is communicated across connections that are not physically direct (such as a direct USB or serial connection) must be protected against disclosure through cryptographic means.

Requirement enhancement 1: L3–L5

Additionally for level 3 and up, storage of such sensitive data must also be protected as customers are likely to re-use passwords across different devices, or even re-purpose online passwords for home use. This includes ensuring that such data is not easily accessible with internal access to the device (e.g., through monitoring an internal serial bus). It is understood that sometimes such data must be displayed for business and user interface reasons (e.g., to display and receive a user password as it is entered), but a business justification for each exposure must be provided. Passwords must never be stored in plain text, but instead always in hashed form. When possible, the hashing process should include the usage of a salt, where the salt is defined as a unique, randomly generated string that is added to each password before hashing. Weak or broken hashing algorithms, such as MD5, must not be used.

Furthermore, industry-standard cryptographic algorithms, outlined in Appendix B, must be used to protect sensitive data. Development of proprietary, or bespoke, algorithms or protections weaken systems as such algorithms will not have undergone the many years of academic review and attack that is performed on those industry-standard methods. Therefore, protections can only be assumed when such standard algorithms are used.

At this level, brute-force attacks via network interfaces need to be protected against, limiting the rate of authentication attempts to a value that does not exceed twice that of what can be reasonably be expected in a typical use scenario.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

Requirement enhancement 2: L4, L5

For an implementation that targets security level 4 and up, the hashing process must incorporate key stretching algorithms such as scrypt or PBKDF2 to reduce the susceptibility against offline brute-force attacks.

6.2.3 DC-PWD: Passphrase complexity enforcement

When passphrases are used to authorize the use of services, they must fulfill minimum strength criteria

Base requirement: L2–L5

Passphrases are often required and implemented to provide authentication of users. If not set to a sufficiently secure value, they can be easily guessed or brute-forced to bypass this authentication, allowing a bad actor to gain access to the services the passphrases are supposed to protect. Many attacks on devices are based on exploiting insecure, or default, password values. The strength of a passphrase typically depends on two key factors: The first factor is the set of characters that passphrase characters are chosen from, known as the *alphabet*. The second factor is the length of the passphrase in characters. Strengths of passphrases are typically given in equivalent bit lengths, i.e., the binary logarithm of the number of possible combinations.

This requirement deals with the complexity of such passwords. Note that the requirement does only apply to scenarios in which they are technically feasible. For example, a numeric number verification for Bluetooth pairing would be out of scope regarding this requirement, because in principle the underlying system prevents using sufficiently secure passcodes. Similar situations arise where a passphrase may only use a limited alphabet because the HMI does not allow other characters (e.g., a device with a keypad that only has digits 0-9).

We differentiate in this requirement passphrases that are chosen by a user (i.e., a passphrase that the user can change themselves) against those that are chosen by a device or machine entirely at random (e.g., an API key that is chosen by software and cannot be changed by a user). For the latter, plausibility check routines used for human-choice passwords must not be used, since that would counterintuitively decrease the strength of passphrases.

Users should always be allowed to at least use the 26 special characters that correspond to ASCII codepoints 0x21 - 0x2f (!"#$%&'()*+,-./), 0x3a - 0x3f (:;<=>?), and 0x5b - 0x5f ([\]^_) if they so choose. Furthermore, the maximum length of a password shall not be restricted below 127 characters of length, meaning that any system shall be able to accept a password up to 127 characters (but may of course support longer passphrases).

For the base requirement, these are the minimum criteria regarding passphrase complexity:

- User-chosen passphrases: length at least 10 characters, at least one uppercase, and at least one lowercase character. Example of valid passphrases: "Achievement", waterFaLL5". Example of invalid passphrases: "fooBar123" (too short), "achievement" (no uppercase character), "ACHIEVEMENT" (no lowercase character).
- Machine-chosen passphrases: alphabet at least [A-Za-z], length at least 10 characters (approximately 57 bit of security)

Requirement enhancement 1: L4, L5

For level 4 and up, those rules become stricter:

- User-chosen passphrases: length at least 12 characters, at least one uppercase, at least one lowercase character, and at least one digit. Examples of valid passphrases: "Y3ll0whamm3r", "M1croorgan1sm". Example of invalid passphrases: "foobarF0BAR" (too short), "ucroorgan1sm" (no uppercase character), "UCROOGRAN1SM" (no lowercase character), "FOOBARfoobar" (no digit).
- Machine-chosen passphrases: alphabet at least [A-Za-z0-9], length at least 12 characters (approximately 71 bit of security)

6.2.4 DC-RECVR: Credential recovery

Provide recovery mechanisms for credentials

Base requirement: L2–L5

It is expected that users will forget or otherwise fail to correctly enter passwords from time to time. This should not present a complete barrier to the further use of the device, and therefore password recovery mechanisms must be put in place to ensure that the device can be recovered, and the user can continue to use the system.

Requirement enhancement 1: L4, L5

For devices that fulfill level 4 and up, such recovery mechanisms must be protected against exploitation by confirming user intent and authenticity. For example, where automated methods without human interaction are used, they must provide two-factor/out-of-band authentication of the user. Alternatively, a physical “reset” button may be implemented that requires physical interaction, preventing the exploitation of this feature through malware and validating the customer intent and authenticity through physical proximity to the system.

6.2.5 DC-TRNG: Cryptographically strong random number generation

Random number generation must ensure sufficient entropy

Base requirement: L3–L5

Random numbers are often used as part of security protocols and for generating cryptographic keys and passwords. Many vulnerabilities in systems have been caused by a lack of strong random numbers – including some attacks on standard protocols such as TLS. Generating random numbers on an embedded system can be difficult, as there is often little source “entropy” to be used to generate the numbers. It is recommended that multiple sources of entropy, such as network data timing, least significant bits on analog to digital input pins, and real-time clock data are used as a seed for any random number generation – which should then use an industry-standard pseudo-random bit generator. Ideally, each device should be shipped with some random, unique values injected during the manufacturing process to start this seed value.

