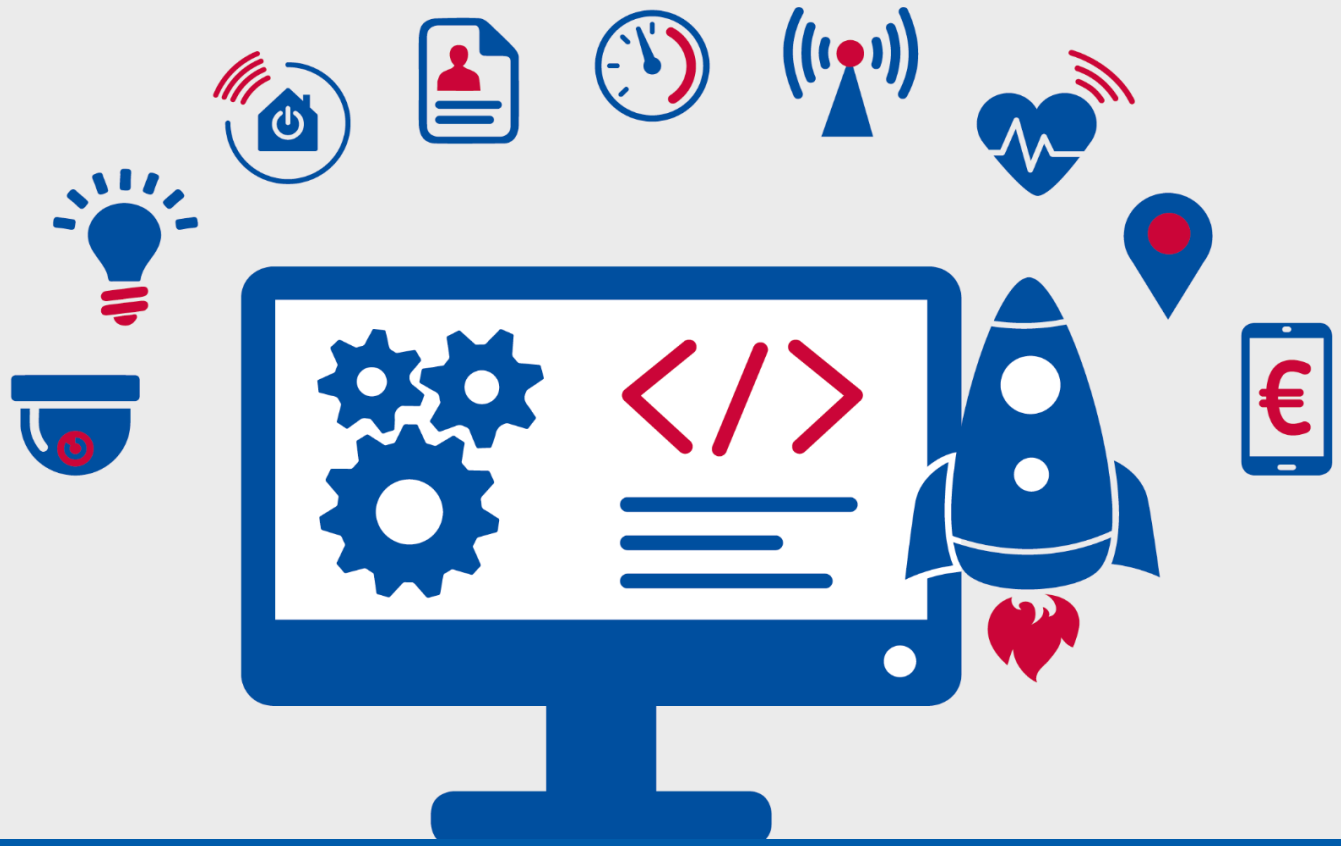
EUROPEAN UNION AGENCY
FOR CYBERSECURITY



GOOD PRACTICES FOR SECURITY OF IOT

Secure Software Development Lifecycle

NOVEMBER 2019

ABOUT ENISA

The European Union Agency for Cybersecurity (ENISA) has been working to make Europe cyber secure since 2004. ENISA works with the EU, its member states, the private sector and Europe's citizens to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. Since 2019, it has been drawing up cybersecurity certification schemes. More information about ENISA and its work can be found at www.enisa.europa.eu.

CONTACT

For contacting the authors please use iot-security@enisa.europa.eu
For media enquiries about this paper, please use press@enisa.europa.eu

AUTHORS

ENISA

ACKNOWLEDGEMENTS

Alessandro Cosenza	Bticino S.p.A
Arndt Kohler	IBM
Benedikt Abendroth	Microsoft Corporation
Carlos Valderrama	Geomantis Corporation Limited
Alex Cruz Farmer	Cloudflare
Cédric Lévy-Bencheton	Cetome
Eric Vetillard	NXP
Filip Chytry	Avast
Hannes Tschofenig	ARM Ltd.
Hagai Bar-El	ARM Ltd.
Ian Smith	GSM Association (GSMA)
Antonio Jara	HOP Ubiquitous S.L. (HOPU)
Julio Hernandez-Castro	University of Kent
Mirko Ross	asvin.io
Mark Harrison	Pentestpartners
Sylvie Wuidart	STMicroelectronics
Tiago Da Costa Silva	Cisco
Jeff Schutt	Cisco
Evangelos Gazis	Huawei Technologies Co., Ltd.
Viacheslav Zolotnikov	Kaspersky
Ekaterina Rudina	Kaspersky
Wolfgang Klasen	Siemens AG
Pierre Kobes	Siemens AG
Yun Shen	Symantec
Adrien Becue	Airbus
Dharminder Debisarun	Palo Alto Networks
Denis Justinek	Biokoda
Ernie Hayden	Jacobs
Georges-Henri Leclercq	Engie
Gisele Widdershoven	Accenture
Jalal Bouhdada	Applied Risk
Jens Mehrfeld	BSI
Konstantin Rogalas	Honeywell

Pascal Oser	CERN
Pirmin Heinzer	Reporting and Analysis Centre for Information Assurance MELANI
Rafal Leszczyna	Gdansk University of Technology
Roberto Minicucci	BHGE
Samuel Linares	iHackLabs
Stefano Zanero	Politecnico di Milano
Victor Fidalgo Villar	INCIBE (The Spanish National Cybersecurity Institute)
Vytautas Butrimas	NATO Energy Security Center of Excellence
Roger Jardí-Cedó	Nestlé S.A.
Aaron Guzman	OWASP
Maor Vermucht	VDOO
Tommy Ross	BSA
José Alejandro Rivas Vidal	Applus+ Laboratories
Dirk-Willem van Gulik	Web Weaving

LEGAL NOTICE

COPYRIGHT NOTICE

ISBN 978-92-9204-316-2, DOI: 10.2824/742784

TABLE OF CONTENTS

1. INTRODUCTION 7

1.1 OBJECTIVES 8

1.2 SCOPE 8

1.3 TARGET AUDIENCE 8

1.4 METHODOLOGY 8

1.5 STRUCTURE OF THE DOCUMENT 10

2. IOT SECURE SDLC 11

2.1 REQUIREMENTS 12

2.2 SOFTWARE DESIGN 13

2.3 DEVELOPMENT/IMPLEMENTATION 15

2.4 TESTING AND ACCEPTANCE 16

2.5 DEPLOYMENT AND INTEGRATION 17

2.6 MAINTENANCE AND DISPOSAL 18

2.7 SECURITY IN SDLC 19

3. ASSET AND THREAT TAXONOMY 20

3.1 ASSET TAXONOMY 20

3.2 THREAT TAXONOMY 26

3.3 EXAMPLES OF ATTACK SCENARIOS 43

3.3.1 Insecure Credentials in Embedded Devices 43
3.3.2 Lack of Flexibility to Secure Communications 45
3.3.3 Insecure Software Dependencies in Cloud Services 47

4. GOOD PRACTICES FOR SECURE IOT SDLC 49

4.1 SECURITY CONSIDERATIONS 49

4.2 GOOD PRACTICES 50

4.2.1 People 51

4.2.2 Processes 52
4.2.3 Technologies 54

A ANNEX: MAPPING OF SECURITY MEASURES 57

B ANNEX: SDLC STANDARDS AND BEST PRACTICES 119

C ANNEX: SECURITY IN SDLC MODELS 126

D ANNEX: IOT SDLC TESTING 129

EXECUTIVE SUMMARY

This ENISA study introduces good practices for IoT security, with a particular focus on software development guidelines for secure IoT products and services throughout their lifetime. Establishing secure development guidelines across the IoT ecosystem, is a fundamental building block for IoT security. By providing good practices on how to secure the IoT software development process, this study tackles one aspect for achieving security by design, a key recommendation that was highlighted in the ENISA Baseline Security Recommendations study which focused on the security of the IoT ecosystem from a horizontal point of view.

Software lies at the core of every IoT system and service, enabling their functionality and providing value added features. The firmware of IoT devices, implementations of IoT communication protocols and stacks, Operating Systems (OSs) for IoT products, Application Programming Interfaces (APIs) supporting interoperability and connectivity of different IoT services, IoT device drivers, backend IoT cloud and virtualization software, as well as software implementing different IoT service functionalities, are some examples of how software provides essence to IoT. Due consideration to supply chain issues, including integration of software and hardware, is given.

Making use of secure Software Development Life Cycle (SDLC) principles is an effective and proactive means to avoid vulnerabilities in IoT and thus assist in developing software applications and services in a secure manner. Several security challenges of the IoT can be addressed by establishing a baseline of secure development guidelines, such as checking for security vulnerabilities, secure deployment, ensuring continuity of secure development in cases of integrators, continuous delivery etc.

It is therefore important to analyze the relevant IoT cybersecurity threats and accordingly to set forward security measures and specific secure development guidelines to avoid common software vulnerabilities deriving from insecure practices that might be followed throughout the SDLC (requirements analysis, software design, software development, implementation, deployment, integration, maintenance and disposal).

The main contributions of the study include:

- Analysis of security concerns in all phases of IoT SDLC and key points to consider.
- Detailed asset and threat taxonomies concerning the IoT secure SDLC.
- Concrete and actionable good practices to enhance the cybersecurity of the IoT SDLC.
- Mapping of ENISA good practices to related existing standards, guidelines and schemes.

The study is mainly targeted at IoT software developers, integrators and platform and system engineers and aims to serve as a point of reference for secure IoT development. Security considerations and guidelines for all phases of software development are provided, starting from requirements, software design and development/implementation, all the way to testing and acceptance, integration and deployment, as well as maintenance and disposal.

The study underlines the need to consider end-to-end IoT security, not only focusing on smart devices, network protocols and communications, but also taking a step back and methodically integrating cybersecurity by design principles throughout the software development lifecycle.

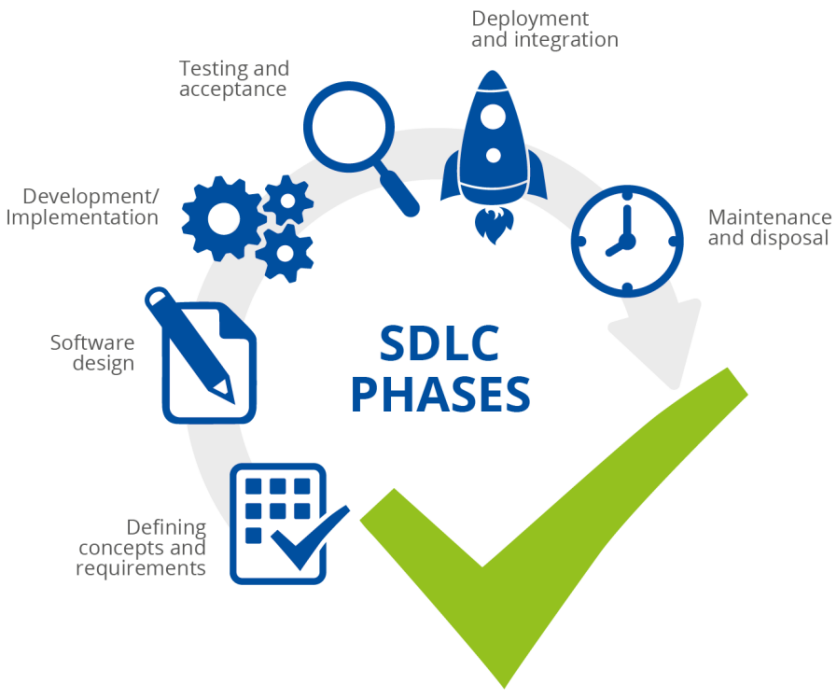
1. INTRODUCTION

IoT is at the core of operations for many Operators of Essential Services (OES), as defined in the NIS Directive, especially considering recent initiatives towards Smart Infrastructures, Industry 4.0, 5G, Smart Grids, etc. With a great impact on citizens' safety, security and privacy, the IoT threat landscape is extremely complex. Therefore, it is important to understand what exactly needs to be secured and to implement specific security measures to protect the IoT from cyber threats. ENISA has published studies on both baseline IoT security recommendations, as well as sectorial IoT security good practices (e.g. smart manufacturing, smart cars, smart hospitals, etc). While the horizontal and vertical IoT security measures greatly assist in reducing relevant risks, the design, development, deployment and configuration of secure IoT solutions should not be neglected.

ENISA strongly recommends security and privacy by design and by default. Accordingly, an effective and proactive means to reduce the number and severity of vulnerabilities in IoT is to develop applications in a secure manner, making use of secure Software Development Life Cycle (sSDLC) principles and developers trained in secure coding. Several security challenges of the IoT can be addressed by establishing a set of secure development guidelines, such as checking for security vulnerabilities, secure deployment, ensuring continuity of secure development in cases of integrators, continuous delivery etc.

In this regard, the aim of this study is to define a set of good practices and guidelines to be applied in the different phases of the secure SDLC of IoT solutions. During this study, experts were asked what the main phases of the SDLC were. The vast majority of them considered that the SDLC comprises up to six phases, as shown in Figure 1.

Figure 1: SDLC phases



1.1 OBJECTIVES

This ENISA study aims to address the cybersecurity challenges related to the SDLC of IoT systems and services. The main objectives were to collect good practices, to foster cybersecurity across the different phases of the IoT SDLC, while also mapping the relevant assets, threats, risks and attack scenarios.

To this end, the following objectives have been set:

- Analyse the different IoT SDLC phases and underline key cybersecurity challenges in each one.
- Identify IoT SDLC assets to protect.
- Identify key cybersecurity threats and attack scenarios targeting the IoT SDLC.
- Map identified threats to assets.
- Identify security measures and map them to attack scenarios and threats.
- Identify SDLC principles for IoT code developers.

Accordingly, the study aims to promote collaboration for IoT security in Europe and to increase awareness of threats and risks, with particular focus on the secure SDLC of IoT systems and services. In addition, the study will serve as a reference point for future developments and provide a solid basis for securing IoT software from the requirements analysis to maintenance and disposal.

1.2 SCOPE

This ENISA study outlines good practices for IoT security with a particular focus on securing SDLC of IoT systems. This entails defining security measures that apply to the entire IoT ecosystem (devices, communications/networks, cloud, etc.) in order to bolster the security of the development process.

During this study, ENISA identified available documentation and standards on IoT security, with a focus on SDLC and its different phases. ENISA also collected inputs from a number of IoT security experts through a questionnaire and a series of interviews. Following a thorough analysis of the identified material and the review of security experts feedbacks, ENISA identified the main IoT assets and threats targeting the SDLC. Based on these threats, a set of security measures and good practices were defined to ensure integration of security across the different phases of the IoT SDLC.

1.3 TARGET AUDIENCE

This study defines good practices for security of IoT, focusing on securing the SDLC of IoT systems and services. Given the diverse phases that SDLC entails and the complexity of the IoT ecosystem, the target audience of this study comprises the following profiles:

- IoT software developers
- IoT platform, Software Development Kit (SDK) and Application Programming Interface (API) developers and consumers
- IoT integrators

1.4 METHODOLOGY

This ENISA study was carried out using a five-step methodological approach as shown in Figure 2.

1. **Scope definition and identification of experts:** The first step was to establish the scope of the study and to pinpoint the main topics to be considered. A concurrent activity involved identifying the relevant IoT subject matter experts to contribute. The experts (members of ENISA informal expert groups on IoT and Industry 4.0 security,

IoTSec and EICS respectively) provided input and expertise in relation to the objectives of this report.

2. **Desktop research:** Extensive research of relevant efforts to gather as much information as possible on securing the IoT SDLC and secure SDLC in general. The identified documents included existing good practices, publications, standards and other initiatives on the topics related to the objectives of the report. This served as support for the analysis of the threats and for the development of the security measures.

3. **Questionnaire and interviews with identified experts**: ENISA reached out to the identified experts in order to collect information and get their point of view. To this end, an online questionnaire covering various security aspects, such as critical assets, key threats targeting IoT SDLC and awareness with respect to IoT SDLC standards and guidelines, was developed. The questionnaire was completed by the identified experts, and interviews were conducted with experts[1] to collect additional valuable inputs to prepare the report.

4. **Analysis and development:** The results from the desktop research, online questionnaire and the interviews were analysed to align them with the objectives of the report, developing the asset and threat taxonomies. This helped to identify the attack scenarios, as well as the IoT SDLC security measures. This led to the development of the first draft of this report.

5. **Report write-up and validation:** ENISA shared the draft of the report with its relevant stakeholder communities and reference groups for review. Taking into account the stakeholders feedbacks, the final version of the report was issued and a validation face-to-face workshop was organized (on the 8th of October 2019 in Brussels, Belgium) to present the study results and discuss relevant cybersecurity recommendations.

Figure 2: Methodology followed in the study

[1] 41 experts filled in the questionnaire. Interviews with experts who expressed availability were conducted with the aim to cover the various aspects of the IoT ecosystem (service providers, hardware manufacturers, developers, integrators, etc.).

1.5 STRUCTURE OF THE DOCUMENT

The report is structured as follows:

- **Chapter 1 - Introduction:** provides introductory information to the report and introduces the scope, objectives, and the methodology followed.
- **Chapter 2 – Secure IoT SDLC:** discusses cybersecurity considerations in the different phases of IoT SDLC. An asset taxonomy is also presented in accordance with their perceived criticality.
- **Chapter 3 – Asset and Threat taxonomy:** identifies the security threats affecting IoT SDLC and details some examples of potential attack scenarios. Detailed description of threats and mapping to the corresponding assets that they might impact.
- **Chapter 4 – Good practices for security of IoT SDLC:** lists and describes good practices and security measures to secure the IoT SDLC.

Further details are provided in the appendix:

- **Annex A:** Comprehensive description of security measures discussed in Chapter 4 and mapping of the security measures to previous work carried out in the field and to the corresponding threats that they are intended to mitigate.
- **Annex B:** List of standards, good practices, security initiatives and other works that have been used in the mapping of Annex B.
- **Annex C:** Introduction of the notion of security across different IoT SDLC models.
- **Annex D:** List of IoT SDLC testing solutions and methodologies.

2. IOT SECURE SDLC

Software lies at the core of every IoT system and service, enabling their functionality and providing value added features. The firmware of IoT devices, implementations of IoT communication protocols and stacks, Operating Systems (OSs) for IoT products, Application Programming Interfaces (APIs) supporting interoperability and connectivity of different IoT services, architectures that enhance the IoT interoperability, such as Manufacturers Usage Description (MUD)[2], IoT device drivers, backend IoT cloud and virtualization software, as well as software implementing different IoT service functionalities, are some examples of how software provides essence to IoT.

However, the pervasive nature of software across the IoT ecosystem and its role as a central cog in the entire IoT supply chain[3], bring security risks. Adversaries may exploit software vulnerabilities to compromise the security of IoT systems and services and impact the proper operation of such systems and services. The entire IoT ecosystem, taking into consideration also the Internet and the external physical systems that make use of IoT need to be taken into consideration when calculating risk. It is therefore evident that systematically securing IoT software is essential throughout the lifetime of IoT systems and services in order to deliver resilient, reliable and failsafe solutions. In this respect, the IoT Software Development Life Cycle (SDLC) as a whole needs to be secured and proper considerations to be taken into account by all involved stakeholders from the beginning of the software development process up to maintenance and disposal.

However, securing IoT, and especially IoT edge-devices, can prove a difficult task for software developers if hardware comes without basic security capabilities. For example, when implementing a strong cryptographic algorithm in the software stack, it is the use of a Trusted Platform Module (TPM) in the hardware that will ensure the private key cannot be exposed. Therefore, software development for IoT cannot neglect the underlying hardware, which in turn entails that the security approach has to conceive it as a set where the design of hardware influences the design of software. Elements such as the Root of Trust or Chain of Trust are good examples of how software and hardware are related and interconnected and result in joint security considerations to confront current IoT vulnerabilities such as vulnerabilities in communication stack derived from hardware implementation (e.g. which could be faced implementing a hardware isolation and secure boot).

SDLC is a process consisting of different phases that aims at delivering effective and efficient systems as per their design and functional requirements. There are many ways to achieve this goal, which are represented by various SDLC models as described in the following. Accordingly, incorporating security considerations takes place in a different manner based on the adopted SDLC model. By methodically considering security across all phases of IoT SDLC and applying appropriate security measures on the corresponding assets that may be affected, the overall security of the IoT ecosystem is improved.

[2] See https://datatracker.ietf.org/doc/rfc8520/
[3] See https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/vc-security-infographic.pdf

Securing the IoT SDLC process involves securing the SDLC process across all elements of the IoT ecosystem, namely IoT end devices, communications, cloud backend and applications for mobile devices for controlling devices[4]. Moreover, all types of software running on the aforementioned elements should be considered, including but not limited to end device firmware, IoT services/software implementations, network protocol implementations, API source code, IoT gateways source code, software running on backend cloud servers, etc.

It becomes clear that the complexity and heterogeneity of the different IoT elements and types of IoT software both exacerbate cybersecurity issues and therefore there is a growing need to come up with homogeneous, good practices for securing the SDLC. Security considerations of the different IoT SDLC phases are discussed in what follows.

2.1 REQUIREMENTS

Requirements are the foundation for all that is to follow in the IoT development cycle. During this phase user, business and functional requirements of the software are being defined.[5] These requirements reflect the intended use of the software and will be translated to specifications that will guide design, development and maintenance/deployment decisions at a later stage. Accordingly, it is essential to consider security from this first phase of software development, in order to ensure as much as possible that security by design principles are taken on board and that security does not come as an afterthought.

In this respect, during the requirements phase it is essential to conduct a preliminary identification of software security aspects taking into account the aforementioned (user, business, legal, regulatory and functional) requirements[6]. Indicative security requirements include user password change policies, the need to implement a business recovery plan, the ability to stay up to date, etc.), cost – benefit and risk analysis results, as well as ones that refer to the external environment. The latter include for example third-party dependencies, security standards and/or certification objectives, potential IoT threats, possible IoT attack vectors, etc. For the sake of quality assurance, it is considered a good practice to monitor and review the requirements periodically throughout the SDLC. The former will ensure that security is in line with the general requirements of the software and ensure consistency in development. Security engineers will need to work with software engineers and business analysts in order to guarantee the optimal convergence of the two fields. The latter are externalities to the software itself and usually outside the realm of control of the software engineers. However, they are used to define security assumptions about the software under development. For example, likelihood of critical threats, levels of trust to be placed on user accounts, likelihood of attack vector realization, etc. should be considered. Therefore, the requirements phase in the context of security yields two outputs: a set of security requirements that depends on the context (connectivity type, target environment specifics, etc.), as well as a set of security requirements that depends on the functionalities offered by the solution (business or use cases)[7].

Additionally, another important aspect to consider during the identification of requirements is the physical or hardware requirements needed for development (functional requirements), since software and hardware are closely related. As part of the system, the security requirements of software may have certain implications when selecting the physical media (hardware), which have to be addressed during the SDLC process. Specifically, the security requirements in the Requirements phase will entail considerations for the selection of hardware in the definition of the architecture during the Design phase. For instance, if the implementation of a secure boot

[4] This classification is based on the IoT high-level reference model introduced in ENISA's Baseline IoT Security Recommendations study, available at: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot
[5] See https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
[6] Personnel training and establishment of clear and well-defined processes is beneficial in meeting certain cases of requirements, e.g. legal or regulatory.
[7] See https://www.securesoftwarealliance.org/FrameworkSecureSoftware_v1.pdf

mechanism is selected as a requirement, it will be necessary for hardware to support this type of Root-of-Trust (RoT) mechanism, potentially requiring the inclusion of physical hardware security modules (HSM) to manage cryptographic keys. Thus, physical requirements become as much an aspect for consideration as requirements for communications, processing capacity required, hard disk space, etc.

To ensure consistency of all above types of requirements, one commonly used technique is that of quality gateways[8]. These gateways receive requirements as input and they check them for completeness, relevance, testability, coherency, traceability and several other qualities. In the context of security, other interesting techniques include bug bars[9]. A bug bar is an example of quality gates, which is used to define the severity thresholds of security vulnerabilities (E.g. no known vulnerabilities in the application with a “critical” or “important” rating at time of release).

Risk analysis, as well as making use of best practices documents (e.g. OWASP IoT Top10[10]) and guidelines[11] also help to secure software development by means of predefined checklists of most common security risks and pitfalls. Risk analysis also involves identification of the assets that will comprise the software system or service, as well as their interactions and external dependencies. This may be used to pinpoint asset criticality, data flows and allowed operations on data, thus yielding significant input to improve software security.

While drawing up requirements, the concept of threat modelling should be borne in mind. It assumes that potential threats, such as structural vulnerabilities, can be identified, enumerated, and prioritised.[12] Most commonly, the STRIDE (Spoofing, Tampering, Information Disclosure, Repudiation, Denial of Service and Elevation of Privilege) methodology is used to identify and classify threats. Threat modelling starts in the requirements phase with the identification of critical assets and is completed in the software design phase, when the risks have been evaluated and ranked, and their mitigation has been planned [13].

With IoT, the digital and physical worlds are no longer kept apart from one another. The cyber physical nature as well as the diverse components and application domains of IoT introduce additional parameters in the threat modelling equation. These include the consideration of industry-specific threats that may apply, such as interoperability with legacy-coded and outdated devices.

Given the dynamic evolution of IoT ecosystems and their inherent adaptation to changing context, it is evident that requirements identification should cater for flexibility throughout the lifetime of IoT products and services. In terms of securing the IoT SDLC, this translates to iterative rounds of security requirements identification and the need to consider all possible use case scenarios of the IoT system or service.

2.2 SOFTWARE DESIGN

During the software design phase the on device architecture and the design of the IoT solution are created. This phase involves the creation of a set of documents that describe how the user/business and functional requirements will be translated to system specifications and essentially how the IoT solution will work. Therefore, it is important to make sure that

[8] See http://ptgmedia.pearsoncmg.com/images/9780321815743/samplepages/0321815742.pdf
[9] See https://docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307404(v=msdn.10)
[10] See https://www.owasp.org/images/1/1c/OWASP-IoT-Top-10-2018-final.pdf
[11] See https://www.owasp.org/images/6/67/OWASPApplicationSecurityVerificationStandard3.0.pdf
[12] See https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html
[13] See https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf

specifications meet all the requirements defined in the previous phase. For instance, in IoT devices the specifications for user passwords should be included as requirement. In turn, during the design phase, this would entail implementing functions to manage user passwords as required (change cycles, minimum password length, special symbols, etc.).

From a security standpoint, a risk-based approach that identifies pertinent threats is followed to incorporate security in the software design. The security requirements established in the previous phase are being reviewed using threat modelling and attack surface analysis techniques. Threat modelling is the backbone of the security activities [14] carried out during the definition of security requirements, but can also be further refined during software design. Additionally, attack surface analysis should consider the attacker's potential motivations, intentions and capabilities, as well as the attack avenues of the system, the impact and the probability. As stated in requirements section, risks can also be entailed by business features. For this reason, it is necessary to design specific security controls to prevent such potential issues.

Effective identification and mitigation of security threats in early design phase needs to be prioritized as it can be very hard to mitigate them in later phases of the development lifecycle. [15] Moreover, other provisions in the software design, such as the chain of trust and recovery plan of the solution and the integration of security mechanisms (FOTA, remote credential management, etc.) in this phase safeguards the operation of IoT systems and prevents costly security implementations into IoT solutions after they have been developed or during the remote sustain/maintenance[16].

Architectures primarily focus on overarching, cross-cutting concerns for the IoT system that pursue mainly high scalability and integration of diverse technologies and systems. A security architecture for IoT relies mainly on the CIA triad (Confidentiality, Integrity, and Availability). In addition, other relevant aspects like access control, policy configuration, or security lifecycle should be considered. Guidelines, secure design patterns and principles aid in this mission. These principles apply to the design of any IoT solution, although there may be variations depending on the specific functionalities or limitations of the component being designed or the context in which is going to operate. For instance, cloud platforms require privilege-based administration roles, and, depending on the use case, IoT devices may require updates and measures that reassure minimum disruption[17].

However, IoT is highly related to the cyber-physical world and, in this respect, in addition to the CIA triad, safety implications are pertinent and particularly relevant when designing an IoT solution. Due to the use of sensors and actuators that act as frontiers between the logical and physical worlds, safety aspects and considerations should also be considered during the design phase. For instance, if the design principles are not used, a software solution could be built without taking into account the "least privilege principle", so an attacker could leverage this vulnerability to take control of an automated process and cause a process malfunction, which in turn could have a significant impact on human safety.

[14] See https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf
[15] See https://www.vdoo.com/blog/integrating-security-into-the-iot-sdlc/
[16] See https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf
[17] See https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf

2.3 DEVELOPMENT/IMPLEMENTATION

In the Development/Implementation phase, specifications and software design diagrams written in an appropriate notation are transposed into code. Therefore, what is defined in the two previous phases plays a crucial role in the successful execution of the development process.

The foundation of IoT secure software development relies on secure code. Code should be built, tested, integrated, maintained, and updated with security aspects in mind. Risk mitigation as evaluated by means of the threat model carried out in the Design phase is implemented in the code. Given the hardware and software constraints of IoT devices, integrating security in the code of this ecosystem poses challenges to developers as they can't build a code base with full-fledged security as in traditional IT systems. Code must be optimised to ensure that every instruction counts but, at the same time, secure development practices (e.g., SANS Top 25 Software Errors[18]) and appropriate security solutions for all different elements, such as access interface, applications, data and device layers, need to be leveraged. For example, in IoT communications, the use of lightweight authentication and encryption have proved to be good security techniques that fit to the specific purpose.[19]

In the IoT space, product releases may vary in frequency. Coding needs to keep the pace and to that end, the use of secure code guidelines and coding standards[20] [21] can prove extremely helpful to the developers to identify known vulnerabilities, avoid insecure coding practices and use the built-in security specific features that certain programming languages may offer. Additionally, there are tools and methods to verify the quality of security for software development languages. Secure IoT frameworks offer to developers a rapid and effective manner to integrate security components, prevent security weaknesses and provide security by design from the beginning of the development.[22]

To ensure continuous secure software coding operations, software development should be accompanied by continuous integration of security best practices and assessments. Fundamental testing activities should be consistently addressed in this phase. This way, only a signed-off build is propagated to the next phase. A realistic way of maintaining security in an environment that grows so rapidly and changes so quickly is to automate it. Automated tools such as Static Code Analysis tools are a useful complement to manual inspection of code to help detect security issues during the software development phase. [23] Static Application Security Testing (SAST) methodology allows the automation of the security process and enables early elimination of application-layer vulnerabilities. Although SAST takes place in the case of continuous integration (Agile / DevOps / DevSecOps) it can be carried out manually in other software methodologies. Conversely, other less automated activities- like code review, build environment, anti-tampering techniques, and configuration management[24] [25], -complement the testing and verification process determined to improve resilience.

Especially for IoT, it is a common practice for developers to consume third-party APIs, frameworks, libraries and tools (either commercial off-the-shelf-components (COTS) or open source software (OSS)) for the compilation and the build process. In a world where there is no need to reinvent the wheel, this approach brings numerous benefits as it allows developers to focus on product-specific features and reduce time to market and development costs. However,

[18] See https://www.sans.org/top25-software-errors/
[19] See https://pdfs.semanticscholar.org/6aec/74231f1716bd350b1c60b2bc3168471e1c13.pdf
[20] See https://www.iotcentral.io/blog/security-first-design-for-iot-devices
[21] See https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf
[22] See https://github.com/zettajs/zetta/wiki
[23] See https://www.owasp.org/index.php/Static_Code_Analysis
[24] See https://www.iotcentral.io/blog/security-first-design-for-iot-devices
[25] See https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf

these third-party components are often treated as black boxes and are less scrutinized than internally developed components; hence they come with risk. It is essential to be aware of the vulnerabilities that derive from these components and they should be considered and evaluated before integrating them into the IoT system. For example, specific parameters should be checked before using these external components, such as active community support for OSS components, current market use of these components, integration-ready state, latest version available, supporting documentation, frequency and number of reported CVEs, etc. It is also a good practice to use well-established and secure libraries and frameworks, so that the final outcome is less prone to inherit security vulnerabilities of the components that it integrates. [26,27,28,29]

IoT development requires a standard approach to developing secure products. When developing software, it is important to organise the process so that developers can work on new versions that are less vulnerable or provide better services. Configuration management integrates processes, policies and tools to make software systems more secure and flexible. In this regard, Version Control Systems (VCS), secure bootstrap capability and other equivalent methods become noteworthy means to achieve these goals.[30]

2.4 TESTING AND ACCEPTANCE

The testing and acceptance phase involves all necessary steps to verify that the developed software actually meets the identified requirements and design principles of the previous phases. For this reason, a variety of tests (each serving a different purpose) are performed on the software. These tests may be automated or manual, and both the source code (static analysis) and the running software (dynamic analysis) need to be checked. Automated tests may significantly reduce the required time to conduct them compared to manual tests and they can also increase consistency and efficiency by being highly scalable. Conversely, they introduce an additional degree of uncertainty in the testing phase and need to be redesigned to be more effective. This uncertainty results from the size of the code base to which the tests are applied and from the poor design for a larger scale of applications, since they can produce varying numbers of false positives/negatives compared to manual testing[31]. During the testing phase, it is therefore important to assess the particular needs of the software product and establish the most suitable and efficient testing strategy and build the appropriate testing environment (e.g. simulated or emulated environment, digital twin, test datasets, capturing of outputs for post-processing, fuzzing, pentesting, sandboxing, etc.).

In terms of security, testing during this phase (e.g. fuzzing testing,- see Annex D) helps to verify the proper and effective use of identified security measures and controls, as well as to identify and highlight potential vulnerabilities and weaknesses already present in the developed software prior to integration and deployment. In particular for the IoT SDLC testing phase, it is important to consider all elements of the IoT ecosystem as previously described, i.e. IoT end devices, firmware and communications[32,33]. There is an added level of complexity in designing and building an appropriate testing strategy and a testing environment for IoT systems and services, given the many interdependencies of the numerous fundamental elements of IoT. It is

[26] See https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf
[27] See https://safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf
[28] Third party components from proprietary source may come with a vendor commitment but less transparency and unverified security properties. Open-source based components come with no such commitment but with full transparency and the advantage that a large community can maintain it in security conditions.
[29] See https://github.com/jeremylong/DependencyCheck/blob/master/RELEASE_NOTES.md#version-523-2019-11-11
[30] See https://www.bosch-si.com/iot-platform/insights/downloads/iot-security.html
[31] See https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf
[32] See https://www.iotcentral.io/blog/security-first-design-for-iot-devices
[33] See https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf

thus important to carefully plan and ensure that all interfaces, data flows and externalities are fully and correctly assessed during testing and acceptance.

An important aspect of testing and acceptance is that of code review. While this may be considered an advanced and time-consuming solution (especially considering the large variety of many IoT systems and services and the frequent use of third-party libraries), this type of test may expose underlying software vulnerabilities and weaknesses that are undetectable by other techniques (e.g. logic bombs).

For a comprehensive listing of SDLC tests, the reader may refer to Annex D. Based on risk and threat assessment and taking into account security requirements and software specifics, the selection of the most suitable testing suite should be performed.

2.5 DEPLOYMENT AND INTEGRATION

The deployment and integration phase follows the acceptance of the software subject to successful testing in the previous phase, i.e. after it has been approved for release. It involves integrating all necessary elements of the software in the production environment and its deployment. Deployment should be carefully planned, executed and communicated to all actors involved (e.g. end users, production teams, development teams, integrators, etc.), in order to ensure a smooth deployment in the target environment. This is particularly challenging in the IoT realm, given the many interdependencies involved and the fact that IoT solutions are usually deployed in widely open environments, where administrators and the support team might not have full control. Another particularity of IoT deployment involves the heterogeneity of deployment environments, e.g. IoT devices for firmware solutions, cloud-servers for back-end IoT services, gateways and network components in the case of IoT communication protocols implementations, etc. It may also be the case that an IoT software project could require aspects of all these possible deployment environments or target environments (e.g. whether an IoT device meets the requirements, or if the security configuration of an IoT device needs to be adapted to the target environment). Therefore, choosing the right deployment strategy (e.g. Canary, A/B testing, Blue/Green, etc.[34]) is of paramount importance and requires to weight options like the impact of change on the system and/or on the end-users, rollout/rollback timings, downtime requirements, etc.

An important step in secure IoT deployment is that of asset and user authorization. To this end, adaptive user-rights administration interfaces, device authentication and user authentication mechanisms are beneficial. Additionally, IoT systems allowing self-enrolment should provide a means for an administrator to check and accept or reject the enrolment request.

Furthermore, security risks involved with deployment should be given proper emphasis in the deployment and integration phase. The aim is to be able to maintain secure, stable and failsafe operation of the software during deployment and for this reason appropriate metrics are monitored to ensure the “health” of the running, “live” software. Such metrics might include number of bugs reported, number of identified vulnerabilities and weaknesses, number of exploit attempts, etc.

An essential part of deployment involves change management and in the case of IoT SDLC this mainly refers to software updates. The latter mainly support configuration and vulnerability management, but may also be needed for other reasons. All patches should follow a structured change management approach and, in the case of security, should ensure that any updates to the system retain at least the same level of security that was provided by the previous solution. In terms of deployment for IoT systems and services and software updates, an added level of

[34] See https://thenewstack.io/deployment-strategies/

complexity derives from the fact that these mostly take place over the air, via wireless channels and making use of backend servers for the propagation of patches. It is evident that all these elements increase the potential attack surface and therefore should be mitigated by appropriate security controls. Software updates and patch management are also among the tasks of the forthcoming phase, namely maintenance and disposal.

2.6 MAINTENANCE AND DISPOSAL

The last phase of IoT SDLC involves maintenance and disposal. It is important to not disregard activities and tasks that fall under this phase. This is because software deployed in production needs to be constantly maintained to ensure availability and integrity of the provided functionality. The pervasive and adaptive nature of IoT solutions and the fact that devices may be appropriated by users, further strengthens the need for maintenance operations. In addition, given the fact that IoT end devices are many times located in uncontrolled or mission-critical environments and even in outdoor settings, maintenance planning should integrate relevant features (e.g. consistent remote secure (over-the-air) update procedures, updates assuming no physical access/no user-interaction, intuitive update timings with minimum disruption, etc.).

Moreover, in terms of security, incident management is an ongoing activity during maintenance. All elements of the software, as well as all other integral parts of an IoT solution need to be continuously monitored to ensure threat detection and response. It is thus a good practice to perform frequent vulnerability assessments, penetration tests, security maintenance and threat intelligence tasks to prevent attacks and threats to the cloud, the network and IoT end devices, as well as the applications developed for such devices.[35] [36] Constrained and low-powered IoT devices may not able to create or store log files, so preparing for accountability is very important. In addition, service continuity planning by means of automatic backups or redundancy helps to prepare for malfunctioning or service disruptions caused by security incidents.

Other maintenance and disposal tasks include management of software updates, regulatory compliance (by monitoring the corresponding legal and regulatory framework) and secure software and device disposal. Similarly to the previous phase, management of software updates needs to cope with the particularities of the IoT ecosystem. One additional element to consider is the lifetime of IoT devices, which in some cases might be long (e.g. smart car). Combined with vulnerability management, security patches and updates should be issued in a timely and reliable manner. Recent interesting related work includes IETF SUIT (Software Updates for Internet of Things)[37].

Additionally, it is important to ensure that when IoT maintenance functions are delegated to third parties (contractors), this takes place in well-looked-after security conditions, i.e. access control, permission handling, audit and accountability.

Lastly, when IoT software becomes obsolete (for example, when a product series of IoT end devices is decommissioned), it is important to ensure a secure disposal to preserve privacy management, providing data erasure mechanisms. The main risk involved is the abuse of the various types of data that the IoT software makes use of and has probably cached for processing. Such sensitive data types could include patient health records, Wi-Fi credentials to access the organisation/enterprise network, or operational knowledge used to gain a competitive advantage.

[35] See https://www.iotcentral.io/blog/security-first-design-for-iot-devices
[36] See https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf
[37] See https://datatracker.ietf.org/doc/draft-ietf-suit-architecture/

2.7 SECURITY IN SDLC

An important aspect that is commonly overlooked when integrating security in a process (such as in the SDLC) is that of assessment and evaluation. Understanding the current cybersecurity posture is the first step towards establishing a plan to maintain this posture and improve it. In this respect, Security Maturity Models (SMM) are a very useful tool since they guide organisations to define their level of security in accordance with the requirements they wish to fulfil[38]. The maturity of security assesses the understanding of the current level of security, its needs, benefits, and the cost of its support. This assessment takes into account specific threats to the regulatory and compliance requirements of an organisation's industry, the unique risks present in an environment, and the organisation's threat profile. There are numerous standards that serve as tools to evaluate the security of a software project. Some of the most widely recognised industry standards are: Common Criteria (CC)[39], Capability Maturity Model Integration (CMMI)[40], Building Security in Maturity Model (BSIMM)[41], Security for industrial automation and control systems Part 4-1 - Secure product development lifecycle requirements (IEC 62443-4-1), or Open Software Assurance Maturity Model (OpenSAMM)[42], among others.

Throughout the document, security will be addressed from the need to define a procedure to carry out a secure software development process and its management (governance), as well as from the implementation and execution of the necessary measures to that security be embedded during the different phases of the SDLC.

Along these lines, security should be a fundamental principle across all six phases of the IoT SDLC. Relevant and applicable controls need to be carried out in each phase to evaluate the state of security (e.g. establishing security gates and metrics). In order to ensure that software meets all required security conditions before proceeding to following phases, it is necessary to implement security gates. The severity thresholds that indicate the completion of each phase are defined by means of metrics. These metrics are used to analyse, detect and correct vulnerabilities throughout the development process.

Complementary to security, a cross-cutting activity during the SDLC process that is often not considered as essential is the documentation process. This is due to the complexity of the IoT solutions, the number of resources involved in a development process, the IoT interconnectivity, the number of external and internal components, the module integrations, configurations, designs, requirements, etc. It is crucial to have a good documentation and a documentation management system that supports the SDLC process to make it understandable, traceable, and subject to monitoring and auditing.

[38] See https://www.iiconsortium.org/pdf/SMM_Description_and_Intended_Use_FINAL_Updated_V1.1.pdf
[39] See https://www.commoncriteriaportal.org/
[40] See https://cmmiinstitute.com/
[41] See https://www.bsimm.com/
[42] See https://www.opensamm.org/

3. ASSET AND THREAT TAXONOMY

3.1 ASSET TAXONOMY

To focus on the details of IoT security in the SDLC it is essential to start from identification and decomposition of assets of such vast and complex environments focusing on software development. Here we provide an overview of the key asset groups and assets that need to be protected. IoT SDLC assets are classified into key groups depicted in Figure 3 and described in Table 1. It should be noted that the lowest level of the taxonomy is indicative and not exhaustive. For instance, not all types of data are listed, just some representative ones.

Figure 3: Asset taxonomy

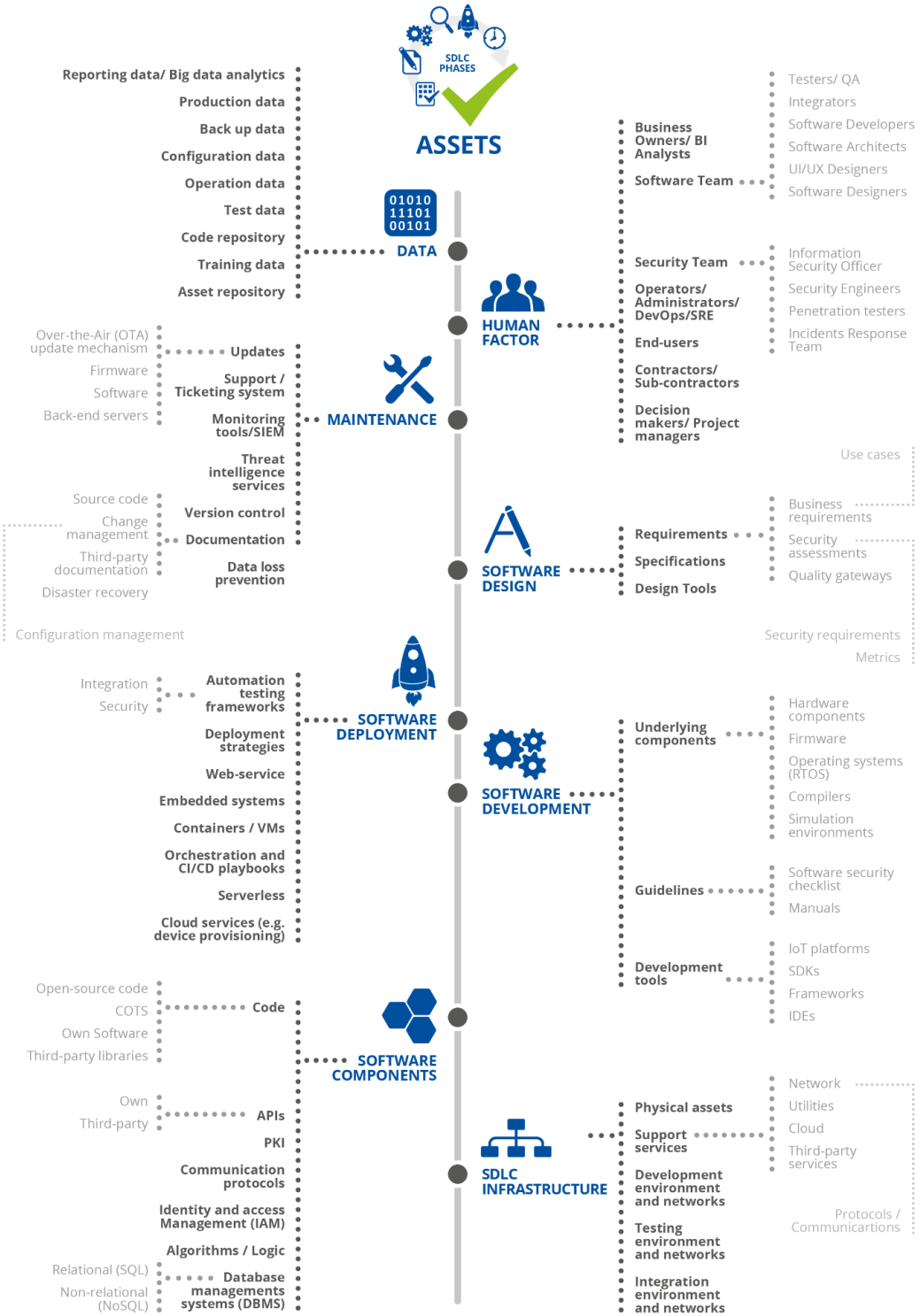


Table 1: **Asset taxonomy**

Asset group	Subgroup	Indicative assets		Description
Human factor	Business owners / BI analysts			Individual or team responsible for analysing data that are used by a business or organisation or a specific business function.
	Software Team	Testers Q/A		People in charge of the quality of the software (QA staff), by means of checking it.
		Integrators		Specialist people in putting different IT components together, working as a whole system
		Software Developers		People that develop software applications
		Software Architects		Expert who makes high-level design choices and dictates technical standards, including software coding standards, tools, and platforms.
		UI/UIX Designers		Designers responsible for the user interface of an IoT application that need to work closely together.
		Software Designers		Software designers that use principles of science and mathematics to develop IoT applications.
	Security Team	(Chief) Information Security Officer		International Standards and Best Practices applicable in the work process management
		Security engineers		Security engineers are responsible for the security aspects in the design of systems that need to be able to deal robustly with possible sources of disruption, ranging from natural disasters to malicious acts.
		Penetration Testers		Professional specialized in security that attempt to crack into a system for the purposes of security testing.
		Incident Response Team		Group of people who prepare for and respond to any emergency incident, such as a natural disaster or an interruption of business operations.
	Operators/Administrators/DevOps/SRE (Operations Team)			People with this role undertake ongoing activities that are required for the provision of IoT software or services.
	End Users			People that use the software applications
	Contractors/Sub-contractors			Entities or companies that provide services or products relevant to the processes of IoT software development.
	Decision makers / Project Managers			Project managers are accountable for the success of a project and their responsibilities include the planning and the execution of a project, building its comprehensive work plan, and managing the budget.
Software design	Requirements	Business requirements		High-level description of what the intended product or services should do based on the business and/or stakeholders needs
		Security assessments	Metrics	Quantifiable measures that are used to track and assess the status of a specific business process.
			Quality gateways	Methodology for the quality assurance of an SDLC process.

			Use cases	Methodology used to identify and analyse the behaviour of a system when responding to an event.
	Specifications			Detailed and technical documents that describe the technical functionalities of the end product or service.
	Design Tools			Tools to aid in the design software or systems, also known as CASE tools: Computer Aided Software Engineering.
Software development	**Underlying components**	**Hardware components**		Components on which the intended software relies or is built on.
		Firmware		
		Operating Systems (ROS)		
		Compilers		
		Simulation environments		
	Guidelines	**Software security checklist**		A set routines or practices that streamline a particular processes.
		Manuals		
	Development tools	**IoT platforms**		A multi-layer technology that enables management tasks and data visualisation.
		SDKs		Software development kits: a set of functionalities and tools to allow developing software in a programming language.
		Frameworks		A set of functionalities and libraries to ease and speed up the software development, being the foundation of software applications.
		IDEs		Integrated development environment: software application that provides a set of tools to aid in software development.
		Algorithm Training tools		Algorithms to perform a task without instructions, resorting to patterns and inference. A subset of artificial intelligence, the algorithms that make a mathematical model from "training data" depend on the kind of problem, the computing resources available, and the nature of the data (supervised, unsupervised, classification, regression, etc.).
Software deployment	**Automation testing frameworks**	**Integration**		A set of guidelines for creating and designing test cases. It is a conceptual part of automated testing that helps testers to use resources more efficiently.
		Security		
	Deployment strategies			Deployment strategies provide a way to change or upgrade an application without downtime in a way that the user barely notices the improvements.
	Web-services			A solution that uses different protocols and standards with the objective of exchanging data between applications.

	Embedded systems		System designed to perform some dedicated functions, typically with low resources, and sometimes located remotely. Embedded Systems with updatable software or firmware include a bootloader which is responsible for verifying the integrity of the software or firmware image on the device before loading it.
	Containers / VMs		Software package that contains everything the software needs to run. This includes the executable program as well as system tools, libraries, and settings.
	Orchestration and CI/CD playbooks		Continuous Integration and Delivery: Continuous Integration is the engineering practice of frequently committing code in a shared repository. Continuous Delivery is the practice to build the software in a way that is always ready to run in their target environment
	Serverless		Applications where the management and allocation of servers and resources are completely managed by the cloud provider
	Cloud services (e.g. device provisioning)		Cloud computing: the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.
	Integrity verification software		Software that protects against unexpected or unauthorised changes in data once it was created by an authorised source.
Data	**Reporting data/ Big data analytics**		These data inform of critical elements concerning an organisation's performance to improve different aspects.
	Production Data		Without these data it would not be possible to complete daily business tasks and processes.
	Backup Data		Security copy of data files and folders to enable recovery in the event of data loss.
	Configuration Data		Data needed to set up the system correctly
	Operation Data		Real data with which the software works
	Code repository		Platform that stores and centralizes all the developed source code. Allows the development team to keep track of versions.
	Test Data		Data used to perform the different tests concerning software, e.g. penetration testing, black box testing, etc.
	Asset repository		This repository provides a single, centralised database to store and track organisational assets.
	Training data		Data used to train Artificial Intelligence/Machine Learning algorithms. Training involves the learning phase where algorithms can make predictions based on the training data that been fed to them.
Maintenance	**Updates**	**Over-the-Air (OTA) update mechanism**	Mechanism to update hardware remotely with new settings, software or firmware.
		Firmware	Software that sets the lowest-level logic to control a device‘s electronic circuits.

		Software	Minor software modifications deployed that provide security or functionality error fix.
		Back-end servers	Software component that provide functionality for other programs such as sharing data or resources
	Support/ Ticketing system		Software designed to organise and distribute incoming customer service requests.
	Monitoring tools / SIEM		Monitoring tools used to continuously keep track of the status of the system in use, in order to ensure the earliest warning of failures, defects or problems, and to improve them. Monitoring tools span from servers, networks, and databases, to security, performance, end-devices and applications.
	Threat Intelligence services		Threat Intelligence Services generate, aggregate and distribute real-time feeds of intelligence data generated and derived from the use of IoT.
	Documentation	**Source Code**	Written text or illustration that accompanies Software and explain how operates or how to use it. Different types of documentation exist such as that for source code, change management, etc.
		Change management	
		Disaster recovery	
		Third-party documentation	
	Data loss prevention		The practice used by organisations to detect and prevent breaches, leakages, or the undesired destruction of sensitive data. Also used for regulatory compliance. An example would a ransomware attack. DLP focuses on preventing illicit transfers of data outside of the organisation.
	Version control		Management of the different changes made to the elements of a product or its configuration.
Software components	**Code**	**Open-source code**	Software readily available for users to build and distribute new solutions.
		COTS	Commercial-off-the-shelf: software and services are built and delivered usually from a third party vendor. COTS can be purchased, leased or even licensed to the general public.
		Own software	Software developed and maintained by the own company.
		Third-party libraries	Software not developed or maintained by the company, but they are part of an application or system of the company
	APIs	**Own**	Application Programing Interface: a set of subroutine definitions, communication protocols, and tools offered for one library to be used by other software
		Third-party	
	PKI		Technology that is used for authenticating users and devices in the IoT ecosystem.
	Communication protocols		Formal descriptions of digital message formats and rules that allow two or more entities of a communications system to transmit information.

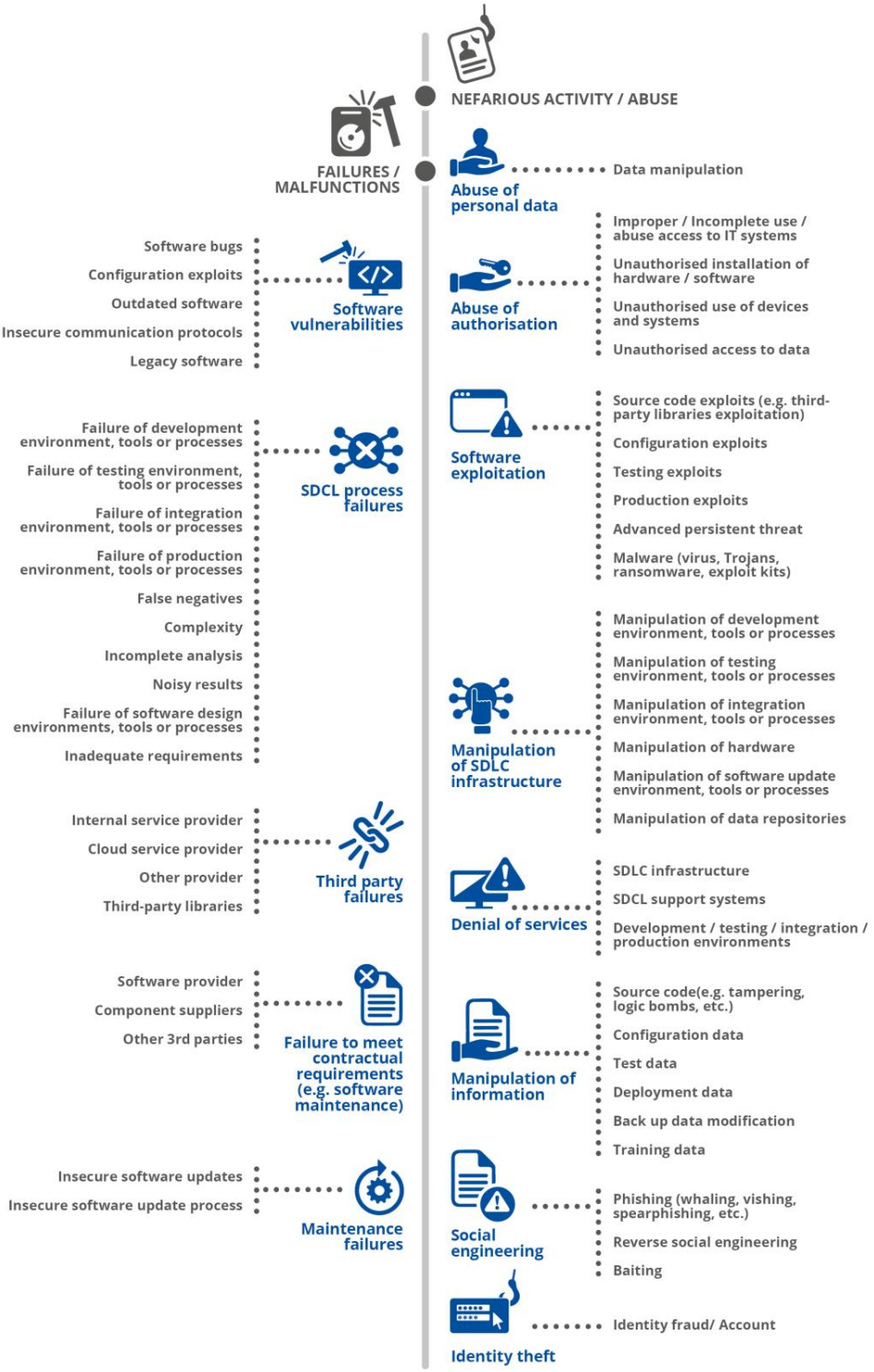
	Identity and Access Management		A framework of business processes, policies and technologies that facilitates management access control.
	Algorithms/Logic		A set of unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks.
	Database management systems	**Relational (SQL)**	Software packages designed to define, manipulate, retrieve and manage data in a database
		Non-relational (NoSQL)	
SDLC infrastructure	**Physical assets**		Any type of tangible asset that is used to support the SDLC process (e.g. computers, wires, etc).
	Support servicers	**Network**	Intangible assets in the form of internal or external services that support the operation of the SDLC infrastructure.
		Utilities	
		Cloud	
		Third-party services	
	Development environment and networks		Environment and networks used for the development of the IoT applications.
	Testing environment and networks		Environment and networks used for testing purposes of the IoT applications.
	Integration environment and networks		Environment and networks used for the integration of the IoT applications.

3.2 THREAT TAXONOMY

The complexity and large scale of the IoT SDLC process combined with the inherent particularities and challenges of IoT systems and services exacerbate relevant cybersecurity challenges. There exist a series of threats that might affect the IoT SDLC, while it should also be noted that these threats come with a varying level of potential impact if they materialize. In accordance with the ENISA Threat Taxonomy[43], Figure 4 depicts the main threats related to the IoT SDLC. Further details are provided in Table 2, where a detailed description of all threats is listed, alongside the list of assets (as per the asset taxonomy) that each threat may affect.

[43] See ENISA (2016) "ENISA Threat Taxonomy A tool for structuring threat information": https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends/enisa-threatlandscape/etl2015/enisa-threat-taxonomy-a-tool-for-structuring-threat-information

Figure 4: Threat taxonomy


NEFARIOUS ACTIVITY / ABUSE
FAILURES / MALFUNCTIONS
Abuse of personal data
Data manipulation
Improper / Incomplete use / abuse access to IT systems
Abuse of authorisation
Unauthorised installation of hardware / software
Unauthorised use of devices and systems
Unauthorised access to data
Software bugs
Configuration exploits
Outdated software
Insecure communication protocols
Legacy software
Software vulnerabilities
Software exploitation
Source code exploits (e.g. third-party libraries exploitation)
Configuration exploits
Testing exploits
Production exploits
Advanced persistent threat
Malware (virus, Trojans, ransomware, exploit kits)
Failure of development environment, tools or processes
Failure of testing environment, tools or processes
Failure of integration environment, tools or processes
Failure of production environment, tools or processes
False negatives
Complexity
Incomplete analysis
Noisy results
Failure of software design environments, tools or processes
Inadequate requirements
SDCL process failures
Manipulation of SDLC infrastructure
Manipulation of development environment, tools or processes
Manipulation of testing environment, tools or processes
Manipulation of integration environment, tools or processes
Manipulation of hardware
Manipulation of software update environment, tools or processes
Manipulation of data repositories
Internal service provider
Cloud service provider
Other provider
Third-party libraries
Third party failures
Denial of services
SDLC infrastructure
SDCL support systems
Development / testing / integration / production environments
Software provider
Component suppliers
Other 3rd parties
Failure to meet contractual requirements (e.g. software maintenance)
Manipulation of information
Source code(e.g. tampering, logic bombs, etc.)
Configuration data
Test data
Deployment data
Back up data modification
Training data
Insecure software updates
Insecure software update process
Maintenance failures
Social engineering
Phishing (whaling, vishing, spearphishing, etc.)
Reverse social engineering
Baiting
Identity theft
Identity fraud/ Account

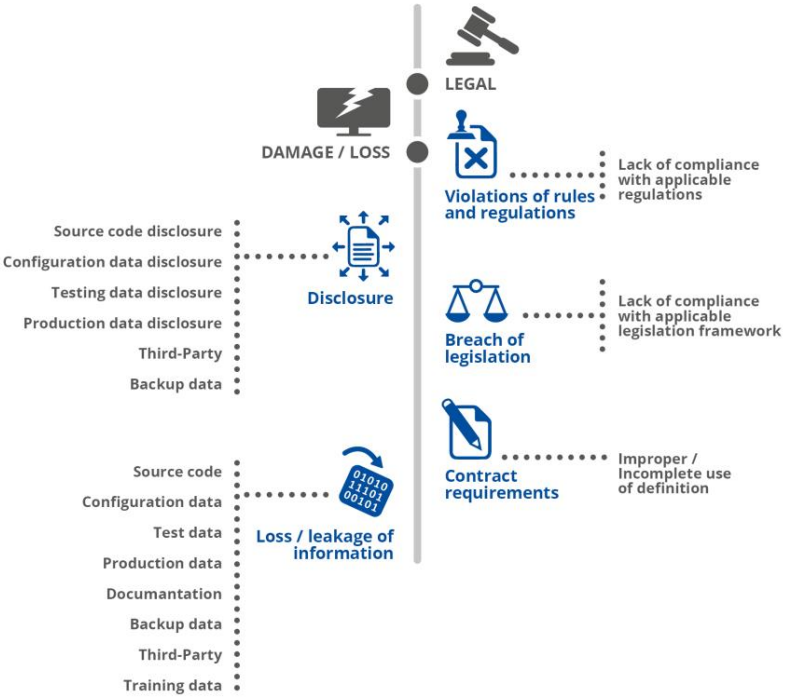

LEGAL
DAMAGE / LOSS
Violations of rules and regulations
Lack of compliance with applicable regulations
Source code disclosure
Configuration data disclosure
Testing data disclosure
Production data disclosure
Third-Party
Backup data
Disclosure
Breach of legislation
Lack of compliance with applicable legislation framework
Contract requirements
Improper / Incomplete use of definition
Source code
Configuration data
Test data
Production data
Documantation
Backup data
Third-Party
Training data
Loss / leakage of information

Table 2: Threat taxonomy and mapping to assets

Category	Sub-Category	Threat	Description	Assets Affected
Personnel	Insider Threat	**Corporate Espionage**	Theft of data to gather critical and valuable information, by an employee or by some other company (competitors), throughout the development lifecycle process, affecting the final product, intellectual property, time to market, etc.	Software Component Data Human Factor Software Development SDLC infrastructure
		Sabotage	Intentional unauthorised actions (non-fulfilment or defective fulfilment of personal duties) aimed at causing a disruption or damage during the software development, to obstruct the process, to affect the integrity of the software or to ultimately compromise the objective of the software.	Data Human factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Fraudulent activities	A team member or an attacker may use confidential information or exploit system vulnerabilities to carry out fraudulent activities (theft of sensitive information, industrial espionage, or extortion) that may affect the integrity of the software or cause damages to third parties.	Data Human factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Blackmailed staff	A member of the team is under duress from a malicious third party to carry out certain actions that could compromise the security of software in exchange of not revealing embarrassing, disgraceful or otherwise damaging information about the employee. It is a form of extortion.	Human Factor
		Disgruntled staff	A disgruntled employee may deliberately use his or her privileges in order to seek revenge by leaking sensitive information to competitors or other companies that offer some kind of incentive to the employee to compensate for this dissatisfaction.	Human Factor
		Corrupted staff	A corrupt employee may deliberately seek to exploit his or her privileges in relation to corporate resources to his or her own benefit, leveraging the said resources for personal gain despite not being dissatisfied with the situation at the organisation.	Human Factor
	Teamwork Issues	**Incompetent / Inexperienced / Demotivated Staff**	An incompetent/inexperience/demotivated may pose a threat to the organisation due to absentmindedness or to a lack of knowledge and awareness of security, resulting in accidental risks.	Human Factor
		Issues in communication/coordination	Lack of a proper communication between project members, either internals or communications with service providers, may lead to errors such as misunderstandings, duplication of tasks,	Human Factor

			undefined scope, lack of systems integration, use of obsolete versions, etc.	
	Internal Limitations	**Absence of personnel / Limited resources**	A lack or unavailability of necessary personnel (strike, unexpected events, disasters, or staff turnover) may lead to an inability to ensure the level of security required due to the excessive workloads burdening other staff members and preventing them from paying the necessary attention to security throughout the process.	Human Factor
	Hacktivism	**The use of illegal logical tools**	A way of activism that uses and/or abuses technology to spread ideas or to punish organisations or people based on their beliefs. This threat can be posed either by isolated individuals or by organised professionals taking advantage of an organisation's security flaws.	Data Human factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
Outages	**Loss of support services**	**Business software**	Unavailability of business software required for development of the software, failure of business software, failure of the support services, or loss of the license.	Data Human factor Software Development Maintenance SDLC infrastructure Software components
		Cloud (online storage)	Unavailability, interruption or failure of online storage on the cloud. Depending on the communication, and on the time required to recover, the importance of this threat ranges from high to critical.	Software Deployment SDLC infrastructure
		Third-party services (e.g. API, Communication Brokers, Cloud)	The failure or malfunction of a service or support that has been assigned to a third party (supplier), thus creating a dependency, can affect the whole product lifecycle, from the development process (e.g. drawing the project out) to the release of the product in the market (e.g. unavailability of the service).	Data Human Factor SDLC infrastructure Software components Software Deployment Maintenance
		Third party documentation	Threat of unavailability documents of private company archives, often a failure of document management control affects the specifications of the software, information leakage / sharing caused by inadequate security measures of the third-party.	Data Human Factor Software Design SDLC infrastructure Maintenance
		Code repositories	Unavailability of the code repositories, due to a lack support of the repository, failure of the third parties, failure of communications, etc.	Data Software Development SDLC infrastructure Software components Maintenance

		Software configuration management	Unavailability of proper software configuration management, unawareness of the correct version of the code or modification of the code.	Data Software Deployment Software Components Maintenance
		Subcontracting of development services	Unavailability of subcontracting of development services required for development process. Unavailability of key personnel and their competences, unavailability of the business development, etc.	Data Human Factor Software Development SDLC infrastructure Software components Software Deployment Maintenance
	Utility outage	**Power (Electricity and Gas)**	Interruption or failure in the supply of power (electricity or gas), either intentional or accidental, and the time required to recover. The importance of this threat ranges from high to critical.	Human Factor Data SDLC infrastructure Maintenance
		UPS (uninterruptible power supply)	Interruption or failure in the UPS, either intentional or accidental, and the time required to recover. The importance of this threat ranges from high to critical.	Human Factor Data SDLC infrastructure Maintenance
		Cooling	An interruption or failure in the cooling services (air-conditioning in server room), either intentional or accidental, may affect hardware support and prevent access to project information (loss of information, file deletion, etc.)	Human Factor Data SDLC infrastructure Maintenance
	Network outage	**Communication issues**	A lack of communication links (wireless, mobile, fixed network, internet) prevents information flows due to problems with networks blocking file updates, repository access, teamwork communications, information exchanges, etc.	Human Factor Data SDLC infrastructure Maintenance
Unintentional Damages (Accidental)	**Unintentional modifications**	**Source code**	A member of the team unconsciously makes a mistake in any of the tasks, causing an unwanted modification of source code (and probably damaging it).	Data Software Components Maintenance
		Configuration data	A member of the team unconsciously makes a mistake in any of the tasks, causing an unwanted modification of configuration files (and probably damaging them).	Data Maintenance
		Test data	A member of the team unconsciously makes a mistake in any of the tasks, causing an unwanted modification of test reports (and probably damaging them).	Data Maintenance
		Deployment data	The information about how to put the software into production, or about how to launch the system (start scripts) is quite sensitive. Errors concerning these data could leave the software in a vulnerable state (security measures not activated, etc.)	Data Software Deployment Software Development Software components

		Documentation	A member of the team unconsciously makes a mistake in any of the tasks, causing an unwanted modification of project documentation (and probably damaging it).	Human Factor Data Maintenance
		Backup Data Modification	An unexpected modification that affects the backups could put at risk the system's operation or even bring about a loss of the application in case of a system failure.	Data Maintenance
		Perturbation of environment	A change of the environmental work conditions can cause the failure of results in the SDLC process (testing results, maintenance and operation environment, etc.)	Data Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
	Erroneous use or administration of devices and systems	**Development environment**	Information leakage / sharing / damage or system management misuse that could affect the programming process and tools during the development phase.	Data Software Development SDLC infrastructure Software Components Maintenance
		Integration environment	Information leakage / sharing / damage or system management misuse that could affect the process and tools when all software components are put together and tested as a whole.	Data Software Design Software Development SDLC infrastructure Software components Software Deployment
		Testing environment	Information leakage / sharing / damage or system management misuse that could affect to the validation process and tools (automated checks or non-automated techniques) causing failed tests, or false test results.	Data Software Development SDLC infrastructure Software components Software Deployment
		Production environment	Information leakage / sharing / damage or system management misuse that could modify the current conditions of the software (such as configuration) during its production phase.	Data Software Development Maintenance SDLC infrastructure Software Components Software Deployment
	Damage caused by a 3rd party	**Discontinued third-party products /services for development**	A failure on the part of a service provider on which the project depends puts at risk the proper operation of the software development process because the corresponding dependency (service or product) will no longer be provided.	Software Components SDLC infrastructure Maintenance
	Information leakage	**Data disclosure**	A sensitive information exposure occurs when, due to an accidental event, an application or program does not adequately protect information such as passwords, payment info, or health data. With this information, cybercriminals can make	Data Software Components

			fraudulent purchases, access a victim's personal accounts, or even blackmail someone.	
Physical Attack	Sabotage	**Internal**	Intentional actions by internal people aimed at causing a disruption or damage of the physical components or facilities	Data Human factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
		External	Intentional actions by external people aimed at causing a disruption or damage of the physical components or facilities	Data Human factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
	Vandalism and theft	**Theft of equipment (hardware)**	Theft of information or IT assets that support the development process	Data SDLC infrastructure Software Components
		Physical damage to equipment	Incidents such device thefts, bomb attacks, vandalism or sabotage could damage the equipment	Software Deployment SDLC infrastructure Maintenance
		Modification of equipment/ devices	Intentional attacks on development process support (servers, laptops, mobile) of development software, dependencies thereof, and on IoT devices that are closer to the physical process.	Software Deployment SDLC infrastructure Software Development Maintenance
		Theft of documents	Theft of documents from private company archives, often for the purpose of re-sale or to obtain personal benefits.	Data Maintenance
		Theft of backups	Stealing media devices on which copies of essential information are kept.	Data Maintenance
	Attacks with physical access	**Side-channel attacks**	Attack based on the collection of information about what the system does when performing cryptographic operations to reverse-engineer it instead of on cryptographic weaknesses.	Data SDLC infrastructure Software components Software Deployment Maintenance
		Radio Frequency attacks	Theft or data tampering by an attacker leveraging the vulnerabilities of Radio Frequency (RF) communications in order to access facilities or physical components.	Data SDLC infrastructure Software components Software Deployment Maintenance

		Communication layer attacks	Attacks that could involve the modification of messages, identity theft, repudiation, data analysis, etc. when a communication among different entities is performed with the aim of accessing facilities or physical components.	Data SDLC infrastructure Software components Software Deployment Maintenance
	Forced Access	**Unauthorised physical access / Unauthorised entry to premises**	Unapproved access to facilities that could be leveraged for malicious actions.	Data SDLC infrastructure Software components Software Deployment Maintenance
Legal	**Violation of rules and regulations**	**Lack of compliance with applicable regulations**	Threat of financial or legal penalties or loss of trust of customers and collaborators due to a violation of applicable law or regulations.	Human Factor Data
	Breach of legislation	**Lack of compliance with applicable legislative framework**	Lack of compliance with international standards and Best Practices (e.g. BSA, ISO, CSA, NIST, etc.) applied to the Software Development Process introducing known failures in the system.	Human Factor Data
	Contract Requirements	**Improper / Incomplete use of definition**	The lack of specific security clauses in provider contracts means that there may not be contractual obligations at service or product level.	Human Factor Data Software Components Maintenance
Failures/ Malfunctions	**Software vulnerabilities**	**Software bugs**	Flaws or errors in the software programming or system that produce an incorrect or unexpected operation or result.	Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Configuration exploits	Due to a failure in the configuration system, an attacker can leverage the vulnerability to launch an attack on the system.	Software Development SDLC infrastructure Software Components Software Deployment Maintenance Data
		Outdated software	Software that is not up to date may trigger severe risks for a software solution. Potential issues may arise from vulnerabilities that are present in the software dependencies, or legacy systems.	SDLC infrastructure Software Components Software Deployment Maintenance Software Development
		Insecure communication protocols	Use of insecure communication protocols that an attacker could leverage in order to cause a malfunction of the system or capture sensitive information.	Data SDLC infrastructure Software components Software Deployment Maintenance

		Legacy software	Software that is obsolete and presents a vulnerability due to a lack of support, updates or patches.	Software Components Software Deployment Maintenance Software Development SDLC infrastructure
	SDLC process failures	**Failure of development environment, tools or processes**	These kind of failures can stop the development process, or delay it, or bring about a loss of control over it.	Human Factor Software Design Software Development Software Deployment Maintenance
		Failure of testing environment, tools or processes	Issues in the testing environment could affect the veracity of testing results (not tested correctly, not tested uniformly, etc.), or may make it impossible to carry out some tests, stopping or delaying the development process.	Data Software Development SDLC infrastructure Software components Software Deployment
		Failure of integration environment, tools or processes	The integration phase is critical, since all software pieces are put together to ensure they work as expected. If potential issues arise, they may result in software integration issues or bad results in integration testing.	Software Development Data Software Components SDLC infrastructure Software Deployment
		Failure of production environment, tools or processes	The production environment is critical because it is the real scenario to work with. Failures can affect the availability of the whole solution, as well as the way to measure or control how software behaves. It also may result in a leakage of sensitive information due to errors.	Software Development Data Software Components Software Deployment SDLC infrastructure
		False Negatives	The ratio of false negatives in the security tools is too high to rely on the results.	Data SDLC infrastructure Software Development Software Deployment
		Complexity	The security tools are too complex, leading to their incorrect use and results that are difficult to interpret.	Software Development Software Design SDLC infrastructure Software Deployment
		Incomplete Analysis	The tool does not analyse the full project or the tool is used when the software is not finished, leaving parts of the software unanalysed from the point of view of security.	Data SDLC infrastructure Software Development Software Deployment
		Noisy Results	Either the means by which the results are presented or the high volume of false positives make the results hard to process, causing vulnerabilities that may go unnoticed.	Data SDLC infrastructure Software Development Software Deployment

		Failure of software design environments, tools or processes	In the design phase, security requirements are included in the solution as design features. Potential failures at this point can lead many vulnerabilities to go unnoticed during following stages of development.	Data SDLC infrastructure Maintenance Software Design Software Development Software components Software Deployment
		Inadequate requirements	Establishing security requirements that are not appropriate for the solution or for the development process can lead to the emergence of vulnerabilities.	Software Development Data Software Design Software Components Software Deployment
	Third party failures	**Internal service provider**	A failure of a service such as the programming of a code part or component design that has been developed by IT departments/service providers within an organisation	Software Design SDLC infrastructure Software components Software Deployment Maintenance
		Cloud service provider	A failure of a service that is supported by a cloud provider, such as an application through the Internet (SaaS).	SDLC infrastructure Software Components Software Deployment Maintenance
		Other provider	A failure or unexpected result of any part that has been outsourced and whose operation has an impact on software development.	SDLC infrastructure Software Components Software Deployment Maintenance
		Third-party libraries	It is necessary to adopt risk management for these assets. These risks must be mitigated to prevent various types of threats from being executed.	SDLC infrastructure Software Components Software Deployment Maintenance
	Failure to meet contractual requirements (e.g. software maintenance)	**Software providers**	Contractual requirements for software providers can manage many different aspects, such as how software is developed, when it has to be delivered, security in workstations of developers, how to deliver the software, security maturity of the software, maintenance of the software, etc. In case of failure, it may have severe consequences, such as intellectual property loss, inability to provide software when needed, immature security for the software delivered, etc.	Human Factor Software Design SDLC infrastructure Software components Maintenance
		Component suppliers	Many different aspects can be included in the contract, and they depend on the component that they provide. SDLC may be impacted if components are not needed, stopping or delaying the process, or not providing the functionalities that they need, or lacking maintenance when it is required (SLAs)	Human Factor Software components SDLC infrastructure Software Deployment Maintenance

		Other 3rd parties	Security is an aspect to consider globally in an organisation, and any organisation provider may result in security issues such as an information leakage or damage to information integrity, if security clauses of the contract are not correctly followed (for instance, cleaning service staff may expose sensitive information when they manage an organisation's residues)	Human Factor Software Deployment Software Components Maintenance SDLC infrastructure
	Maintenance failures	**Insecure software updates**	New updates (new software versions) need to be tested thoroughly to ensure that they not impact the properties of the software. Insecure updates can make a software that was safe in the previous version vulnerable.	Software Deployment Maintenance Software Components
		Insecure software update process	The process to update software is not secure enough (hardcoded credentials for maintenance, backdoors, integrity is not checked), allowing potential attackers to compromise the software by abusing the updating process.	Software Deployment Maintenance
Nefarious activity/ Abuse	**Abuse of personal data**	**Data Manipulation**	In this case, the objective is to manipulate the data in order to modify data, cause the failure of the software, or acquire monetary gains. By accessing the operation data of the system, an attacker may modify them to alter the operation of the application for malicious purposes	Data
	Abuse of authorisation	**Improper / Incomplete use /abuse access to IT systems**	Abuse of authorised access systems that support the infrastructure, making it possible to modify the version of the software and the tools during the process of software,	Software Deployment SDLC infrastructure Software Development Data Maintenance
		Unauthorised installation of software/ hardware	Threat of unauthorised manipulation of hardware and software that can be used to modify source code for malicious purposes, posing threats such as bomb injections, backdoor generation, or the destruction of source code.	Software Deployment Software Development SDLC infrastructure Software components Maintenance
		Unauthorised use of devices and systems	An unauthorised modification of configuration data could cause the system to work incorrectly or the security measures implemented may not act correctly, allowing attacks against the system.	Data Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Unauthorised access to data	Unauthorised modification of code or data, attacking its integrity. In this case, it can result in the manipulation of information, unauthorised access to confidential information, and access to source code.	Data Software Development Software components
	Software exploitation	**Source code exploits (e.g. third-party libraries exploitation)**	Unauthorised modification of source code for malicious purposes such as bomb injections, backdoor generation, or the destruction of source code.	Software Components Software Development Data Maintenance

		Configuration exploits	The default configuration is vulnerable, containing weak/default passwords, software bugs, and configuration errors. This threat is usually connected to others, like exploit kits	Software Components Software Deployment Maintenance Data
		Testing exploits	Threat leveraging the use of default configuration of the testing environment, with default passwords, software bugs, and configuration errors.	Software Deployment SDLC infrastructure Software Components
		Production exploits	Threats leveraging the use of outdated software versions, bugs, improper configurations, zero-day vulnerabilities or specific software components, such as weak cryptographic algorithms or vulnerable open source libraries	Software Deployment SDLC infrastructure Maintenance Software Components
		Advanced Persistent Threat	In Advanced Persistent Threat (APT) attacks, eavesdropping and information gathering comprise one of the first stages carried out in order to identify weak spots and potential entry/attack points	Data Human Factor Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Malware (virus, Trojans, ransomware, exploit kits)	Exploit Kit Code designed to take advantage of a vulnerability in order to gain access to a system. This threat is difficult to detect and during the software development process its impact ranges from high to crucial, depending on the assets affected	Software Deployment SDLC infrastructure Software components Maintenance
	Manipulation of SDLC infrastructure	**Manipulation of development environment, tools or processes**	Threat of unauthorised manipulation of development environment, tools or processes to intentionally manipulate the information systems or review process of the development to cover other nefarious activities (false results, modification of the information, information integrity loss, no testing updates), or to obtain information about the software under development (intellectual property, etc.)	Software Development Maintenance SDLC infrastructure
		Manipulation of testing environment, tools or processes	Unauthorised modification of testing elements (environment, processes, tools) with malicious intentions (modifying test results, obtaining intellectual property or other sensitive information, etc.). For instance, an attacker could modify the test data in order to allow a system that has not passed the security tests to be accepted and continue to the production phase with security flaws	Software Development Maintenance Software Deployment SDLC infrastructure
		Manipulation of integration environment, tools or processes	Threats that aim to modify the integration environment to obtain intellectual property (the whole solution), or modify the results of the tests when all software components are put together.	Software Development Maintenance Software Deployment SDLC infrastructure
		Manipulation of production environment, tools or processes	Production environment is critical because it is the real scenario to work with. Malicious modifications can affect the availability of the IoT solution, as well as the way to measure or control how software behaves (try to cover other malicious activities). It may also have severe effects, for	Software Development Maintenance Software Deployment

			instance providing access to sensitive information (personal data, code, configuration data, operation data, etc.), or modifying it.	SDLC infrastructure
		Manipulation of hardware	Unauthorised manipulation of hardware elements of the solution, affecting the integrity of hardware elements (which are the basis of the rest of technologies: infrastructure technologies, support systems, etc.)	Maintenance Software Deployment SDLC infrastructure
		Manipulation of software update environment, tools or processes	Threat of unauthorised manipulation of software update environment Patched the lack of a formal update management procedure entails that the urgency of fixing an application or system may bring about errors that cause vulnerabilities in the system.	Maintenance Software Deployment SDLC infrastructure
		Manipulation of data repositories	Threat of manipulation of data repositories with the objective of manipulating source code for malicious purposes such as bomb injections, backdoor generation, or the destruction of source code.	Data Maintenance Software components
	Denial of Service	**SDLC infrastructure**	Understanding IT infrastructure as the set of technologies that provide the needed environment (networks, operating systems, etc.) for the systems and applications, these threats aim to make them unavailable, affecting all technologies that need them. It can have severe consequences.	Maintenance Software Development SDLC infrastructure
		SDLC support systems	Threats that aim to compromise the availability of all type of systems and middleware that sustain the software development process, stopping or delaying the development process.	Maintenance SDLC infrastructure Software components Software Deployment
		Development/ testing/ integration/ production environments	When an environment is not available due to malicious activities, the development process may be stopped or delayed (tests cannot be performed, etc.). In the case of the production environment, the availability of the IoT solution may be partially or completely impacted.	Maintenance Software Deployment Software Development SDLC infrastructure
	Manipulation of information	**Source code (e.g. tampering, logic bombs, etc.)**	Unauthorised modification of source code for malicious purposes such as bomb injections, backdoor generation, or the destruction of source code.	Software Components Maintenance Data
		Configuration data	The unauthorised modification of this type of data may result in an alteration of software parameters, which can affect the security of the solution (disabling security measures, etc.).	Data Software Deployment Maintenance
		Test data	Threat of intentional manipulation of test data with the objective to modify the test data in order to allow a system that has not passed the security tests to be accepted and continue to the production phase with security flaws.	Data Maintenance
		Deployment data	Threat of intentional manipulation of deployment data. A lack of an adequate testing environment affects the validity of the security tests, since the environment should be as similar to production as possible.	Data Software Deployment

		Backup Data Modification	Not adequately protecting backups could allow an attacker to access and modify or destroy data, compromising the system's operation in the event of a failure if access to the backups is required.	Data Maintenance Software Deployment
		Poisoning training / testing data	Training data tampering could cause a diversion from expected data, highly impacting the final results of the SDLC process.	Data Software Development Software Deployment Maintenance
	Social engineering	**Phishing (whaling, vishing, spearphishing, etc.)**	Threat of an e-mail fraud method in which the perpetrator sends out legitimate-looking email in an attempt to gather personal and financial information from recipients. Typically, the messages appear to come from well-known and trustworthy Web sites. The main object in this case it is obtain information of the member of the team development and get identify, passwords and could be modification of the source code	Human Factor
		Reverse social engineering	A reverse social engineering attack is a person-to-person attack in which an attacker convinces the victim that he/she has or will have a problem, and the attacker is the key to solve it.	Human Factor
		Baiting	It is a technique to drive the victim into a trap by resorting to his/her curiosity and interest (like putting rouge USBs on the floor of a parking area).	Human Factor
	Identity theft	**Identity Fraud/ Account**	This threat aims to steal the identity of a legitimate user of the system to perform actions on behalf of the original user, or to access information that the user can access.	Human Factor
Damage/ Loss	**Disclosure**	**Source code disclosure**	Source code is one of the most important assets for a software development project. It is an investment of the organisation in creating technology. A loss of confidentiality regarding this asset may have severe consequences for the organisation due to an intellectual property loss, and potential attackers can discover security holes while reviewing code.	Software Components Maintenance Data
		Configuration data disclosure	Configuration data is an important type of data where many parameters of the software are defined, as well as data about connections to other systems.	Data Maintenance Software Deployment
		Testing data disclosure	Test data contain sensitive information about the system and its status (current and past), since these data are composed of results of the different tests done, including security tests. This information is confidential, and an unauthorised access may have severe consequences for the software, as well as for the organisation.	Data SDLC infrastructure
		Production data disclosure	Production data is an important asset that is quite valuable for the organisation. If these data are exposed in any way, sensitive information can be compromised (personal information, intellectual property).	Data Software Deployment Maintenance
		Third-Party	Third-party components that are part of the solution are as important as your own. The information they hold may be accessed by unauthorised individuals if the third party does not observe due diligence.	Data SDLC infrastructure Software Components

				Software Deployment Maintenance
		Backup Data	An unauthorised access to backup data may expose sensitive information, intellectual property, etc.	Data Software Deployment Maintenance
	Loss/leakage of information	**Source code**	Source code is one of the most important assets for a software development project, it is an investment of the organisation for the creation of technology. For proprietary software, a loss of confidentiality regarding this asset may have severe consequences for the organisation due to an intellectual property loss. Moreover, potential attackers may discover security holes by reviewing code. In the case of a loss of source code, the company will lose efforts and resources. It would need new resources (equal or more than in the previous situation) to be at the same point as it was.	Data Maintenance Software Components
		Configuration data	Configuration data is an important type of data where many parameters of the software are defined, as well as data about connections to other systems. A loss of these data may affect internal system connections or may cause any other issue affecting the availability of the solution.	Data Maintenance Software Deployment
		Test data	Test data contain sensitive information about the system and its status (current and pasts), as these data are composed of results of the different tests done, including security tests. This information is confidential and an unauthorised access may have severe consequences for the software, as well as for the organisation. Its loss represents a loss of the technological memory of the organisation.	Data SDLC infrastructure
		Production data	Production data is an important asset that is quite valuable for the organisation. If these data are exposed in any way, sensitive information may be compromised (personal information, intellectual property). A loss of this kind of data may have a severe impact on the daily operations of the organisation, and possibly make it difficult to continue with regular operations.	Data Software Deployment Maintenance
		Documentation	Documentation is a valuable asset that contains information about processes, software, designs, etc. In general, private information can have negative consequences for the organisation if these data are accessed by unauthorised parties. A loss of documentation or a failure to document changes to it represents a loss of knowledge for the organisation.	Software Design Software Development SDLC infrastructure Software components Software Deployment Maintenance
		Backup Data	Backup data are as critical as operation data, since it is the main means to restore operations in the event of an issue. If these data were leaked, the impact might be even greater than that of a leakage of operation data, since it may also entail the leakage of all history data.	Data Software Deployment Maintenance
		Third-Party	A security breach in a service provider on which the project depends exposes sensitive information about its own system. This endangers sensitive data, such as data about its operation, personal	Data SDLC infrastructure

			data of users, test reports, intellectual property, etc.	Software Components Software Deployment Maintenance
		Training data	Training all personnel is essential to ensure that the development process is addressed in a correct manner. A data modification or leakage could be leveraged by an attacker to cause an interruption of the SDLC process or to obtain sensitive information about security to perform reverse engineering.	Data Software Design Software Development Software components Software Deployment Maintenance

3.3 EXAMPLES OF ATTACK SCENARIOS

The different cases presented here are examples of potential scenarios that can affect IoT software development. Taking into account the overall picture of IoT applications previously discussed (embedded devices, communications and cloud services), each scenario focuses on one of these three elements. These use cases show a high-level overview of different attacks, whereby it should be clarified that they serve only as indicative illustrative cases.

3.3.1 Insecure Credentials in Embedded Devices

One of the most critical issues in relation to IoT solutions is the use of default or insecure credentials. Many such solutions impose hardcoded passwords that cannot or do not need to be changed, or do not include mechanisms to ensure the use of secure authentication.

This means that oftentimes a user lacking sufficient awareness of security may choose to use default credentials or create weak ones on their IoT solutions simply because they are easier to remember. Even when there is a certain level of awareness, traditional restrictions (such as password length restrictions or character use impositions) often lead to exasperation or annoyance on the part of users, who opt to use insecure passwords to overcome these abrasive burdens.

In turn, an attacker might be able to scan the exposed devices using resources like Shodan and other tools (e.g. insecam.org or online databases containing exposed or default credentials to different solutions). The attacker would eventually be able to determine, by means of testing, if the solution lacks a password or uses a default or weak one, thus taking control of the device to gain further privileges and, ultimately, use it for malicious purposes, such as creating botnets (one famous example would be Mirai). These botnets are often used to launch Distributed Denial of Service (DDoS) and cryptojacking attacks on other networks, among others.

Such attacks can be prevented by means of different measures throughout the SDLC process, and essentially entails ensuring that such weak of guessable credentials cannot be used in the solution. For instance, it should be impossible to preserve default credentials after initialization of the IoT product, and users should be provided with guidelines on how to create secure ones.

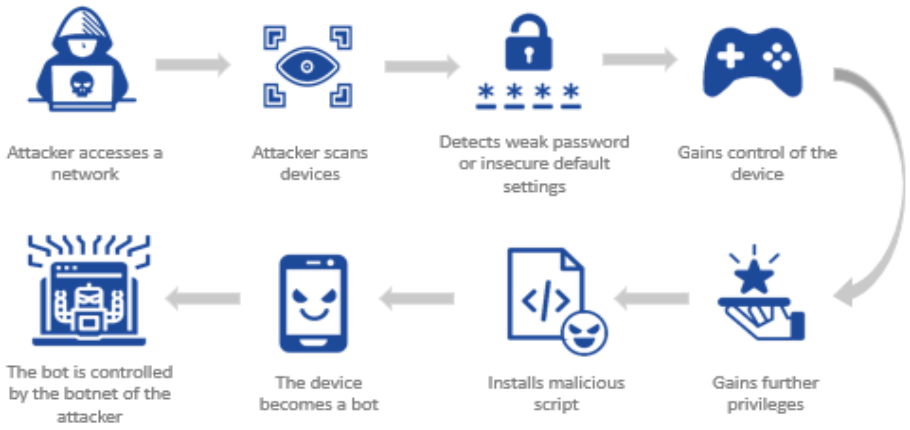
When it comes to creating safe passwords, first of all, it is necessary to use strong authentication mechanisms (for example, based on challenge-response authentication, with an SSH signature) whenever possible. Additionally, it is also important to ensure that the authentication mechanism prevents users from creating weak credentials (such as keywalk passwords, that is, passwords based on adjacent keyboard keys, e.g. ‘qwertyuiop’, or obvious

ones, like 'Aa12345!'). There are plenty of international authorities[44] that have created guidelines for this purpose, but essentially, users should be given freedom to be creative and create custom passwords that are both secure and user-friendly, without too many format restrictions.

Online databases of exposed or default credentials[45] are a good resource to avoid the use of weak authentication mechanisms. This would hinder the malicious actions of the attacker, since the tools available would be less effective to guess weak or default passwords. Another recommendation is to implement security mechanisms like multiple-factor authentication during the SDLC or mechanisms forcing to change or set up a new password before using the device for the first time.

On a similar note, in order to address publicly exposed services and harden the available services, it would be necessary to reduce service to the minimum and remove all unnecessary functions/services/libraries.

Figure 5: Attack 1 – Insecure Credentials in Embedded Devices



[44] See https://pages.nist.gov/800-63-3/sp800-63b.html
[45] For instance, see haveibeenpwned.com. This database includes API to check if a password has been compromised during a leak, so that during the sign-up or password change processes, it is possible to check, securely, if the password created by the user is publicly available or too common/weak.

Table 3: Insecure Credentials in Embedded Devices

Threats	Assets Affected
Personnel: Insider threat; Incompetent / Inexperienced / Demotivated Staff; Absence of personnel / Limited resources.	• Software Component • Data • Software Development Tools • Software Deployment • Maintenance • Human Factor
Unintentional damages: Unintentional modifications; Erroneous use or administration of devices and systems.	• Software Development Tools • Data • Software Components • Software Deployment • Human Factor • Maintenance
Failure / Malfunctions: SDLC infrastructure failures; Maintenance failures.	• Software Development Tools • Data • Software Components • Software Deployment • Human Factor
Nefarious activity / abuse: Manipulation of SDLC infrastructure; Manipulation of information.	• Software Development Tools • Maintenance • Data • Software Deployment • Software Components
Prevention actions during SDLC phases	
• **Requirements: Make secure device and user authentication a formal security requirement.** • **Software Design: Include a mature authentication mechanism, and perform threat modelling as a security exercise to detect potential issues.** • **Development/Implementations: Build/implement the secure authentication design appropriately.** • **Testing & Acceptance: Test the software to detect potential issues and assess the results from the point of view of security.** • **Deployment & Integration: Ensure configuration errors or mistakes (software, infrastructure, and third-party services) have been fixed to prevent deployment failures.**	

3.3.2 Lack of Flexibility to Secure Communications

In this scenario, the security issue arises in relation to the use of rigid communication protocols and the software components that provide this functionality. This scenario especially affects IoT interfaces, where the software uses the communication functionalities to exchange information with other elements. These interfaces are mostly software-based, where the protocol and its corresponding options are used by the software.

This issue arises in relation to the software development process, since flexibility is designed, implemented and configured during the creation of technology. Depending on this, the operation mode of the software may vary in the maintenance phase. In this regard, flexibility is a software decision that is present in the whole lifecycle of the software development.

When it comes to real life, the problem of this scenario would materialise if an insufficiently flexible communication protocol prevents the user from applying additional security measures to communications or from changing to another new protocol that increases the security. Such an incompatibility could cause a security gap in the solution, rendering it vulnerable.

An attacker may detect this lack of security and try to compromise the communications, because if the type of communication used is obsolete, it will no longer be appropriate for the functionality of the system, as it uses insecure ecosystem interfaces or insecure data transfers and storage. The attacker might then use the insecure communication protocol in combination with different types of attacks (man-in-the-middle attacks, leakage of sensitive information, identity theft, etc.).

Ultimately, the objective of the malicious user is to gain further privileges and to obtain sensitive information such as personal data, credit card information, authentication credentials, etc. This issue can be easily prevented by assessing and observing the necessary level of flexibility throughout the SDLC, making it a requirement for the solution.

Figure 6: Attack 2 – Lack of Flexibility to Secure Communications

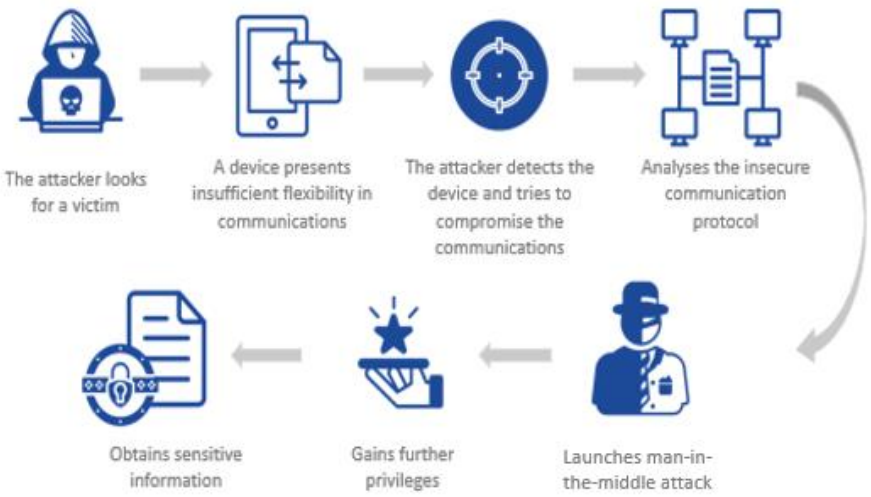


Table 4: Lack of Flexibility to Secure Communications

Threats	Assets Affected
Outage: Loss of support services; Communication issues	• Software Deployment • Data • Software Components • Maintenance • Human Factor
Unintentional damages: Unintentional modifications; Erroneous use or administration of devices and systems; Damage caused by a third party	• Software Development Tools • Data • Software Components • Software Deployment • Human Factor • Maintenance
Legal: Contract requirements	• Human Factor • Data • Software Components • Maintenance
Failure / Malfunctions: Third party failures; Failure to meet contractual requirements; Maintenance failures	• Software Components • Software Deployment • Maintenance • Human Factor

Prevention actions during SDLC phases
• **Requirements:** Establish the flexibility of software communications as a requirement. • **Software Design:** Perform a good design of the solution, and perform threat modelling as a security review to detect potential issues. • **Development/Implementations:** Choose a secure library, perform a secure implementation of the functionality, and use peer code reviews to identify potential problems. • **Testing & Acceptance:** Use Dynamic Application Security Testing - DAST (see Annex D) to detect insecure communications. • **Deployment & Integration:** Perform hardening activities. • **Maintenance & Disposal:** If a communication mechanism is secured during deployment but, later on, it proves to be vulnerable, the lack of flexibility makes it impossible to solve this issue easily. This issue can be addressed with flexibility to secure communication by patches / software updates.

3.3.3 Insecure Software Dependencies in Cloud Services

As previously studied by ENISA[46] cloud services are developed using software components (dependencies) already available, to reduce development time. These components (usually from third parties) are as important as the code developed by the project team (in terms of security). They are usually software frameworks or libraries that provide common functionalities to the software, of the same nature regardless of the business logic. If a dependency is vulnerable and exposed on the Internet, there may be severe consequences. Successful attacks[47] based on outdated components are clear indicators that software dependencies should be taken seriously.

Cloud services could be affected in a similar way. For example, a developer or organisation may not ensure that the dependencies used for a solution are not vulnerable. Even if they do verify this, they may not keep track of new vulnerabilities detected over time in order to ensure prompt patching. If an attacker is aware of this negligence, he or she may study the endpoints exposed for such IoT solution. Analysing the underlying technologies of this endpoint (producing errors, responses, etc.), the attacker may be able to detect dependencies and versions.

This means that, ultimately, simply by checking public vulnerabilities for the dependencies exposed, the attacker may find one to exploit. If such a vulnerability is detected, the malicious agent could then take advantage of it to launch, for example, an unrestricted file upload. The impact of such attack is high, supposed code can be executed in the server context or on the client side remotely, gaining access to the solution, and potentially obtaining sensitive or confidential data.

This type of issue might have been easily prevented, had the organisation behind the solution ensured the use of vulnerability-free components for development, and periodically reviewed their security and applied patches and updates accordingly during the SDLC. The integrity of the cloud service reflects the security of the components it comprises; one vulnerable dependency may compromise the entire solution. In this context, it is important to implementing, for important/critical business functionalities, end-to-end security mechanisms to mitigate this risk.

[46] See https://www.enisa.europa.eu/news/enisa-news/towards-secure-convergence-of-cloud-and-iot
[47] For instance, see the Equifax attack: https://www.netsparker.com/blog/web-security/how-equifax-data-breach-hack-happened/

Figure 7: Attack 3 – Insecure Software Dependencies in Cloud Services

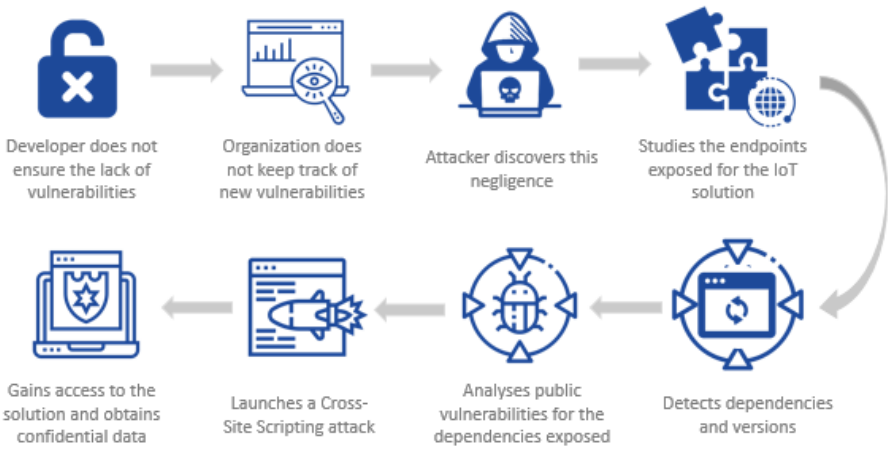


Table 5: Insecure Software Dependencies in Cloud Services

Threats	Assets Affected
Personnel: Incompetent / Inexperienced / Demotivated Staff	• Human factor
Failure / Malfunctions: Outdated software; Third party failures	• Software Components • Software Deployment • Maintenance
Prevention actions during SDLC phases	
• **Requirements: Establish software component dependency analysis against CVE feeds as a security requirement.** • **Development/Implementations: When selecting a FOSS dependency, assess if the community behind it takes security seriously, and if it is big enough to ensure that the software is not discontinued.** • **Testing & Acceptance: Perform software composition analyses to detect potential problems with software components using SAST and DAST.** • **Deployment & Integration: Test the different software artefacts of the cloud service, as well as their corresponding dependencies. Do not put risky dependencies in the production environment when software is going to be deployed, especially for software that is exposed to the Internet.** • **Maintenance & Disposal: Monitor software dependencies periodically to detect new vulnerabilities prior to the cloud services being deployed into the production environment.**	

4. GOOD PRACTICES FOR SECURE IOT SDLC

4.1 SECURITY CONSIDERATIONS

One of the most significant objectives of this document is to address the main security considerations to take into account during the SDLC. During the execution of this study, a group of experts was asked about the main challenges that they have to overcome in order to provide greater security during IoT SDLC.

As a result of this consultation, a non-exhaustive list of security considerations is shown in Table 6.

Table 6: Security considerations

Phases	Security considerations	Description
Analysis and requirements	Security Requirements	Identification of security requirements according to data classification, business requirements and legislative or standardisation objectives.
	Hardware limitations	Alignment of security requirements with hardware limitations taking into consideration additional aspects resulting from software requirements.
	Protocols	Identification of the appropriate protocols for the solution, taking into account its security features and the IoT solution's security requirements.
	Threat modelling	Application of threat modelling methodologies to identify the software threats and the associated countermeasures to mitigate them.
Software design	Attack surface analysis	Identification of the IoT solution's attack surface by taking into consideration architecture aspects and utilising security user stories.
	Secure design	Use / application of secure design patterns and principles. Security architectures determine when and where to apply them.
Development / Implementation	Frameworks	Use of known security guidelines to ease the implementation of security controls during the software development process in order to enhance security throughout the software lifecycle.
	Libraries	Use of trusted security libraries when third-party resources are used, ensuring that they are widely tested based on certain security criteria so as to not compromise the software.
	Built-in Security	OS as well as communication protocols come with built-in security functions which can be leveraged to implement security features in applications.
	Guidelines	Use of the Secure Code guidelines and standards to alleviate from most common application layer vulnerabilities.
	External checks	Use of mechanisms to ensure that external libraries, tools or APIs used during the SDLC phases such as development, deployment and maintenance are proven, secure and updated.

Testing / Acceptance	Design review	Activities aimed at verifying that the design used follows the specifications defined during the design stage so that all security requirements are met
	Code verification	Review of the code/quality of the code, preferably using automated tools in order to look for errors introduced in the implementation phase.
	Security requirement tests	Performance of security tests to ensure that software is free of known vulnerabilities and to detect risks related to security requirements.
	Penetration tests	Testing to identify potential vulnerabilities that could exist in IoT solutions and could be exploited by an attacker.
Deployment / Integration	Hardening environment	Secure the environment adding protection layers as a part of the in-depth defence strategy in order to reduce the system's attack surface.
	Configuration and Vulnerability management	Use of the control activities to guarantee an artefact's quality, monitoring and controlling changes made during the development lifecycle, and identifying and repairing potential flaws affecting the software.
	Change management	Definition of the procedure to document, monitor and track all changes that may be made in the software development process
Maintenance / Disposal	Incident management	Procedure to address the steps to be taken in order to ensure a normal operation when a security issue takes place in the SDLC process.
	Management of the end of life-disposal	Secure management process of software components, artefacts and data once the IoT solution is going to be retired from production.
	Remote SW updates	Delivery management to push new versions of software in a remote environment securely when it is necessary to apply an update, either to add new functionalities or to mitigate vulnerabilities.

4.2 GOOD PRACTICES

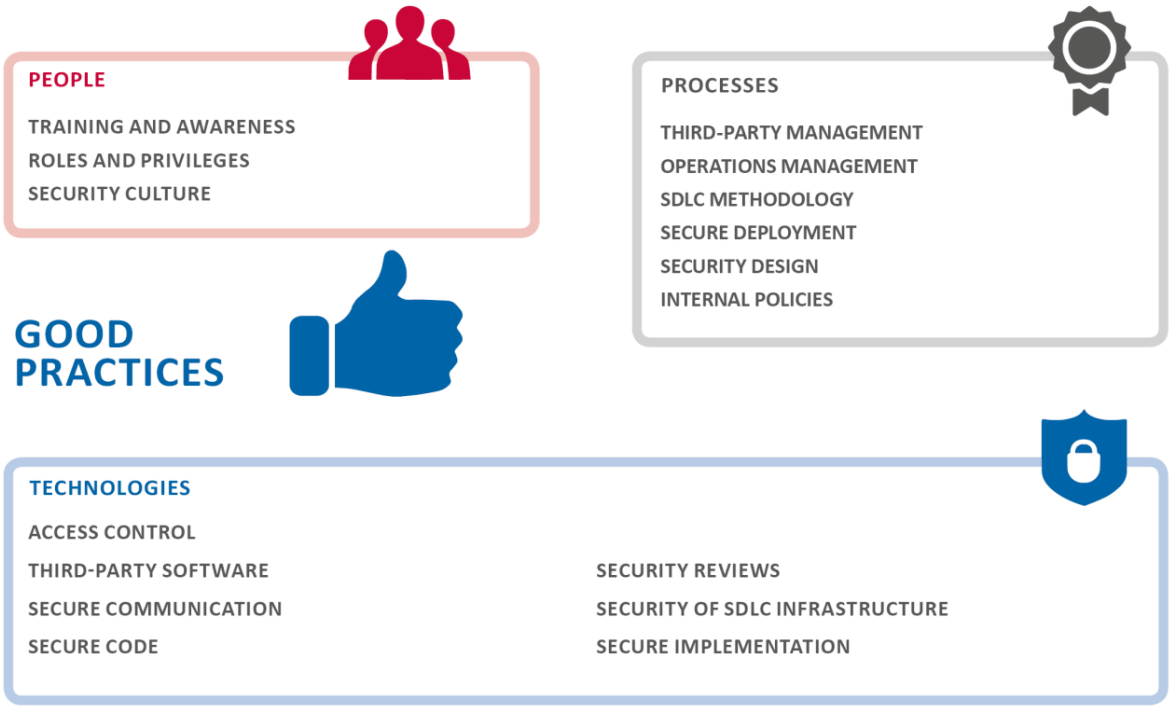
Development of security measures for the IoT SDLC is one of the key objectives of this report. The aim is to provide guidelines and recommendations for the target audience to assist in countering and mitigating the threats that might impact IoT SDLC.

Firstly, extensive desktop research was conducted. Thorough analysis of relevant sources (listed in Annex C) allowed distinguishing frequently mentioned topics in IoT SDLC security. These topics were then aggregated to create an initial list of security domains. Final set of domains was clarified and adapted based on the interviews conducted with the stakeholders resulting in a list of 16 domains that provide a comprehensive view of the Secure IoT SDLC landscape and indicate areas that require protection. To organise the domains in a logical manner, they were classified into three main groups:

- People: security considerations that affect all stakeholders involved in the life cycle of IoT solutions, from the software developers, to the end users of the product.
- Processes: secure development addresses security in the process of software development when a software project is conceived, initiated, developed, and brought to market.
- Technologies: technical measures and elements used in order to reduce vulnerabilities and flaws during the software development process.

Measures included in this section provide a short description but further details and a mapping with the threats and the corresponding references to back up each measure may be found in Annex A.

Figure 8: Security measures



4.2.1 People

4.2.1.1 Training and Awareness

- **PE-01. Define a corporate strategy for specific security training:** Ensure that all personnel participate in awareness-raising activities and training, focusing on the responsibilities of each role in applying security throughout the customised development process.
- **PE-02. Promote security awareness at all organisation levels:** Ensure that the entire organisation structure is made aware of the importance of safeguarding security from the first stages of development, including decision-makers.
- **PE-03. Assess the security skills to be updated:** Assess the internal security knowledge of the organisation to determine if the resources are aligned with the latest security advances by means of activities, exams, certifications, etc.
- **PE-04. Allocate resources to stay up to date with security topics:** Stay up to date with the latest industry trends as technology progresses in order to anticipate increasing risks and be prepared to face any new threats that may arise.

4.2.1.2 Roles and Privileges

- **PE-05. Establish security roles and privileges within the development project:** Define roles and responsibilities within the process so that the minimum sufficient level of privilege for each duty can be identified and assigned to the relevant person.
- **PE-06. Implement a separation of duties in the work team:** Carry out a segregation of duties in order to enable the collusion-resistant processes in SDLC and to minimise the risk exposure of its processes.
- **PE-07. Protect the process against privilege abuse:** Implement security controls to prevent the SLDC process from being compromised by any user with legitimate rights.
- **PE-08. Allocate resources for process monitoring:** Propose improvements to ensure that a problem during SDLC process can not cause an interruption of business continuity.

- **PE-09. Designate a physical security officer:** As a part of SDLC process, the physical facilities must be correctly protected and veiled by a security responsible.

4.2.1.3 Security Culture

- **PE-10. Establish governance and controls for critical security know-how:** Allocate efforts to ensure that certain critical skills stay at company by means of promotions, rewards, etc.
- **PE-11. Consult with security experts to improve the process:** Define a roadmap of new actions to be carried out in order to improve or complete the development process.
- **PE-12. Designate a Security Champion figure:** Centralize all security related aspects security in a cross-functional profile. This figure must manage all security topics during the SDLC.
- **PE-13. Monitor and respond to the supporting security incidents:** Security is not absolute and risks can materialise. Allocate resources to ensure the correct performance of SDLC support infrastructure.

4.2.2 Processes

4.2.2.1 Third-Party Management

- **PR-01. Implement a Supply chain management plan:** Ensure the integrity of the supply chain with a plan containing security frameworks, risk management, contract guidelines, etc.
- **PR-02. Assess the software dependency process:** Properly manage third parties and dependencies by means of risk management and security requirements.
- **PR-03. Test the Third Party process:** Define a process to test the security of all third party components prior to integration.
- **PR-04. Verify Third Party software and services:** Ensure that all third-party components meet the security and contract requirements stipulated.
- **PR-05. Disseminate a communication procedure to request external support:** Ensure that the whole organisation is informed of how to proceed if support is required from third-parties.
- **PR-06. Protect data against leakages:** Specify confidentiality clauses in order to prevent and avoid sensible data disclosures.
- **PR-07. Contractually require controlling and monitoring the external services through KPI's:** Apply controls to guarantee that external stakeholders involved in the SDLC process implement security into its processes.

4.2.2.2 Operations Management

- **PR-08. Define an Incident Management Plan:** Define a plan to manage vulnerabilities and updates, including for third-party components, with the necessary roles, responsibilities and activities to effectively respond to security incidents during development.
- **PR-09. Define a Change Management Plan:** Define a plan to manage changes to the development process with an informed view of the associated impact on the budget, schedule, scope, communication and resources.
- **PR-10. Implement Vulnerability and Patch Management:** Define a plan to manage vulnerabilities and updates during development, in a manner informed to the associated impact upon the security assurances attainable by the SDLC outcome.
- **PR-11. Implement Configuration Management:** Adequately manage the integrity of the system by ensuring that no unauthorised changes are made to the configuration.

4.2.2.3 SDLC Methodology

- **PR-12. Establish a Control Access and Authorisation Policy:** Define a privilege-based scheme to prevent unauthorised users from accessing restricted resources.
- **PR-13. Automate the SDLC process:** Automate processes supported by tested tools to reduce costs, human efforts and errors.
- **PR-14. Define Security Metrics:** Define, implement, and monitor security metrics to ensure fulfilment of the security requirements throughout lifecycle.
- **PR-15. Adopt a Maturity model:** Adopt a maturity model to improve security best practices during software development.
- **PR-16. Define and document the SDLC process:** Define a guide identifying security best practices and testing in each phase of SDLC. In addition, document a secure development process with security requirements and development guides and update it regularly based on the impact and criticality of newly discovered vulnerabilities.

4.2.2.4 Secure Deployment

- **PR-17. Define a disposal strategy:** Define a plan to dispose of the solution and all its data and components adequately at the end of the lifecycle.
- **PR-18. Establish process for SDLC vulnerabilities follow-up, monitoring and updates:** As threats progress, new vulnerabilities can affect the SDLC process. Perform a procedure to be updated is crucial for so as not to incur emerging issues.
- **PR-19. Implement a testing strategy:** Leverage automatic tools to ensure that minor errors are eliminated.
- **PR-20. Define a secure deployment strategy:** Establish a procedure for the deployment steps and ensure all stakeholders follow it.

4.2.2.5 Security Design

- **PR-21. Provide a secure framework:** Define a framework to implement security by design throughout the lifecycle of the solution.
- **PR-22. Apply least privilege principle:** Allocate only the user privileges required to perform the necessary operations of each role in the solution.
- **PR-23. Verify security controls:** Verify that the security controls implemented are reusable, sufficient, effective, reliable, audited, managed, and governed.
- **PR-24. Perform a design review:** Review the security of design periodically throughout development to ensure requirements are met and identify the attack surface.
- **PR-25. Specify security requirements:** Identify security requirements prior to development to implement features that ensure regulatory compliance and avoid vulnerabilities throughout the process.
- **PR-26. Perform risk assessment:** Identify risks throughout the software development process, analysing the sources, data storage, applications or third parties.
- **PR-27. Implement Threat Modelling:** Identify security objectives and define a threat model to implement countermeasures from the early stages of SDLC. Ensure that the threat modelling process is notified of changes by integrating its communication channel to that of the change management process.
- **PR-28. Implement Data classification:** Classify data based on their level of sensitivity to establish protection measures accordingly.
- **PR-29. Ensure that the hardware requirements derived from software requirements are considered:** Define and document requirements stemming from hardware, namely requirements that the software needs to meet in order to be consistently deployed on the targeted hardware.

4.2.2.6 Internal Policies

- **PR-30. Establish a communication plan for security measures:** Develop a communication plan to ensure that the whole organisation is informed about security considerations: security policies, security procedures, new updates, etc.
- **PR-31. Control the process against information disclosure:** Implement security controls to prevent any disclosure of information about security in the process that may compromise it.
- **PR-32. Verify and ensure the availability of updated security documents:** Periodically verify that the organisation's security documentation is available, updated, not tampered with, and strictly relevant to the matter of concern.
- **PR-33. Plan an alternative for unavailability cases:** Resources are not for all time and unavailability times can come up, plan and have a second option is essential.

4.2.3 Technologies

4.2.3.1 Access Control

- **TC-01. Implement authorisation:** Implement access control in IoT systems (and other underlying infrastructure) to ensure that the system verifies that users and applications have the right permissions.
- **TC-02. Secure storage of users' credentials:** Ensure that user credentials of IoT systems (and other underlying infrastructure) are protected from disclosure (e.g. using hash functions to store passwords, centralising password storage, using hardware with TPM, etc.).
- **TC-03. Deploy physical protection for systems:** Deploy security measures to prevent physical damages (intentional and unintentional) to IoT systems (and other underlying infrastructure).
- **TC-04. Implement key management and authentication mechanisms (e.g. FIDO):** Ensure that SDLC systems' service credentials are stored securely (not accessible for non-authorised parties) for those systems that need to use (e.g. using a secrets vaults).
- **TC-05. Control the physical access to the critical facilities:** Information cannot be accessible by any resource thus it must be protected against unauthorised access.

4.2.3.2 Third-Party Software

- **TC-06. Use libraries and third-party components that are patched for latest known vulnerabilities:** Verify that software libraries and other frameworks to be included in the software development project are patched for the latest known vulnerabilities. Establish an upgrade roadmap for libraries and third-party components. In the context of fully developed third-party software, ensure that the manufacturer monitors databases (e.g. CVE) and notifies new vulnerabilities.
- **TC-07. Use known secure frameworks with long-term support:** During the design, implement/develop and test the software under development, ensuring that the foundation technologies of the software will be maintained in the long term.

4.2.3.3 Secure Communication

- **TC-08. Use secure communication protocols:** Ensure that communications cannot be compromised by using encrypted channels, integrity protection and authenticated connection to share information between IoT systems.
- **TC-09. Use proven encryption techniques:** Protect data using encryption algorithms considered secure in any IoT systems and underlying infrastructure.
- **TC-10. Implement secure web interfaces:** Any web interface, including openness (i.e. publicly accessible interface) or technologies of implementation (i.e. HTTP/HTTPS/QUIC stack), in use for IoT systems should require security in order to be used (authentication and authorisation check).

- **TC-11. Secure session management:** Due to the number of communications that take place in the IoT systems, many sessions are established between them. It is necessary to implement secure sessions in order to ensure security of communications.

4.2.3.4 Secure Code

- **TC-12. Implement secure coding practices:** During the design, implement/develop and test the software under development, ensuring that an authentication mechanism referenced in globally accepted best practices is in place, that errors are handled correctly, that all input/output data are validated before accepting it, and that queries use parameterisation (or other equivalent security measure) to avoid code injections.
- **TC-13. Provide audit capability:** During the design, implement/develop and test the software under development, ensuring that relevant security events are registered in software logs.
- **TC-14. Follow the principles of security by design and by default:** During the design, implement/develop and test the software under development, ensuring that these principles are the foundation of the software.
- **TC-15. Implement software development techniques:** Choose software development techniques (e.g. microservices) or architecture that produce clean and maintainable code.
- **TC-16. Verify production code:** Ensure that the production environment is controlled and resources are securely utilised.
- **TC-17. Ensure security for patches and updates:** Ensure that the SDLC model always allows for modification/patching/update of software in a secure fashion (tested, reviewed, etc.) before deploying any software change.
- **TC-18. Implement measures against rogue code and fraud detection:** Deploy the countermeasures required in your SDLC process to detect potential rogue code in all relevant phases.
- **TC-19. Implement anti-tampering features:** Deploy the countermeasures required in your SDLC process to prevent unauthorised code modification in all steps of the process.

4.2.3.5 Security Reviews

- **TC-20. Apply secure code review:** Ensure that, in SDLC, the source code is reviewed in terms of security before accepting it.
- **TC-21. Perform an attack surface analysis:** Ensure that, in SDLC, the attack surface is analysed and documented.
- **TC-22. Perform IoT SDLC tests:** Ensure that throughout the SDLC, at least a penetration test is carried out when the software is complete. Additional testing should also be considered based on functional needs and risk assessment.
- **TC-23. Design a contingency plan:** Ensure that the contingency plan is aligned with software development process in order to avoid an interruption during the SDLC phases.
- **TC-24. Monitor requirements to ensure the SDLC success:** Implement a system to guarantee the requirement fulfilment.

4.2.3.6 Security of SDLC Infrastructure

- **TC-25. Ensure secure logging and implement monitoring:** Ensure that logs of IoT systems and logs of the various tools used throughout the SDLC are stored in a secure place, and that they are constantly monitored.
- **TC-26. Implement physical detection systems:** Implement a detection systems to control the state of the data hosting facilities (temperature, fire, etc.) to protect the SDLC support infrastructure against unavailability scenarios.

- **TC-27. Define a mitigation plan for physical damages:** Ensure that the possible risks that can affect and danger to the SDLC physical infrastructure are covered by a countermeasure in an action plan.
- **TC-28. Use whitelists for allowed applications:** Control the applications that can be used by providing an authorised list and denying the rest.
- **TC-29. Audit the access to the SDLC infrastructure:** Implement control systems based on logs in order to manage and track all accesses to physical and logical systems**.**
- **TC-30. Implement an identification protocol in your facilities:** Provide personal and non-transferrable identification to all personnel, both internal and external.

4.2.3.7 Secure Implementation

- **TC-31. Enforce the change of default settings:** Ensure change of default settings at first user interaction.
- **TC-32. Use substantiated underlying components:** Restrict component customizations in order to avoid loss of security functionalities.
- **TC-33. Provide secure configuration options for end users:** Ensure that end users have options to continuously improve security through the solution's settings.
- **TC-34. Implement interoperability open standards:** Implement interoperability open standards (e.g. OCF) to enhance secure integration processes.
- **TC-35. Enable devices to advertise their access and network functionality:** By enabling devices to advertise their intended and supported functionality, the threat surface can be significantly reduced. An indicative practical example involves the use of IETF RFC 8520 on Manufacturer Usage Description Specification[48].

[48] See https://www.rfc-editor.org/info/rfc8520, March 2019

A ANNEX: MAPPING OF SECURITY MEASURES

4.3 PEOPLE

Security domain	Title	Description	Threats	SSDLC phases involved	Reference title
Training and Awareness	PE-01. **Define a corporate strategy for specific security training**	Ensure that all personnel participate in awareness-raising activities and training, focusing on how to apply security in a SDLC process. These activities must be customised depending on roles and responsibilities in the software lifecycle. Security knowledge must be a requirement before starting any SDLC project. The training should include information about best practices to ensure a safe work environment, security roles and responsibilities within the project phases, and security tasks, as well as security policies, standards, applicable regulations and legislation.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - Software Assurance Maturity Model (SAMM 1.5v). OWASP. - Fundamental Practices for Secure Software Development – SAFECODE - Systems and software engineering - Software life cycle processes. ISO 12207 - BUILDING SECURITY IN MATURIT Y MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST

					- Information Security Management. ISO 27001 - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - The BSA Framework for Secure Software. BSA Software Alliance
Training and Awareness	PE-02. **Promote security awareness at all organisation levels**	Include security activities to raise awareness among the employees (courses, simulations, talks, security capsules via e-mail, round tables, etc.) about how to address security during the development process. If the entire organization is sensitised to security, it will be easier to implement the necessary measures to achieve a process as secure as possible. These activities may be focused on different profiles: rewards depending on developers security training, promotion for specific project manager security training, decision-makers security awareness by performing attack simulations (financial impact simulations), etc.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - Secure Coding. Practical steps to defend your web apps. SANS. - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM. - Software Assurance Maturity Model (SAMM). OWASP SAMM. - Security Assurance in the SDLC for the Internet of Things. ISACA. - IoT Security Maturity Model. Industrial Internet Consortium (IIC). - Secure Software Development Life Cycle Processes. CISA. - THE DZONE GUIDE TO 2015 EDITION APPLICATION SECURITY. DZONE. - The BSA Framework for Secure Software. BSA Software Alliance

					- "Systems and software engineering -Software life cycle processes. ISO 12207" - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - "Information technology - Security techniques - Information security management systems - Requirements. ISO 27001"
Training and Awareness	PE-03. **Assess the security skills to be updated**	An organisation must stay up to date with the latest security knowledge and certifications of its employees. This assessment must be a matrix covering both specific and cross-cutting security aspects by means of exams, the renewal of certifications, internal assessments, training providers in order to classify the staff by levels in function of their security preparation and background, thus making it possible to assign them specific tasks according to their level during the different SDLC phases. At least, once a year must be updated this information.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - "Systems and software engineering - Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Software Assurance Maturity Model (SAMM). OWASP SAMM - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1

Training and Awareness	PE-04. **Allocate resources to stay up to date with security topics**	Appoint resources and promote the implementation of monitoring, tracking and update activities by means of threat intelligence in order to be aware of the status of current vulnerabilities and new types of attacks that may affect your industry. Along with security lessons learned, this information must be centralised in an internal repository. The result of these tasks will help to prevent future security issues.	PERSONNEL LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM. - Strategic Principles for Securing the Internet of Things (IoT). U.S. Department of Homeland Security. - Code of Practice for Consumer IoT Security. UK. - "Systems and software engineering -Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - GSMA IoT Security Assessment Checklist - Reference CLP11_5
Roles and Privileges	PE-05. **Establish security roles and privileges within the development project**	Ensure that development teams work alongside security teams by means of the definition, identification and allocation of functions, responsibilities and tasks in relation to security in all phases of development. This measure ensures that security is addressed when required.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance. - Systems and software engineering — Software life cycle processes. ISO 12207. - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST. - Security for industrial automation and control systems. Part 4-1: Secure product development

					lifecycle requirements. IEC 62443-4-1. - CLASP Concepts – OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Software Assurance Maturity Model (SAMM 1.5v). OWASP - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53
Roles and Privileges	PE-06. **Implement a separation of duties in the work team**	It is essential to ensure a proper separation of duties during the development process, implementing security controls in order to prevent security impacts. Without a separation of duties, people could carry out fraudulent activities in any phase by leveraging their privileges. The goal is to avoid the possibility of users having admin rights or inadequate profiles for critical tasks.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security Design Principles. OWASP. - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - Implementing Segregation of Duties. ISACA. - Secure Coding. Practical steps to defend your web apps. SANS. - Information technology — Security techniques — Information security management systems — Requirements. ISO 27001 - Technical Considerations White Paper. FCC TAC.

					- "Systems and software engineering — Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53
Roles and Privileges	PE-07. **Protect the process against privilege abuse**	The integrity of the development process must be guaranteed. Implement security measures to access project resources so as to prevent any team member (insider, third-party) with privileges from disabling security controls, establishing or modifying policies and guides, collecting sensitive data, etc. Perform audits periodically to ensure the integrity of information.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - The BSA Framework for Secure Software. BSA Software Alliance - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM

					- Fundamental Practices for Secure Software Development, SAFECODE - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - Software Assurance Maturity Model (SAMM). OWASP SAMM
Roles and Privileges	PE-08. **Allocate resources for process monitoring**	Designate a person to perform, review and put forth improvement actions for the business continuity plan: preventing network crashes and service redundancies, safeguarding critical points that may slow down or compromise the development process (SDLC), like the unavailability of third-party services, the uncontrolled access to sensitive locations where information is stored, the lack or expiry of software licences involved in the SDLC, etc.	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - The BSA Framework for Secure Software. BSA Software Alliance - "Systems and software engineering — Software life cycle processes. ISO 12207" - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE

			NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS		SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Software Assurance Maturity Model (SAMM 1.5v). OWASP - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Information security management. ISO 27001 - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1
Roles and Privileges	PE-09. **Designate a physical security officer**	Designate a resource responsible for fulfilling the plan or procedure defined to take actions when risks have to be mitigated and to contain them and prevent them from resulting in additional risks if information regarding the SDLC or spaces where it is stored are compromised due to a fire, flood, electric show, etc.	OUTAGES PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - The BSA Framework for Secure Software. BSA Software Alliance - "Systems and software engineering — Software life cycle processes. ISO 12207" - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE

					SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Software Assurance Maturity Model (SAMM 1.5v). OWASP - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Information security management. ISO 27001 - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1
Security Culture	PE-10. **Establish governance and controls for critical security know-how**	The profiles with security knowledge are essential in the development process. Design a policy to reward and stimulate this kind of profiles in order to reduce the risk of security knowledge being lost in the organisation, as well as internal threats: staff turnover, espionage, sabotage, etc.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- "Payment Card Industry - Software Security Framework. PCI Security Standards Council" - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - "Systems and software engineering - Software life cycle processes. ISO 12207" - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001"

					- The BSA Framework for Secure Software. BSA Software Alliance - Software Assurance Maturity Model (SAMM). OWASP SAMM - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1
Security Culture	PE-11. **Consult with security experts to improve the process**	Engage internal or external security support to complement, support, or cover security aspects and to contribute during specific activities, such as: - Use of external penetration testers during the testing phase to provide with different perspectives, adding robustness to the process. - Use of specific expert in security tools to control access to the process resources, increasing the confidentiality and integrity throughout all phases - Use of a coach to bring security into the SDLC phases, etc.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - "Systems and software engineering — - Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Software Assurance Maturity Model (SAMM). OWASP SAMM
Security Culture	PE-12. **Designate a Security Champion figure**	Designate a role to centralise all issues related to software development security. This figure should not be responsible for the implementation of security functions, but for coordination, follow-up, planning, and monitoring efforts and activities related to security. This position should be understood as a	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- Software Security Takes a Champion. SAFECODE - The BSA Framework for Secure Software. BSA Software Alliance

		bridge, a security catalyst among organisation statements (developers, team leaders and decision-makers).	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Security Champions. OWASP - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Security Culture	PE-13. **Monitor and respond to the supporting security incidents**	Designate a resource (internal or through a third-party service) to monitor, operate and respond to alarms generated by events resulting from the loss or poor performance of the infrastructures that support the secure development process (sSDLC), which are essential for correct functioning. This would be the case of communications slowing down or being lost, preventing the exchange of documentation between team members or with third parties, as well as the loss or unavailability of data	OUTAGES UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP

		repositories, be they owned or through a cloud service, etc.	DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- "Systems and software engineering —Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - The BSA Framework for Secure Software. BSA Software Alliance - NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO

4.4 PROCESSES

Security domain	Title	Description	Threats	SSDLC phases involved	Reference title
Third-Party Management	PR-01. **Implement a supply chain management plan**	During a SDLC process, components or services are outsourced to a third-party (external supplier). A supply chain management plan should be implemented and integrated into this process to ensure the integrity of the SDLC. This plan should include information related to security frameworks to be used, risk management, third-party acquisition management, purchasing contract definition, etc., as well as controls that prevent third-party tampering or compromise, e.g. avoiding reverse engineering to better understand how a software works and to leverage this knowledge to carry-out malicious activities (anti-reverse engineering).	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM 1.5v). OWASP - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)

					- Application Security Verification Standard 4.0 (ASVS). OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Third-Party Management	PR-02. **Assess the software dependency process**	Ensure a proper management of third parties and dependencies of the software development using risk management and integrating security requirements in contracts, ensuring the visibility and traceability of components, documenting all components and subcomponents acquired, managing incidents, scanning dependencies, etc. CVSS must be consulted when choosing a third-party software/library and dependencies must be checked periodically or every time they are updated. An open-source update plan for IoT must be considered and followed, monitoring and managing third-party vulnerabilities.	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Software Assurance Maturity Model (SAMM 1.5v). OWASP - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - OWASP Dependency-Check. OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development

					lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Third-Party Management	PR-03. **Test third-party processes**	Define a process to secure third-party code by performing controls in order to detect vulnerable components, such as penetration tests, fuzzing tests, validation tests, etc. The objective is to verify that the technical, functional, and business requirements are met. It is recommended to execute this process in every iteration (sprint).	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	TESTING AND ACCEPTANCE	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - The BSA Framework for Secure Software. BSA Software Alliance - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Application Security Verification Standard 4.0 (ASVS). OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

					- The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM 1.5v). OWASP - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Third-Party Management	PR-04. **Verify third-party software and services**	Verify that the software provided by third parties meets the security requirements, including the requirements specified in contracts. Define requirements for commercial software and verify the evidence provided by the supplier (audits, testing, software certifications, etc.). It is advisable to check that the requirements have been met at least every time a delivery occurs.	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- "Systems and software engineering —Software life cycle processes. ISO 12207" - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Proactive Controls for developers v3.0. OWASP - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - Software Assurance Maturity Model (SAMM 1.5v). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)

					- Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST
Third-Party Management	PR-05. **Disseminate a communication procedure to request external support**	Establish a procedure for the organisation to know the steps to be taken in the event of requiring support from external providers to face events or incidents concerning cloud services, testing services, etc., indicating at least the person of contact in charge of the service, the request model and communication channel, incident follow-up and management, the remediation, documentation updates, version, etc.	OUTAGES UNINTENTIONAL DAMAGES (accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM 1.5v). OWASP - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE

					- Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS) . OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Third-Party Management	PR-06. **Protect data against leakages**	Require, audit, and specify confidentiality clauses for all internal and external personnel with access to the facilities with a view to preventing leakages of information on topics such as software design or architecture, which may represent sabotage or espionage attempts during any of the phases of the product development (SDLC).	PERSONNEL OUTAGES PHYSICAL ATTACK LEGAL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM 1.5v). OWASP - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207"

					- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS) . OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Third-Party Management	PR-07. **Contractually require controlling and monitoring the external services through KPI's**	By means of contractual clauses, ensure that both internal and external service providers implement security controls to measure the quality of the service (e.g. service incident response time, unavailability terms, etc.) and detect potential flaws, stipulating a reporting period (e.g. on a weekly basis) for the KPIs to assess the service, along with measures to be taken to prevent impacts on the SDLC phases, such as, for instance, the maintenance phase.	OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS	- Software Assurance Maturity Model (SAMM 1.5v). OWASP - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - NIST SP 800 53r5: Security and Privacy Controls for Federal

					Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Operations Management	PR-08. **Define an Incident Management plan**	Provide guidance for the definition and allocation of roles, responsibilities and activities to be implemented by the organisations in the event of security incidents. Security incidents pose a higher impact as the SDLC process reaches the last stages, so it is crucial to manage it following an established resolution process. This process should contain at least: - Incident detection and registration. - Classification and initial support. - Research and diagnosis. - Solution and service restoration. - Incident closure. - Monitoring, follow-up and communication of the incident. Maintain, to the greatest extent feasible, a full inventory of third party components and dependencies, and track vulnerabilities, patches, and updates to those components to preserve security. An Incident Management Plan should be defined and periodically updated.	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES PHYSICAL ATTACKS LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - The BSA Framework for Secure Software. BSA Software Alliance - NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

					- ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP11_5
Operations Management	PR-09. **Define a Change Management plan**	A Change Management Plan should be defined to manage any changes that may be take place during the SDLC process. This entails ensuring control over the budget, schedule, scope, communication, and resources. The main focus is to minimise the impact a change throughout the process could have on the different assets: business, team, users, and other important stakeholders. Change management is a highly important activity both in the development and integration phases (changes may affect the requirements) as well as in the maintenance and disposal, during updates, patches or functionalities changes. The plan should detail a procedure containing at least: - Identification and formal request. - Impact analysis and assessment. - Validation. - Planning and testing. - Implementation. - Monitoring, follow-up and communication of the change.	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - "Payment Card Industry - Software Security Framework. PCI Security Standards Council" - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_5

Operations Management	PR-10. **Implement Vulnerability and Patch Management**	Develop a process for vulnerability and update management as well as for vulnerability disclosure from external and internal parties to reduce the risk of system failures, especially in operation. This process must encompass identification and patching processes and the communication process with the relevant stakeholders when a vulnerability is discovered. This guide should document the process and controls to be carried out by the project team, such as: - Vulnerability discovery/disclosure - Identification of the affected asset - Development of the solution or patch - Testing, solution compliance - Patch implementation, update - Update follow-up	PERSONNEL LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP12_5
Operations Management	PR-11. **Implement Configuration Management**	Configuration management focuses on maintaining the integrity of the system, ensuring that uncontrolled changes are implemented during the deployment and maintenance phases of the SDLC process. It must be configured in a restrictive way to guarantee maximum resistance against malicious or unintentional attacks (changes to a file or code element, adaptation of security settings on the operating environment, etc.).	PERSONNEL LEGAL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST

					- Application Security Verification Standard 4.0 (ASVS) . OWASP - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - "Systems and software engineering —Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP11_5
SDLC Methodology	PR-12. **Establish a Control Access and Authorisation policy**	The access to resources and processes should be protected to prevent users without authorisation from accessing restricted resources (e.g. data repository, password storage, test reports, etc.) at any stage of the SDLC process. By establishing user access privileges, it is possible to ensure the confidentiality, integrity and	PEPERSONNEL UNINTENTIONAL DAMAGES OUTAGES PHYSICAL ATTACKS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION	- The BSA Framework for Secure Software. BSA Software Alliance - Fundamental Practices for Secure Software Development, SAFECODE

		availability of data and process: - Only authorised persons (based on their privileges) will be able to access the resources (phases, information, systems, equipment, programs, applications, databases, networks, etc.). - The control access will make it possible to identify and audit the accesses that have taken place, establishing internal security controls.	LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	MAINTENANCE AND DISPOSAL	- Proactive Controls for developers v3.0. OWASP - Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - The BSA Framework for Secure Software. BSA Software Alliance - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP11_5
SDLC Methodology	PR-13. **Automate the SDLC process**	Processes supported by tested tools should be automated in order to reduce costs and human efforts and errors. The main objective is to improve the monitoring and measurement of development progress, as well as the implementation of security measures for the process. The result of automated testing must be analysed, since automated tools	PERSONNEL UNINTENTIONAL DAMAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST

		are based on patterns that can suffer modifications, which may not be detected and produce false positives. In cases where this is not possible, manual tools should be used. It is recommended to execute this process in every iteration (sprint).	ACTIVITY / ABUSE DAMAGE / LOSS		- BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - ISO-IEC 27001. ISO - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - The BSA Framework for Secure Software. BSA Software Alliance
SDLC Methodology	PR-14. **Define security metrics**	Implement security metrics, which should be defined and tracked in order to verify that the specified security requirements have been fulfilled during the Software Development process. Checking the security metrics should be a necessary requirement to: - Evaluate the security maturity and identify actions to improve the process (SMM). - Reassure quality for all SDLC phases. - Assess the status of an ongoing process. - Track potential risks. - Discover process issues before they become critical. - Evaluate the ability of the project team to control the quality of software products.	PERSONNEL UNINTENTIONAL DAMAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Software Assurance Maturity Model (SAMM 1.5v). OWASP - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - The BSA Framework for Secure Software. BSA Software Alliance

SDLC Methodology	PR-15. **Adopt a maturity model**	Adopt a software assurance maturity model for software development to identify security best practices during the process (e.g. OWASP SAMM and BSIMM). The implementation of security is considered mature if the mechanisms used effectively achieve the security requirements. The analysis takes account of specific threats to the regulatory and compliance requirements of an organisation's industry, the unique risks present in an environment, and the organisation's threat profile.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- "Payment Card Industry - Software Security Framework. PCI Security Standards Council" - "Software Assurance Maturity Model (SAMM) - BUILDING SECURITY IN MATURITY MODEL (BSIMM). OWASP SAMM - BSIMM" - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - BSIMM. - The BSA Framework for Secure Software. BSA Software Alliance.
SDLC Methodology	PR-16. **Define and document the SDLC process**	Define security guides establishing the performance of security tests during the different phases of development, defining best practices such as the generation of use cases, the performance of penetration tests during development, the use of tools, the performance of security tests at the end of the process, etc. It is recommended to execute this process in every iteration (sprint) or when a modification is implemented.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Proactive Controls for developers v3.0. OWASP - The BSA Framework for Secure Software. BSA Software Alliance

Secure Deployment	PR-17. **Define a disposal strategy**	A plan for the withdrawal of the solution at the end of the life-cycle must be considered. The plan must include measures to formally retire stored data according to the needs (organisational, data privacy, regulatory compliance) including third-party components and the communication to the stakeholders. To ensure the disposal process, an audit log must be maintained.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) / LEGAL	MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering —Software life cycle processes. ISO 12207" - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP12_5 and CLP13_8
Secure Deployment	PR-18. **Establish process for SDLC vulnerabilities follow-up, monitoring and updates**	Establish a procedure to inform of new published vulnerabilities (e.g. establishing mechanisms to receive feedback from security research community) that may affect the software development life cycle (including those that affect third party components), so that they can be taken into account in all phases. This information measure can help the organisation not to incur into known errors, and to take them into account as security requirements in the requirements phase of the SDLC for future developments. This includes not only for software development projects but also as requirements to be stipulated with third parties for services or infrastructure, such as external data repository services for backups or access control systems for server rooms.	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - "Systems and software engineering —Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM

Secure Deployment	PR-19. **Implement a testing strategy**	Define testing strategy that eliminates trivial bugs by utilizing automated tools for both static and dynamic analysis. Use infrastructure-as-code or digital twins to ensure accuracy of testing processes for IoT in critical infrastructure. This strategy should contain considerations such as test scope definition, criteria to be used, quality control points, procedures to solve errors, etc.	PERSONNEL UNINTENTIONAL DAMAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1. - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM. - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST. - Fundamental Practices for Secure Software Development, SAFECODE. - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA. - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST. - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM. - Application Security Verification Standard 4.0 (ASVS). OWASP. - The BSA Framework for Secure Software. BSA Software Alliance.
Secure Deployment	PR-20. **Define a secure deployment strategy**	Define effective and secure deployment strategy, weighing the options in terms of the impact of change on the targeted systems, and the end-users. It must be considered that only qualified personnel must have access to deployment environment, audit systems for all deployments establishing versions control, acceptance threshold, person who conducted it, etc.	PERSONNEL LEGAL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance. - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST. - Application Security Verification Standard 4.0 (ASVS). OWASP.

					- Proactive Controls for developers v3.0. OWASP. - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA. - "Systems and software engineering — - Software life cycle processes. ISO 12207. " - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE. - Fundamental Practices for Secure Software Development, SAFECODE. - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - Application Security Verification Standard 4.0 (ASVS). OWASP. - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1. - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST. - ISO-IEC 27001. ISO.
Security Design	PR-21. **Provide a secure framework**	Adopt a security framework encompassing the necessary requirements in order to define and provide guides and policies to be implemented throughout the Software Development Life Cycle process. Known frameworks minimise risks and threats that could affect the process. Define a secure framework to ensure in-depth defence and observe security by design considering the entire life cycle of the solution and comprising the design, maintenance, and disposal phases.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - "Payment Card Industry - Software Security Framework. PCI Security Standards Council"

					- ISO-IEC 27001. ISO - "Systems and software engineering —Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Software Assurance Maturity Model (SAMM). OWASP SAMM - Security-First Design for IoT Devices - IoTCentral. IoTCentral - Application Security Verification Standard 4.0 (ASVS). OWASP
Security Design	PR-22. **Apply least privilege principle**	Ensure that user and software privileges are strictly limited to features required to carry out the operations. Limiting permissions and rights in the tasks to be performed is an important activity during the SDLC process, gaining greater relevance in the Design and Testing phases. Privileges must have a resilient configuration against unauthorised changes, and must be in line with authorisation and access control policies.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Proactive Controls for developers v3.0. OWASP

					- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - GSMA IoT Security Assessment Checklist - Reference CLP12_5
Security Design	PR-23. **Verify security controls**	Allocate a project resource (i.e. a data repository) to centralise security control management activities (security control updates, tracking, monitoring) to be carried out during the SDLC process. Verify that the security controls implemented are accessible, controlled regularly, safe, and reusable, avoiding duplicates and ensuring they are efficient, reliable, and based on international best practices. It is recommended to review and update them periodically, at least once a year or upon every important change (new technologies, project's lessons learned, etc.).	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Application Security Verification Standard 4.0 (ASVS). OWASP - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - The BSA Framework for Secure Software. BSA Software Alliance - Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)
Security Design	PR-24. **Perform a design review**	During the Design phase of the SDLC process, solutions must be reviewed from the point of view of security, ensuring that security requirements, which have been previously defined, have been met, identifying the attack surface, carrying out a threat modelling, providing security mechanisms, and scheduling periodic reviews throughout the development process based on milestones. It is recommended to execute this process in every iteration (sprint).	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DESIGN	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - "Systems and software engineering —Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - The BSA Framework for Secure Software. BSA Software Alliance

					- NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - ISO-IEC 27001. ISO - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - "Software Assurance Maturity Model (SAMM). OWASP SAMM - CSA Guidance - CSA" - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Security Design	PR-25. **Specify security requirements**	Establishing security requirements prior to development makes it possible to implement security functionalities that ensure compliance with standards and laws and avoid known vulnerabilities. The definition of these security requirements makes it possible to industrialise the security standards that apply to different developments, complying with a series of standard security controls, making it possible to fix past problems, and helping to prevent future flaws. Some best practices would be the performance of security and requirement compliance assessments, the specification of requirements based on known risks, the definition of requirements in agreement with providers, the implementation of security user stories, and the performance of security audits. They must be reviewed periodically, at least every time known best practices and regulations are updated.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS	- Proactive Controls for developers v3.0. OWASP - Software Assurance Maturity Model (SAMM). OWASP SAMM - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS) . OWASP

					- The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Security Design	PR-26. **Perform a risk assessment**	Identify risks throughout the software development process, analysing the sources, data storage, applications or third parties. As part of the analysis, make sure that the data to be protected are reliable, and that there are measures in place to prevent the unauthorised access, loss, destruction or manipulation thereof. A security risk assessment should include: - The analysis of the potential risk if the security of each of the following components were compromised: sources, storage, sensitive data, applications, data stores, cloud services. - The analysis of data classification mechanisms and data security capabilities in order to protect sensitive data from unauthorised use, access, loss, destruction or sabotages. - The analysis of the potential for trusted insiders to misuse their privileged access to data. Based on these analyses, implement best practices for the mitigation of each potential security threat. This process must be periodically reviewed, at least once a year.	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Software Assurance Maturity Model (SAMM). OWASP SAMM - The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework Cloud Security Alliance (CSA) - International Organization for Standardization (ISO). ISO27001 - "Systems and software engineering —Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_5

Security Design	PR-27. **Implement Threat Modelling**	In the software design phase, it is necessary to study the architecture and the design of the system by means of threat modelling techniques. Threat modelling thoroughly identifies key assets thus far hidden, as well as their associated risks. Through this technique, developers can focus their efforts on subsequent phases, applying tools oriented to the uncovered risks. Developers should regard the following aspects as best practices: - Building and maintaining threat models for each application, defining the profile of potential attackers by means of the software architecture. - Building and maintaining abuse case models per project, establishing threat assessment systems. Explicitly evaluate the risk of third-party components and generate threat models with security controls.	OUTAGES NEFARIOUS ACTIVITY / ABUSE	DESIGN	- Security Assurance in the SDLC for the Internet of Things - ISACA. - "Software Assurance Maturity Model (SAMM) - Open Reference Architecture for Security and Privacy Tactical Threat Modeling Veracode - OWASP SAMM. Open Reference Architecture for Security and Privacy - Security in the Software Development Lifecycle- USENIX " - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Security Design	PR-28. **Implement data classification**	Data are a critical asset from the point of view of security. Based on the classification of information (status, use, owner, risk, etc.), assign a level of sensitivity to the data in the requirement phase to establish the corresponding protection measures throughout the SDLC process (ensuring the privacy of data at	PERSONNEL FAILURES / MALFUNCTIONS LEGAL DAMAGE / LOSS	REQUIREMENTS DESIGN	- Proactive Controls for developers v3.0. OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)

		rest by means of encryption, preventing unauthorised access by means of control access, etc.).			- ISO-IEC 27001. ISO - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Application Security Verification Standard 4.0 (ASVS). OWASP - GSMA IoT Security Assessment Checklist - Reference CLP12_5
Security Design	PR-29. **Ensure that the hardware requirements derived from software requirements are considered**	Bear in mind that, as part of the functional requirements, it is essential to take into account the implications for hardware derived from software security requirements. Implement controls during the Requirements phase in order to associate/map software security requirement and hardware requirements and ultimately fulfil them. For instance, associate secure boot mechanisms with the use of chips/modules supporting this technology (Root-of-Trust), identifying hardware needs based on the communication protocol chosen in order to determine the power source depending on consumption, etc.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS	- Proactive Controls for developers v3.0. OWASP - Software Assurance Maturity Model (SAMM). OWASP SAMM - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance

					- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Internal Policies	PR-30. **Establish a communication plan for security measures**	Develop a communication plan targeted at all persons involved in the development process (specially third-parties) in order to report on the security measures that must be observed for a proper development, such as applicable regulations, security frameworks and methodologies to be used, security best practices, etc. This plan must be reviewed, validated and disseminated in the organisation at least once a year.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Fundamental Practices for Secure Software Development, SAFECODE - Software Assurance Maturity Model (SAMM 1.5v). OWASP - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM

Internal Policies	PR-31. **Control the process against information disclosure**	Ensure that process information is not disclosed or tampered with by any stakeholder throughout the lifecycle without prior authorisation, as it could result in a compromise of intellectual property, a breach of regulatory compliance, reputational losses, etc. Security measures should be considered such as role-based access control, authorisation, permission assignment, non-disclosure clauses in the contracts, etc.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance. - Proactive Controls for developers v3.0. OWASP. - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - "Information technology- Security techniques - Information security management systems - Requirements. ISO 27001" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53
Internal Policies	PR-32. **Verify and ensure the availability of updated security documents**	Ensure the availability of security policies, procedures, guides, applicable regulations and requirements for developers. Throughout the process, a centralised repository must be accessible. Organisations have to implement change management to guarantee the integrity of data and avoid introducing errors in the process.	PERSONNEL UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - Proactive Controls for developers v3.0. OWASP

					- "Systems and software engineering -Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Software Assurance Maturity Model (SAMM). OWASP SAMM
Internal Policies	PR-33. **Plan an alternative for unavailability cases**	Distribute your resources so as to not centralise security knowledge in a single resource, be it internal or through a third party, with a view to avoiding cases of unavailability that may bring the secure development (sSDLC) process to a standstill in any of the phases. This would be the case, for instance, when there is only one security pentesting specialist during the Testing phase. This measure focuses on providing an alternative for SDLC critical points (resources redundancy).	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - "Systems and software engineering —Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Software Assurance Maturity Model (SAMM). OWASP SAMM - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1

4.5 TECHNOLOGIES

Security domain	Title	Description	Threats	SSDLC phases involved	Reference title
Access Control	TC-01. **Implement authorisation**	Implement access control in IoT systems and software to ensure that the system verifies that users and applications have the right permissions allocated to their roles to access system resources. This can be done by means of the least privilege principle and a strategy regarding authorisation policies, controls, and design principles for different categories of data. If a password is being used for authentication, the asset should force the user to change the password at first use. Furthermore, typed characters should be masked.	PERSONNEL PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION DEPLOYMENT AND INTEGRATION	- The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - GSMA IoT Security Assessment Checklist - Reference CLP12_5 and CLP13_6

Access Control	TC-02. **Secure storage of users' credentials**	Ensure that user credentials of IoT systems (and other underlying infrastructure) are secured. Passwords must always be hashed with a salt. Password bolts are often used to hard code credentials for system communications, so that the system has to request the credentials before accessing a resource. This measure prevents to access to sensitive functionalities and data (e.g. source code).	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Proactive Controls for developers v3.0. OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_6
Access Control	TC-03. **Deploy physical protection for systems**	Systems and their corresponding hardware must be protected against unauthorised modification attempts and direct access, as well as other dangers (fire, water, cooling issues, etc.). Physical access must be controlled and unused physical interfaces must be disabled or inaccessible. Removing unnecessary items helps to reduce the attack surface.	PERSONNEL OUTAGES PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Security Design Principles. OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1

					- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - The BSA Framework for Secure Software. BSA Software Alliance - GSMA IoT Security Assessment Checklist - Reference CLP13_7
Access Control	TC-04. **Implement Key management and authentication mechanisms (e.g. FIDO)**	Ensure the secure management of service credentials for your SDLC systems, especially in the context of web and cloud services. They must be temporary and single-use, and the right communication privileges have to be allocated for the different service credentials (e.g. user credentials vs. System credentials).	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DESIGN DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - GSMA IoT Security Assessment Checklist - Reference CLP12_5 and CLP13_6
Access Control	TC-05. **Control the physical access to the critical facilities**	Implement a physical access control system with authorisation mechanisms to identify users and their privileges. This system should be monitored and provide event logs for all accesses, including unauthorised access attempts. The access to physical facilities storing information concerning the SDLC or systems that support the process (repositories, network equipment, documentation files, etc.) must be adequately protected. This measure can be stipulated in contracts with external providers concerning the control of facilities containing information about the service hired. Additionally, a CCTV surveillance system could be configured to communicate with an alarm	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS	- The BSA Framework for Secure Software. BSA Software Alliance - Fundamental Practices for Secure Software Development, SAFECODE - Proactive Controls for developers v3.0. OWASP - Application Security Verification Standard 4.0 (ASVS) . OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)

		system (e.g. SIEM) and send signals alerting to unauthorised access attempts.			- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Third-Party Software	TC-06. **Use libraries and third-party component that are patched for latest known vulnerabilities**	Ensure that your SDLC model enforces the use of the latest versions of third-party libraries to safeguard their integrity. The most costly and extensive attacks have been caused by this issue. Check the versions of your dependencies at least quarterly once the software under construction is in production.	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE MAINTENANCE AND DISPOSAL	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Software Assurance Maturity Model (SAMM). OWASP SAMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST

					- The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1
Third-Party Software	TC-07. **Use known secure frameworks with long-term support**	For the foundation technologies of the software under development, use and verify known software security frameworks from third party providers supplying LTS (Long Time Support) or similar. Some software have associated security flaws, so it is essential to make sure that these components can be trusted in the long term. These components should be chosen considering if they are maintained by a private organisation or an active group, if security patches are available in short time when a vulnerability is disclosed and if developers can be contacted if a vulnerability is identified.	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DESIGN DEVELOPMENT/IMPLEMENTATION	- The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Application Security Verification Standard 4.0 (ASVS) . OWASP - ISO-IEC 27001. ISO - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)
Secure Communication	TC-08. **Use secure communication protocols**	Ensure that communications are always encrypted between IoT systems and the underlying infrastructure they are integrated. Additionally, it is	PERSONNEL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION	- Application Security Verification Standard 4.0 (ASVS). OWASP

		also recommended to implement mechanisms to authenticate communications.		TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- NISTIR 8200 - Interagency Report on the Status of International Cybersecurity Standardization for the Internet of Things (IoT). NIST - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - The BSA Framework for Secure Software. BSA Software Alliance - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - GSMA IoT Security Assessment Checklist - Reference CLP12_6 and CLP13_6
Secure Communication	TC-09. **Use proven encryption techniques**	In any IoT system, data must be encrypted, both at rest and in transit, using a recognised encryption algorithm. However, even resilient algorithms are not efficient if they are not properly used (e.g. sufficient key length). It is necessary to use an initialisation vector and to guarantee a minimum level of entropy. It is highly recommended to apply hashes to protect electronic signatures. These measures apply both to original data and to any existing backups. Potential legal consequences may arise if due diligence it is not in place for data protection.	LEGAL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP - The BSA Framework for Secure Software. BSA Software Alliance

					- BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Fundamental Practices for Secure Software Development, SAFECODE - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)
Secure Communication	TC-10. **Implement secure web interfaces**	Any web interface of IoT components must implement technical measures to reduce the exposure of management interfaces and detect potential unauthorised accesses, making the web interfaces hard to use for an attacker. This measure prevents the access to sensitive functionalities and data (e.g. source code).	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - GSMA IoT Security Assessment Checklist - Reference CLP12_5

Secure Communication	TC-11. **Implement secure session management**	For all sessions that take place in IoT, it is essential to ensure that active sessions are unique and cannot be shared or guessed, and that they are timed out and invalidated when no longer necessary. Session tokens should be unique for each session, guaranteeing a minimum level of entropy. They must never be disclosed in URLs or error messages. Cookie-based sessions must have the 'Secure', 'SameSite', and 'HttpOnly' attributes enabled.	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- Application Security Verification Standard 4.0 (ASVS). OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO - Proactive Controls for developers v3.0. OWASP - GSMA IoT Security Assessment Checklist - Reference CLP12_5 and CLP13_6
Secure Code	TC-12. **Implement secure coding practices**	In the SDLC process, secure coding practices must be implemented during different phases, including at least: Proven strong authentication mechanism to access the software (e.g. two-factor authentication, minimum password length, secure transfer, secure connection, secure credential management, etc.). Handling all errors and anomalous conditions that can compromise of sensitive information about the application Parameterisation of queries by binding the variables in the corresponding languages to prevent code injections in the query language, and Validation of input and output for forms' submissions such as with respect to language, characters, etc. (e.g. whitelisting mechanisms). These should be addressed in the SDLC to ensure the design, implementation and testing take this into account.	UNINTENTIONAL DAMAGES (Accidental) PERSONNEL PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION DEPLOYMENT AND INTEGRATION TESTING AND ACCEPTANCE	- The BSA Framework for Secure Software. BSA Software Alliance - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - Fundamental Practices for Secure Software Development, SAFECODE - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1

					- ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - "Systems and software engineering -Software life cycle processes. ISO 12207" - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - GSMA IoT Security Assessment Checklist - Reference CLP13_6
Secure Code	TC-13. **Provide audit capability**	Your SDLC model must ensure that the software under development (and IoT systems) include non-repudiation features (design, implementation, testing, etc.). High-value functionalities must be tracked to control critical aspects of the software. This could be mandatory, or highly advisable for regulatory compliance.	UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL NEFARIOUS ACTIVITY / ABUSE	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - The BSA Framework for Secure Software. BSA Software Alliance - Application Security Verification Standard 4.0 (ASVS). OWASP - Fundamental Practices for Secure Software Development, SAFECODE - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - GSMA IoT Security Assessment Checklist - Reference CLP11_7
Secure Code	TC-14. **Follow the principles of security by design and by default**	Many decisions are made during the design phase, when the final functionality of the solution is devised, including access verifications. These	PERSONNEL UNINTENTIONAL DAMAGES	DESIGN DEVELOPMENT/IMPLEMENTATION	- STRATEGIC PRINCIPLES FOR SECURING THE INTERNET OF

		decisions apply to the entire scope of the SDLC, implemented in the implementation/development phase, and tested before and after the production environment. The fail-safe principle must be taken into account to prepare the device for errors, anticipate potential disruptions of the service, and respond appropriately to ensure recovery. The principle of least privilege must also be observed to prevent unnecessary or unauthorised accesses. This set of measures is aimed at safeguarding data from being compromised. Implement strong user authentication by enforcing the change of passwords upon first use, and the periodic renewal of passwords (e.g. at least once in 90 days to every 6 months) and session / time lockout upon multiple failed authentication attempts (password, or other).	(Accidental) FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	TESTING AND ACCEPTANCE	THINGS (IoT). U.S. Department of Homeland Security - Application Security Verification Standard 4.0 (ASVS). OWASP - Security Design Principles. OWASP - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - ISO-IEC 27001. ISO - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1
Secure Code	TC-15. **Implement software development techniques**	Use development techniques that make application architecture more flexible. Modular architectures provide great benefits, not only during the operation to speed up updates or identify and troubleshoot, but during development. Developing large and indivisible blocks implies having a large team and making it difficult to define the scope. However, using techniques such as micro-services, a large block can be broken down into several to make the development agile, increase flexibility and scalability, facilitate the definition of scopes and functionalities, and decrease errors.	LEGAL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. - Application Security Verification Standard 4.0 (ASVS). - Proactive Controls for developers v3.0. - The BSA Framework for Secure Software. - BUILDING SECURITY IN MATURITY MODEL (BSIMM). - Security Assurance in the SDLC for the Internet of Things - ISACA. - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF).

					- SAFECode_Fundamental_Practices_forSDLC. - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. - ISO-IEC 27001. - CSA IoT Security Controls Framework.
Secure Code	TC-16. **Verify production code**	Ensure that production code comes with secure compiler options (compiled with security flags) and does not contain forgotten debug code or debug symbols. At production environment, security is crucial and it must be carefully controlled by ensuring not only the integrity of the tools but person competence conducting these activities.	PERSONNEL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- Application Security Verification Standard 4.0 (ASVS). OWASP. - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA). - ISO-IEC 27001. ISO. - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST. - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM. - Application Security Verification Standard 4.0 (ASVS). OWASP.
Security Code	TC-17. **Ensure security for patches and updates**	Patches must be carefully managed and deployed to prevent additional issues with update capabilities. It is necessary to ensure that all IoT elements can be updated and patched, and developers enable notifications of updates and security patches so that users can receive them for having information if, when and how patch software. The installation of security patches and updates should be user-friendly (e.g. automatic or in a few clicks).	LEGAL NEFARIOUS ACTIVITY / ABUSE	MAINTENANCE AND DISPOSAL	- Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NISTIR 8200 - Interagency Report on the Status of International

		Update mechanisms include secure/encrypted delivery of updates, validation of signatures on the device before installing the patch (secure boot), etc. Secure over-the-air updates should be considered through a secure mechanism that is cryptographically signed. This must be considered for all IoT systems, as well as for the software under construction already in production (patching as soon as possible for critical vulnerabilities). This measure prevents CVEs exploited by threat agents, and potential legal consequences may arise if due diligence is not in place to keep the systems in a well-fit state.			Cybersecurity Standardization for the Internet of Things (IoT). NIST - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - The BSA Framework for Secure Software. BSA Software Alliance - ISO-IEC 27001. ISO - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - GSMA IoT Security Assessment Checklist - Reference CLP12_6
Secure Code	TC-18. **Implement measures against rogue code and fraud detection**	Ensure malicious code is adequately managed (perform manual reviews, protect the code repository against tampering, etc.) in your SDLC model. Validate the application source code and third-party libraries (e.g. lack of backdoors, time bombs), and that the application does not grant unnecessary permissions. This measure includes the review of all changes before the deployment of the change.	PERSONNEL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

					- BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Application Security Verification Standard 4.0 (ASVS). OWASP
Secure Code	TC-19. **Implement anti-tampering features**	There must be logical tamperproof measures in IoT systems, that is, measures to monitor and ensure that the most critical assets (e.g. code) have not been tampered with (e.g. code-signing). Tampering could ease the access to sensitive functionalities or data for threat agents, and allow the insertion of rogue code in the software under construction.	UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - Application Security Verification Standard 4.0 (ASVS) . OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - GSMA IoT Security Assessment Checklist - Reference CLP13_6
Security Reviews	TC-20. **Apply secure code review**	Ensure that your SDLC model includes source code reviews. Code reviews can be manual or automated. Good practices recommend performing it manually for each candidate release (i.e. a member of the development team reviews what another team member has developed to ensure quality and share knowledge about the	PERSONNEL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Software Assurance Maturity Model (SAMM). OWASP SAMM

		development with the team). This is the only tool available to detect malicious code. Automated code reviews are commonplace and more cost-effective compared to manual ones.			- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Fundamental Practices for Secure Software Development, SAFECODE - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance
Security Reviews	TC-21. **Perform an attack surface analysis**	Carry out this activity during the design phase to detect any potential threats resulting from weaknesses. Ensure that your SDLC model includes this activity to provide value in other phases. It ensures the control of what is susceptible to be misused in the software under development, as well as of potential entry points. It helps to avoid unauthorised activities and data leakages.	NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DESIGN	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM). OWASP SAMM - Proactive Controls for developers v3.0. OWASP - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

					- The BSA Framework for Secure Software. BSA Software Alliance - GSMA IoT Security Assessment Checklist - Reference CLP11_5
Security Reviews	TC-22. **Perform IoT SDLC tests**	Ensure that your SDLC model makes software undergo testing prior to production to ensure it has no vulnerabilities before deployment. This can be done by means of an audit, and it should be performed at least, annually (for software under construction and IoT systems) or for each candidate release.	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Fundamental Practices for Secure Software Development, SAFECODE - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - GSMA IoT Security Assessment Checklist - Reference CLP13_7

Security Reviews	TC-23. **Design a contingency plan**	Take into consideration contingency plans designed to be integrated into the SDLC. Some activities of the contingency plan, such as the development of contingency planning policy and completion of the business impact analysis, must be executed in the initial phase of the SDLC. However, all the activities of the contingency plan are involved in all the SDLC phases but the last one, since once the system is operational, the contingency planning becomes a core part of continuous supervision and other ongoing security management tasks.	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- ISO-IEC 27001. ISO - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST
Security Reviews	TC-24. **Monitor requirements to ensure the SDLC success**	Implement a system to monitor the requirements agreed by contracts. During the SDLC, a partial or full breach of compliance with a requirement is a critical aspect. It would entail an increase in the project vulnerabilities and might even lead the project to fail. It is essential to perform a correct follow-up of the level of compliance reached by the requirements. To this end, key compliance indicators can be used (regarding quality, result required, scope, etc.) by means of a requirement matrix.	OUTAGES UNINTENTIONAL DAMAGES (Accidental) LEGAL FAILURES / MALFUNCTIONS	REQUIREMENTS	- MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Software Assurance Maturity Model (SAMM 1.5v). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - The BSA Framework for Secure Software. BSA Software Alliance
Security of SDLC Infrastructure	TC-25. **Ensure secure Logging and Monitoring Implementation**	The software under construction and the IoT systems have to generate high-quality logs, preventing the inclusion of sensitive information. Logs have to be monitored (if possible, in real time using automatic systems) and, reviewed and analysed by security staff. There are logging services that send the logs to a remote location instead of storing them locally so that, if the software is compromised, the data are not compromised.	PERSONNEL PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Proactive Controls for developers v3.0. OWASP - Application Security Verification Standard 4.0 (ASVS). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA)

					- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - Fundamental Practices for Secure Software Development, SAFECODE - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - The BSA Framework for Secure Software. BSA Software Alliance - GSMA IoT Security Assessment Checklist - Reference CLP12_5 and CLP13_6
Security of SDLC Infrastructure	TC-26. **Implement physical detection systems**	Deploy detection systems to control the critical physical environment (workplace, server rooms, etc.) where the SDLC infrastructure supports as temperature control, fire/smoke detection, alimentation loss, etc.) in order to avoid the loss of essential support for the SDLC such as organisation network, external communication,	PERSONNEL OUTAGES PHYSICAL ATTACK FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE	REQUIREMENTS	- The BSA Framework for Secure Software. BSA Software Alliance - Fundamental Practices for Secure Software Development, SAFECODE - NIST SP 800 53r5: Security and Privacy Controls for Federal

		external services as cloud, internet, surveillance, etc. Deploy backup systems for critical points.	DAMAGE / LOSS		Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Security of SDLC Infrastructure	TC-27. **Define a mitigation plan for physical damages**	Implement a procedure describing the steps to be taken in order to mitigate the damages that could be caused to the systems where data are stored during the SDLC process (communication systems, network equipment, servers, disks, data repositories, computers, etc.), as well as the spaces where they are hosted, to prevent them from being compromised due to a fire, flood, electric shock, etc. It is also important to have a redundant system in place to provide support and prevent alterations in the SDLC process.	OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Software Assurance Maturity Model (SAMM). OWASP SAMM - The BSA Framework for Secure Software. BSA Software Alliance - "CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - International Organization for Standardization (ISO) ISO27001 6 Planning" - "Systems and software engineering —Software life cycle processes. ISO 12207" - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Security of SDLC Infrastructure	TC-28. **Use whitelists for allowed applications**	Whitelist-based monitoring makes it possible to strengthen the security of connections and servers by controlling the applications. Only authorised applications can be run, thus preventing the execution of unauthorised software or malware. Whitelists must be periodically updated in order to include the latest applications, software has to be patched and tested to verify their functionality, etc.	OUTAGES UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION	- CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

				MAINTENANCE AND DISPOSAL	- SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP
Security of SDLC Infrastructure	TC-29. **Audit the access to the SDLC infrastructure**	Collect security logs to audit access to the SDLC resources, such as access to information in servers, files, data stored in physical rooms, etc. Regardless of whether the accesses are physical or logical, they have to be analysed with security tools (e.g. SIEM) to register the events (access to information, downloads, modifications, erasure attempts, etc.), identify users, and monitor the correct functioning of the process in order to generate alarms if security is compromised. These logs must be stored in a safe location and erased once the period of time stipulated by the industry elapses (e.g. erasure of financial data after 5 years).	PERSONNEL OUTAGES UNINTENTIONAL DAMAGES (Accidental) PHYSICAL ATTACK LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - The BSA Framework for Secure Software. BSA Software Alliance
Security of SDLC Infrastructure	TC-30. **Implement an identification protocol in your facilities**	Disseminate among internal and external employees of the organisation a policy on how to adequately identify themselves in the facilities, and on how to act and where to go if they detect unauthorised individuals attempting to access the facilities of the organisation for malicious purposes such as sabotage, industrial espionage, or the theft of confidential information.	PERSONNEL OUTAGES PHYSICAL ATTACK LEGAL NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS	- "Information technology — Security techniques — Information security management systems — Requirements. ISO 27001" - "Systems and software engineering —Software life cycle processes. ISO 12207" - Recommended Security Controls for Federal Information Systems and Organizations. NIST 800-53 - Software Assurance Maturity Model (SAMM). OWASP SAMM - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1

Secure Implementation	TC-31. **Enforce the change of default settings**	Security does not end once the software is produced. During the operation it is necessary to enforce the end users to safely utilise the application. Therefore, mechanisms must be established during the SDLC process to ensure it, namely: not allowing operation with password and user by default, ensuring that passwords have a minimum level of security (length, characters, etc.), including functions to manage user passwords (e.g. enforcing change cycles every 90 days, etc.), closing the user session after an inactivity time, locking the access out after multiple authentication fails, enable user notifications of updates, etc.	PERSONNEL LEGAL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - "Systems and software engineering — - Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST

					- ISO-IEC 27001. ISO
Secure Implementation	TC-32. **Use substantiated underlying components**	Choose well-supported underlying components that do not require customizations that may lead to losing security oversight and use proven tools to apply security hardening practices (e.g. metasploit).	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE	- "Systems and software engineering — - Software life cycle processes. ISO 12207" - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Proactive Controls for developers v3.0. OWASP - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - Software Assurance Maturity Model (SAMM 1.5v). OWASP - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - The BSA Framework for Secure Software. BSA Software Alliance - Security for industrial automation and control systems. Part 4-1:

					Secure product development lifecycle requirements. IEC 62443-4-1 - ISO-IEC 27001. ISO - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST
Secure Implementation	TC-33. **Provide secure configuration options for end users**	Ensure that the SDLC process addresses the provision of adequate measures in order to include different setting options for end-users upon first usage of an IoT solution to enable a continuous improvement of security, such as, for instance, the ability to disable features or functionalities that are not going to be used or to add automatic security check mechanism.	PERSONNEL LEGAL UNINTENTIONAL DAMAGES (Accidental) FAILURES / MALFUNCTIONS DAMAGE / LOSS	DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- The BSA Framework for Secure Software. BSA Software Alliance - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - Application Security Verification Standard 4.0 (ASVS). OWASP - Proactive Controls for developers v3.0. OWASP - Security Assurance in the SDLC for the Internet of Things - ISACA. ISACA - Fundamental Practices for Secure Software Development, SAFECODE - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - NIST SP 800 53r5: Security and Privacy Controls for Federal Information Systems and Organizations. NIST - ISO-IEC 27001. ISO

	Secure Implementation	TC-34. **Implement interoperability open standards**	One of the key problems in the IoT world is the lack of standardization. This fact causes that the interconnectivity between different devices is not easy and autonomous, which makes their integration difficult. Implement technologies based on open standards (e.g. OCF, oneM2M, etc.) to ensure that communication and integration between different devices is secure and reliable.	PERSONNEL UNINTENTIONAL DAMAGES OUTAGES LEGAL FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE DAMAGE / LOSS	REQUIREMENTS DESIGN DEVELOPMENT/IMPLEMENTATION TESTING AND ACCEPTANCE DEPLOYMENT AND INTEGRATION MAINTENANCE AND DISPOSAL	- Security for industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements. IEC 62443-4-1 - Software Assurance Maturity Model (SAMM 1.5v). OWASP - MITIGATING THE RISK OF SOFTWARE VULNERABILITIES BY ADOPTING A SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF). NIST - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - The BSA Framework for Secure Software. BSA Software Alliance - Proactive Controls for developers v3.0. OWASP - "Systems and software engineering — - Software life cycle processes. ISO 12207" - SECURE CODING BEST PRACTICES HANDBOOK. VERACODE - Fundamental Practices for Secure Software Development, SAFECODE - BUILDING SECURITY IN MATURITY MODEL (BSIMM). BSIMM - CSA IoT Security Controls Framework. Cloud Security Alliance (CSA) - Application Security Verification Standard 4.0 (ASVS). OWASP - NIST SP 800 53r5: Security and Privacy Controls for Federal

					Information Systems and Organizations. NIST - ISO-IEC 27001. ISO
Secure Implementation	TC-35. **Enable devices to advertise their access and network functionality**	By enabling devices to advertise their intended and supported functionality, the threat surface can be significantly reduced. An indicative practical example involves the use of IETF RFC 8520 on Manufacturer Usage Description Specification.	FAILURES / MALFUNCTIONS NEFARIOUS ACTIVITY / ABUSE	DESIGN DEVELOPMENT/IMPLEMENTATION	- RFC 8520, Manufacturer Usage Description Specification

A ANNEX: SDLC STANDARDS AND BEST PRACTICES

The following table summarises the main standards and best practice guides used during the development of the security measures previously introduced. These standards and guides bring together the security considerations to take into account through the entire SDLC process and they are analysed in this study as a baseline for the proposed security measures.

Publisher	Title	Reference
ISO (International Organization for Standardization)	ISO 12207:2008 Systems and Software Engineering – Software Life Cycle Processes	https://www.iso.org/standard/43447.html
	ISO 30141: Internet of Things - Reference Architecture	https://www.iso.org/standard/65695.html
	ISO 27001:2013: Information security management	https://www.iso.org/standard/54534.html
IEC/ISA (International Electrotechnical Commission/Standard for Automation)	IEC 62443-4-1: Security for Industrial automation and control systems. Part 4-1: Secure product development lifecycle requirements	https://webstore.iec.ch/publication/63337
OWASP (Open Web Application Security Project)	IoT Security Guidance	https://www.owasp.org/index.php/IoT_Security_Guidance#Developer_IoT_Security_Guidance
	Application Security Verification Standard (ASVS) 4.0	https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project#tab=Downloads
	Security Champions Playbook	https://www.owasp.org/index.php/Security_Champions_Playbook
	Security by Design principles	https://www.owasp.org/index.php/Security_by_Design_Principles

	Software Assurance Maturity Model (SAMM 1.5v)	https://www.owasp.org/index.php/OWASP_SAMM_Project#tab=Main
	CLASP Concepts	https://www.owasp.org/index.php/CLASP_Concepts
	Internet of Things Top 10	https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10
	API Security Top 10 2019	https://www.owasp.org/index.php/OWASP_API_Security_Project
	Proactive Controls for developers v3.0	https://www.owasp.org/index.php/OWASP_Proactive_Controls
	OWASP Dependency Check	https://www.owasp.org/index.php/OWASP_Dependency_Check
BSA (BSA Software Alliance)	The BSA Framework for Secure Software	https://ww2.bsa.org/~/media/Files/Policy/BSA_2019SoftwareSecurityFramework.pdf
CSA (Cloud Security Alliance)	IoT Security Controls Framework	https://cloudsecurityalliance.org/artifacts/iot-security-controls-framework
ETSI	ETSI TS 103 645 Cyber Security for Consumer Internet of Things	https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf
NIST (National Institutes of Standards and Technologies)	NIST Cloud Computing Standards Roadmap	https://www.nist.gov/publications/nist-cloud-computing-standards-roadmap
	Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework	https://csrc.nist.gov/publications/detail/white-paper/2019/06/11/mitigating-risk-of-software-vulnerabilities-with-ssdf/draft
IoT Central	Security-First Design for IoT Devices	https://www.iotcentral.io/blog/security-first-design-for-iot-devices
ISACA	Security Assurance in the SDLC for the Internet of Things	https://www.isaca.org/Journal/archives/2017/Volume-3/Documents/Security-Assurance-in-the-SDLC-for-the-Internet-of-Things_joa_Eng_0517.pdf
	Implementing Segregation of Duties	https://www.isaca.org/Journal/archives/2016/volume-3/Pages/implementing-segregation-of-duties.aspx

DHS (U.S. Department of Homeland Security)	Strategic Principles for Security the Internet of Things	https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf
OMG - CSCC (Object Management Group - Cloud Standards Customer Council)	Cloud Customer Architecture for IoT	https://www.omg.org/cloud/deliverables/CSCC-Cloud-Customer-Architecture-for-IoT.pdf
IIC (Industrial Internet Consortium)	IoT Security Maturity Model	https://www.iiconsortium.org/pdf/SMM_Description_and_Intended_Use_FINAL_Updated_V1.1.pdf
USENIX (Advanced Computing Systems Association)	Security in the Software Development Lifecycle	https://www.usenix.org/system/files/conference/soups2018/soups2018-assal.pdf
ONEM2M (Standards for M2M and the Internet of Things)	Security in the Software Development Lifecycle	http://onem2m.org/cache/mod_roksprocet/1fdfd821aa
CISA (US-CERT)	Secure Software Development Life Cycle Processes	https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes
IEEE (Institute of Electrical and Electronics Engineers)	IoT Security Principles and Best Practices	https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_of_things_feb2017.pdf
IBM (International Business Machines)	IBM Point of view: Internet of Things Security	https://www.ibm.com/downloads/cas/7DGG9VBO
TRENDMICRO	IoT Security Whitepaper	https://www.trendmicro.com/us/iot-security/content/main/document/IoT%20Security%20Whitepaper.pdf
SAFECode	Fundamental Practices for Secure Software Development	https://safecode.org/wp-content/uploads/2018/03/SAFECode_Fundamental_Practices_for_Secure_Software_Development_March_2018.pdf

	SAFECode Comments on EU Cybersecurity Legislation	https://safecode.org/wp-content/uploads/2018/11/SAFECode_Comments_on_EU_Cybersecurity_Legislation_Oct_2018_v2.pdf
	Tactical Threat Modeling	https://safecode.org/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf
	The Software Supply Chain Integrity Framework	http://safecode.org/publication/SAFECode_Supply_Chain0709.pdf
	Managing security risks inherent in the use of third-party components	https://safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf
	Software Security Takes a Champion	http://safecode.org/wp-content/uploads/2019/02/Security-Champions-2019-.pdf
Australian Information Security Management Conference	Source Code Embedded (SCEM) Security	https://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1134&context=ism
BSIMM (Building Security In Maturity Model)	BSIMM9	https://www.bsimm.com/content/dam/bsimm/reports/bsimm9.pdf
DELLEMC	IoT Security: Challenges, solutions and future prospects	https://education.emc.com/content/dam/dell-emc/documents/en-us/2018KS_Gloukhovtsev-IoT_Security_Challenges_Solutions_and_Future_Prospects.pdf
IoTAC (IoT Acceleration Consortium Ministry of Internal Affairs and Communications)	IoT Security Guidelines	http://www.iotac.jp/wp-content/uploads/2016/01/IoT-Security-Guidelines_ver.1.0.pdf
MSRUAS - SASTech Journal	Analysis of SDLC Models for Embedded Systems	https://www.researchgate.net/publication/322138583_Analysis_of_SDLC_Models_for_Embedded_Systems
GRAMMATECH	A Four-Step Guide to Security Assurance for IoT Devices	http://codesonar.grammatech.com/a-four-step-guide-to-security-assurance-for-iot-devices
NCC Group (National Computing Centre)	An Implementers' Guide to Cyber-Security for Internet of Things Devices and Beyond	https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2014/april/security-of-things-an-implementers-guide-to-cyber-security-for-internet-of-things-devices-and-beyond/

BOSCH	Holistic IoT Security	https://www.bosch-si.com/iot-platform/insights/downloads/iot-security.html
GSMA	Security Guidelines for Product Categories – IoT GW	https://www.ccds.or.jp/english/contents/CCDS%20Security%20Guidelines%20for%20Product%20Categories%20IoT-GW_v2.0_eng.pdf
	IoT Security Guidelines and Assessment	https://www.gsma.com/iot/iot-security/iot-security-guidelines/
CISCO	Secure Development Lifecycle	https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-secure-development-lifecycle.pdf
DZONE	The DZONE Guide to Application Security 2015 Edition	https://dzone.com/guides/application-security-2015-edition
FCC TAC (Federal Communications Community-Technological Advisory Council)	Technical Considerations White Paper	https://transition.fcc.gov/oet/tac/tacdocs/reports/2015/FCC-TAC-Cyber-IoT-White-Paper-Rel1.1-2015.pdf
Secure Software Foundation	Framework Secure Software	https://www.securesoftwarealliance.org/FrameworkSecureSoftware_v1.pdf
PCI Security Standards Council	Payment Card Industry Software Security Framework	https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf?agreement=true&time=1566543843174
IJETMAS	Security Secure Software Development Life Cycle	http://www.ijetmas.com/admin/resources/project/paper/f201509231443005088.pdf
	Threat Modeling for Secure Embedded Software	https://pdfs.semanticscholar.org/d3a8/8f79f3baf7c1f3ad75fada8ec2b71b27ca99.pdf
IPA	IoT Safety/Security Design Tutorial	https://www.ipa.go.jp/files/000053921.pdf
VDOO	Integrating Security into the IoT SDLC	https://www.vdoo.com/blog/integrating-security-into-the-iot-sdlc/

ERNW	Security Bsides Ljubljana IoT and SDLC	https://0x7df.bsidesljubljana.si/wp-content/uploads/sites/9/ERNW_Scheuring_Security_BSides_Ljubljana_IoT_and_SDLC_2015.pdf
Advanced Science and Technology Letters	A Study of Developing Security Requirements for Internet of Things	https://pdfs.semanticscholar.org/6aec/74231f1716bd350b1c60b2bc3168471e1c13.pdf
WIND RIVER	Managing the IoT Lifecycle from Design through End-of-Life	https://www.windriver.com/whitepapers/managing-iot-lifecycle/2434-Cloud_white_paper.pdf
DCMS-UK (Department for Digital, Culture, Media and Sport, UK Government)	Code of Practice for Consumer IoT Security	https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf
SECURA	Source Code Analysis	https://www.secura.com/pathtoimg.php?id=1213&image=source_code_analysis.pdf
	Security Testing Compliance for IoT v0.5	https://www.secura.com/pathtoimg.php?id=1199&image=security_testing___compliance_for_iot_v0.5.pdf
BISEC (International Conference on Business Information Security)	The Role of Software testing in a Security-Oriented IoT Software Development Process	https://www.metropolitan.ac.rs/files/2018/01/BISEC2017-Zbornik-ilovepdf-compressed.pdf
Zephyr	Continuous Testing Agility 2020	https://conference.eurostarsoftwaretesting.com/wp-content/uploads/ContinuousTestingAgility2020-1-1.pdf
ScienceDirect	Secure IoT Devices for the Maintenance of Machine Tools	https://reader.elsevier.com/reader/sd/pii/S2212827116309878?token=626E3AF21033F555197704E40AF514E48A61FEAA92C01B196F9CB9390B6B292E1C1119714454FAE438EAD6E14434EDF5
GitHub	Secure Software Development in the Financial Services Industry	https://resources.github.com/downloads/GitHub_eBook_FSI_Secure_Development.pdf
INTEL	Secure Solutions for the IoT	https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/developing-solutions-for-iot.pdf

SANS Institute	A Security Checklist for Web Application Design	https://www.sans.org/reading-room/whitepapers/securecode/security-checklist-web-application-design-1389
	Secure Coding. Practical steps to defend your web apps.	https://software-security.sans.org/resources/paper/cissp/application-security
CMU (Carnegie Mellon University)	Threat Modeling: A Summary of Available Methods	https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=524448
	Security Quality Requirements Engineering Technical Report	https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=7657
VERACODE	Secure Coding Best Practices	https://info.veracode.com/secure-coding-best-practices-hand-book-guide-resource.html
INDUSA	10 Challenges Every Software Product Developer Faces	http://www.indusa.com/articles/10-challenges-every-software-product-developer-faces/
LANDesk	Resolving the Top Three Patch Management Challenges	https://www.infosecurityeurope.com/__novadocuments/20559
IEEE	Proposed Embedded Security Framework for Internet of Things.	https://www.researchgate.net/profile/Jaydip_Sen/publication/252013823_Proposed_Embedded_Security_Framework_for_Internet_of_Things_IoT/links/00b495278c92a797d9000000/Proposed-Embedded-Security-Framework-for-Internet-of-Things-IoT.pdf
UL	UL 2900 series Standard for Software Cybersecurity for Network-Connectable Products	https://standardscatalog.ul.com/standards/en/standard_2900-1_1

B ANNEX: SECURITY IN SDLC MODELS

SDLC models are conceptual frameworks that are used to detail all activities relating to software development and the interrelations between these activities. This conceptualization allows for a structured, coordinated and well-communicated software development approach among all members of the team. Different SDLC models have been proposed over the years, including Waterfall, Agile, Spiral, DevOps, DevSecOps, etc.

In general, SDLC models fall under one of the following three categories: sequential, iterative or agile (this includes the agile methods). The difference lies in the transition from one SDLC phase to another. Reflecting the diversity and intricacies of the different SDLC models, security considerations are taken into account and are incorporated in different ways. There exist several possible categorizations of the SDLC phases. In the context of this study and focusing on IoT, six distinct phases are identified (further details may be found in Section 2.2), namely:

1. Requirements, (Definition and Identification).
2. Software design.
3. Development/implementation.
4. Testing and acceptance.
5. Deployment and integration.
6. Maintenance and disposal.

The Waterfall SDLC model represents the 6 phases in a sequential, linear flow. Accordingly, requirements are defined at the beginning of the process. While this process might be useful for rigid and highly structured projects, it might not be very useful when considering modern development cycles that require high degrees of adaptation, extensibility and flexibility (e.g., IoT, cloud-based, etc.). Conversely, the Agile SDLC model is meant to represent a more flexible means to define and capture new requirements. For this reason, it makes use of short, time-constrained development cycles that facilitate the adaptability of the final software product or service by means of validation of the outputs of these agile cycles. The Waterfall and Agile models are the foundations for other, more recently introduced models.[49]

[49] See https://www.theseus.fi/bitstream/handle/10024/135804/Williams_Paivi.pdf?sequence=1&isAllowed=y

When considering security, the Waterfall model plans for security at the beginning of the SDLC, but actual security tests are not carried out until the final phases (Development/Implementation or Integration and Testing). This implies that security is not an integral part of the development process and is addressed only when the final product is nearing completion. In the Agile model, security (and other types of) testing take place in the context of every cycle and not as a whole, similarly to the definition of security requirements. Moreover, it is normally the case that different teams are in charge of different cycles and therefore capturing of security requirements and testing might differ in-between cycles. In the context of agile models, the most critical aspect for security is to ensure the consistency of security requirements, design setup and testing in each iteration.

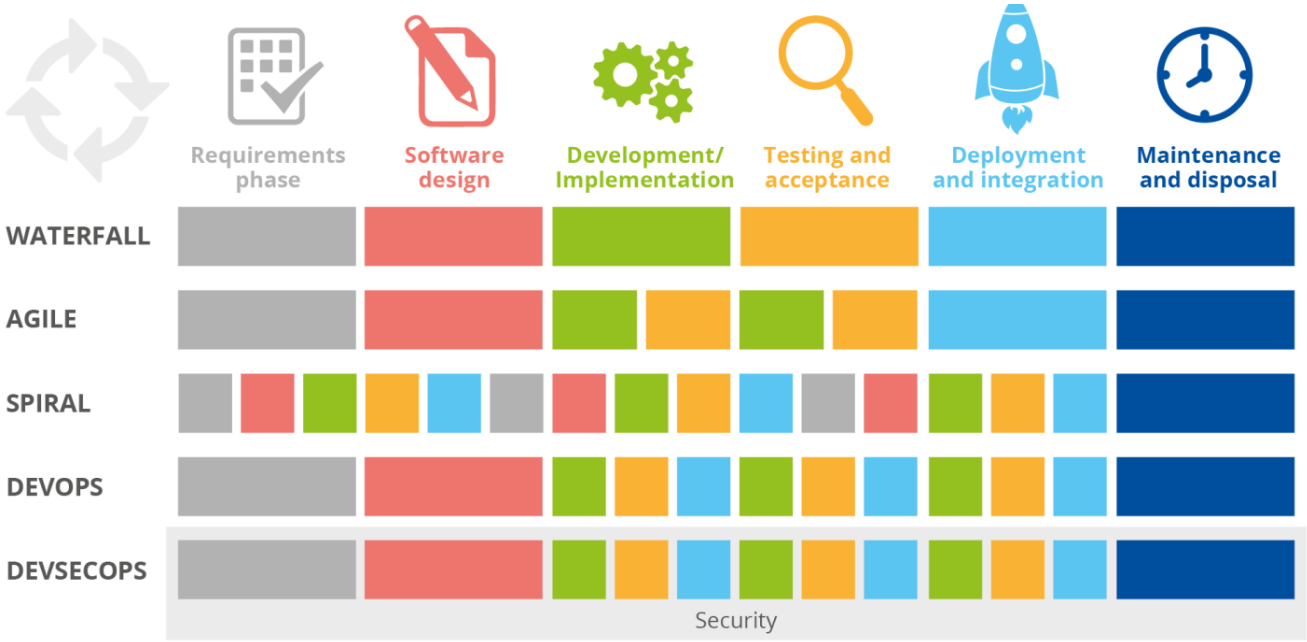
Building on the Waterfall model, more iterative SDLC models were proposed such as the Spiral one. Software engineers using the Spiral SDLC model collect a series of requirements at the beginning of the process, which are then checked at every stage of development. This allows the inclusion of additional requirements as necessary in every iteration or "spiral", so that by the time the application reaches the deployment and maintenance phases it has considered additional security requirements that were not planned for in the beginning.

Building on the Agile SDLC model, the DevOps model provides faster release cycles since deployment and integration are also a part of the cycle (in traditional Agile, it is only software design and development/implementation). DevOps was introduced to eliminate the barriers between development and operations, bringing together professionals from both teams and accordingly leads to integrated and more complete security tests. In addition, other well-known SDLC models based on Agile include XP (Extreme Programming), Scrum and Kanban. Of particular interest to secure SDLC is the DevSecOps SDLC model, which integrates security practices in the DevOps methodology. In this way, security is considered in all phases of software development in an agile manner.

In the context of IoT software development, the particularities of this dynamic and adaptive ecosystem should be taken into account when selecting the most appropriate SDLC model. Given the diversities of the different models and the way in which security is addressed in each one of them, it is evident that the decision on which SDLC model to adopt should be given considerable thought, since it will affect the overall outcome. Figure 10 depicts different SDLC models and how they manage the different SDLC phases[50].

[50] Figure 9 and discussion in Annex C are indicative when it comes to the listed SDLC models, referring to the most well-known ones. For a full coverage and description of available SDLC models the reader is referred to relevant, up-to-date software engineering textbooks.

Figure 9: Overview of SDLC models[51]



[51] Figure based on and adapted from https://analyze.co.za/the-transition-to-devops/

C ANNEX: IOT SDLC TESTING

Tests	Description
Dynamic Analysis Security Testing (DAST)	DAST allows to study the software when it runs, by implementing a package of prebuilt attacks (less limited and more automated than a human attacker) and aiming them against the executed software. DAST can find vulnerabilities when all of the components are integrated and if the test is successful, an attacker can carry out the attack. It is recommended to use DAST with SAST to obtain more complete test analysis.[58]This kind of test can be fully automatic using security scanners and/or vulnerability scanning tools, or complemented with a manual review to check the results and perform complex tests manually.
Static Analysis Security Testing (SAST)	SAST involves the use of tools and techniques to analyse all elements of software (including source code, bytecode and any used binaries). SAST tests software that is not currently executed and is thus complementary to DAST. The aim is to identify coding and design software aspects that might indicate the existence of possible known vulnerabilities.
Interactive Analysis Security Testing (IAST)	This type of security testing aims to be an evolution of DAST testing with the knowledge of the SAST testing (information flows), where the test checks the execution flow during runtime. It is usually automated and used in DevOps scenarios.
Software Composition Analysis (SCA)	Software dependencies are as important as the code itself, and vulnerabilities detected within them can have severe consequences. This is particularly the case with IoT software, which is commonly comprised of several third-party components and libraries. It is thus recommended as a good practice to check whether software dependencies can be trusted before deployment in a production environment. This provides more confidence in the software as a whole.
Validation (Acceptance) Testing	Validation acceptance testing is focused on realistic test scenarios, such as sequences of actions performed by the user (use cases). The outcomes of this type of testing yields whether the system can be securely used in real-world scenarios, Validation acceptance tests are the final and most critical test prior to deployment and integration, because it will determine whether the software is ready to roll out to the market. This type of testing aims at assessing whether the software does what it was envisaged to do at the beginning of the project, i.e. based on the security requirements.
Fuzzing test	Fuzzing generates (or mutates) large sets of data and passes it to an application's data parsers to check its reaction. This kind of testing technique can reveal potential security vulnerabilities and weaknesses and help to mitigate attacks such as ones based on buffer overflows and data input validation (e.g. cross-site scripting). In the case of IoT, complexity of fuzzing increases in accordance to the large number of available protocols and data formats, so this is a particularity that needs to be catered for by the testing team.
Security verification and validation	Software verification is the process used to determine whether the outcome of a given stage of product development (i.e. software development) conforms exactly to the requirements set at the beginning of the stage. Software validation is the process used to determine whether the product (computer program, operating system, appliance etc.) satisfies its intended use and user needs. Requirements, lifecycle processes and other supporting artifacts can also be validated for their conformance to the expected results. Careful consideration has to be given to IoT software,

	since the logic and functionality of the software depends on the context of use, therefore appropriate tests should be devised to ensure full coverage of the expected functionality.
Manual Code Review	Despite being a time-consuming activity, the manual review of code is one of the best options to detect potential security issues that might reside within the code. Automated tools can provide a solid basis of understanding, however employing experienced security experts to review the code may lead to the detection of more complex security issues (e.g. logic bombs) following the information flow. A slight variation of this testing technique is peer code review, where a different member of the team checks the code developed to ensure that best practices (quality and security) have indeed been followed.
Load Testing	This type of testing aims to check whether the software is able to operate in high load conditions (e.g. Internet rush hour), taking into account the constraints in terms of resources, which many IoT solutions face. This might provide interesting results in terms of availability of the systems. It is important to keep in mind to check for all flows in an IoT software solution and to load test all of them, including the ones referring to cloud and possible backend servers.
Stress Testing	Stress testing test is similar to load testing, but in this case the objective is to find the point where the tested software will fail, i.e. the maximum load that it is able to manage. This kind of test reveals the potential impact on availability of the software, however in the context of IoT it should be taken with a grain of salt, since due to limited resources and the need to have lightweight solutions the results will need to be seen under this perspective.
Regression Testing	Regression testing type is one of the most useful ones in IoT ecosystems, given that they are subject to frequent software updates to enhance their functionalities and security levels. Accordingly, regression is used to ensure that functionalities already deployed in the software or solution work after a modification. This way, not only new modules or functionalities are checked, but previous ones are also tested for compatibility or any other inadvertent changes in their behavior after a new type of functionality has been deployed. Regression testing is particularly common in DevOps scenarios, where software is continuously tested prior to its deployment.
Integration Testing	Integration testing is used in software testing cases where the solution is made up of different elements. In such cases, it is necessary to check whether the different elements can work together as expected. IoT software development by definition falls under this category, since it integrates components of end devices, communications/networks, cloud-based ones, etc. Accordingly, integration testing is significant in the context of IoT since it is meant to verify that no security issues will arise when all elements of the IoT solution are integrated.
Penetration Testing	Penetration testing is by definition related to security. Such tests consist of simulated attacks on the developed software solution in order to evaluate its security solution. The level of sophistication of the testing team influences the success of penetration tests, since the more sophisticated the attack, the more it is expected to uncover security issues. Pentesting IoT software has recently received attention with the proliferation of IoT end devices and the popularity of publicized attacks on them, as well as the publication of relevant automated tools[52].
Safety Testing	IoT is inherently linked to cyber-physical deployments and this implies the need to consider safety in tandem with cybersecurity, as emphasized in the ENISA studies on Smart Manufacturing[53] and Industry 4.0[54]. Safety testing is aimed at addressing safety aspects related to the developed software solutions and examine any potential adverse effects. All elements of the software that may impact the physical aspects of the IoT system or service should be examined during these tests, as well as any interdependencies of said elements.

[52] See https://www.darkreading.com/threat-intelligence/new-metasploit-extension-available-for-testing-iot-device-security/
[53] See https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot
[54] See https://www.enisa.europa.eu/publications/industry-4-0-cybersecurity-challenges-and-recommendations

TP-02-19-880-EN-N

ABOUT ENISA

The European Union Agency for Cybersecurity (ENISA) has been working to make Europe cyber secure since 2004. ENISA works with the EU, its member states, the private sector and Europe's citizens to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. Since 2019, it has been drawing up cybersecurity certification schemes. More information about ENISA and its work can be found at www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str
151 24 Marousi, Attiki, Greece

Heraklion office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu

ISBN 978-92-9204-316-2
DOI: 10.2824/742784