IEC 62443-4-1:2018 Secure product development lifecycle for industrial automation

Secure development lifecycle, eight practices

47 requirements. This standard can be certified against by an accredited body. As of 07/2026.

IEC 62443-4-1 can be certified against, but the object is unusual: what gets certified is the development process of a named development organisation, meaning one concrete documented version of the secure lifecycle. Neither a product nor a person is certified. The routes are ISASecure SDLA and the IECEE CB Scheme. As things stand today, such a certificate does not create a presumption of conformity with the EU Cyber Resilience Act. Do not confuse it with IEC 62443-4-2, which places requirements on the component itself. This list is a working aid for self-assessment, not a proof.

Company	Date	Prepared by

5 Practice 1: Security Management (SM) 13

SM-1 **A defined product development lifecycle exists that carries security through every phase, from the initial concept through to decommissioning of the product, and this lifecycle is the one actually followed in practice.** Document

What proves it: Process description or work instruction listing the phases, the deliverables per phase and the handover points, together with evidence from an ongoing project showing that the phases were genuinely worked through. A small organisation can manage with one page per phase plus a checklist in the ticketing system; what matters is that the documented process and daily practice match.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SM-2 **Every task in the secure development lifecycle has an assigned owner, and this assignment is known to the people involved.** Document

What proves it: Role and responsibility matrix mapping each process step to a named role, supplemented by the current staffing of those roles. With only a handful of people a table with role, task and person is sufficient; the important part is the deputy arrangement for holidays and sickness.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SM-3 **It is traceably defined which products, variants, releases and components fall under the secure development lifecycle and which do not.** Document

What proves it: List of the products and version states in scope with a rationale for inclusions and exclusions, maintained with every new release. In practice a column in the product overview works well, carrying a yes or no per row plus one sentence of justification.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SM-4 **The people who take on security tasks hold the technical competence required for them, and that competence is deliberately built up and refreshed.**

What proves it: Competence profile per security role, qualification records, training plan and attendance records, supplemented by a gap overview. Small organisations evidence this through certificates, conference or webinar attendance and a short annual review of who closes which gap by when, buying in external expertise where needed.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SM-5 **The scope of the secure development lifecycle is delimited and names the sites, organisational units, supplier contributions and development environments involved, along with the boundaries of that scope.** Document

What proves it: Scope statement covering sites, teams, outsourced portions and systems, plus a sketch of the external interfaces. Anyone developing at a single site records this in a few sentences and explicitly names what lies outside, such as externally developed assemblies or third party hosting.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SM-6 **For every element delivered to users, software, firmware, configurations and accompanying files, an integrity check is possible so that recipients can detect undetected modification.**

What proves it: Signatures or published checksums per delivery package, a description of the method and one example from a real release, together with instructions for the customer on how to perform the check. For small teams a signed checksum file next to the download plus a paragraph in the release notes is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-7 **The environments used for development, build and test are protected against unauthorised access and against tampering with source code, build tooling and artefacts.**

What proves it: Access control concept for repository, build server and test systems, access lists with the date of the last review, records of security relevant changes and evidence that production and development access are separated. Without an in house IT department, two factor sign in, a protected main branch with mandatory review and a half yearly review of who still has access will do.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-8 **Private keys used to sign deliveries are protected against disclosure and misuse, and their use is restricted to authorised people and systems.**

What proves it: Description of key storage, access, renewal and revocation, evidence of protected storage such as a hardware token or key management service, plus records of signing operations. A small organisation keeps the signing key on a hardware token, documents the two people with access and holds a procedure ready for the loss case.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-9 **For purchased or adopted components, including open source software, the security expectations placed on the supplier are defined and are applied during selection.** Document

What proves it: Security requirements in supplier specifications or contracts, selection criteria for third party components, an inventory of all third party components with version and source, plus the assessment of whether the supplier meets the expectations. Without a purchasing department, a maintained component list with a column recording whether the supplier's security updates and reporting channels were checked is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-10 **Components developed to order by a third party are subject to the same security requirements as in house development, and adherence is verified.**

What proves it: Purchase order or contract documents stating the required security obligations, evidence from the service provider on the practices applied, acceptance records and results of your own checks on the delivered work. Without an audit budget, a contractual assurance, submission of the provider's test results and one sample check of your own before acceptance are enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-11 **Security related findings from all sources, tests, reviews, user reports and third party notifications, are captured, assessed for their impact, assigned an owner and a due date, and tracked through to closure.** Document