Often, the simple random number generation functions provided by common programming languages are not cryptographically sound and should not be used. Using /dev/urandom on Unix-like systems is regarded as an acceptable solution for random number generation. To pass this requirement, it is necessary to use a cryptographically secure random number generator (CSPRNG or CSRNG). When a pseudo-random number generator is used, it must be seeded by a secure source of randomness and may not rely solely on predictable data such as a time stamp.

The use of a CSPRNG especially also is required for the generation of any key material, such as the creation of session keys or an asymmetric keypair.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.3 Logical Security

6.3.1 LS-DBG: Disable debug interfaces

Debug interfaces must be disabled or protected against misuse

Base requirement: L3–L5

Often devices will come with some interfaces that are either specifically designed or can be used, for “debugging” purposes. For example, local JTAG ports can often be used to extract the software from devices and start the reverse engineering process which allows for the determination of vulnerabilities within the device. Other examples are serial connections (e.g., via RS232) which may output sensitive information or provide direct access to the system. Such “debug” interfaces must be disabled in production devices. If it is not possible to do so, mitigations must be implemented to prevent the extraction of sensitive data or exploitation of the device. Often microcontrollers provide a feature called CRP which prevents the internal flash readout via JTAG. Devices that support this feature must have it enabled.

To fulfill level 3, at least serial connections must be deactivated or protected.

Requirement enhancement 1: L4, L5

Additionally, for level 4 and up, JTAG connections must be deactivated or protected.

6.3.2 LS-SECDEF: Systems configured to secure defaults

Systems must be configured to secure defaults

Base requirement: L3–L5

The default configuration of the system must ensure that the device is secure “out of the box”. Where deprecated or vulnerable features may be required, or desirable, for a specific market or customer segment, these features must be disabled by default. Examples of such features may be to allow for WPS wireless key negotiation, which is known to be vulnerable but may be desirable in some instances, or the usage of Modbus TCP which does not provide authenticity or confidentiality of data but is needed for a particular network environment.

Features that are essential to the device functionality must always have a secure alternative to an insecure optional feature. For example, a CCTV camera may offer the optional, disabled-by-default insecure RTSP protocol, but it must always have a secure alternative such as RTSPS.

When features that are commonly regarded as insecure are present in the device, sufficient user guidance must be provided to the user when they opt to enable such features. This allows users to understand the risk associated with enabling said features of the device. The guidance must be phrased in unambiguous language, discouraging the use of the insecure feature, and recommending a secure alternative. Guidance must not obfuscate the security risk by using technical jargon that cannot be understood by the device's intended target audience.

Devices must also be free from undocumented features that may allow for a “takeover” of the device by someone other than the intended end-users. The system must provide clear documentation of its features, and such “back-door” access or control features must not be implemented or possible within production devices.

Whenever information the device offers is publicly accessible without authentication – especially if this information might be security-relevant – this information must be kept to the minimum that still retains device functionality. For example, a device may have an API that publicly displays the progress of an ongoing firmware update or the respective API version, but it may not contain information that is not operationally required such as detailed version numbers of contained software packages.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.3.3 LS-FDIS: Unwanted functionality can be disabled

Customer access to disable unwanted features

Base requirement: L3–L5

Many products are shipped with large feature sets to allow for the broadest appeal. Thought should be given to allow for users to disable any features they do not want or do not need, as this may reduce the attack surface of the product, or even allow for easy remediation of vulnerabilities that are found in the field before those vulnerabilities being actively patched or fixed. It is understood that not all features of a system can be deactivated without having an impact on the usability or operation of the device. This requirement aims at those optional features which have no or limited impact on the operation of the device. An example may be a smart sensor that supports multiple network protocols (such as MQTT, XMPP, or HTTP) to retrieve measurement data where users must be able to choose their preferred protocol and disable those which they do not require.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.3.4 LS-LPP: Least Privilege Principle

Systems must implement the least privilege principle and utilize hardware-supported features such as memory containment

Base requirement: L3–L5

All software of sufficient complexity has vulnerabilities, and “defense in depth” measures must be used to protect against the successful exploitation of any newly discovered flaws. The goal is to have multiple layers of defense so that if one protection mechanism fails in practice, this does not lead to a full system compromise. One good measure is the application of the “least privilege principle” in systems. This means that software is assigned only the execution privilege and access rights that are sufficient and essential for its required operation.

Modern processing and operating systems provide many different methods for this to be achieved, and this requirement is not intended to mandate a specific implementation, but instead ensure that the device vendor has considered what access rights are necessary and put in place measures to ensure that additional access is prevented, or at least mitigated. For example, “sandboxing” or virtualized environments may be used, or access between assets and functions may be managed through assigning lower processor and/or operating system privilege levels to all code that does not require full access to the hardware of the device.

Typical means of implementation of this on an embedded system would include that processes should run as unprivileged users (e.g., the “nobody” user of a Linux system), use of chroot environments, and using file system permissions that disallow access to any data that needs not to be read or written by the respective processes. This requirement would be considered failed if one or more processes were running with root privileges even though they do not require these privileges at runtime. Another failure to meet this requirement would be world-readable (or group-readable) data that is potentially sensitive such as cryptographic keys.

6.3.5 LS-MPF: Memory Protection Features

Runtime and compile-time memory protection features must be implemented

Base requirement: L4, L5

Modern processing systems and compilers provide multiple methods to assist in the exploitation of any vulnerabilities which may exist in the source code of the device. By correctly enabling and implementing such protections, the security posture of the system can be increased. This requirement does not seek to mandate which protections should be implemented, as this will depend on the specific processing system/operating system/compiler used – for example, Address Space Layout Randomisation may be implemented in many modern, complex operating systems, but is often not used in smaller Real-Time Operating Systems which can have other protection methods. However, the vendor must demonstrate an understanding of the protections that are available and justify the use (or lack of use) of the protections that they have chosen to implement.

6.3.6 LS-KVUL: Software free from known vulnerabilities

System software should be free of publicly disclosed vulnerabilities

Base requirement: L4, L5

It is increasingly common for systems to be composed of several types and sources of software – from internally developed, to externally developed open-source or commercial software. For any externally developed software component, it is possible – and indeed likely – that there are previously disclosed vulnerabilities that have been patched and/or mitigated in further updates to the software. Therefore, it is an essential part of securing software to first identify what externally developed software components exist and using this list to confirm that these components are up to date and sufficiently mitigate any previously identified vulnerabilities.

It should be noted that – although it is desirable – it is not an absolute requirement that the very latest version is always used if existing vulnerabilities have been mitigated in other ways.

6.3.7 LS-LOGS: Logs or errors do not expose sensitive data

Logging and error messages must not expose sensitive data without authentication

Base requirement: L4, L5

It is often necessary for systems to be placed into a “debug” or “logging” mode to facilitate the identification and remediation of problems with the device. However, such data may be used to gain information about the system or to obtain data that should otherwise remain confidential. Therefore, it is important that any functions that allow for the logging of sensitive data are disabled by default and can only be temporarily enabled after suitable authentication. Once enabled, such logging should get deactivated automatically after a reasonable amount of time to ensure that the logging state is not accidentally left active. As guidance, this timeout should typically not exceed one hour, but the applicant can give written case-by-case justification if there is good reason to use a longer timeout period.

It is also strongly recommended that any sensitive data that is logged is secured with cryptography (e.g., through encryption using a public key on the device). Any upload or exfiltration of user-identifiable data from the customer premises in such logs must be covered under the privacy policy of the system and require an opt-in from the customer to accept the transfer of this data.

Error messages may also result in the exposure of information – for example, detailing an error with the padding in a cryptographic message can sometimes help attackers determine the values of sensitive information. Therefore, error messages must be carefully designed to not expose details that are too specific about the error state, and instead simply inform the user that an error has occurred. Timing of error messages must also be carefully managed; for example, common compare functions will return the result as quickly as they can, and therefore if used in comparison functions on sensitive data (e.g., passwords) could accidentally expose information about how many characters of the sensitive data are correct. For this reason, non-timing dependent compare functions are recommended for use with sensitive information, and passwords must not be compared directly with stored plaintext. Instead, they must be compared against a hashed value, such as that calculated through the scrypt or PBKDF2 function.

6.3.8 LS-UVUL: Software tested for unknown vulnerabilities

System software must be tested to check for undisclosed vulnerabilities

Base requirement: L5

Although much software may be re-used from other sources, a device will likely contain some applicant-developed code. In addition, the combination of different software components can open new threat vectors and potential vulnerabilities. Therefore, it is important that some checking is performed against the software of a device to identify such vulnerabilities. The intent of this testing is not to perform an exhaustive penetration test against all features and code of the device, as this would be expensive in terms of both time and direct costs – but to confirm that simple attacks are not possible on the system.

6.4 System Management

6.4.1 SM-AUTH: Sensitive services require authentication

Sensitive services must require authentication and ensure the confidentiality and integrity of data

Base requirement: L1–L5

Sensitive services within a device are services that allow for the allocation or changing of security settings, or which allow for access to customer personal information (such as authentication data, email addresses, etc.). Such access is inherently security-sensitive and therefore requires authentication to be performed to ensure that any changes are being correctly performed by the customer and are not being accessed or altered by a bad actor. This includes ensuring that access, once authenticated, ensures the integrity of data as it is passed into the device, as well as ensuring the confidentiality of any customer data during transport.

Likely, many systems will rely on standard protocols such as TLS to provide these features, and testing will include validating that such protocols are correctly configured and used, along with ensuring that weak modes of operation – such as insecure cipher suites – are disabled by default.

Requirement enhancement 1: L4, L5

For devices that target security levels 4 and up, sensitive services must be protected against brute force attacks by providing limits for authentication attempts.

6.4.2 SM-ERASE: Permanent erasure of sensitive data

Permanent erasure of sensitive data must be supported

Base requirement: L1–L5

Devices must protect sensitive data even during decommissioning (e.g., to prevent the exposure of customer Wi-Fi passwords after disposal or resale), and therefore implement either a “factory reset” which permanently erases all data and configuration from the device or provide strong protections to the data even given unrestricted physical access to the device. Where the device supports a network interface, it must be possible to “remotely decommission” the device. At all times, a local decommission procedure must always be provided – this may be passive. For instance, erasure of RAM storage after disconnection from power, but where passive mechanisms are implemented, they must operate within less than 8 hours and be shown to ensure permanent erasure.

6.4.3 SM-SFTY: Manual override for safety-critical operations

Manual backup/override must be provided for safety-related services

Base requirement: L2–L5

Safety-related services, such as those performed by door locks, are increasingly being automated and enabled through digital systems. This requirement outlines the need for such systems to provide is a safety mechanism that ensures any failure of the device – either through malware, lack of power, or coding flaw – does not result in a safety issue that could lead to a risk to life. For example, door locks should provide a manual method for locking and unlocking (such as a “standard” key).