What proves it: Defect or ticket system with a dedicated marker for security topics, and per entry the impact assessment, the owner, the due date and the closure note, plus a report on open items. A small team uses a label in the existing ticket system and a standing agenda item in the weekly meeting.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-12 **Before a product is released it is verified and evidenced that the intended steps of the secure development lifecycle were actually carried out and that the associated records were produced.** Document

What proves it: Release review per release comparing against the process steps, references to the respective records and a documented release decision with date and signature. In practice a release checklist that allows no shipment without complete ticks works well; open items are justified and given a date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SM-13 **The secure development lifecycle itself is regularly challenged and improved, drawing on insights from incidents, test results, customer feedback and the threat landscape.**

What proves it: Records of the process review, for example a management review or retrospective, with the process changes triggered, their implementation status and a reference to the root cause. Anyone using the maturity levels of the standard records the current state and the target per practice; maturity level 4 demands exactly this evidence of continuous improvement. For a small organisation one meeting per year with minutes and three concrete improvements is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6 Practice 2: Specification of Security Requirements (SR) 5

SR-1 **The intended product context is described: expected operating environment, security measures assumed to be provided by that environment, intended user groups and roles, assumed trust boundaries, and the uses for which the product is explicitly not intended.** Document

What proves it: A product security context description, either as a standalone document or as a fixed section of the product specification, covering the network environment, assumed external protections such as a firewall or physical access control, a list of roles, and an explicit list of out of scope uses. A small organisation keeps this to two pages, adds a simple sketch of the plant environment, and has the product owner date and approve it.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SR-2 **A threat model exists for the product that maps data flows, interfaces, trust boundaries and attack paths, names the resulting threats and their possible impact, and assigns countermeasures. It is reviewed and updated on relevant changes and at least once per product release, and open items are tracked until they are closed.** Document

What proves it: Threat model with a data flow diagram, a threat list per interface, assigned countermeasures and the status of open items, plus a revision history with date and author. A small organisation can work from one table per interface, with columns for what could go wrong there, how bad it would be, what we do about it, and who reviews it.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SR-3 **The security requirements for the product are defined and documented. They cover the results of the threat model, the described product security context, applicable legal and contractual obligations and the target security level, and they include the security capabilities the product itself has to provide.** Document

What proves it: Requirements list or requirements management tool with a unique identifier per requirement, the source of each entry such as a threat, a customer contract or a legal obligation, and the target security level. A small organisation runs this as a single table with a fixed ID column that verification and testing later hook into.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SR-4 **Each individual security requirement is worded so that it is unambiguous, free of contradictions, testable and traceable to a specific product function or interface, and it carries the information needed to verify it later without further clarification.**

What proves it: A sample from the requirements list showing, for each requirement, a measurable acceptance criterion and the affected function or interface, together with the requirement to test case mapping. A small organisation checks this with a fixed wording template and the control question of whether a test case can be written from the requirement without asking back.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SR-5 **The security requirements are reviewed before they are used in design and implementation, with the affected parties involved, such as development, test, product ownership and, where useful, sales or the customer. The review covers completeness against the threat model and the product security context, correctness and testability. Findings are recorded and tracked to closure.**

What proves it: Review record with date, participants and their roles, the reviewed version of the requirements, a list of findings with an owner and closure status, and an approval note. A small organisation handles this as a two person review between development and test, with a short record and an approval line in the requirements list.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

7 Practice 3: Secure by Design (SD) 4

SD-1 Document

Every product has a design description that captures the security relevant aspects of the architecture: which components exist, how they communicate with each other, where the trust boundaries run, which external interfaces are exposed and by which mechanisms the previously defined security requirements are realised in the design.

What proves it: Architecture document or design specification containing a diagram that shows components, data flows and trust boundaries, plus a mapping table from each security requirement to the design element that implements it. A small supplier gets by with one maintained diagram plus two or three pages of explanation; what matters is that it is version controlled in the repository and updated whenever the architecture changes.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SD-2 Document

The design is systematically analysed for threats. The analysis names attacker objectives, the affected trust boundaries and interfaces, the resulting threats together with their rating, and the countermeasures taken in the design. It is repeated whenever architecture or environment changes materially, and identified threats are tracked until a decision has been taken on each of them.