6.4.4 SM-INPT: Input validation and sanitization

External inputs must be validated and sanitized before evaluation or execution

Base requirement: L3–L5

Functionality that allows for the direct execution of code or commands by the device or system can often be exploited by a malicious party. Such functionality should not be natively supported, and any method which passes user-supplied inputs to a system shell or parses and evaluates it directly with its native interpreter must validate and sanitize it beforehand. This not only covers direct inputs such as form fields or file uploads but also any other input data the method receives (HTTP headers, cookies, query strings, SQL queries, formatted payload data such as JSON, XML, CSV, JPEG, etc.). In this context, input validation ensures that inputs conform to requirements such as length and data type of the receiving method, whereas input sanitization ensures that inputs conform to requirements of the underlying system to which the inputs are passed. This may include the elimination of unwanted characters through removing, replacing, encoding, or escaping characters. If possible, it is preferred to use the command interpreters or parsers provided functionalities for input validation and sanitization over custom implementations.

This requirement covers all interfaces and services which receive and handle device external inputs.

6.4.5 SM-SESS: Sensitive services implement session management

System management services accessible over wireless and IP interfaces must implement session management to limit multiple sessions, and ensure ongoing authentication

Base requirement: L4, L5

Authentication controls over system services may often be bypassed if the authentication is not correctly managed so that it provides “session management”. This ensures not only that there is not a process of “staying authenticated forever”, but also that the communications process is secured once the authentication is performed so that bad actors cannot interpose their data, or change data being transferred by the correctly authenticated user.

To ensure that users do not accidentally leave a security-sensitive interface open/accessible, there must be an inactivity timer that disconnects/de-authenticates the user on that interface after a reasonable time of inactivity. As guidance, this timeout should typically not exceed 15 minutes, but the applicant can give written case-by-case justification if there is good reason to use a longer timeout period.

This requirement covers system management services that may affect the security of the system – such as changing customer data storage and tracking preferences, or installing certificates, etc. General user interface functions (such as altering the level of light output from a bulb, or changing the temperature on a thermostat), and functions which are only accessible through manual interfaces, where that interface is contained within the customer premises (e.g., not on an external door lock), are not in the scope of this requirement.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.5 Communication Security

6.5.1 CS-XMIT: Cryptographically Secure Data Transmission

Communication channels need to be protected via cryptographic means to achieve various security properties

Base requirement: L1–L5

Any communication channel through which unintended actions can be triggered must be secured in a way that achieves secure communication even when the medium used for transmission cannot be considered secure. For instance, communication over the Internet could potentially be read and modified by anyone on the routing path. An end-to-end security implementation would ensure that the communication still retains important security properties, namely:

- Confidentiality of data: An eavesdropper on the connection is unable to make sense of the transmitted information
- Integrity of data: It is possible to determine with exceeding likelihood if received data was modified in transit
- Peer validation: The respective peer on the other end of the connection can be verified to be the correct party with whom communication is intended
- Downgrade protection: The protocol, if it supports multiple versions, must always use a version both peers agree on and may not be artificially downgraded by an adversary
- Replay protection: Data that has previously been recorded by an adversary and that is repeated by that adversary is detected as a duplicate and properly rejected

Typically, this is achieved by using TLS as the foundational transport protocol, which, in a correct configuration, fulfills all these security properties. Note, however, that even a TLS configuration can be susceptible to attacks on these security goals; most notably if poor choices in the protocol parameterization are used (e.g., weak cipher suites), specific security mechanisms are disabled (e.g., peer validation). Replay protection may be deliberately sacrificed in specific scenarios as well. One example of this would be the use of the 0RTT feature of TLSv1.3. This is permissible if and only if the concerned software has other means of ensuring the replay of messages does not impact the overall security of the system.

For details on TLS parameterization and acceptable cipher suites, refer to Appendix A.

Specific resource constraints lead to a situation in which deeply embedded devices may not have the resources to fulfill a full TLS handshake; they still need to make sure that the desired security properties are met.

Requirement enhancement 1: L2–L5

For devices that target security level 2 and up, the implementation must either follow an industry-standard security protocol (such as TLS) or proof of the security properties must be provided that has been vetted by experts in the field. The protocol must be described in a verbose and unambiguous manner and it must include test vectors of at least one complete message exchange. Note that precisely describing and vetting a custom cryptographic protocol is generally difficult because of the required expertise in the field of theoretical cryptography and cryptanalysis.

Requirement enhancement 2: L3–L5

For devices that target security level 3 and up, custom cryptographic constructions are allowed only if a formal security proof is given and has been reviewed by a FIPS-accredited testing laboratory. Otherwise, industry-standard protocols must be used. Furthermore, for these devices, it is required that all secured communication that falls under this clause also achieves Perfect Forward Secrecy (PFS).

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.5.2 CS-RFXMIT: Wireless Communication Security

The device must support industry-accepted security defaults for any wireless connection

Base requirement: L1–L5

Where devices implement Wi-Fi connections, these devices must not force a reduction in the security of the customer's Wi-Fi implementation. For example, a poorly designed system may force the customer to change from WPA2 security to using WEP, which is considered insecure.

Where devices implement Bluetooth connections, it is required that these devices implement pairing before sensitive data is exchanged. As a pairing mechanism “Secure Simple Pairing” shall be supported. “Secure Simple Pairing” provides four different pairing methods: “JustWorks”, “Passkey Entry”, “Out of band” and “Numeric Comparison”. For devices that have no input nor output capabilities, the “JustWorks” pairing method is acceptable. In any other case, the device shall support one of the other three pairing methods which provide authenticated connections.

In use cases where the implementation of Bluetooth pairing is not feasible the communication must be protected by other means (e.g., custom cryptographic constructions).

For any level, it is permissible for the radio frequency communication to provide no security at all if the requirements of CS-XMIT are fulfilled at least to level 3.

Requirement enhancement 1: L2–L5

For devices that fulfill level 2 and up, the Bluetooth implementation must either use the industry-standard Bluetooth pairing mechanism or proof of the security properties protecting the communication by other means must be provided that has been vetted by experts in the field. Note that this is typically a task that is exceedingly difficult to achieve because of the required expertise in the field of theoretical cryptography and cryptanalysis.

Requirement enhancement 2: L3–L5

For devices that target security level 3 and up, custom cryptographic constructions are disallowed and the industry-standard Bluetooth pairing mechanism must be used.

Requirement enhancement 3: L4, L5

For devices that fulfill levels 4 and 5, for Bluetooth connections additionally the “Secure Connections” feature, which upgrades “Secure Simple Pairing” algorithms, shall be enabled.

6.6 Process & Documentation

6.6.1 PD-DEVID: Product Identification

The model designation of the device must be available to the end-user

Base requirement: L1–L5

The device must have an identifier that uniquely identifies it. Additionally, the device must have the capability to show the currently running firmware version to the end-user. The purpose of making this information available to the end-user is to enable them to determine:

- Published security vulnerabilities affecting the device.
- The most recent firmware version the vendor offers for a particular device.
- If the device is currently running this latest published firmware version.

A device identifier may be put on the device itself in physical form (such as a printed or etched label) or it may be accessible through a network API (such as a web interface or companion app). Due to the nature of the firmware version and its volatility, it needs to be available through some form of HMI (e.g., an app or a display), and having it printed on the device itself is not sufficient.

6.6.2 PD-COLL: Data Collection and Handling

Data collection by the device must be documented

Base requirement: L1–L5

Many devices and systems allow for the collection, processing, and storage of user personally identifiable information (PII). Such data includes address/contact details and biometric data (e.g., audio or video), and must be securely managed to ensure that the use of digital applications by the user does not lead to physical safety issues or identify theft.

Where the system collects/processes/stores user-identifiable data, the user must be provided with a complete list of this data and a description of how this data is used, by whom it is being used, and for what specific purposes it is being used. This requirement also extends to parties handling data on behalf of the applicant.

This requirement also covers any data that may be logged through special “debug” modes.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

Requirement enhancement 1: L2–L5

For level 2 and up, methods must be available to the user which allows to opt-in/opt-out of user-identifiable data collection. When deciding to opt-out, already collected user identifiable data must be erased if requested by the user.

Where the collection of user data is stored externally, the user must be provided with an opt-in to accept the external transfer of this data. This opt-in must outline exactly what data is being collected and how this data is used.

Requirement enhancement 2: L3–L5

For level 3 and up, all sensing capabilities must be described in user-facing documentation (e.g., presence of microphones, sensory equipment, network equipment, radio equipment).

Requirement enhancement 3: L4, L5

For level 4 and up, users must be kept up to date on the uses made of their user-identifiable data, so that they can make informed decisions about their opt-in behavior. Privacy policy changes must be clearly communicated to the customer.

Requirement enhancement 4: L5

For level 5, any user identifiable data that is remotely collected or stored outside of user-specific accounts must be anonymized so that it cannot be linked back to an individual.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.6.3 PD-PROCS: Documented patch/update process

A documented process for the distribution of patches/updates must be maintained

Base requirement: L1–L5

The final step to fixing a known vulnerability is to issue the patch to the customer/device. This must follow a clear process – which need not be complex but must clearly outline the steps involved in approving, signing, and distributing the new code.

This is required because it is often when there is a “rush” to fix a problem that important security steps are missed, resulting in an even worse situation and more potential exposure of the systems which were being patched.

Requirement enhancement 1: L4, L5

For devices that fulfill levels 4 and 5, additionally, a process step shall be included which ensures that any debug functionality or any other unnecessary feature is removed before release.

6.6.4 PD-EOL: End-of-life policy

Information on the minimum support period must be available to end-users

Base requirement: L1–L5

End-users shall be able to obtain information on the minimum support period where the manufacturer of the product shall continue to provide software updates to the product. This period is expected to be appropriate to the device, where e.g., devices with a long product lifecycle will continue to receive updates for several years after purchase.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.6.5 PD-VMGMT: Vulnerability management program

A vulnerability management and disclosure program must be maintained

Base requirement: L1–L5

It can be expected that new issues will become apparent in systems after evaluation and shipping to the customer. Therefore, system vendors must ensure that they have a vulnerability management and disclosure program to maintain the security of their products once shipped. This program must include processes for:

- Monitoring for new vulnerabilities in all code that is contained in the software composition list.
- Testing if vulnerabilities affect the vendor systems, and how they can be mitigated if the system is affected
- Triaging the vulnerability following a commonly accepted methodology such as CVSS to judge their impact.
- The creation and testing of a patch for the vulnerability, if required.
- Informing customers of an already published vulnerability, and any mitigating steps they can take whilst a patch is being created. As long as the vulnerability has not become public knowledge yet, it is acceptable to delay informing customers until after the patch has been created.

Additionally, contact information and details about the vulnerability disclosure process for external security researchers should be published on a publicly available website.

Requirement enhancement 1: L3–L5

For level 3 and up, the vendor needs to demonstrate evidence they are performing vulnerability monitoring of all system components at least biannually. Such monitoring can be done manually by the vendor, deferred to tools that perform such scans, or passed to a third-party vendor.

Requirement enhancement 2: L5

For level 5, the vendor must have a system in place that performs continuous and proactive anomaly detection to catch vulnerabilities early on. Such a feature can be provided by the vendor or by a third party.