What proves it: Threat model per product or per major release, for example a table with columns for threat, affected interface, rating, countermeasure and status, together with the minutes of the modelling session listing participants and date. A small team models pragmatically in a workshop of two to three hours along the architecture diagram and records open points as tickets, so the evidence of tracking is produced without extra effort.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SD-3 **The design demonstrably follows recognised secure design principles, among them least privilege, minimal attack surface, defence in depth, secure defaults, fail secure behaviour and security mechanisms that are as simple and verifiable as possible. The application of these principles is visible in the design and justified.**

What proves it: A section in the design description or an in house design guideline that names the principles, plus concrete evidence per principle in the product, such as the privilege matrix of the services, the list of open ports with justification, the shipping defaults or the documented error handling behaviour. A small supplier uses a short checklist of these principles as a fixed item in the design review and files the completed sheet with the design.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SD-4 **The design is reviewed by qualified peers before implementation starts. The review covers completeness against the security requirements, the correctness and effectiveness of the security mechanisms and the results of the threat analysis. People with sufficient security competence take part, findings are recorded and tracked to closure.**

Document

What proves it: Review record with date, item under review including its version, participants and their roles, list of findings with status and the release decision. A small team attaches the review to the pull request of the design documents and leaves the findings as comments with a closure note, which evidences tracking without any additional tooling. Where security competence is missing in house, an external reviewer engaged on an hourly basis for the critical designs is the pragmatic route.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

8 Practice 4: Secure Implementation (SI) 2

SI-1 **A defined process exists for verifying the actual implementation against the secure design: the review confirms that the protective measures foreseen in the design are genuinely present and effective in the code, that defense in depth is preserved, and that no new vulnerabilities have been introduced during implementation. The review is carried out by suitably qualified personnel, and any deviations found are recorded and tracked through to closure.**

What proves it: A written procedure for implementation review together with review records per component or release: the module reviewed, the reference to the design document, the date, the reviewer, the findings and their resolution status. Suitable evidence includes pull request reviews with a mandatory security checklist, static code analysis results with a documented assessment of each hit, and minutes of review sessions. A small team covers this with the four eyes principle, a review template held in the repository in which every design measure is ticked off individually, and one ticket per open finding. The essential point is that the author of the code is never the reviewer of that same code.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SI-2 **Binding secure coding rules apply to all security related development work and cover at least the avoidance of known insecure constructs and functions, the handling of inputs and errors, the management of memory and resources, and the use of vetted security code rather than self written implementations. The rules apply per programming language in use, are known to all developers, and are updated when needed, for example when new attack patterns or new tooling emerge.**

Document

What proves it: An approved secure coding standard with version status, scope per language and approval date, supplemented by evidence that it takes effect in daily work: the configuration of linters and analysis tools in the repository, a build rule that surfaces violations, and the attendance list of the briefing. Small organisations do not write a standard from scratch; they declare an established public rule set to be binding, add a short list of their own supplementary rules, and anchor the check in the pipeline so that adherence is demonstrated automatically.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

9 Practice 5: Security Verification and Validation Testing (SVV) 5

SVV-1 **Every security-related product requirement has at least one test case assigned to it, demonstrating that the protective function works as intended during normal operation, that it behaves as specified when given invalid input or subjected to misuse, and that the default settings shipped with the product correspond to the secure state. Execution and outcome are recorded for each software version tested.**

Document

What proves it: A traceability matrix linking requirement to test case to result, plus test records showing the date, the version or build under test, and the name of the person performing the test. A small supplier simply adds a test case ID column to its requirements list and archives the output log of the automated test run from the build pipeline as a file attached to the respective version.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SVV-2 **The countermeasures derived from the threat model are specifically tested to confirm that they actually close the assumed attack path. Testing is performed against the identified threats, not only against well-behaved use, and open items are fed into the defect handling process.**

What proves it: Test cases that reference the entries of the threat model directly, for example one abuse case per threat ID with the expected defensive behaviour, together with the records of those runs. Without heavy machinery it is enough to add a column to the threat model referencing the test case and the date of the last successful check.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SVV-3 **The product is systematically examined for vulnerabilities before release. The scope covers at least: examination of the externally reachable interfaces and services, testing with malformed and randomly mutated input, review of the third party components used against known reported vulnerabilities, and a check for known classes of weakness in the supplier's own code. Findings are recorded together with their assessment.** Document

What proves it: Reports from the tools used, stating tool name, version and the data status of the vulnerability database, fuzzing records, a list of third party components with the date of the last comparison, and the findings list with assessment and decision. A small supplier combines free standard tools, a dependency scan and a port and service scan against the installed product, and files the outputs per release in one folder.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SVV-4 **The release candidate is subjected to an attack attempt under realistic conditions that actively tries to bypass the protective mechanisms rather than merely query them. Scope, boundary conditions and limits of the test are agreed in advance, and the weaknesses found are handled and tracked like defects.**

What proves it: The engagement or test order stating scope and test environment, a report covering the approach, the weaknesses exploited and their assessment, plus the entries in the defect or ticket system with evidence of remediation or of a justified risk acceptance. Small suppliers typically buy this test externally as a bounded time budget, or have it carried out by a person who did not develop the product.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