6.6.6 PD-CLMON: Regularly monitored cloud/app environment(s)

Cloud and App environments are regularly monitored for vulnerabilities

Base requirement: L3–L5

It is expected that many systems that integrate with IoT systems will have their own proprietary back end, or “cloud” systems as well as App interfaces and that these will be an important part of the overall security posture of a device. It is a requirement that the vendor implements vulnerability scanning at least every quarter for these environments – and remediate any high severity issues that are found. During the assessment, the vendor must submit evidence of their scanning results, such that it can be confirmed that there is a clear “find and fix” process for online vulnerabilities that is being maintained.

6.6.7 PD-SBOM: Software Bill-of-Materials

A Software Composition List must be maintained

Base requirement: L4, L5

Any software of sufficient complexity contains bugs. It is not possible for any amount of testing to find and allow for the remediation of, all bugs in any nontrivial code basis – which is why ongoing maintenance of such code is so important. However, it is increasingly common today for the software in a device to be created from various “software components” – open-source code, third-party libraries, and external binary files. Therefore, to maintain code, it is not sufficient to simply maintain the code that has been created directly by the product vendor; it is necessary to ensure that all additional “software components” are maintained and updated as well.

To achieve this, it is necessary to create and keep up to date a “software composition list” (sometimes called a “software bill of materials”) which indicates all the different software components used in a particular build, as well as their versions. This list must be exhaustive; think of it as an ingredient list for your software: if not all ingredients are listed, the recipe will not turn out correctly. In this instance, if not all software is listed, you will not be able to securely maintain your device.

Using this software composition list is a prerequisite for the establishment of a vulnerability management program. Using such a program, it is possible to ensure that when there is a new vulnerability found in some third-party or open-source code that is used in the device, it can be noted, investigated, and patched where necessary.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

6.6.8 PD-PHYIF: Physical Interface Documentation

All physical interfaces present in the device hardware must be documented and justified

Base requirement: L5

The security posture of a system is often described as its “attack surface” – the amount of code that can be interacted with is generally directly related to the potential vulnerabilities a system may have. The more code, the more potential vulnerabilities. However, access to this code is of course also important, and the interfaces of a device are the “front line” of the device security, and by definition attacks on devices generally start with these interfaces. Indeed, any device can be summarized by the totality of its inputs, outputs, and internal processing (where the inputs and outputs are the interfaces).

Therefore, all interfaces of the devices need to be clearly understood and justified as to their purpose, as an unnecessary interface may be the one that is used to compromise the system. This list of interfaces must include both physical ports (USB, Serial, Ethernet, etc.) and protocols that are supported over these interfaces.

It is recognized that documenting all protocols supported can be quite complex; for example, a USB interface may support many different protocols, classes, and types of devices. However, the goal is to ensure that the totality of the interfaces is well understood.

6.6.9 PD-SDOC: All services documented

All services present in the device must be documented and justified

Base requirement: L5

For this standard, a service is considered a super-set of a protocol, in that it actively “listens” for connections across switched or wireless connections. Direct physical interfaces, such as serial or JTAG, are generally considered not to be a “service”.

As with protocols, listening services are often the first point of attack on a device, and therefore can be the first line of defense to prevent such attacks. Justification of enabled services is vital to understand the security posture of the system and ensure that sufficient security measures are put in place to protect these interfaces.

It is understood that additional services may be included in a device as a product differentiator, or to provide value-added services to specific market segments. It is recommended that consideration be given to limiting the functionality of the system “out of the box” and instead providing options for users to enable features where they see a need.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

7 Component Qualification

UL MCV 1376 is intended to be a device-centric assessment. Vendors may apply for a specific level of their device and, if all the requirements for that level are fulfilled, the claim is accepted as verified. Service providers or sub-component manufacturers may however choose to have their service or component evaluated. In this case, a subset of requirements would be evaluated that applies to the service or sub-component. Examples of a service could be a platform that provides continuous vulnerability scanning (vulnerability management as a service); an example of a sub-component could be a TLS library or IoT platform.

For component qualification, the applicant receives a Delta Document after evaluation of the component under test. This record describes which clauses are fulfilled by the component and describes how the component or service needs to be integrated by an OEM to fulfill the respective requirement with their end products. Note that it could also be possible for a component to violate specific requirements, which would mean it cannot be integrated into a product that wants to meet a UL MCV 1376 level. Therefore, during component qualification, it is checked:

- That the component or service is not in direct conflict with any requirements of UL MCV 1376.
- Which requirements the component fulfills for the end product when it is integrated in the way intended by the component manufacturer or service provider.

Concretely, when reviewing a UL MCV 1376 component, the pass/fail criterion therefore always is whether the end product would be able to fulfill the respective requirements when the component is integrated as described. For such a component qualification, the applicant needs to provide a sample application (e.g., a demonstrator that uses the component) so that it can be validated as if it were an end product. Examples of what could be assessed as a component in respect to UL MCV 1376 are:

- Services
  - Third-party service that scans/tracks packages and regularly informs customers of recently discovered vulnerabilities.
  - Third-party service that provides PKI services, such as X.509 enrollment, device provisioning, and secure certificate update.
  - Third-party service that provides firmware updates in a secure (encrypted and authenticated) manner over the Internet.
- Platform or Library
  - TLS library that ensures secure communication over untrusted channels.
  - RTOS that provides secure storage mechanisms and/or secure erasure of data when requested.

    - Chipset that provides specific functionality (e.g., custom microcontroller with an embedded Wi-Fi stack).
- Solution
    - Complete IoT device that runs the actual customer application merely as a payload (“App”) and which handles firmware upgrade, or backend communication for the OEM.

When such a component is integrated into a final product, the OEM can submit their device under test along with the Delta Document of the integrated (and previously qualified) component or components so that the assessment does not have to cover the nature of the component or service, but only validates its proper integration as laid out in the Delta Document.

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

8 Mapping to ETSI EN 303 645

UL MCV 1376 serves as the guiding document to aid ETSI EN 303 645 testing and makes it more concrete and transparent what UL is looking for when evaluating ETSI EN 303 645. Note that not all ETSI EN 303 645 provisions might apply to a given device because the applicability of provisions is conditional. As such, it needs to be determined first which conditions a device satisfies to be able to tell which provisions (and their mapped UL MCV 1376 requirements) apply.

Name	Text	MCV1376
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.	6.2.1 Data & Cryptography: No default credentials or secret keys (L1)
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.	6.2.3 Data & Cryptography: Passphrase complexity enforcement (L2)
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.	6.2.2 Data & Cryptography: Protect sensitive data (L1)
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.	6.2.1 Data & Cryptography: No default credentials or secret keys (L1)
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.	6.2.2 Data & Cryptography: Protect sensitive data $^{+}$ (L3)
Provision 5.2-1	The manufacturer shall make a vulnerability disclosure policy	6.6.5 Process & Documentation: Vulnerability management program

	publicly available.	(L1)
Provision 5.3-2	When the device is not a constrained device, it shall have an update mechanism for the secure installation of updates.	6.1.3 Software Update: Software Update Authentication (L1)
Provision 5.3-3	An update shall be simple for the user to apply.	6.1.2 Software Update: Automatic Software Update Tracking (L1)
Provision 5.3-7	The device shall use best practice cryptography to facilitate secure update mechanisms.	6.1.3 Software Update: Software Update Authentication (L1)
Provision 5.3-8	Security updates shall be timely.	6.1.2 Software Update: Automatic Software Update Tracking (L1)
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.	6.1.3 Software Update: Software Update Authentication (L1)
Provision 5.3-13	The manufacturer shall publish, in an accessible way that is clear and transparent to the user, the defined support period.	6.6.4 Process & Documentation: End-of-life policy (L1)
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.	6.6.1 Process & Documentation: Product Identification (L1)
Provision 5.4-1	Sensitive security parameters in persistent storage shall be stored securely by the device.	6.2.2 Data & Cryptography: Protect sensitive data (L1)
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.	6.2.2 Data & Cryptography: Protect sensitive data (L1)
Provision 5.4-3	Hard-coded critical security parameters in device software source code shall not be used.	6.2.1 Data & Cryptography: No default credentials or secret keys (L1)
Provision 5.4-4	Any critical security parameters used for integrity and authenticity checks of software updates and for protection of communication with	6.1.3 Software Update: Software Update Authentication (L1)

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

	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.	
Provision 5.5-1	The consumer IoT device shall use best practice cryptography to communicate securely.	6.5.1 Communication Security: Cryptographically Secure Data Transmission (L1)
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.	6.4.1 System Management: Sensitive services require authentication (L1)
Provision 5.5-7	The consumer IoT device shall protect the confidentiality of critical security parameters that are communicated via remotely accessible network interfaces.	6.5.1 Communication Security: Cryptographically Secure Data Transmission (L1)
Provision 5.5-8	The manufacturer shall follow secure management processes for critical security parameters that relate to the device.	• 6.2.1 Data & Cryptography: No default credentials or secret keys (L1) • 6.2.5 Data & Cryptography: Cryptographically strong random number generation (L3)
Provision 5.6-1	All unused network and logical interfaces shall be disabled.	6.3.2 Logical Security: Systems configured to secure defaults (L3)
Provision 5.6-2	In the initialized state, the network interfaces of the device shall minimize the unauthenticated disclosure of security-relevant information.	6.3.2 Logical Security: Systems configured to secure defaults (L3)
Provision 5.6-4	Where a debug interface is physically accessible, it shall be disabled in software.	6.3.1 Logical Security: Disable debug interfaces (L3)
Provision	The confidentiality of sensitive	6.5.1 Communication Security:

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

5.8-2	personal data communicated between the device and associated services shall be protected, with cryptography appropriate to the properties of the technology and usage.	Cryptographically Secure Data Transmission (L1)
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.	6.6.2 Process & Documentation: Data Collection and Handling $^{++}$ (L3)
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.	6.4.2 System Management: Permanent erasure of sensitive data (L1)
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.	6.4.4 System Management: Input validation and sanitization (L3)
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.	6.6.2 Process & Documentation: Data Collection and Handling (L1)
Provision 6-2	Where personal data is processed on the basis of consumers' consent, this consent shall be obtained in a valid way.	6.6.2 Process & Documentation: Data Collection and Handling $^{+}$ (L2)
Provision 6-3	Consumers who gave consent for the processing of their personal data shall have the capability to withdraw it at any time.	6.6.2 Process & Documentation: Data Collection and Handling $^{+}$ (L2)
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.	6.6.2 Process & Documentation: Data Collection and Handling (L1)

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

A Acceptable Cipher Suites

TLS implementations for levels L1 and L2 must use TLS version 1.1 or above, for L3 and up the TLS version must be version 1.2 or above.

Cipher suites that truncate authentication tags can be viewed as equivalent to their non-truncated counterparts if the implementation can demonstrate that the TLS session is fully severed upon reception of a single wrong MIC (meaning that session resumption is prohibited). For example, `TLS_AES_128_CCM_8_SHA256` can be considered in the same category as `TLS_AES_128_CCM_SHA256` if the implementation performs aforementioned session abort.

The following non-TLSv1.3 cipher suites are considered acceptable for all levels:

- `TLS_DHE_PSK_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_PSK_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256`
- `TLS_ECHDE_RSA_WITH_AES_128_GCM_SHA256`

Additionally, cipher suites below are considered acceptable for levels L1 and L2:

- `TLS_DHE_DSS_WITH_AES_128_CBC_SHA256`
- `TLS_DHE_DSS_WITH_AES_128_CBC_SHA`
- `TLS_DHE_DSS_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_DSS_WITH_AES_256_CBC_SHA256`
- `TLS_DHE_DSS_WITH_AES_256_CBC_SHA`
- `TLS_DHE_DSS_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_PSK_WITH_AES_128_CBC_SHA256`
- `TLS_DHE_PSK_WITH_AES_128_CBC_SHA`
- `TLS_DHE_PSK_WITH_AES_256_CBC_SHA384`
- `TLS_DHE_PSK_WITH_AES_256_CBC_SHA`
- `TLS_DHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_DHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`

git: d661b9771623cf9f2982f45ff21d7c0212c68517 / 2021-09-23 09:41:46 +0200

- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA
- TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_PSK_WITH_AES_128_CBC_SHA256
- TLS_PSK_WITH_AES_128_CBC_SHA
- TLS_PSK_WITH_AES_128_GCM_SHA256
- TLS_PSK_WITH_AES_256_CBC_SHA384
- TLS_PSK_WITH_AES_256_CBC_SHA
- TLS_PSK_WITH_AES_256_GCM_SHA384
- TLS_PSK_WITH_CHACHA20_POLY1305_SHA256
- TLS_RSA_PSK_WITH_AES_128_CBC_SHA256
- TLS_RSA_PSK_WITH_AES_128_CBC_SHA
- TLS_RSA_PSK_WITH_AES_128_GCM_SHA256
- TLS_RSA_PSK_WITH_AES_256_CBC_SHA384
- TLS_RSA_PSK_WITH_AES_256_CBC_SHA
- TLS_RSA_PSK_WITH_AES_256_GCM_SHA384
- TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CCM
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_CCM
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA
- TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA
- TLS_SRP_SHA_WITH_AES_128_CBC_SHA
- TLS_SRP_SHA_WITH_AES_256_CBC_SHA

For any level, use of the following cipher suites is disallowed:

- TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_DH_DSS_WITH_AES_128_CBC_SHA256
- TLS_DH_DSS_WITH_AES_128_CBC_SHA
- TLS_DH_DSS_WITH_AES_128_GCM_SHA256
- TLS_DH_DSS_WITH_AES_256_CBC_SHA256

- TLS_DH_DSS_WITH_AES_256_CBC_SHA
- TLS_DH_DSS_WITH_AES_256_GCM_SHA384
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA

For TLSv1.3, the following cipher suites are permissible for all levels:

- TLS_AES_128_CCM_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

For TLSv1.3, the following cipher suites are permissible only for levels L1 and L2:

- TLS_AES_128_CCM_8_SHA256

For ephemeral TLS key agreements on elliptic curves, the curves listed in Appendix B are considered acceptable for all levels.

Cipher suites which are not included in this enumeration can be permissible but need to be rationalized on a case-by-case basis. In particular, the requirements for inclusion in all levels are:

- Cipher suite must provide PFS.
- Cipher suite must not use any cryptographically weak or broken algorithm or operation mode (such as SHA-1, CBC, or MD5).
- No truncation of authentication tags is performed unless it can be demonstrated any TLS session is fully terminated upon reception of a single wrong MIC.
- Key agreement must be cryptographically strong (e.g., using randomly generated DH parameters or 2048 bits or more or using an elliptic curve of field size roughly equivalent to 256 bits such as X25519 or secp256r1).

For L1 and L2, additional some weak primitives are permissible for legacy purposes (in particular, SHA-1 and the CBC operation mode). PFS does not necessarily have to be provided but is highly encouraged.

B Acceptable Cryptography

Where cryptography is relied upon as a security feature within the device, for example, to encrypt communication or to authenticate firmware images, it shall implement a combination of the following cryptographic algorithms. Deviations of these choices are acceptable if and only if it can be conclusively proven that the security of the chosen algorithm is equivalent to the level of the shown algorithms.

- Block Ciphers
  - AES: at least 128 bit key size
  - TDES: 168 bit key size
- Stream Ciphers
  - ChaCha20: at least 128 bit key size
- Hash Functions
  - SHA-2 family: 256 bit output (SHA-256, SHA-384, SHA-512)
  - SHA-3 family: 256 bit output
  - BLAKE family: at least 256 bit output
- KDF
  - scrypt
  - Argon2
  - PBKDF2
- Asymmetric Algorithms
  - RSA: at least modulus size of 2048 bit
  - FFC/DH: at least modulus size of 2048 bit
  - FFC/DSA: at least L = 2048, N = 224
  - ECC with one of the following domain parameters:
    - Curve25519
    - Curve448
    - secp256r1
    - secp384r1
    - secp521r1
    - Brainpool curves of 256, 320, 384, or 512 bits and their respective twists (8 curves total)
- Modes of Operation
  - CBC only if authentication is added in EtM fashion
  - CTR, GCM
  - CCM
  - EAX
- Authentication
  - AEAD (e.g., GCM, CCM, etc.)
  - Poly1305 (typically in combination with ChaCha20)
  - HMAC with an approved hash function

C Acceptable Plaintext Protocols

It is a requirement that an industry-standard security protocol is implemented where possible. Some standard protocols do not allow for use of encryption, and the following is a non-exhaustive list of these.

- ARP,
- DNS,
- DHCP,
- NTP,
- Initial browser connections (before immediate redirection to a TLS encrypted page)