SVV-5 **The independence of the testing personnel is defined and appropriate to the intended maturity level. Whoever developed a component does not release it on the strength of their own testing alone, and as the maturity level rises the testing is owned by an organisationally separate unit or an external party. The actual assignment is traceable for each test.**

What proves it: A definition of roles within the development process showing the separation of creation and testing, plus, for each test run or test report, the recorded name and role of the person performing the test. A two person business handles this through mutual review with a documented four eyes note, and brings in an external person for the in depth security testing.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

10 Practice 6: Security Defect Management (DM) 6

DM-1 **A defined channel exists through which reports of security weaknesses in the product are received and recorded from every direction: from the development team itself, from testing and reviews, from customers and integrators, from external reporters, and from advisories on third-party and open-source components in use. The reporting channel is discoverable by external parties, and every report is captured with its date of receipt and its source.** Document

What proves it: A description of the reporting channel, the publicly reachable point of contact (a security page, a security.txt file, or a mailbox such as security@), and the intake register listing date, source, and a short description for each report. A small organisation can work with a shared mailbox and a single spreadsheet or ticket label, as long as every report ends up there, including those raised in the internal developer chat.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

DM-2 **Every incoming report is triaged and examined to determine whether it is security relevant and which products, versions, and components it affects. Reports that turn out not to be security relevant, or not to apply, are also closed with a stated reason rather than silently discarded.**

What proves it: A triage note for each report stating the outcome (security relevant yes or no), the affected versions, and the reason for rejection where applicable, visible in the ticket or in the intake list. Teams with a low report volume can run triage in a short recurring session and record the outcome in two sentences per entry.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

DM-3 **Confirmed security defects are assessed using a consistent method: root cause, affected versions and configurations, potential impact on operation and security, exploitability, and severity against a defined scale. The result of the assessment is recorded and traceable.** Document

What proves it: An assessment record or set of ticket fields for each defect covering root cause, severity, and the scale applied (for example a common scoring system with a documented vector), together with the list of affected versions. Small teams can rely on a fixed field template in the ticket system plus half a page describing the scale and the thresholds that trigger immediate action.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

DM-4 **For every assessed security defect, a decision is made on how it will be handled: remediation, compensating measure, or a justified acceptance of the risk. The decision follows from the severity, it is assigned to a responsible person and a due date, and the defect is tracked through to closure.** Document

What proves it: An action list or ticket status showing owner, target date, type of measure, and a closure note, linked to the code baseline or the release in which the fix becomes effective. Accepted residual risks need a short written justification approved by product ownership. A small organisation can hold this in a ticket board with due dates, without any additional tooling.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

DM-5 **Affected users, integrators, and other stakeholders are informed about security defects as soon as the assessment calls for it, stating the affected versions, the impact, and the available fix or compensating measure. Timelines and the recipient scope of such disclosure are defined in advance, as is the approach to third-party reports that come with a disclosure deadline of their own.**

What proves it: Published security notices or advisories carrying a date and a version reference, the distribution list or subscriber channel, and the written rule on timelines and coordinated disclosure. Vendors with few customers can use a single mailing to a maintained customer list plus a dated advisory page whose changes are versioned.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

DM-6 **The handling of security defects is reviewed at fixed intervals. The review looks at open and overdue cases, the time from report to remediation, recurring root causes, and the effectiveness of the measures taken. Improvements to the process that follow from the review are implemented and their implementation is tracked.** Document

What proves it: Minutes of the recurring review with date, participants, figures on open and overdue cases, and a list of agreed improvements with owner and due date. A small organisation can attach this item to an existing quarterly meeting and keep the figures and decisions on a single page.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

11 Practice 7: Security Update Management (SUM) 5

SUM-1 **Every security update for the product is verified against the supported product versions and operating environments before it is released to customers, and there is a record showing that the update closes the vulnerability without disrupting intended product function.** Document

What proves it: Test record per patch covering the product version tested, the platform, the test cases (effectiveness against the vulnerability plus regression of core functions) and the release approval. A small supplier can work with a short patch approval entry in the ticket: ticket ID, verified by, verified on, result, with the log of the automated test run attached.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SUM-2 **For all product components that were not developed in house (third party libraries, operating system, runtime environments), it is defined and evidenced how their security updates are assessed and either adopted into the product or rejected with a documented rationale.** Document

What proves it: Component inventory of the third party parts with version and source, plus one assessment entry per reported third party vulnerability stating the decision (adopt, not affected, covered by a compensating measure) and the rationale. Small teams run an automated dependency scan in the build and keep the decisions as comments on the scan result or in a lean exception list.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SUM-3 **For every security update that is shipped, users receive understandable accompanying information: which product versions are affected, which vulnerability is fixed, how severe it is, whether side effects or compensating measures need to be observed, and how the update is installed.** Document

What proves it: Published security advisories or release notes stating the vulnerability identifier, the affected versions, the severity rating, the installation steps and guidance on rolling back to the previous state. Small suppliers keep a single public page with a chronological list of security advisories and link to it from within the product.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SUM-4 **There is a defined and tamper protected route through which users obtain security updates, and the authenticity and integrity of an update can be verified on the user side before it is installed.**

What proves it: Description of the delivery channel (download area, update server, physical media) together with its protective measures, plus the digital signature or checksum per package and instructions telling the user how to verify it. Small suppliers achieve this by signing on the build server, publishing the checksum next to the download and naming one clear location where the public key is held.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SUM-5 **Defined time targets exist for making security updates available, graded by the severity of the vulnerability; the times actually achieved are tracked, and any target that is missed is justified and backed by an interim measure for users.** Document

What proves it: Internal specification with a time window per severity category (for example critical <= 30 days, high <= 90 days) and an evaluation per vulnerability: report date, patch release date, elapsed days, reason for deviation. Small suppliers keep this as two date fields in the vulnerability ticket and review the list once a quarter instead of setting up a dedicated metrics system.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

12 Practice 8: Security Guidelines for Product Deployment and Operation (SG) 7

SG-1 **Every product comes with a description of its defense in depth strategy: it names the protective layers built into the product itself, maps them to the threats they counter, and shows which protective effect only emerges from the interplay of several layers.** Document

What proves it: A section in the delivered product documentation, or a separate security concept, that states for each protective layer the mechanism, the threat covered and the limits of its effect. A small organisation keeps this as a two page table: column protective layer, column threat taken from its own threat model, column residual risk. A cross reference to the threat model from SR-2 is sufficient, the content does not have to be maintained twice.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-2 **The documentation names the security measures the product does not provide itself but expects from the deployment environment, for example network segmentation, upstream firewalls, physical access control or an external time service, and makes clear that the assured security level is not reached without these measures.** Document

What proves it: A deployment prerequisites chapter in the manual, or a reference architecture drawing with zones and conduits in which the measures to be supplied by the asset owner are explicitly marked. A list of assumptions about the environment has proven useful in practice, derived directly from the product security context and re-read for every product version.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-3 **A secure baseline configuration guide exists that lists all security relevant settings, services, ports and interfaces, states the recommended as delivered condition and describes the impact of deviating from that recommendation.** Document

What proves it: Hardening guide with a settings table: parameter, recommended value, effect, side effect when deviating. Plus the list of services and ports active in the default configuration with a rationale for why they stay enabled. A small team keeps this table right next to the configuration file in the repository and exports it into the delivery documentation, so it stays aligned with the product.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-4 **For ongoing operation there is a guide describing how the product is operated securely, monitored and kept in its secure state, including backup and restore, logging and the handling of security relevant event messages.** Document

What proves it: Operating guide with sections on logging, backups, restart and event messages. As additional evidence, a list of the security relevant messages the product emits, each with a short recommended action, so the asset owner does not have to interpret them alone.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-5 **The documentation explains how user accounts, roles and permissions of the product are set up and maintained, in particular separation of roles, changing default credentials and removing accounts that are no longer needed.** Document

What proves it: An account management chapter with an overview of roles and the rights assigned to each role, plus an explicit note on default credentials and on changing them at commissioning. A short commissioning checklist in the manual works well in practice, one the asset owner can tick off and file.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-6 **A guide for secure decommissioning exists that describes the complete removal of sensitive data from the product, such as credentials, keys, configuration data and logs, before the device is handed on, reused or disposed of.** Document

What proves it: A decommissioning section listing every storage location of sensitive data inside the product and the corresponding erasure procedure, including the cases where erasure is technically impossible and the hardware has to be physically destroyed. A list of persistent storage taken from the architecture description helps here, it prevents a storage location from being forgotten.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

SG-7 **The security documentation delivered to users is reviewed before release: it is technically correct, consistent with the delivered product version, complete against the intended content and understandable for the intended audience, and it is updated whenever the product changes.**

What proves it: A review record per document and product version with reviewer, date, findings and their closure, linked to the release approval. A small organisation solves this with a four eyes principle in the documentation pull request: one developer checks the technical content, a second person checks understandability, and the approval is recorded in the ticket. A version is only shippable once the review is documented.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification