UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Lecture 4 – Legal and standard frameworks, Secure Design, Development and Coding

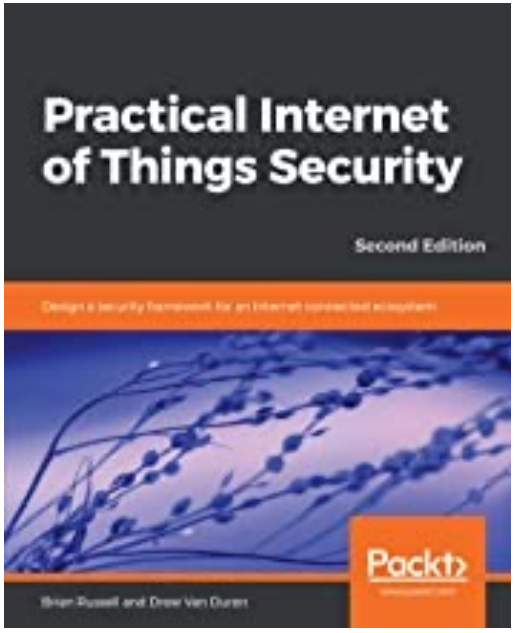
Prof. Esposito Christian

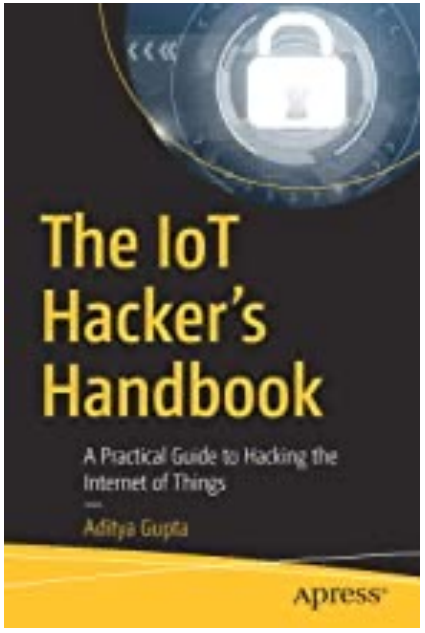
::.. Summary

- Standards and Legislation relevant to IoT Security
- Secure Design & Development
- IoT Product Development Process
- Secure Coding
- GDPR and IoT - analysis of the effects
- Privacy-preserving methods.

::.. References

- Brian Russell, Drew Van Duren, “Practical Internet of Things Security: Design a security framework for an Internet connect-ed ecosystem”, Packt Publishing; 2ª edizione - Novembre 2018;
- Aditya Gupta, “The IoT Hacker's Handbook: A Practical Guide to Hacking the Internet of Things”, Apress - Aprile 2019.





::.. Key Lectures (1/5)

- Neshenko, Nataliia, et al., "Demystifying IoT security: an exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations", IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2702-2733, thirdquarter 2019.
- Alaba, Fadele Ayotunde, et al., "Internet of Things security: A survey", Journal of Network and Computer Applications, vol. 88, pp. 10-28, Giugno 2017.
- Yang, Yuchen, et al., "A survey on security and privacy issues in Internet-of-Things", IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250-1258, Ottobre 2017.
- Mosenia, Arsalan, and Niraj K. Jha, "A comprehensive study of security of internet-of-things", IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 4, pp. 586-602, Ott.-Dic. 2016.

::.. Key Lectures (2/5)

- Lin, Jie, et al., "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications", IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125-1142, Ottobre 2017.
- Ammar, Mahmoud, Giovanni Russello, and Bruno Crispo, "Internet of Things: A survey on the security of IoT frameworks", Journal of Information Security and Applications, vol. 38, pp. 8-27, Febbraio 2018.
- Porambage, Pawani, et al., "The quest for privacy in the internet of things", IEEE Cloud Computing, vol. 3, no. 2, pp. 36-45, Mar.-Apr. 2016.
- Chiara, Pier Giorgio. "The IoT and the new EU cybersecurity regulatory landscape." International Review of Law, Computers & Technology 36.2 (2022): 118-137.

::.. Key Lectures (3/5)

- Stellios, Ioannis, et al. "A survey of Iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services", IEEE Communications Surveys & Tutorials, vol. 20, n. 4, pp. 3453-3495, 2018.
- Alnaeli, Saleh M., et al. "Vulnerable C/C++ code usage in IoT software systems", Proceedings of the IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016.
- Alnaeli, S., et al. "Source Code Vulnerabilities in IoT Software Systems», *Advances in Science, Technology and Engineering Systems Journal, vol.* 2, no. 3, pp. 1502-1507, 2017.
- Chris Jones, "Getting started with IoT security", report on line at https://www.iar.com/globalassets/security-solutions/getting-started-whitepaper.pdf

::.. Key Lectures (4/5)

- ENISA, "Good Practices for Security of IoT - Secure Software Development Lifecycle", report on line at https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot-1
- ENISA; "Baseline Security Recommendations for IoT", report on line at https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot
- ETSI, "Cyber Security for Consumer Internet of Things: Baseline Requirements", report on line at https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.00_30/en_303645v020100v.pdf

::.. Key Lectures (5/5)

- S. Wachter, "Normative challenges of identification in the Internet of Things: Privacy, profiling, discrimination, and the GDPR." Computer law & security review 34.3 (2018): 436-449.
- D. Bastos, F. Giubilo, M. Shackleton, F. El-Moussa, "GDPR Privacy Implications for the Internet of Things", In Proc. of the 4th Annual IoT Security Foundation Conference (2018, December).
- M. Seliem, K. Elgazzar, K. Khalil, "Towards privacy preserving IoT environments: a survey", Wireless Communications and Mobile Computing 2018 (2018).

IoT Security Standards and Legislation on IoT Security

::.. Introduction (1/2)

Why are rules and regulations important?

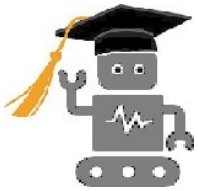
Implement a methodological approach for the assessment of risks in the various contexts, dimensions, complexity, value of the organization, presence of governance and monitoring standards of IT services, internal control systems, preventive evaluation of potential IT incidents and the effects .

Implement a level of control aligned with the degree of risk identified by means of a qualitative analysis.

Implement security and privacy compliance checklists to support DPOs who need to assess the compliance of these systems.

Support the organization in choosing reliable IoT products.

::.. Introduction (2/2)



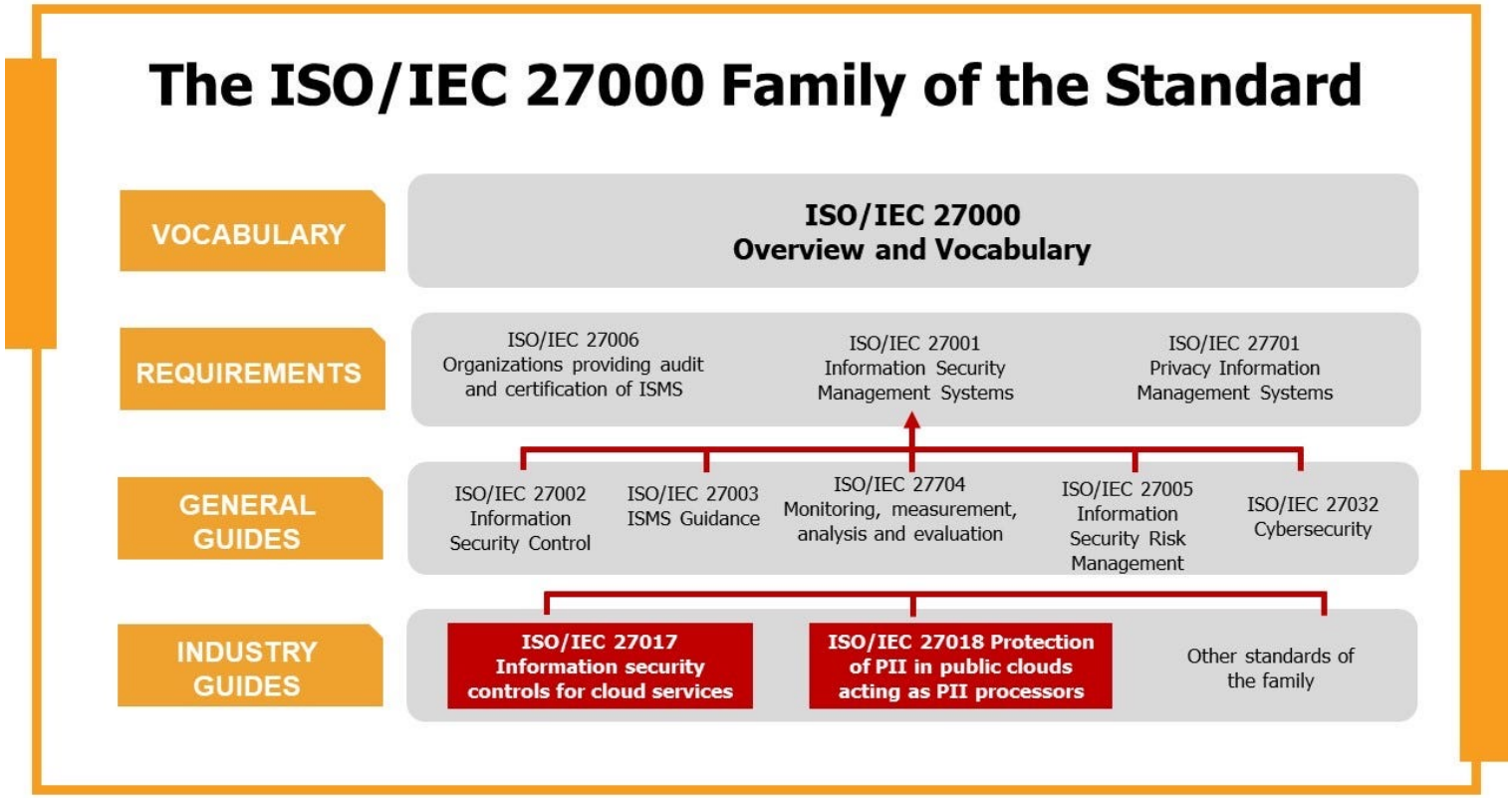
Acquire awareness of the impact of digital on economic activity: incentive to develop a culture of digitalization through training

+169% attacks in Italy in 2022 (Clusit Report 2023) compared to the previous year, affecting by Italian manufacturing companies, technical-scientific sectors and professional services

Adopt best practices: keep systems updated to ensure the software is always intact, use data storage systems and secure communications channels; minimization of attack surfaces, and preparation of response systems to possible attacks.

::.. ISO/IEC 27400:2022 (1/8)

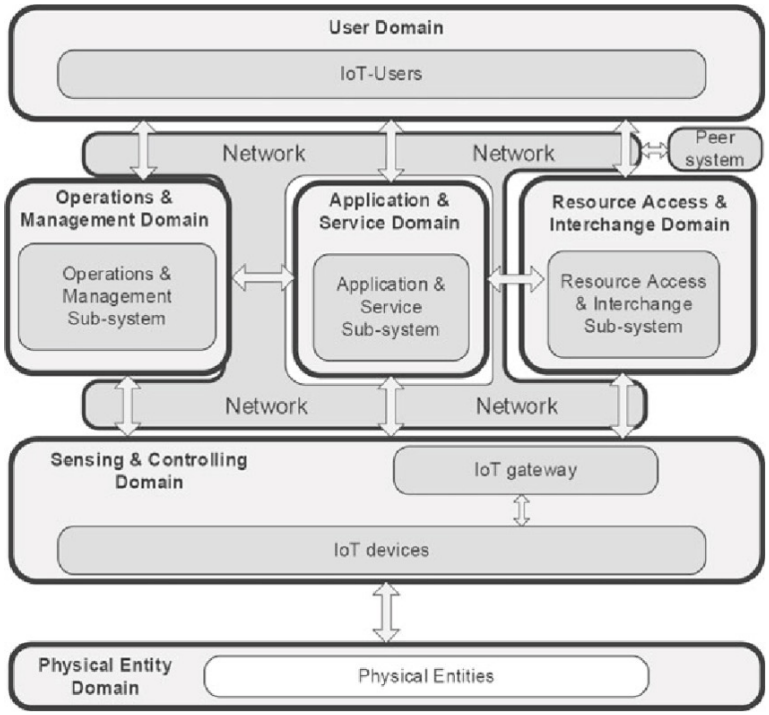
In June 2022, the first version of ISO/IEC 27400:2022 "Guidelines for the security and privacy of IoT devices" was released. This is a long-awaited guideline that integrates the increasingly complex system of ISO/IEC 27000 family standards on information security.



::.. ISO/IEC 27400:2022 (1/8)

In June 2022, the first version of ISO/IEC 27400:2022 "Guidelines for the security and privacy of IoT devices" was released. This is a long-awaited guideline that integrates the increasingly complex system of ISO/IEC 27000 family standards on information security.

The guideline explicitly refers, as regards the reference framework, to ISO/IEC 30141:2018 - Internet of Things (IoT) - Reference Architecture.

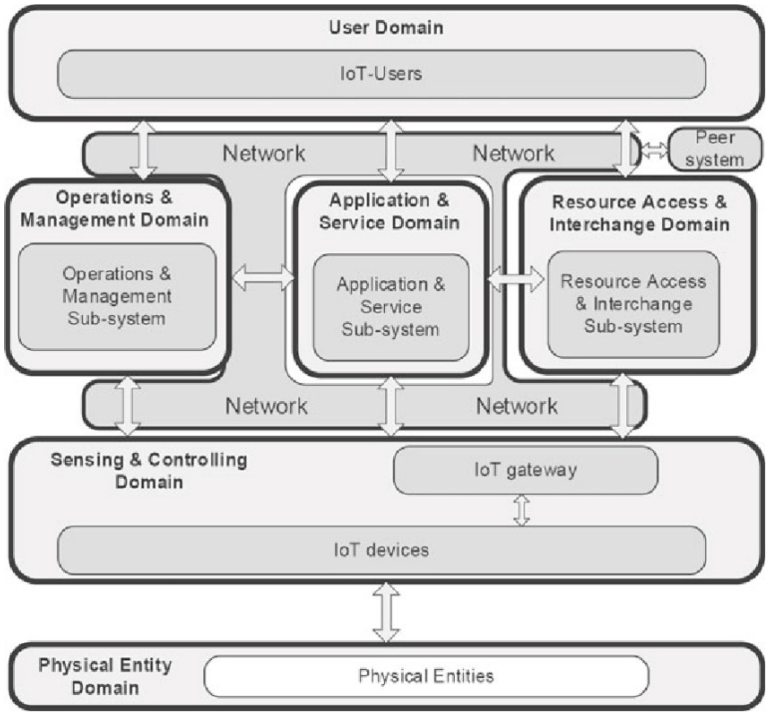


IoT is defined as an infrastructure of interconnected physical entities, systems and information resources together with the intelligent services which can process and react information of both the physical world and the virtual world and can influence activities in the physical world.

::.. ISO/IEC 27400:2022 (1/8)

In June 2022, the first version of ISO/IEC 27400:2022 "Guidelines for the security and privacy of IoT devices" was released. This is a long-awaited guideline that integrates the increasingly complex system of ISO/IEC 27000 family standards on information security.

The guideline explicitly refers, as regards the reference framework, to ISO/IEC 30141:2018 - Internet of Things (IoT) - Reference Architecture.



The IoT Reference Architecture (IoT RA) provides descriptions from different architectural views. IoT RA not only outlines “what” the overall structured approach for the construction of IoT systems by means of the architectural structure description, but also indicates “how” the architecture and its domains or entities will operate.

::.. ISO/IEC 27400:2022 (2/8)

An IoT device is defined by ISO/IEC 27400 as an “IoT system entity that interacts and communicates with the physical world through sensing or actuation”. IoT systems have, in whole or in part, the following characteristics:

- IoT systems include devices, attributable to hardware and software components that are used together or connected to physical media;
- IoT devices are connected to networks and have the ability to transmit and receive data; can use wired and wireless networks;
- IoT devices usually have sensing capabilities;
- IoT devices can have actuation capabilities;
- IoT systems include applications to process data from devices, to generate/send control data, and to be integrated with other systems;
- IoT systems include operational components that enable the configuration and operation of IoT devices and applications;
- IoT systems support human or digital users - insights into ISO/IEC 30141:2018.

::.. ISO/IEC 27400:2022 (3/8)

The main interested parties for which specific controls are defined that they must implement are:

- “IoT service provider” (ISP) - responsible for providing services that allow the IoT system to function;
- “IoT service developer” (ISD) - responsible for the development, design, implementation, production, configuration, assistance and integration of IoT services.
- “IoT user” (IU) - the user (including human and digital users) of an IoT system or service.

The “IoT Device Developer” (IDD) in turn can be considered as a sub-role compared to that of the Service Developer.

::.. ISO/IEC 27400:2022 (4/8)

The domain or components, at various levels, attributable to the IoT are:

- the User Domain (UD) includes digital and human users, at various levels, who configure, manage and use IoT devices and associated services;
- the Physical Entity Domain (PED) includes the physical entities;
- the Sensing and Controlling Domain (SCD) includes IoT devices and communication systems between devices and other components, such as equipment, sensors, actuators, cloud, etc. (gateway);
- the Operations and Management Domain (OMD) includes the operational and management support system; includes organizations that develop, produce, configure, and manage IoT systems, including providing support;
- the Resource Access and Interchange Domain (RAID) considers the resources that provide the systems through which external entities can access IoT systems, including the network infrastructure;
- the Application and Service Domain (ASD) includes the applications and services offered by IoT service providers.

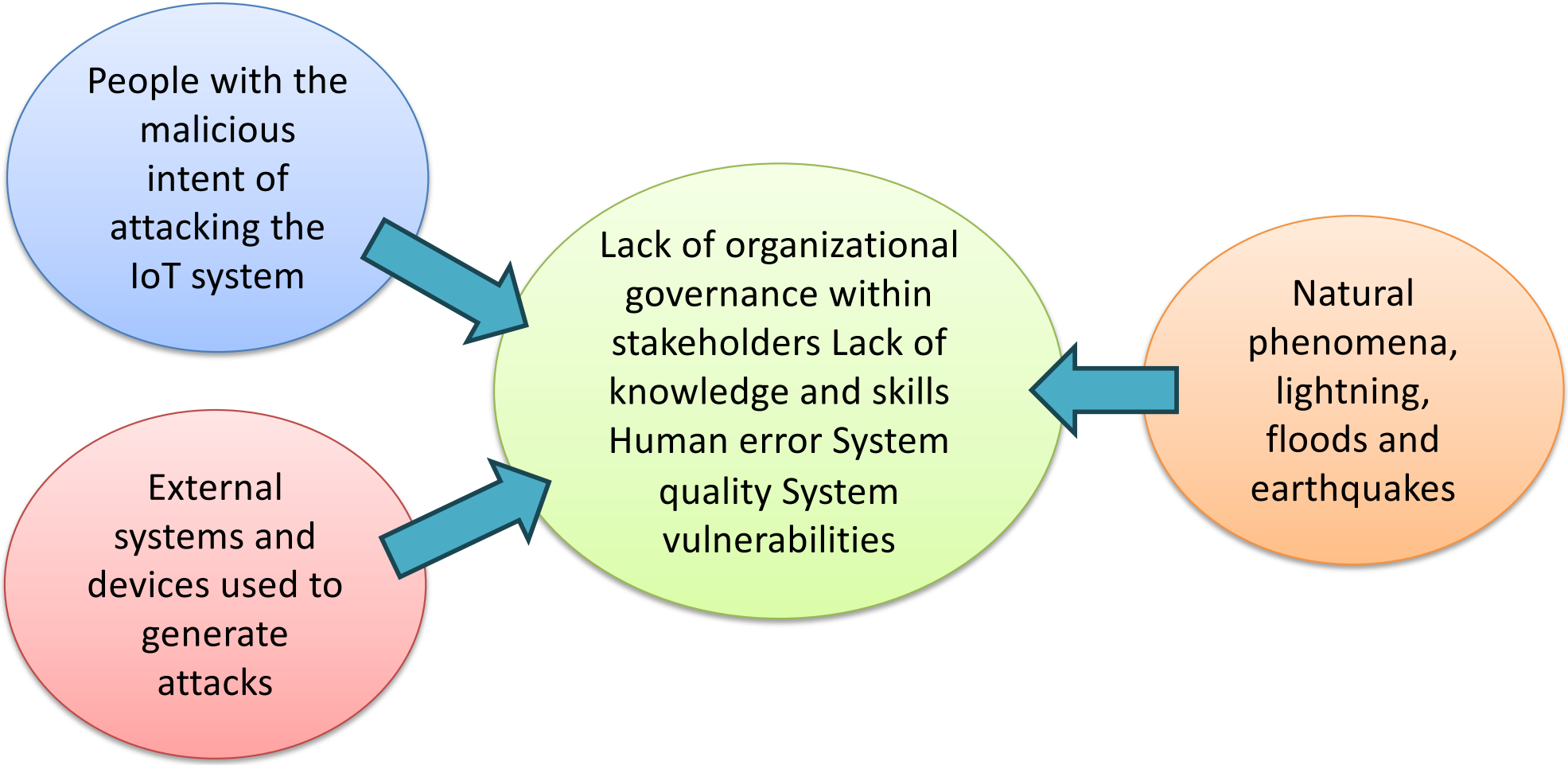
::.. ISO/IEC 27400:2022 (5/8)

The characteristic element of the guideline is the set of controls, both on the security and personal data protection aspects; the risks induced by these systems are different from those foreseen for more traditional devices and similarly are the controls that cover the entire life cycle of the systems. Although there is no specific and precise correlation between the sources of risk and the controls, the latter are identified in relation to the interested parties.

ISO/IEC 27400 identifies 45 security and privacy controls, for which the purpose, responsible stakeholders, IoT domain, and guidance for implementing solutions are defined similarly to other standards in the ISO/IEC 27000 family. With this setting, it is, therefore, possible to filter the controls that belong to a specific domain, or are the responsibility of an interested party.

::.. ISO/IEC 27400:2022 (6/8)

Sources of risk: any element that, alone or combined with others, has the intrinsic potential to generate a risk.



::.. ISO/IEC 27400:2022 (7/8)

Controls at an organizational level, to support company management, such as the definition of specific company policies, the definition of specific roles and responsibilities within the company

Controls applicable to all IT systems, such as monitoring the functioning of the system itself, the creation and protection of logs that track events and the acquisition of information deriving from any incidents

Specific controls of IoT systems. define the principles of secure engineering of IoT systems, design and implementation of security functions during all phases of the life cycle of IoT systems

::.. ISO/IEC 27400:2022 (8/8)

Use of adequate network and communication technologies, which satisfy security needs, as well as performance and other demands of IoT systems, such as mobility and localization

Provide device users with guidance on correct use, highlighting risks and side effects

Implementation of privacy controls - GDPR: Minimizing indirect data collection – Privacy by default

Recommendations for users: be careful in the initial settings of the IoT device, Deactivation of unused devices

::.. Threat Models (1/5)

STRIDE is a model for identifying computer security threats developed by Praerit Garg and Loren Kohnfelder at Microsoft.

	Threat	Property	Property description
IoT systems security objectives expressed as threats to counter	Spoofing	Authentication	The identity of IoT entities or IoT users is established (or you are willing to accept anonymous entities).
	Tampering	Integrity	IoT data and system resources are only changed in appropriate ways by appropriate people.
	Repudiation	Nonrepudiation	IoT users cannot perform an action and later deny performing it.
	Information disclosure	Confidentiality	Data is only available to the IoT users intended to access it.
	Denial of Service	Availability	IoT services are ready when needed and perform acceptably.
	Elevation of privilege	Authorization	IoT users are explicitly allowed or denied access to resources.

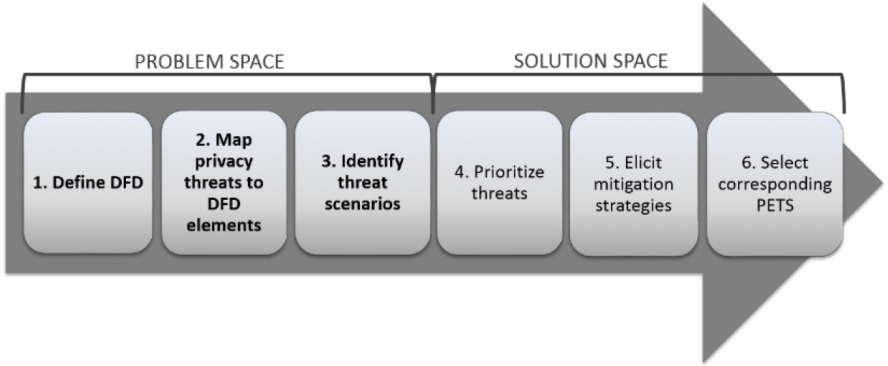
Note that the term IoT systems is according to ISO/IEC 30141 IoT reference architecture. It can refer to a subsystem or a platform.

::.. Threat Models (2/5)

LINDDUN is a privacy counterpart of STRIDE that integrates 7 main privacy threat categories.

	Threat	Property		Property description
IoT systems privacy objectives expressed as threats to counter	Linkability	Hard privacy	Unlinkability	Hiding the link between two or more actions, identities, and pieces of information associated with IoT entities or IoT users.
	Identifiability		Anonymity	Hiding the link between an IoT user or an IoT entity identity and an action or a piece of information
	Non-repudiation		Plausible deniability	Ability for an IoT end user or and IoT entity to deny having performed an action that other parties can neither confirm nor contradict
	Detectability		Undetectability and unobservability	Hiding the activities of an IoT user or an IoT entity.
	Disclosure of information	Security	Confidentiality	Ability of the IoT system to hide the data content or to control the release of data content
	Unawareness	Soft Privacy	Content awareness	IoT user's consciousness regarding his own data
	Non-compliance		Policy and consent compliance	IoT stakeholder who is a PII controller to inform the IoT user who is a PII principal on the IoT system's privacy policy, or allow the IoT user to specify consents in compliance with legislation

::.. Threat Models (3/5)



LINDDUN methodology steps are grouped in two parts. The problem space steps (step 1-3), aim at describing privacy threats.

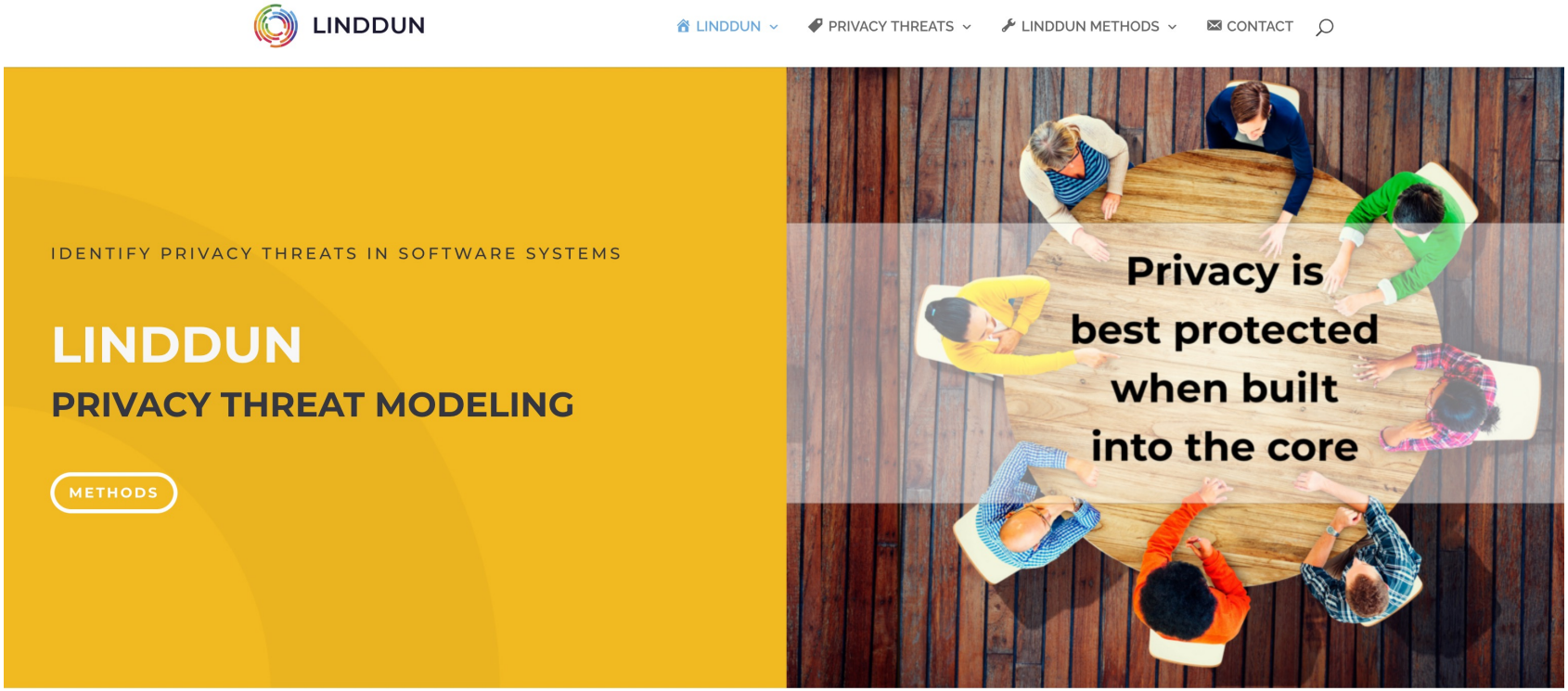
The solution space steps (step 4-6) elicitate mitigation measures and solutions corresponding to the threats identified.

Here's an example of mitigation strategies proposed in LINDDUN, related to the protection of identifiers (the identifiability and linkability of entities in a system).

Mitigation Strategy		**Privacy Enhancing Techniques (PETs)**
Protect ID	Pseudonyms	Privacy enhancing identity management system, User-controlled identity management system
	Attributes	Privacy preserving biometrics Private authentication
	Properties	Anonymous credentials (single show, multishow)

::.. Threat Models (4/5)

The LINDDUN website (https://linddun.org/13) provides a catalogue of mitigation strategies and associated PETs. The catalogue should be considered as a living repository.



A FRAMEWORK FOR
PRIVACY THREAT MODELING

::.. Threat Models (5/5)

Privacy design strategies have been proposed and included in the guidance part of ISO/IEC 27550 (Privacy engineering for system lifecycle processes), useful in a design process.

Design strategy		Description	Privacy control examples
Data oriented strategies	Minimize	Limit as much as possible the processing of PII	Selection before collection, Anonymization
	Separate	Distribute or isolate personal data as much as possible, to prevent correlation	Logical or physical separation; Peer-to-peer arrangement, Endpoint processing
	Abstract	Limit as much as possible the detail in which personal data is processed, while still being useful	Aggregation over time (used in smart grids), Dynamic location granularity (used in location-based services), k-anonymity
	Hide	Prevent PII from becoming public or known.	Encryption, Mixing, Perturbation (e.g. differential privacy, statistical disclosure control), Unlinking (e.g. through pseudonymization), Attribute based credentials
Process oriented strategies	Inform	Inform PII principals about the processing of PII	Privacy icons, Layered privacy policies, Data breach notification
	Control	Provide PII principals control over the processing of their PII.	Privacy dashboard, Consent (including withdrawal)
	Enforce	Commit to PII processing in a privacy friendly way, and enforce this	Sticky policies and privacy rights management, Privacy management system, Commitment of resources, Assignment of responsibilities
	Demonstrate	Demonstrate that PII is processed in a privacy friendly way.	Logging and auditing, Privacy impact assessment, Design decisions documentation

::.. EU Legislation (1/13)

The EU CyberSecurity Act (CSA) entered into force on the 27th of June 2019, strengthening the mandate of the EU Agency for Cybersecurity (ENISA) and launching the European cybersecurity certification framework for ICT digital products, processes, and services (as IoT).

The certification is currently voluntary and the European Commission intends to evaluate whether a mandatory certification is needed for specific categories of products and services.

- It is based on the Common Criteria, the Common Methodology for Information Technology Security Evaluation, and corresponding standards, respectively, ISO/IEC 15408 and ISO/IEC 18045.

::.. EU Legislation (2/13)

The CSA emphasizes on categorizing products and services based on their use cases and computational capabilities.

According to Article 52 of the EU Cybersecurity Act, three assurance levels are specified based on which a certificate can be issued to a product or service.

Assurance Levels	Description of Evaluation Activities
Basic	Evaluation activities to contain at least a technical documentation review.
Substantial	Evaluation activities to contain: (i) a review to confirm the absence of known vulnerabilities and (ii) tests validating the correct implementation of required security functionalities.
High	Evaluation activities to contain (i) a review to validate the absence of known vulnerabilities, (ii) testing to ensure that required security functionalities are correctly applied and (iii) penetration testing to assess its resistance.

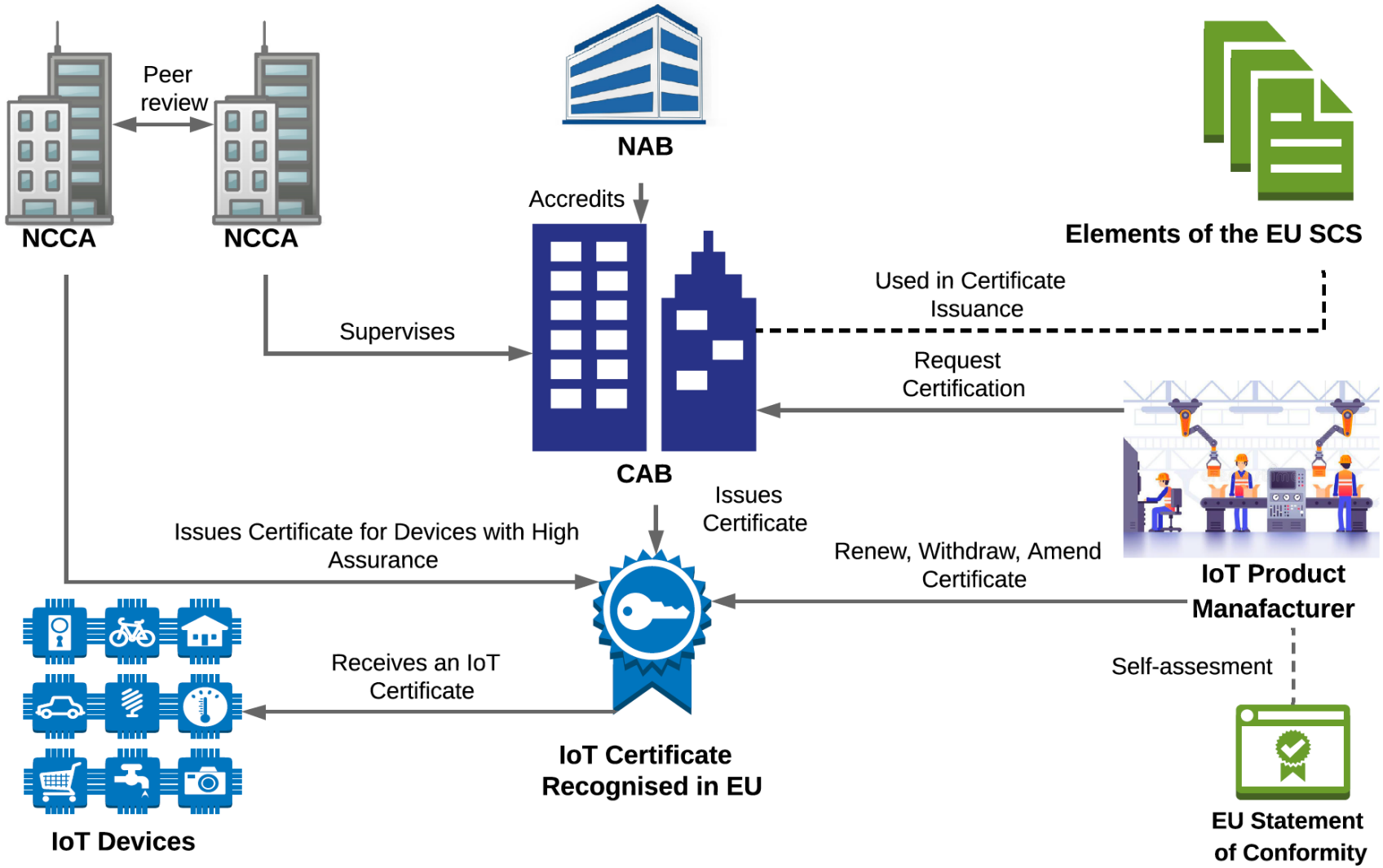
::.. EU Legislation (3/13)

- A Conformity Assessment Body (CAB) is responsible for performing the assessment procedure and is an independent third party which is not the manufacturer or the provider of the product/service/process being assessed.
- A National Accreditation Body (NAB) is responsible for accrediting CABs if they fulfill Requirement specifications.
- The Security Certification Schemes (SCS) should enable the self-assessment feature for Basic assurance level in which the manufacturer/provider itself is responsible for the assessment of the target.
- In case of Self-assessment by manufacturers, it is mandatory that they issue and submit an EU statement of conformity to ENISA and the National Cybersecurity Certication Authority (NCCA).

::.. EU Legislation (4/13)

- An NCCA is a critically important entity as it is responsible for overseeing many aspects of the entire certification process.
  - It should provide the NABs with expertise and information.
  - It should validate CABs that fulfil the requirements set out in the scheme and authorize them to perform assessment and certification.
  - All complaints from legal entities regarding the issued certificates should be directed to and handled by NCCA.
  - NCCA should collaborate with other national authorities or NCCAs, by sharing information on potential noncompliance of products or issues with the SCS or the EU Cybersecurity Act requirements.
  - The certificate issuance request is relayed to the NCCA from the CAB where an EU SCS requires an assurance level High.

::.. EU Legislation (5/13)



::.. EU Legislation (6/13)

The Network and Information Security (NIS) Directive 2016/1148 aimed to achieve a high common level of cybersecurity across EU. While it increased the Member States' cybersecurity capabilities, its implementation proved difficult, resulting in fragmentation at different levels across the internal market.

On 16 January 2023, the Directive (EU) 2022/2555 (known as NIS2) entered into force improving the existing cyber security status across EU in different ways by:

- creating the cyber crisis management structure (CyCLONe);
- increasing the level of harmonization regarding security requirements and reporting obligations;
- encouraging Members States to introduce new interests such as supply chain, vulnerability management, core internet and cyber hygiene their national cybersecurity strategies;

::.. EU Legislation (7/13)

- bringing novel ideas such as the peer reviews for enhancing collaboration and knowledge sharing amongst the Member States;
- covering a larger share of the economy and society by including more sectors which means that more entities are obliged to take measures in order to increase their level of cybersecurity.

The NIS Directive does not specifically and directly cover the IoT, but NIS2 introduces procedural, or 'organisational' requirements that would complement the objectives of protection of EU product safety legislation vis-à-vis (IoT) cybersecurity: (i) the disclosure and handling procedures for vulnerabilities; (ii) cybersecurity requirements in terms of secure supply chain relationships.

::.. EU Legislation (8/13)

- ‘entities that develop such systems should therefore establish appropriate procedures to handle vulnerabilities when they are discovered[.]The manufacturer or provider of ICT products or services should also put in place the necessary procedures to receive vulnerability information from third parties’.
- NIS2 does not foresee an ‘obligation to patch’ within a specific timeframe for manufacturers, thus addressing at the core the problem of unsecure systems.
- When developing patches, ‘should not encounter additional outside pressure which could lead to a deterioration of the quality of the work’, the freedom and trust entitled to manufacturers may come at a cost for consumers in terms of security.

::.. EU Legislation (9/13)

- The majority of IoT devices is comprised of a multitude of components from different hardware and software vendors, part of which could even be accounted for by small companies, including start-ups. This results in a global expansion of the attack surface.
- Supply chain security is crucial in IoT and securing a supply chain represents both a challenge and an opportunity. ENISA mapped out the entire lifespan of the IoT supply chain– hardware, software, and services–by offering security measures for each step.
- NIS2 in Article 18(2) specifies a minimum list of cybersecurity measures that entities have to adopt by addressing supply chain security including security-related aspects concerning the relationships between each entity and its suppliers.

::.. EU Legislation (10/13)

- NIS2 entities ‘shall take into account the vulnerabilities specific to each supplier and service provider and the overall quality of products and cybersecurity practices of their suppliers and service providers; manufacturers will have strong incentives to enhance the security controls in their products.

The EU Cyber Resilience Act, set to roll out in 2024, brings significant changes to the digital product landscape in Europe. This regulatory framework mandates stringent cybersecurity requirements for nearly all products with digital components, from development to end-of-life management. The CRA is designed to work complementary with the EU cybersecurity framework that involves the NIS2, and the CSA.

::.. EU Legislation (11/13)

- All products with digital elements shall only be available in the market if manufacturers ensure that there is an appropriate level of cybersecurity based on the risks. The users shall be informed about the cybersecurity aspects of the digital products.
- The manufacturers have to report after detecting any incident having an impact on the cybersecurity of the digital product within 24 hours.
- Manufacturers are also obligated to perform a conformity assessment of the product with the digital elements and also the vulnerability handling processes that the manufacturer has put in place to comply with the essential requirements set.

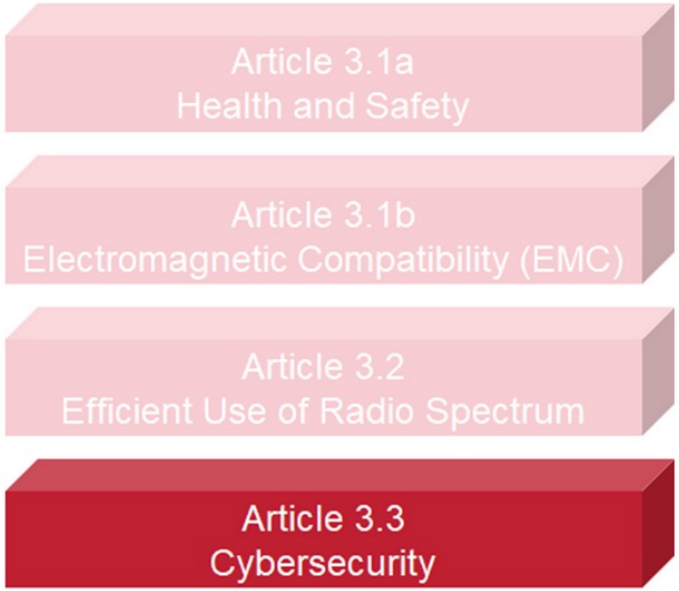
Member States can appoint either existing or new market surveillance authority to effectively implement CRA.

::.. EU Legislation (12/13)

The radio equipment Directive and the delegated regulation 2022/30 (RED) defines ‘radio equipment’ as an ‘electrical or electronic product, which intention-ally emits and/or receives radio waves for the purpose of radio communication’. ‘if IoT devices communicate via radio links, such as Bluetooth or Wi-Fi, they meet the definition of Art. 2 (1) No. 1’.

- It establishes a regulatory framework for placing radio equipment on the market by setting essential requirements for safety and health, electromagnetic compatibility, and the efficient use of the radio spectrum.
- It also provides the basis for further regulation governing technical features for the protection of privacy, personal data and against fraud. Furthermore, additional aspects cover interoperability, access to emergency services, and compliance regarding the combination of radio equipment and software.

::.. EU Legislation (13/13)



Article 3.3(d): manufacturers will have to include features that avoid harming communication networks and prevent disrupting website or services' functionality.

Article 3.3(e): device manufacturers will have to implement measures to prevent unauthorized access or transmission of consumers' personal data.

Article 3.3(f): device manufacturers will have to include features to minimize fraudulent electronic payments and monetary transfers.

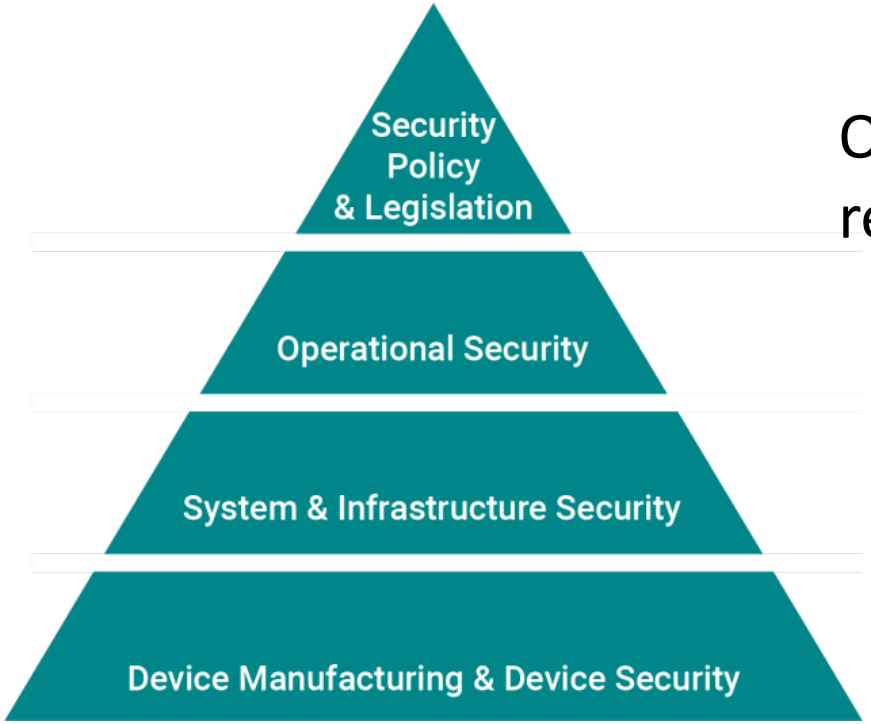
Cybersecurity measures should factor in emerging crime trends in the electronic payments industry such as crypto-jacking, ransomware, near-field communication-related fraud, and biometric authentication tampering.

Secure Design & Development

::.. Security Overview

Product security is fundamental in the development and maintenance of connected systems in both industrial and consumer markets.

The process needs to start with a security policy that provides the basis for enforcement and adherence to appropriate cyber security legislations that apply to the IoT system.



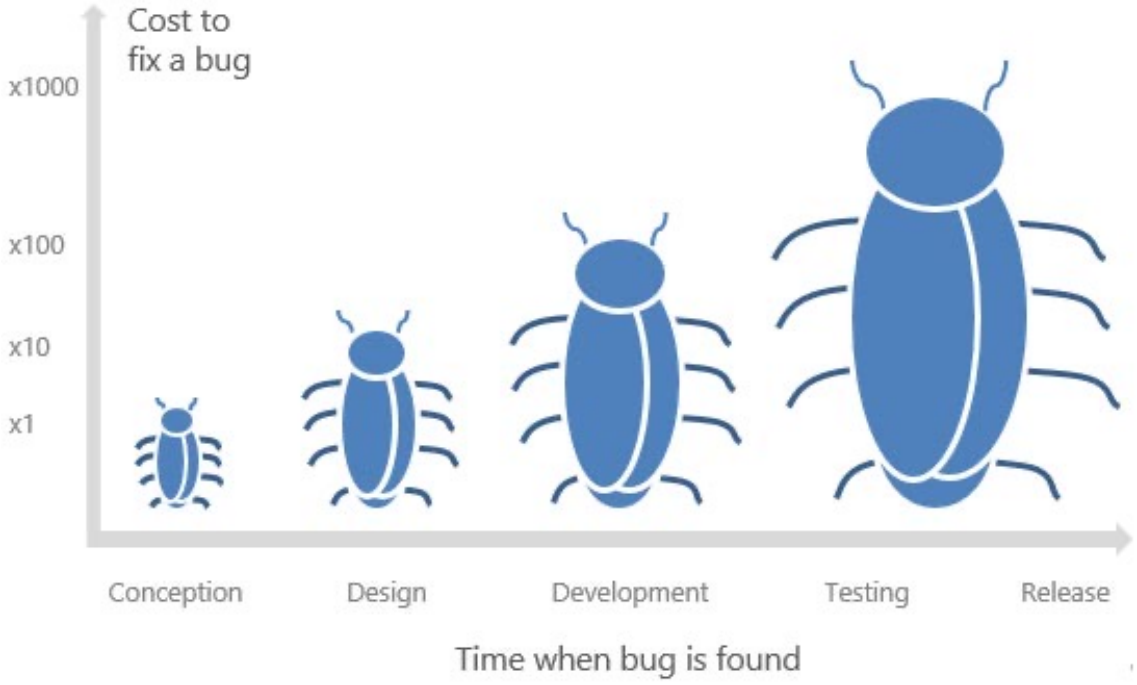
Operational and infrastructure security rely on devices that are inherently secure.

For this to happen, the foundation for robust and reliable IoT infrastructures depends on secure device manufacturing and device security.

::.. Introduction (1/6)

The cost of fixing a security bug varies depending on where it is discovered:

- If it is discovered in the production environment, the cost of fixing it would include the tangible costs of all involved actors, and the intangible cost of reputation and customer trust.



- If it is discovered during the design phase, then it is very easy to correct the design flaw and introduce a security measure during the development phase.
- The cost of fixing a defect postproduction is approximately four times more than fixing it in the development stage.

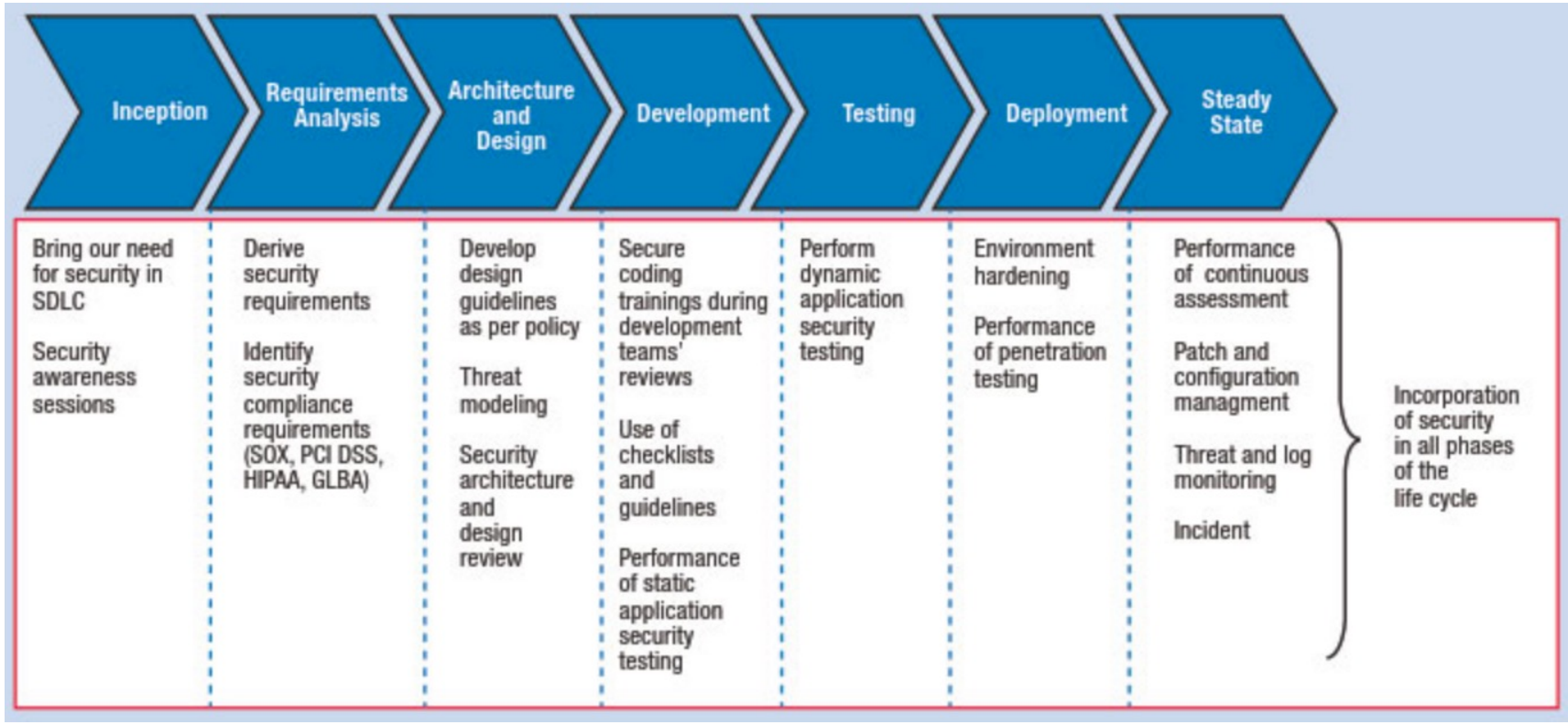
::.. Introduction (2/6)

Security assurance for IoT applications can be best achieved through the adoption of a defense-in-depth strategy, which, in turn, warrants having a Secure Development Life Cycle (SDLC) practice in place.

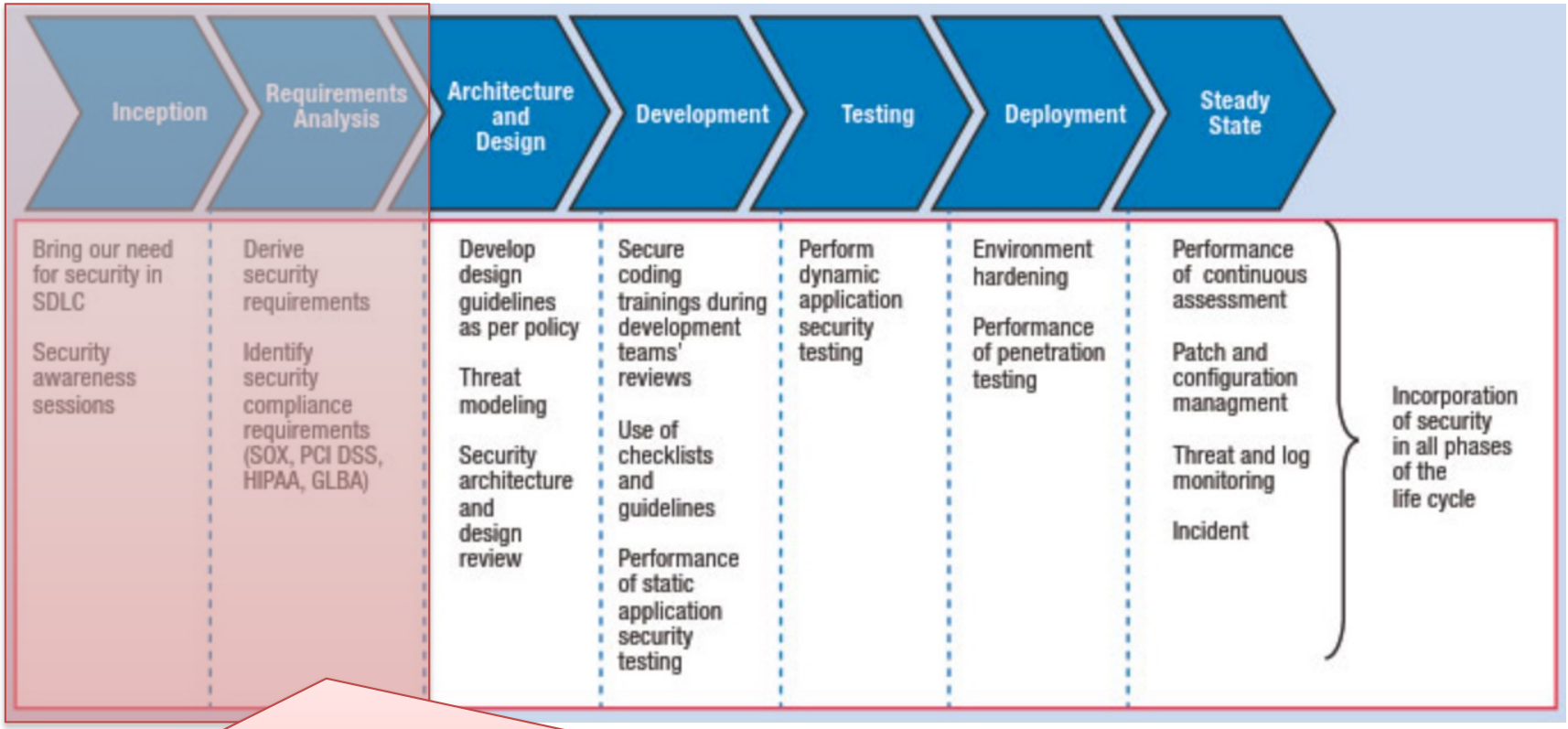
The principal intent is to build security within the life cycle of these applications from ground zero that potentially and gradually reduces the flaws in security, design, implementation and deployment.

Proper adherence to such assurance best practices will result in applications devoid of vulnerabilities that might have been introduced accidentally or intentionally at any point of time in their life cycle.

::.. Introduction (3/6)

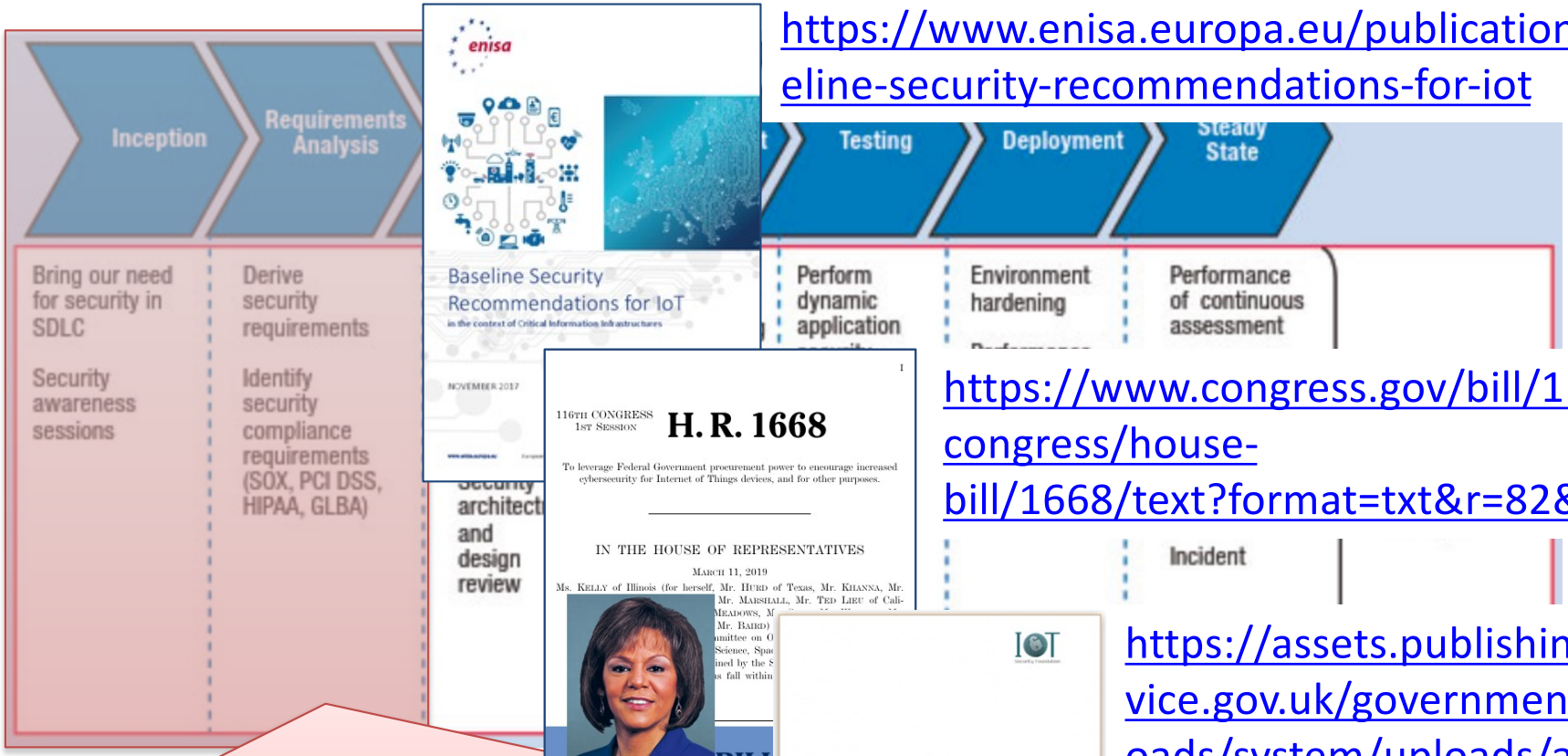


::.. Introduction (3/6)



Effective security management of the IoT components involved within a given solution can be achieved only with a proper inventory of what is to be managed. Moreover, security requirements should be included, by considering standard legislations.

::.. Introduction (3/6)



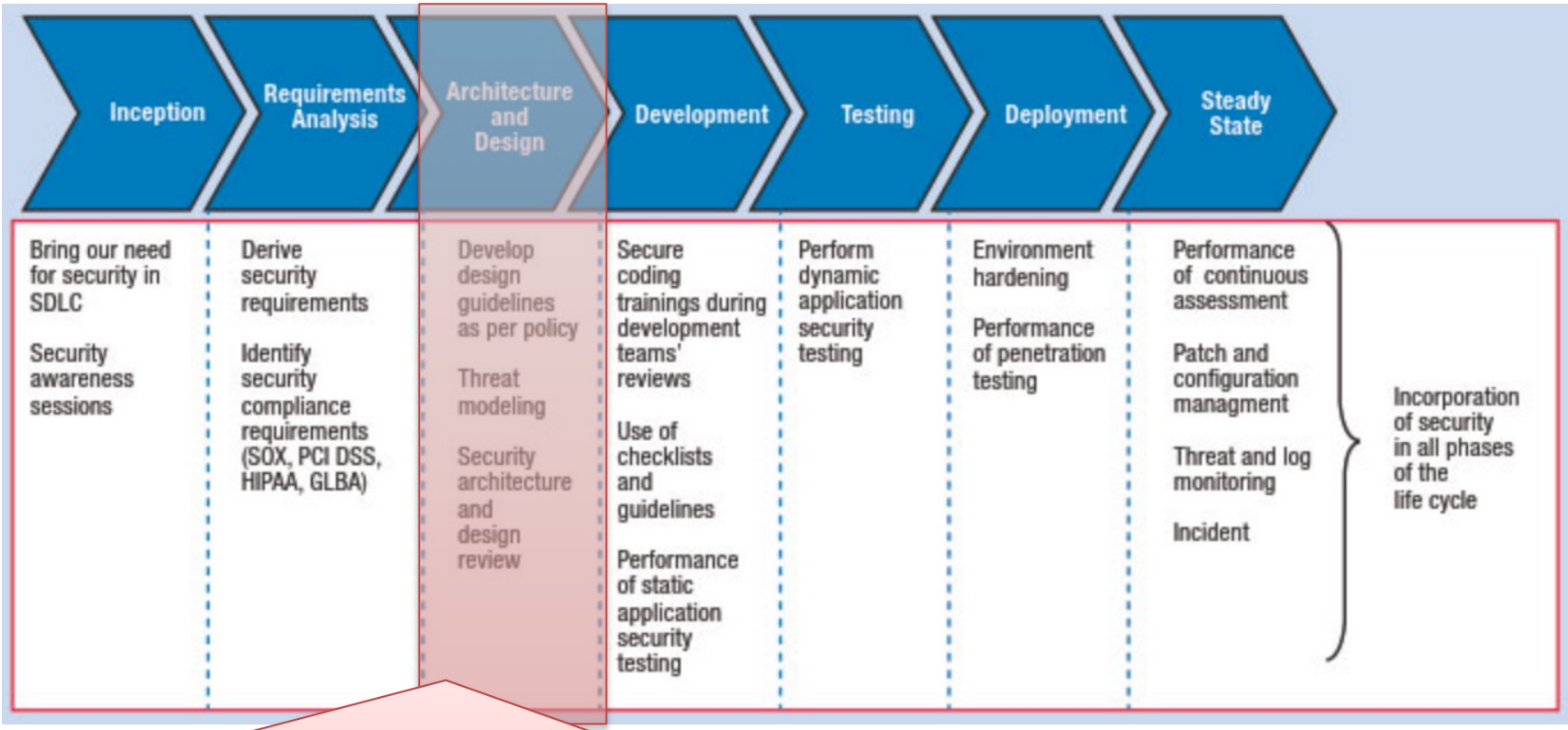
https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot

https://www.congress.gov/bill/116th-congress/house-bill/1668/text?format=txt&r=82&s=1

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/773867/Code_of_Practice_for_Consumer_IoT_Security_October_2018.pdf

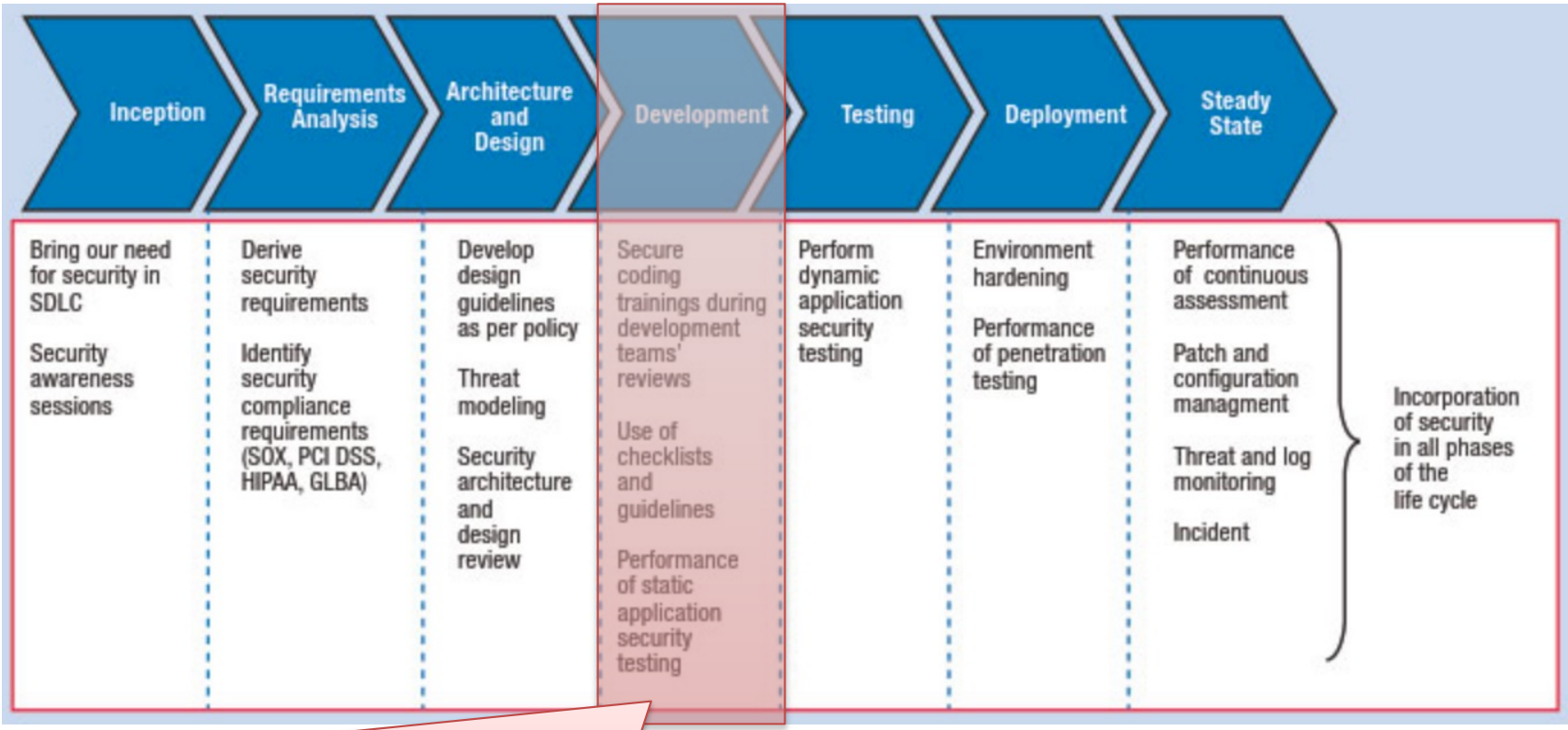
Effective security management within a given solution can be ach of what is to be managed. Morec be included, by considering standa ns.

::.. Introduction (3/6)



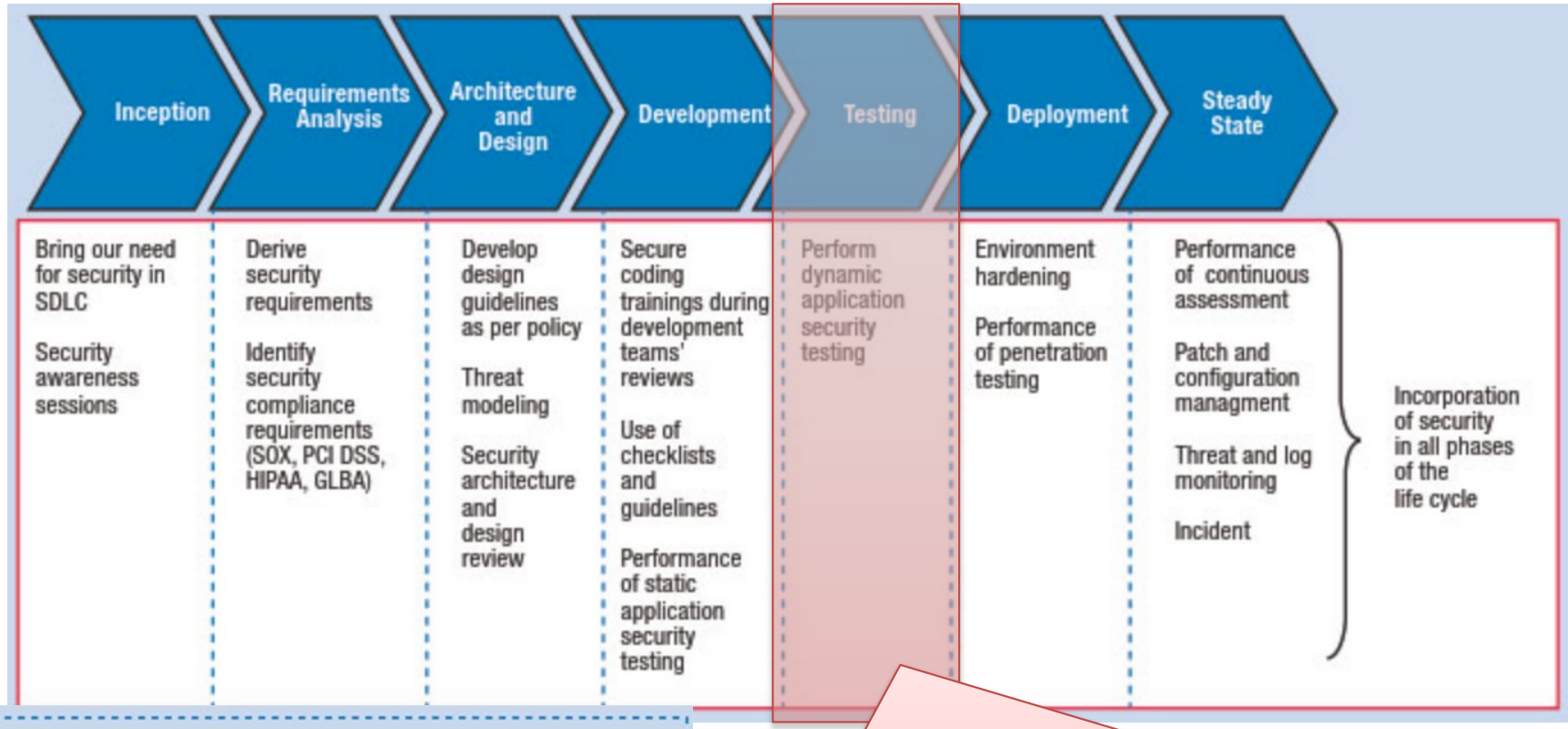
The architecture and design of the IoT solution would be reviewed using threat modeling techniques. Threat actors and scenarios should be enumerated for each of the IoT components. Afterwards, countermeasures can be ranked and recommended.

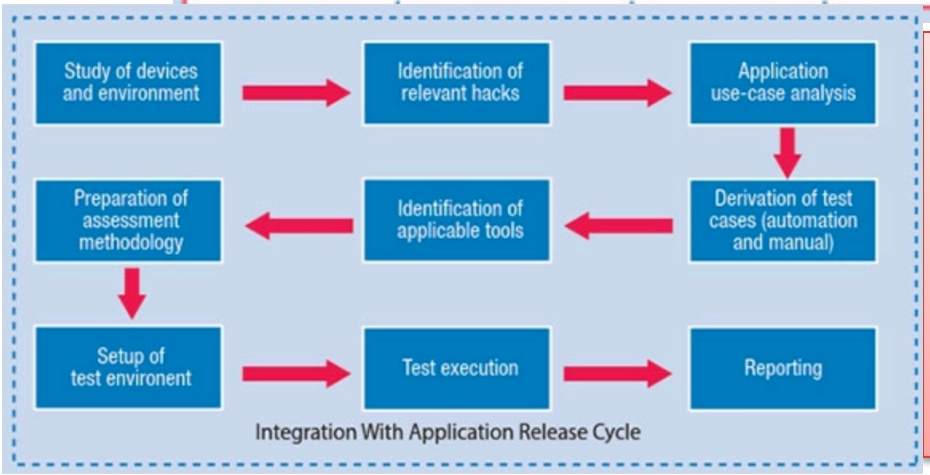
::.. Introduction (3/6)



During the development phase, code needs to be produced according to widely-accepted initiatives and practices in helping programmers circumvent pitfalls and avoid vulnerabilities, such as CERT secure coding standards.

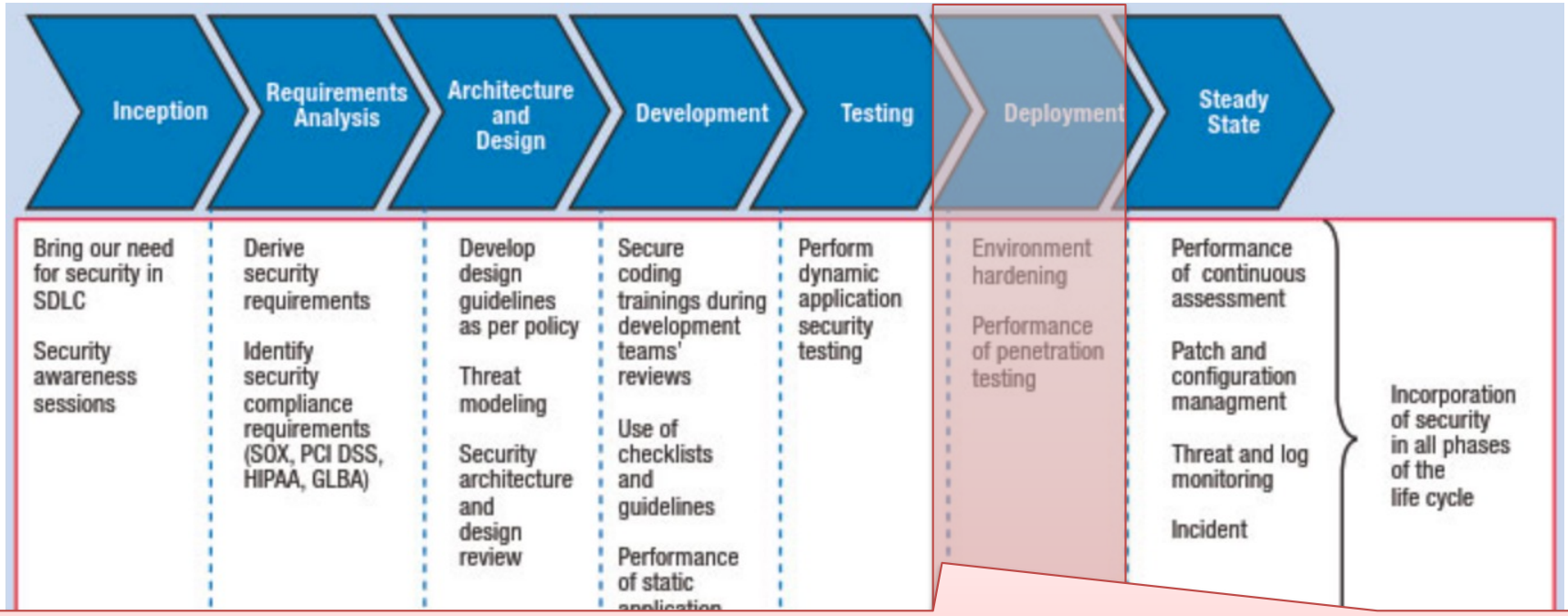
::.. Introduction (3/6)





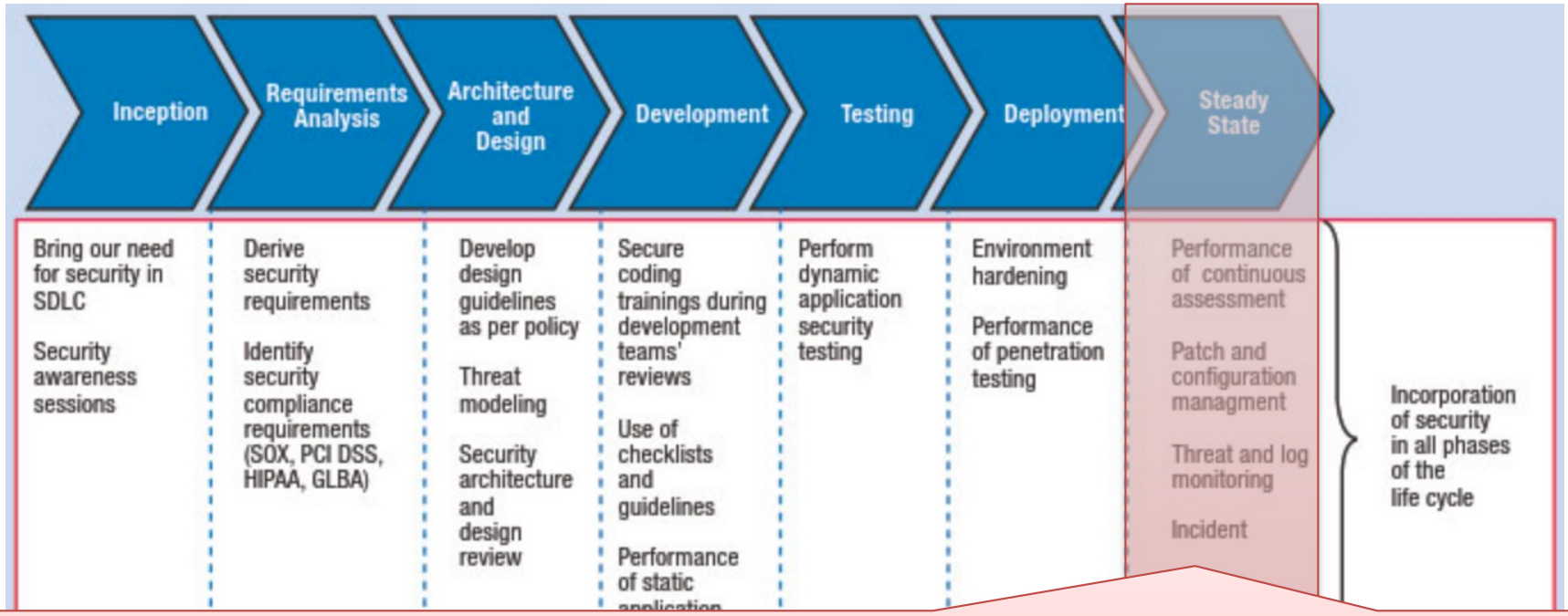
Depending on the type of IoT application and the APIs in place, the necessity for and the type of security assessments to be performed can be decided accordingly.

::.. Introduction (3/6)



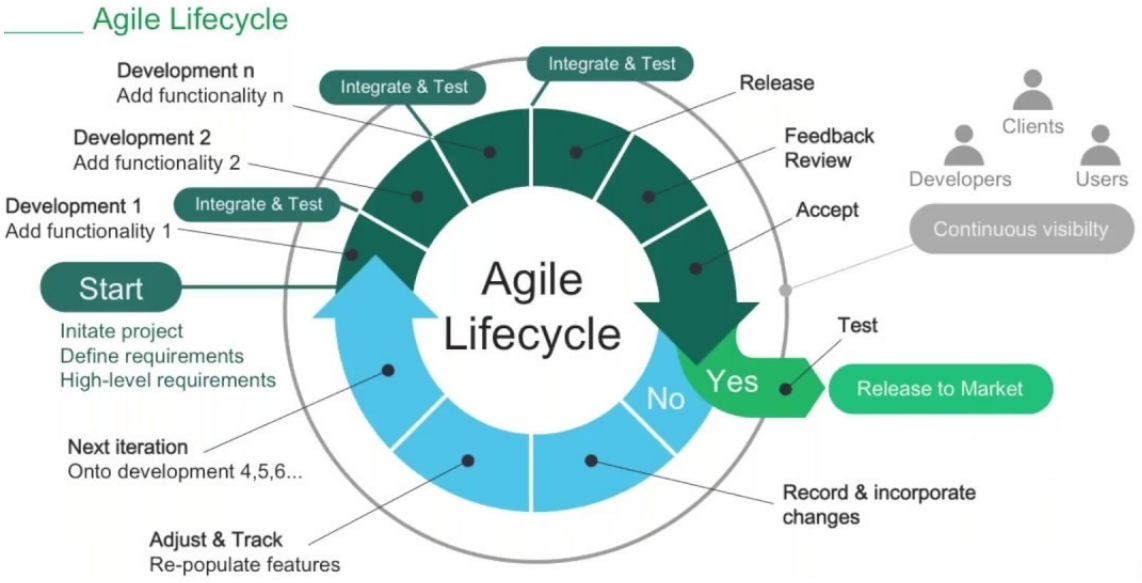
Standard vulnerability assessment and penetration testing have to be conducted on all the IoT environment components, to identify and mitigate the vulnerabilities pertaining to the operating system and platform that could lead to a compromise. Real-world penetration testing should also focus on serving load traffic to the devices and the applications.

::.. Introduction (3/6)



A centralized management and monitoring solution is imperative to track the IoT environment and its components (applications, devices, sensors). This allows realizing security audits, which consist in a systematic evaluation of the security of a company's information system by measuring how well it conforms to a set of established criteria.

::.. Introduction (4/6)



Agile development is widely used in IoT-related projects, and it has different variations, such as Scrum, XP and Kanban.

The Agile Manifesto defines a number of principals that present difficulties for the integration of security engineering approaches.

There are several security requirements that an IoT product must satisfy, and it is difficult to address them in a short development cycle, and they are hard to pin down in user stories, a main feature of the Agile methodology.

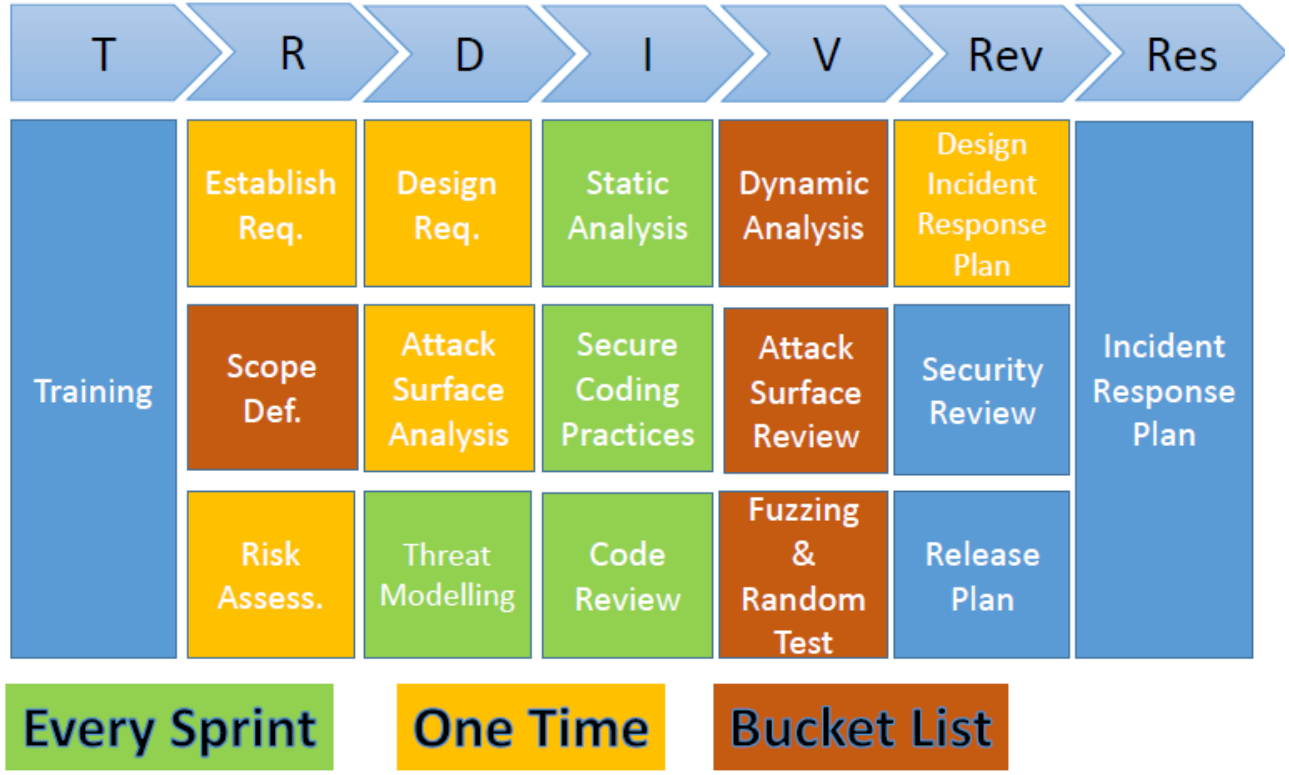
::.. Introduction (5/6)

The idea behind the security in agile is, security requirements are driven by frequency not by phase. Hence, security requirements are divided into three different categories.

1. **Every Sprint** (Most-Critical Requirement) - Some requirements are very critical requirement and the application that failed to meet them will be deemed as incomplete. These are the activities that should be performed in every release and regardless of how long the sprint is, it should be satisfied before releasing the application.
2. **One Time** (Not Repeating Requirement) - These requirements are tasks to be carried out only once in the project lifetime. Usually, these requirements are completed at the beginning stage of an application. There is no necessity to complete the one-time requirement during the first sprint. The grace period is applicable in case the requirements are too large.

::.. Introduction (6/6)

3. **Bucket** (All Other Requirements) - These are the requirements that must be completed several times in the application's lifetime. It includes the important security practices that should be performed at a regular basis. The team does not need to follow any order to complete these requirements.



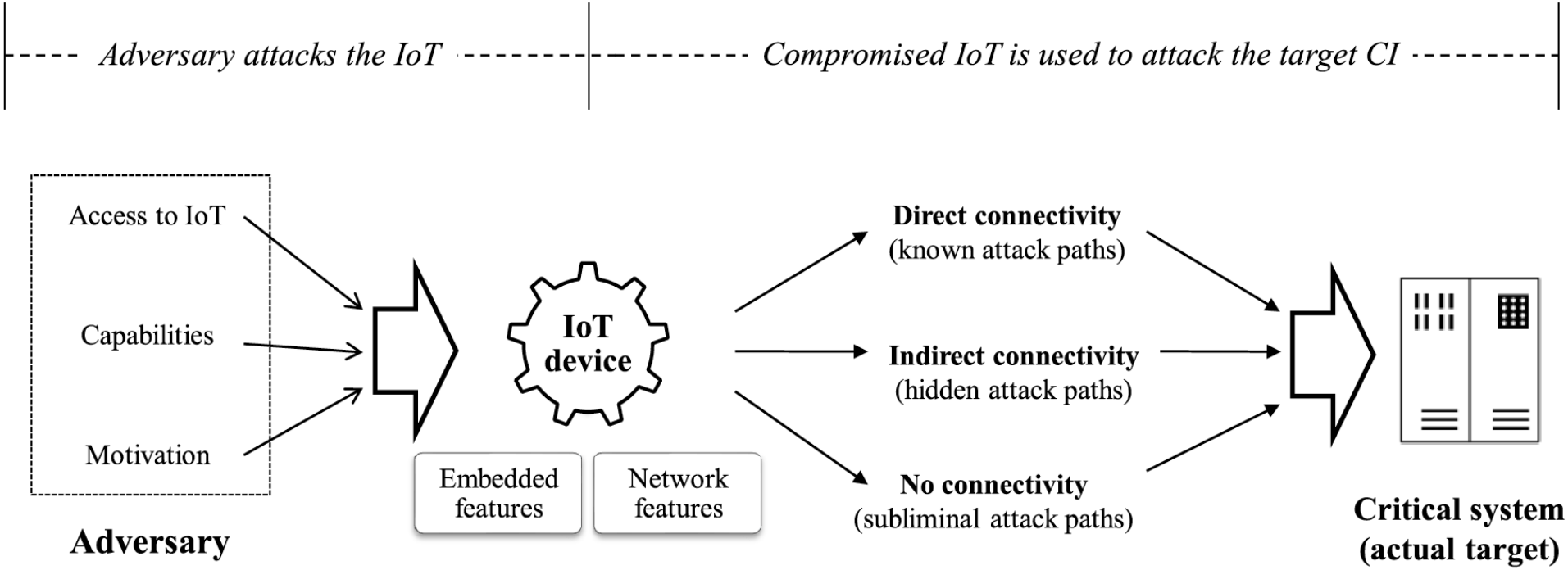
::.. Criticality Assessment (1/6)

An **attack vector** describes the steps that an attacker will undertake in order to realize a threat. The **adversary** represents the actor of the attack. It is the entity whose actual goal is to cause damage to a target system.

The **IoT device** is the enabler (or in some cases the amplifier) of the attack. Being in most cases the weakest link in the security chain, it will usually be used by the adversary as an initial entry point, to gain access to critical services. This can be accomplished by exploiting inherent vulnerabilities, such as lack of embedded security mechanisms or network layer vulnerabilities

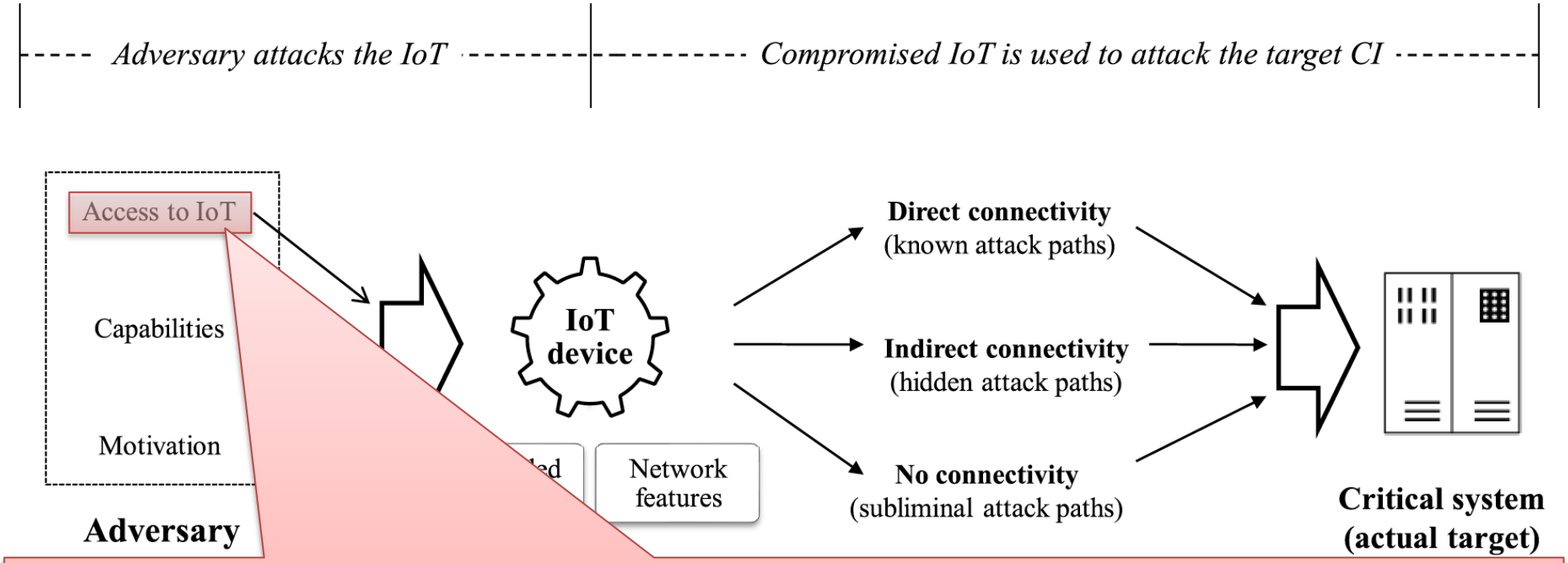
Usually, critical systems of high importance are the actual **targets of attacks**. An adversary with sufficient capabilities and motivation will attempt to abuse existing paths between the vulnerable IoT and a critical system, which may be connected in less obvious, indirect and hidden ways.

::.. Criticality Assessment (2/6)



A motivated adversary will use any potential access to the IoT device and his/her skills in order to compromise the device. Then, by exploiting all the connectivity paths of the IoT device with other systems, he/she will eventually attack the critical system. The connectivity of the IoT device with the critical system may not be obvious.

::.. Criticality Assessment (2/6)



This characteristic examines what type (physical and/or logical) and level of access to the IoT device is required, in order to trigger the attack.

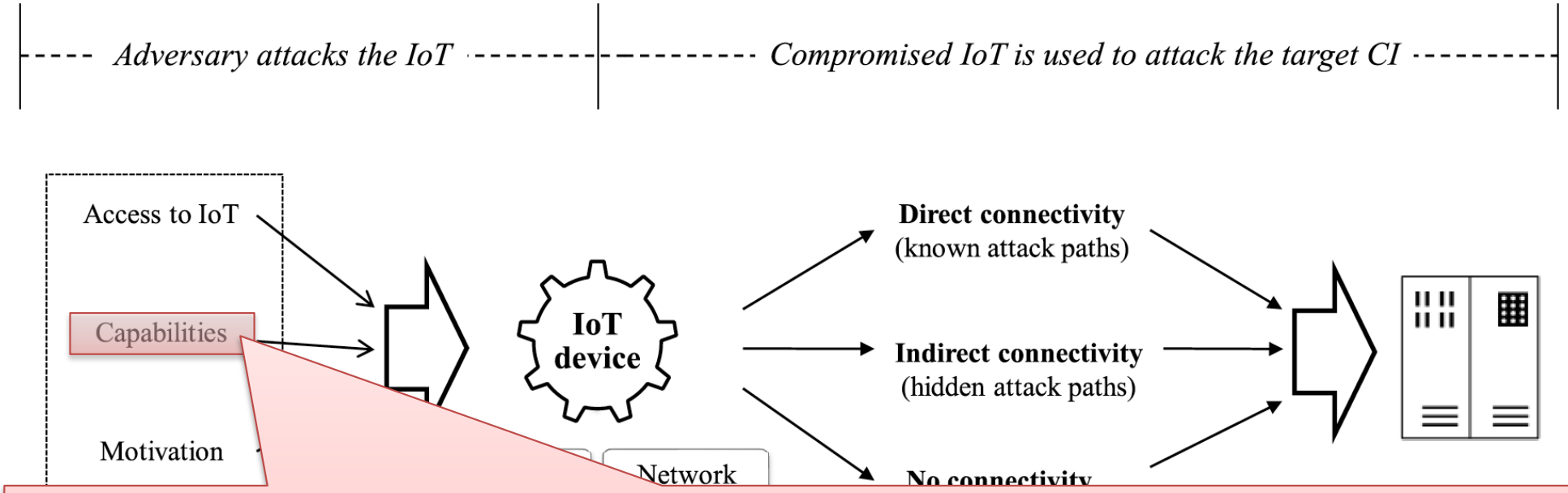
::.. Criticality Assessment (3/6)

1. **Physical access**:
   - An **insider** is an adversary that has direct physical access, or characterised with physical proximity, to the IoT device.
   - An **outsider** has no direct physical access or proximity to the target IoT device, but may try to gain knowledge by tampering another IoT device of the same type.

   In general, if an attack can be realized only by insiders, it is less likely to happen than an attack that could also be triggered by outsiders.
2. **Logical access**:
   - **Privileged access adversaries** are allowed to logically connect to the IoT device through an available interface.
   - **Unprivileged adversaries** does not have a priori logical access to the target device.

In general, attacks that require privileged logical access to the IoT device are less likely to happen, since the adversary will have to bypass authorization controls. On the other hand, attacks that do not require privileged access are more likely to happen.

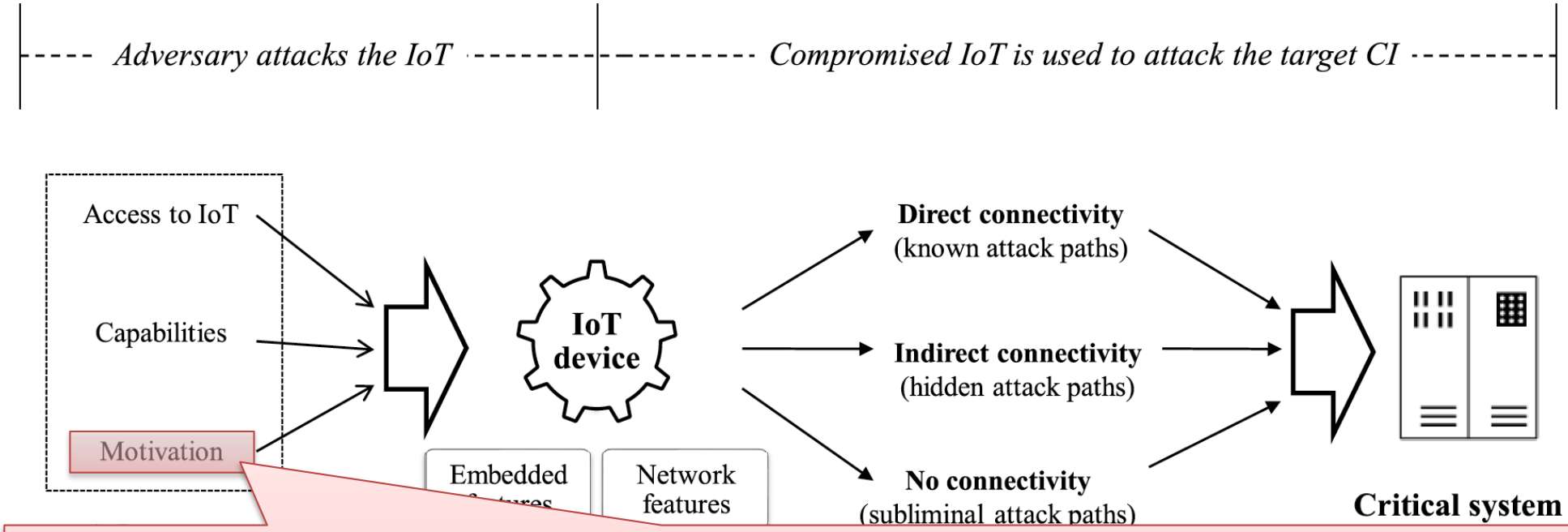
::.. Criticality Assessment (4/6)



This characteristic models the skills and resources required by an adversary to successfully attack the target system:

- Attacks only implemented by technical experts are less likely to happen, than those triggered by novice adversaries.
- Similarly, attacks implemented only by adversaries with high resources, are less likely to happen, in contrast to attacks that require, for example, cheap Off-the-Shelf equipment only.

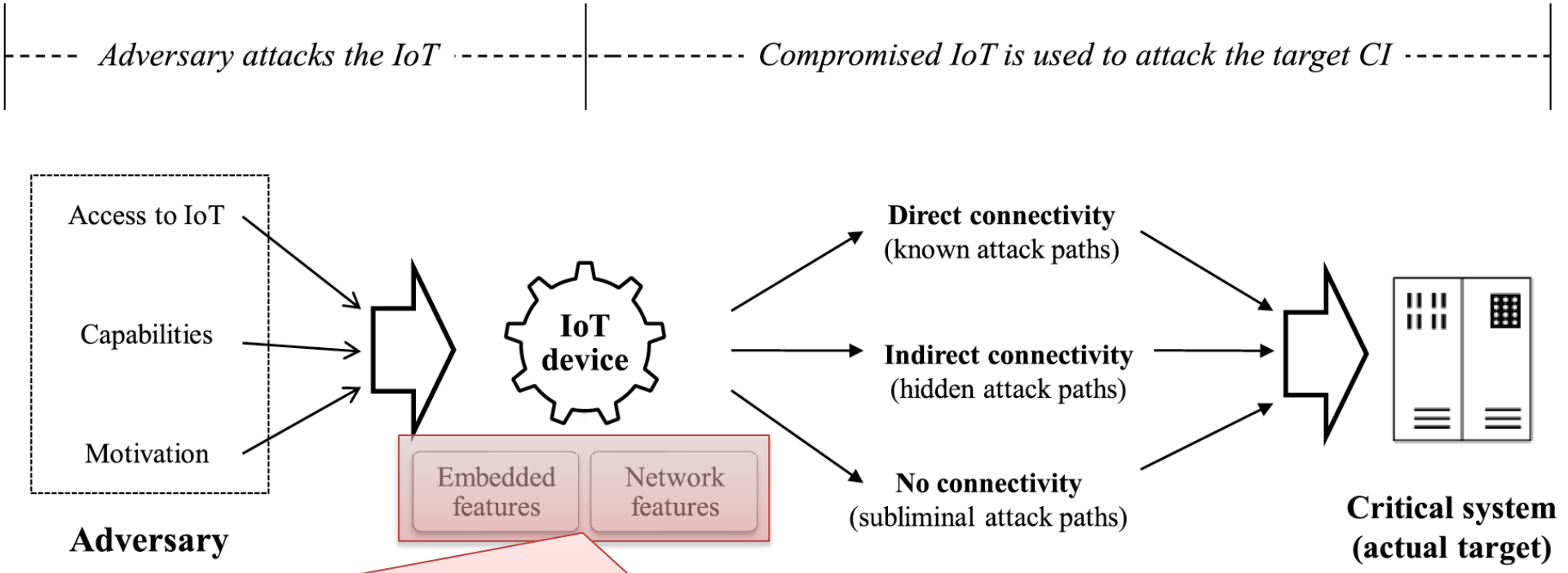
::.. Criticality Assessment (4/6)



Motivation may be seen as an alternative way to describe the potential gain that an adversary would benefit from a successful attack, in combination with the expected penalty for an adversary being traced.

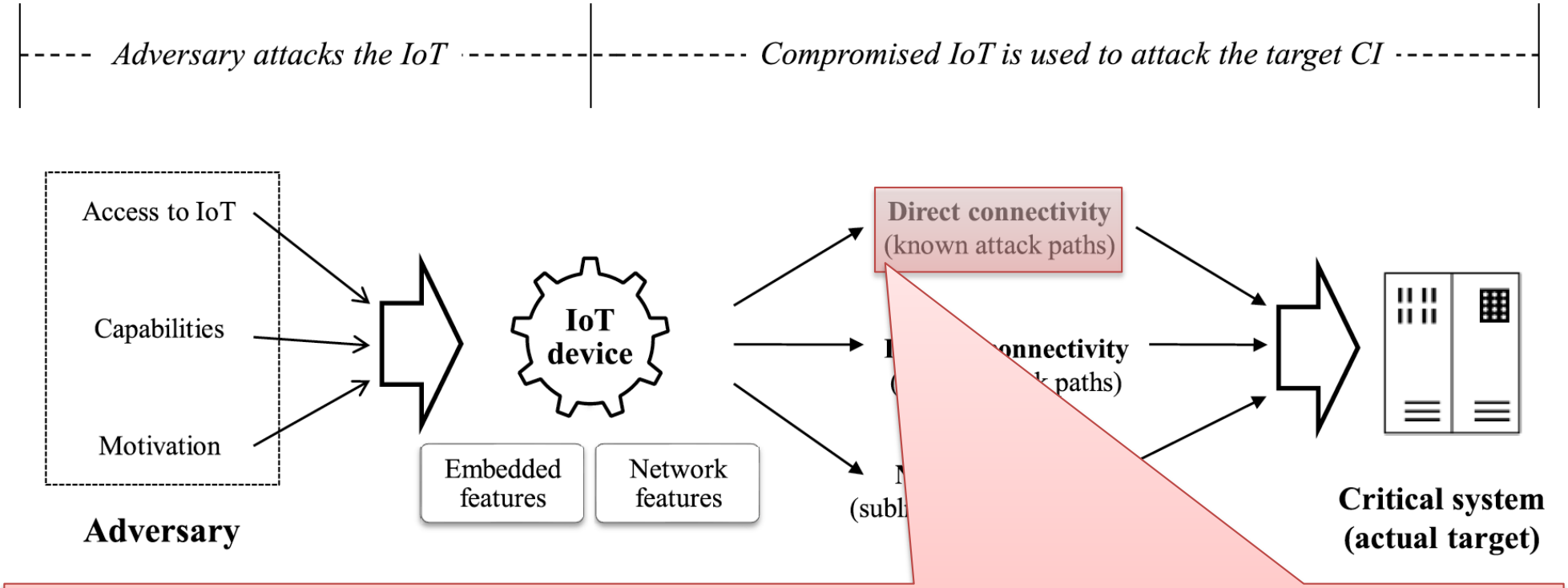
Attacks that can attract adversaries having even a weak motivation are more likely to happen. In contrast, attacks that would be triggered only by strongly motivated adversaries are less possible.

::.. Criticality Assessment (4/6)



Since the IoT device is the enabler/amplifier of the attack, an adversary shall discover and exploit existing vulnerabilities grouped in two main categories: **Embedded vulnerabilities**, i.e., design and implementation flaws at the IoT hardware (HW) and the software (SW) layers, and **Network vulnerabilities**, i.e., flaws in the network protocols and the supporting mechanisms of IoT communications.

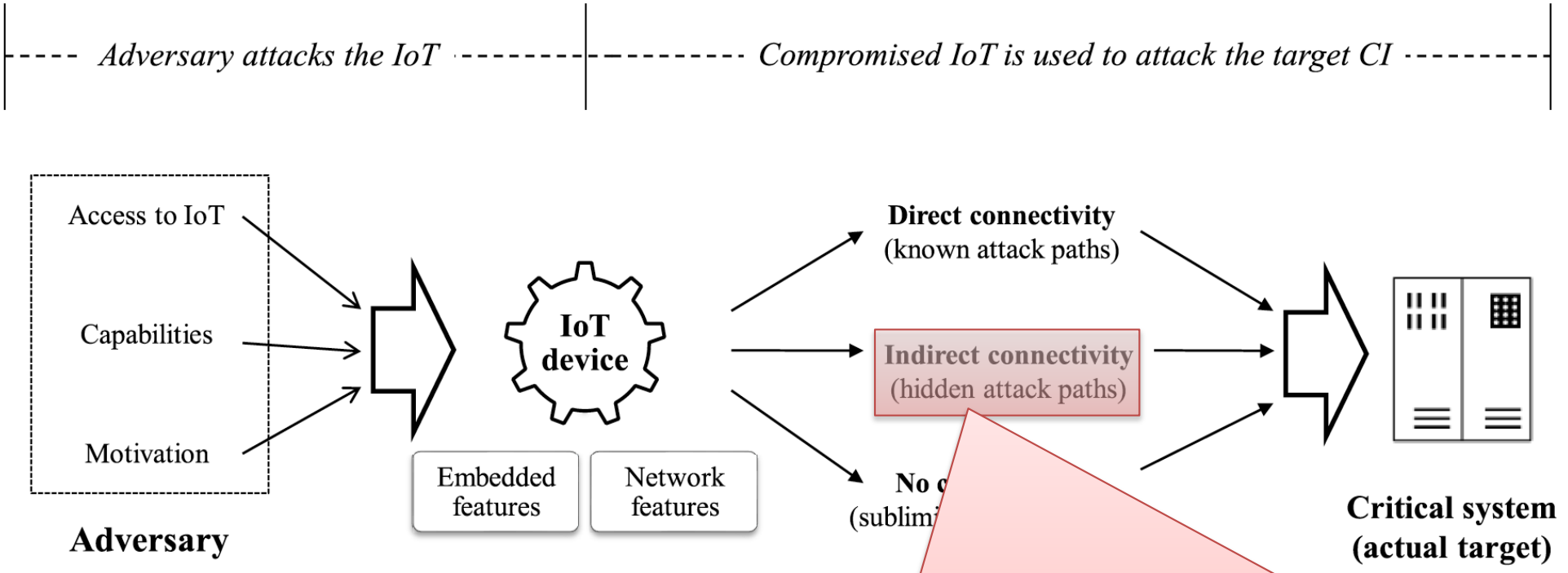
::.. Criticality Assessment (4/6)



IoT device is physically and/or logically connected with a critical system.

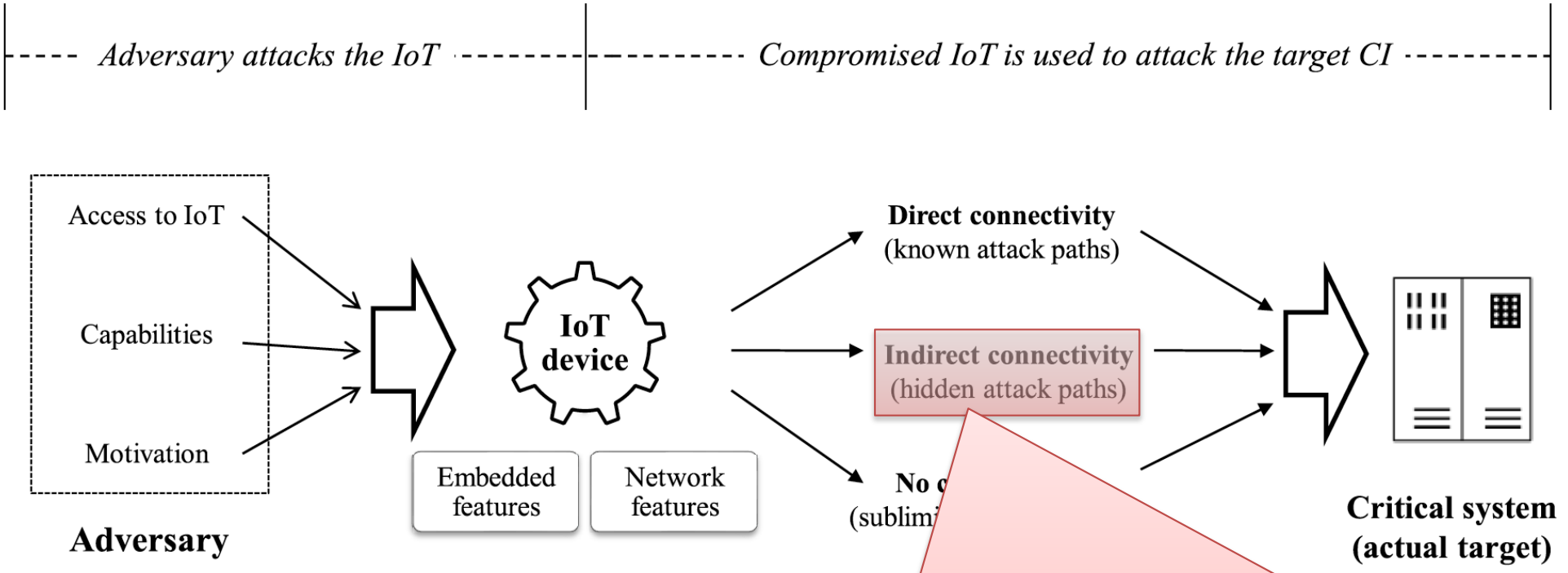
- A physical connection usually implies that the IoT device is installed inside a secured physical perimeter.
- Direct logical connection: A logical connection may refer to IoT devices that are either inside or outside the CI premises.

::.. Criticality Assessment (4/6)



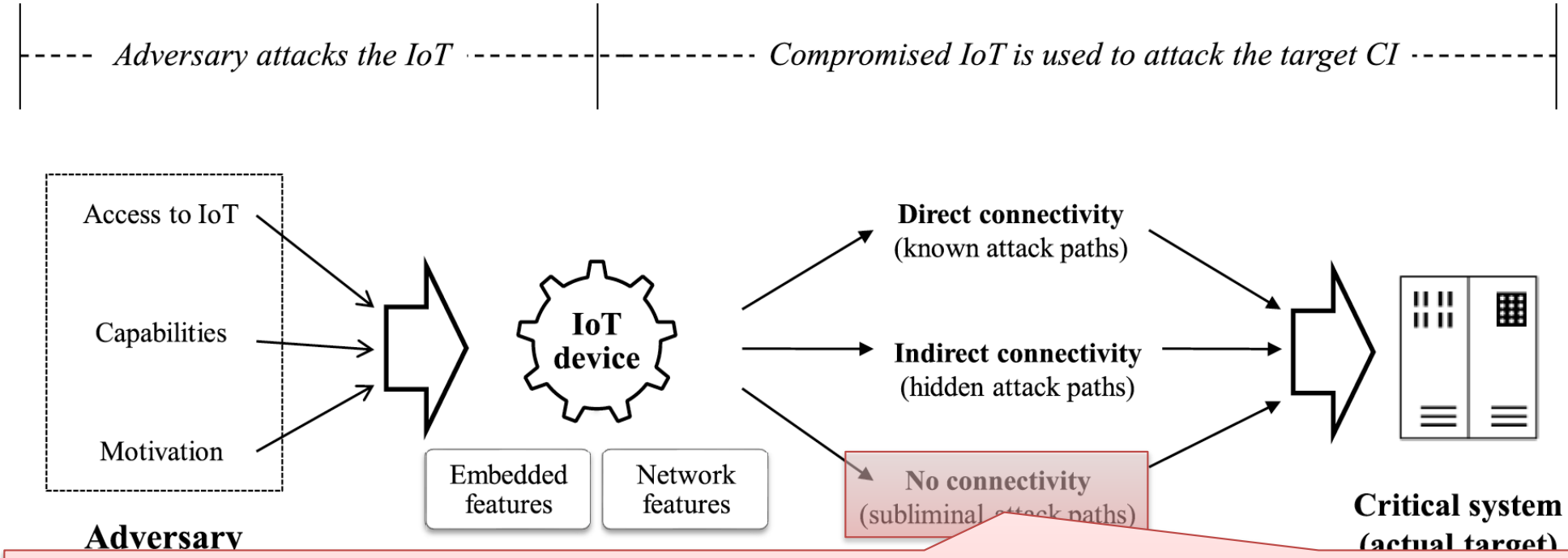
IoT devices that are connected with a critical system in an indirect and non-obvious way, have been used to attack the system.
They can be very dangerous, mostly because they are overlooked and therefore underestimated; if such indirect connections are not identified, then a threat with a potentially high impact will be neglected.

::.. Criticality Assessment (4/6)



- **Physical proximity:** An auxiliary and usually low importance IoT device that resides near a critical system, may be used to create a hidden attack path.
- **Indirect logical connectivity:** IoT devices may be connected to an auxiliary system that is logically connected to a critical system.

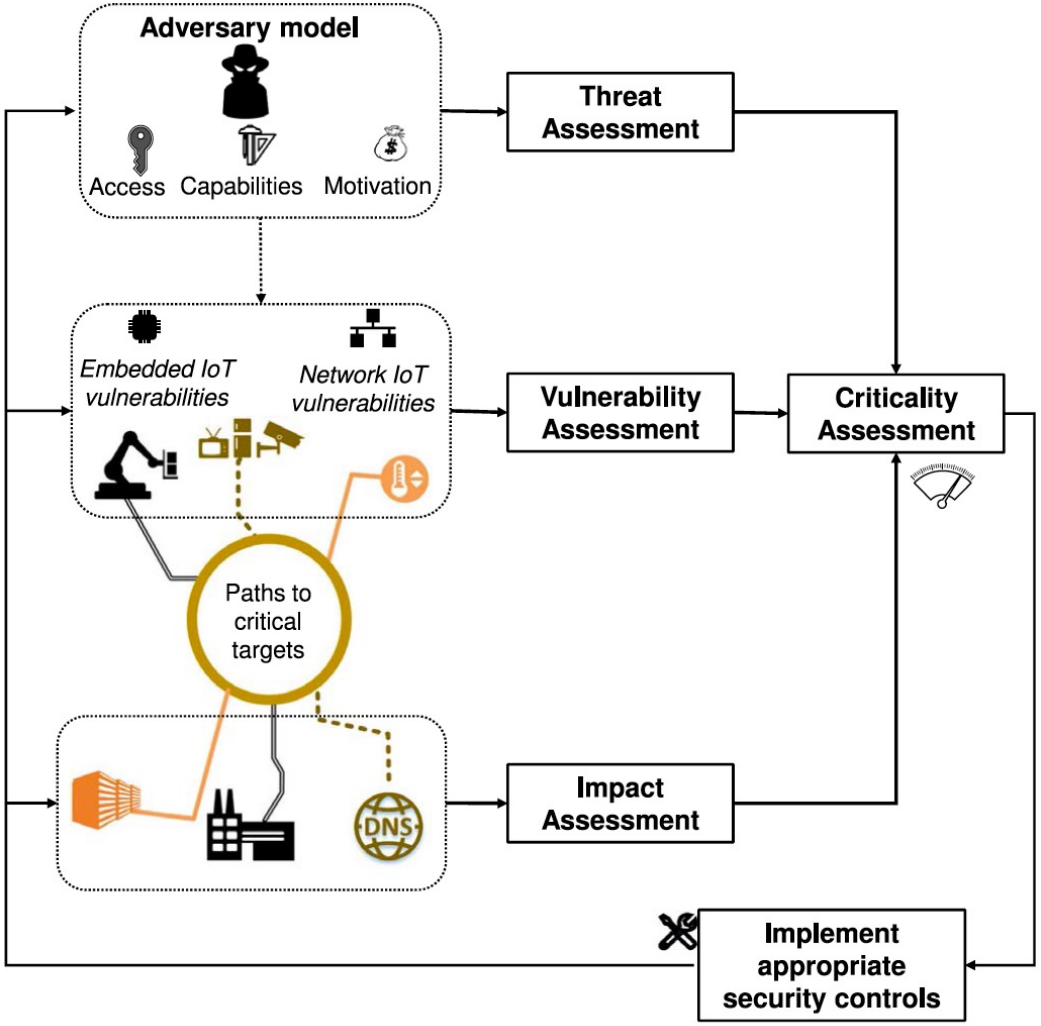
::.. Criticality Assessment (4/6)



Smart IoT devices that are not connected with critical systems have also been used to attack critical systems and services.

- An adversary can exploit built-in vulnerabilities in a plethora of end-user IoT devices to control them and create a botnet.
- The attack is actually targeting against a large number of end-user IoT devices, and its massiveness may lead to very important consequences.

::.. Criticality Assessment (5/6)



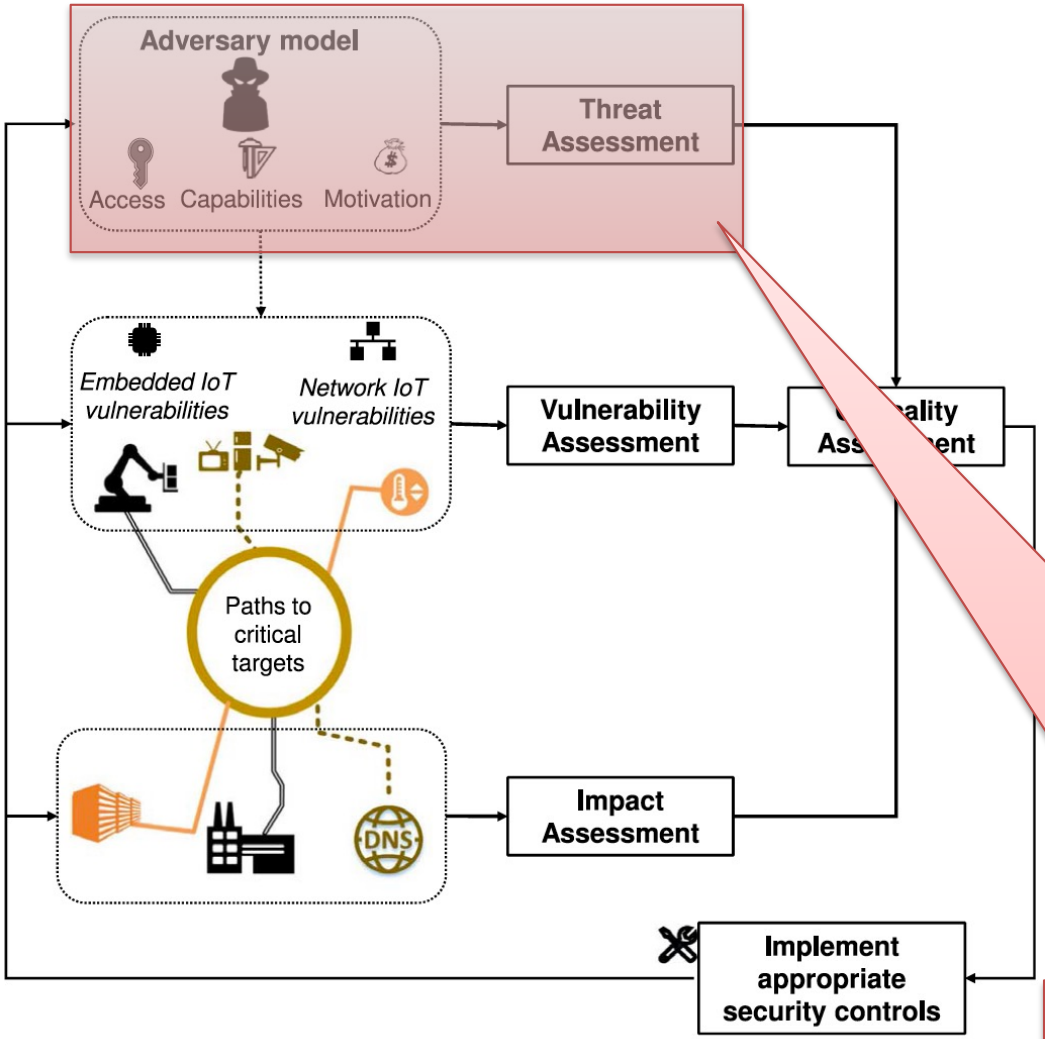
Examples of attack paths against critical infrastructures and services

- *Direct : An internet connected industrial robot used to disrupt factory production*
- *Indirect: A thermostat in the data center causing DoS on nearby servers*
- *No connectivity: Home appliances creating a botnet against a DNS server*

In order to assess IoT-enabled cyber attacks in terms of their severity, generic risk based methodologies have been formulated by various security standards.

A security attack can be assessed based on the security risk that it may cause to a target system.

::.. Criticality Assessment (5/6)



Examples of attack paths against critical infrastructures and services

Direct : An internet connected industrial robot used to disrupt factory production

Indirect: A thermostat in the data center causing DoS on nearby servers

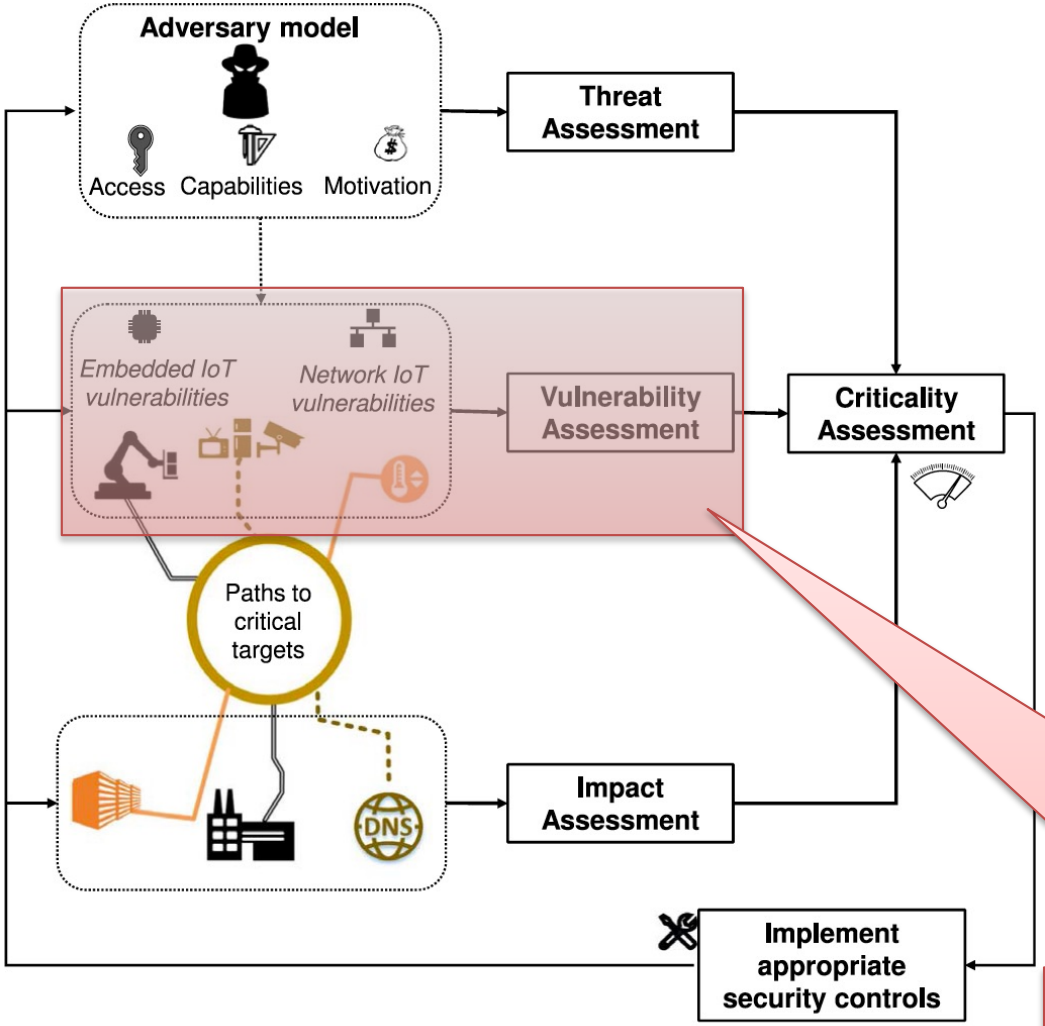
No connectivity: Home appliances creating a botnet against a DNS server

In order to assess IoT-enabled cyber attacks in terms of their severity, generic risk based methodologies have been formulated by various security standards.

security attack can be
ed based on the security
it may cause to a
n.

The threat level measures the extent to which a system is threatened by the attack.

::.. Criticality Assessment (5/6)

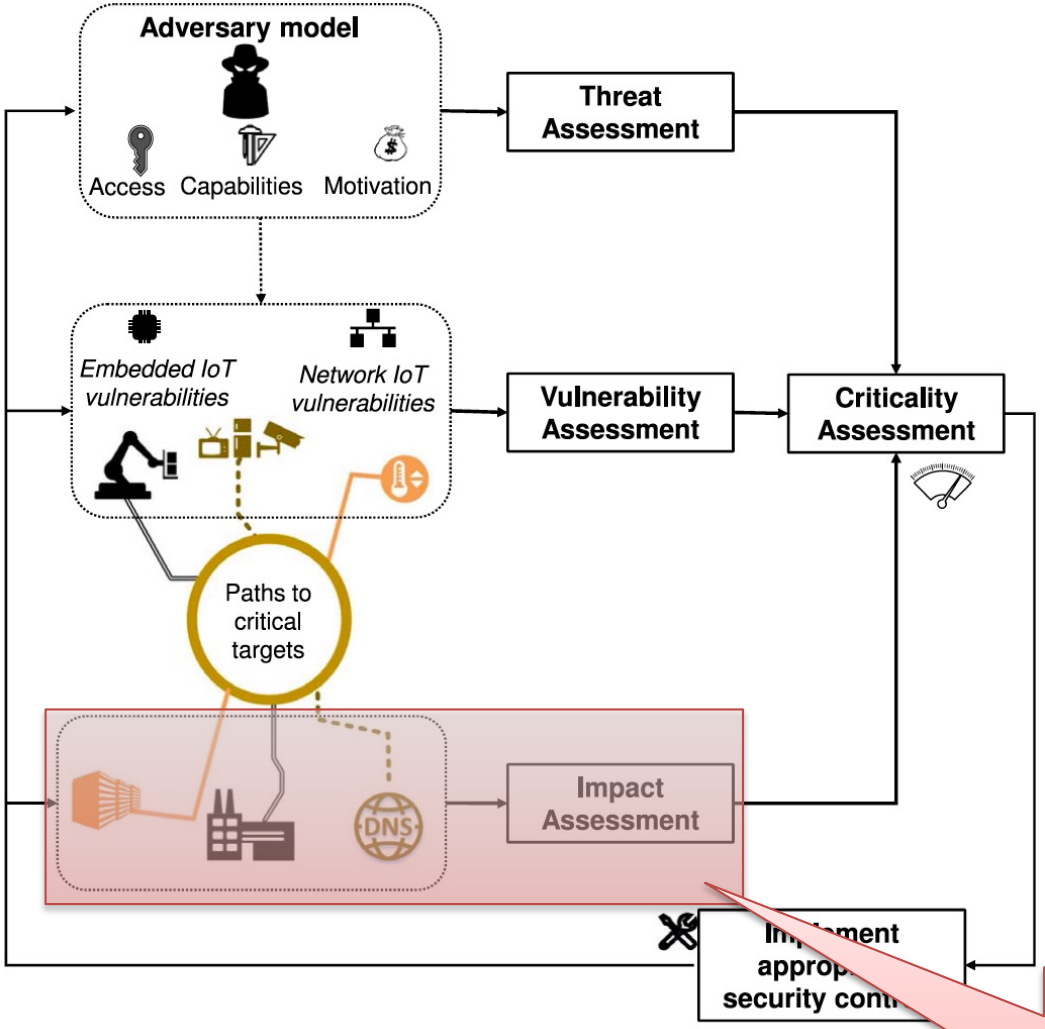


In order to assess IoT-enabled cyber attacks in terms of their severity, generic risk based methodologies have been formulated by various security standards.

A security attack can be assessed based on the security [illegible]hat it may cause to a targ[illegible]m.

The vulnerability level measures the weaknesses exploited by an adversary to realize the attack.

::.. Criticality Assessment (5/6)



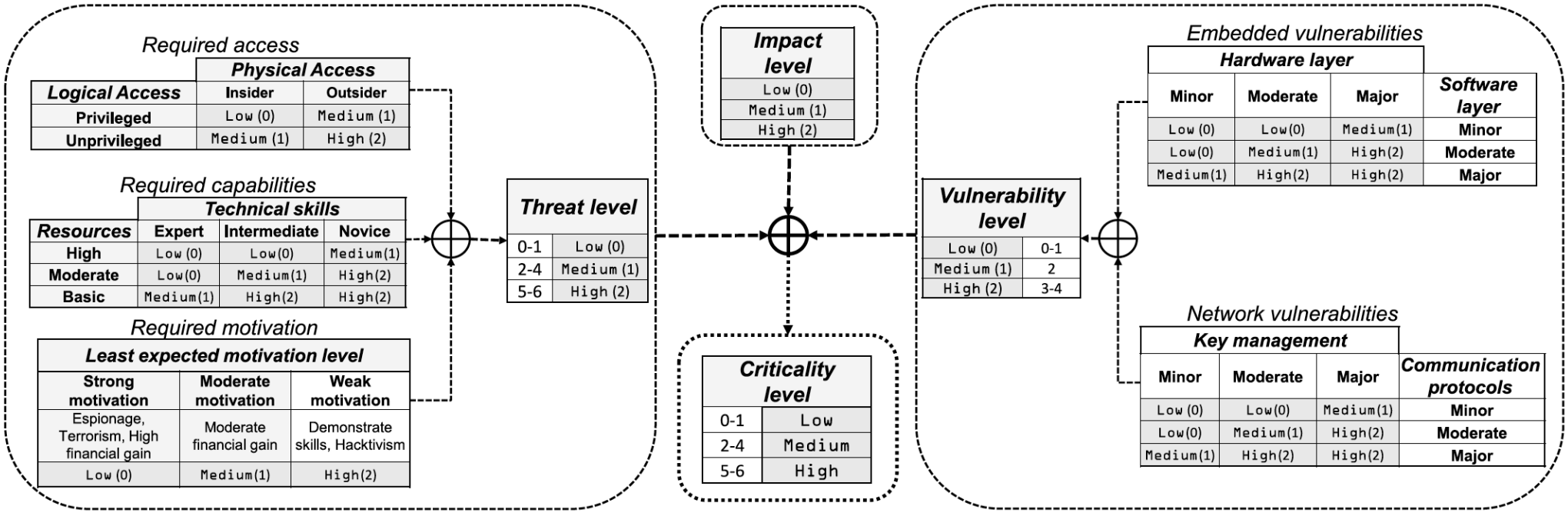
In order to assess IoT-enabled cyber attacks in terms of their severity, generic risk based methodologies have been formulated by various security standards.

A security attack can be assessed based on the security risk that it may cause to a target system.

The impact level represents the potential damage that would be caused by the attack.

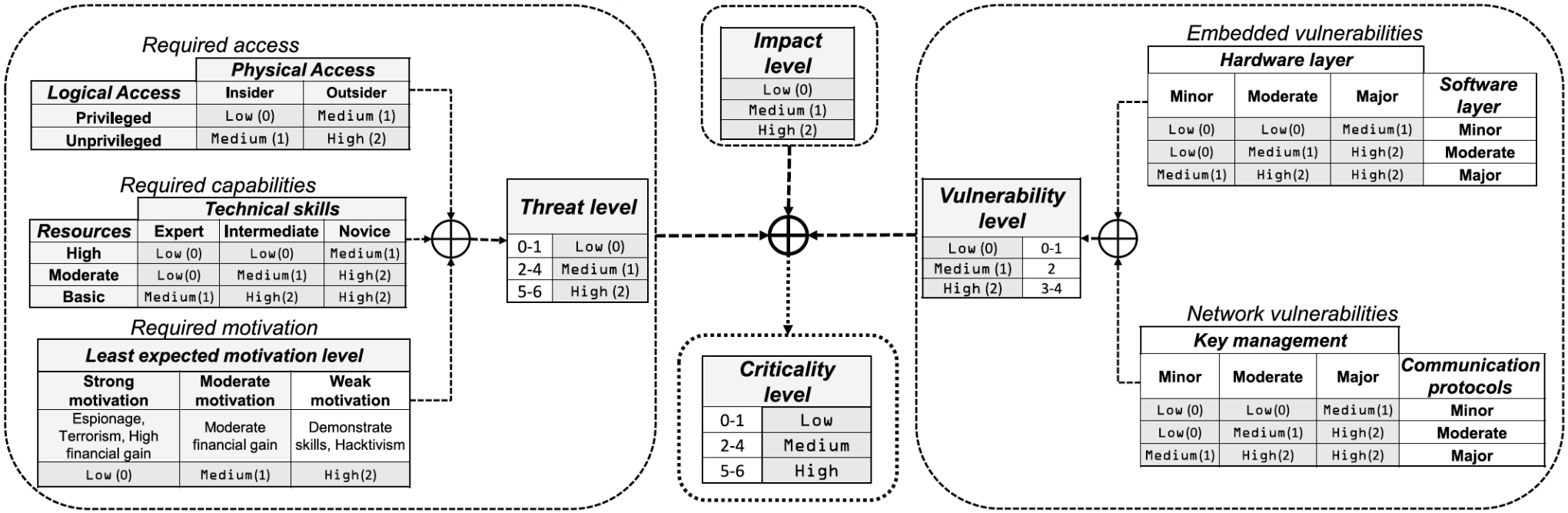
::.. Criticality Assessment (6/6)

For each risk factor (threat, vulnerability and impact level) and eventually for their combined outcome, the criticality level, we will use a three level qualitative scale [Low, Medium, High], where each level is also assigned to an arithmetic value in the range [0, 1, 2].



::.. Criticality Assessment (6/6)

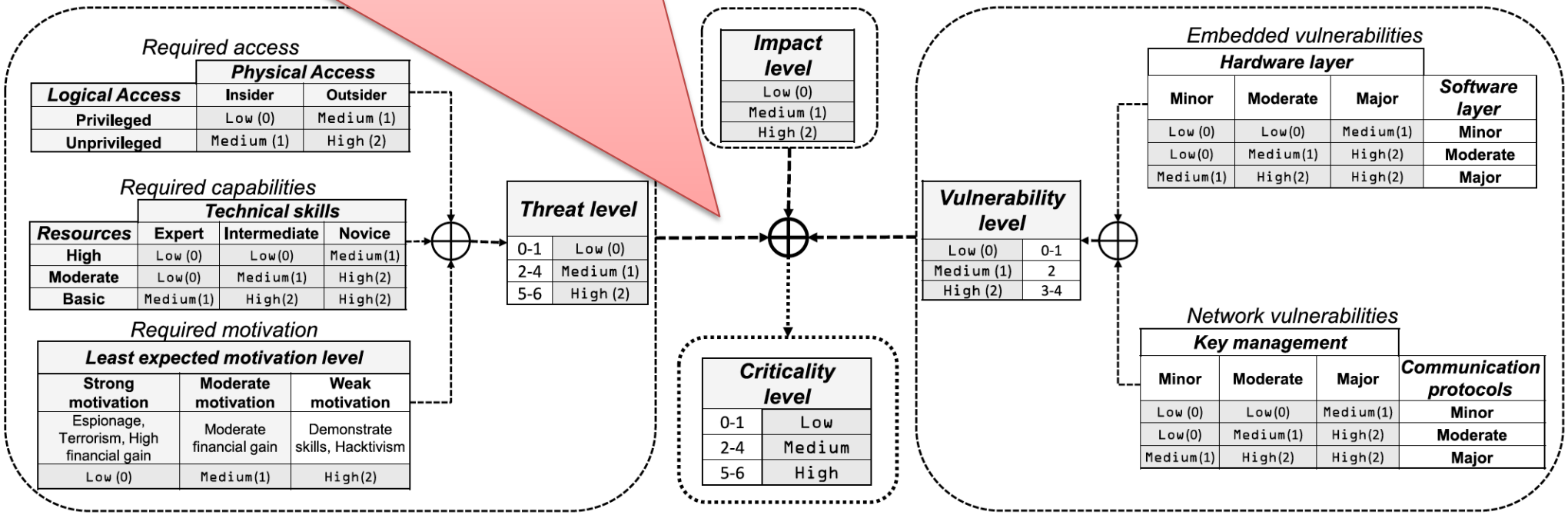
Values	Threat scale	Vulnerability scale	Impact scale	Criticality
Low (0)	Attack requires adversaries having full access to IoT, advanced capabilities and motivation	The involved IoT devices have (at most) minor embedded (HW, SW) and network-layer vulnerabilities	Attack may cause limited damages for any possible attack path	Attacks of low importance and priority
Medium (1)	Attack requires adversaries having some access to IoT, moderate capabilities and motivation	The involved IoT devices have moderate embedded (HW, SW) or network-layer vulnerabilities that are exploitable	Attack may exploit known, hidden or subliminal paths to cause at most moderate damages	Attacks that should be considered with medium priority
High (2)	Attack may be realized by adversaries with no access to IoT, low capabilities and motivation	Highly exploitable HW, SW and network-layer vulnerabilities	Attack may exploit known, hidden or subliminal paths to cause severe damages to a critical system	Highly important attacks that require immediate mitigation



::.. Criticality Assessment (6/6)

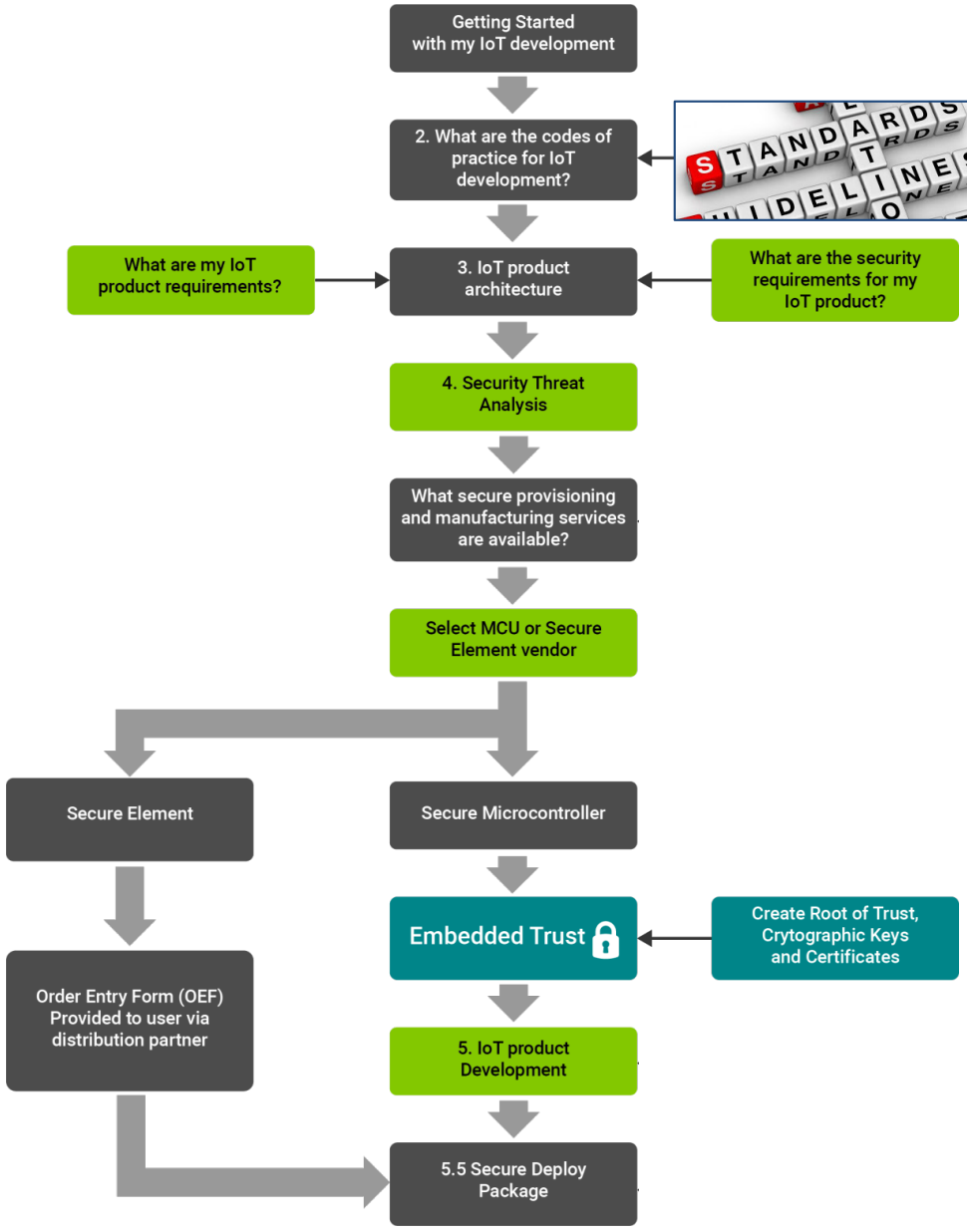
The Threat level and the Vulnerability one are made by combining their respective characteristics level by a simple addition; the the Criticality level is obtained by using a simple “addition-and-reduction” rule:

- The three levels are summed up;
- The obtained arithmetic value is then reduced (mapped) in the [Low(0), Medium(1), High(2)] scale.



IoT Product Development Process

::.. IoT Product Development Process



Government agencies are increasingly becoming aware of the threat posed by non-secure IoT devices. Several standardization bodies have prepared guidelines and standards to provide pragmatic guidance to businesses developing devices and services that have network connectivity to enhance their functionality.

Security best practices require choices in design, features, implementation, testing, configuration and maintenance. There are a great many considerations including protocols, encryption, technology, software, application programming interfaces (APIs), platforms and more.

::.. Security Requirements (1/4)

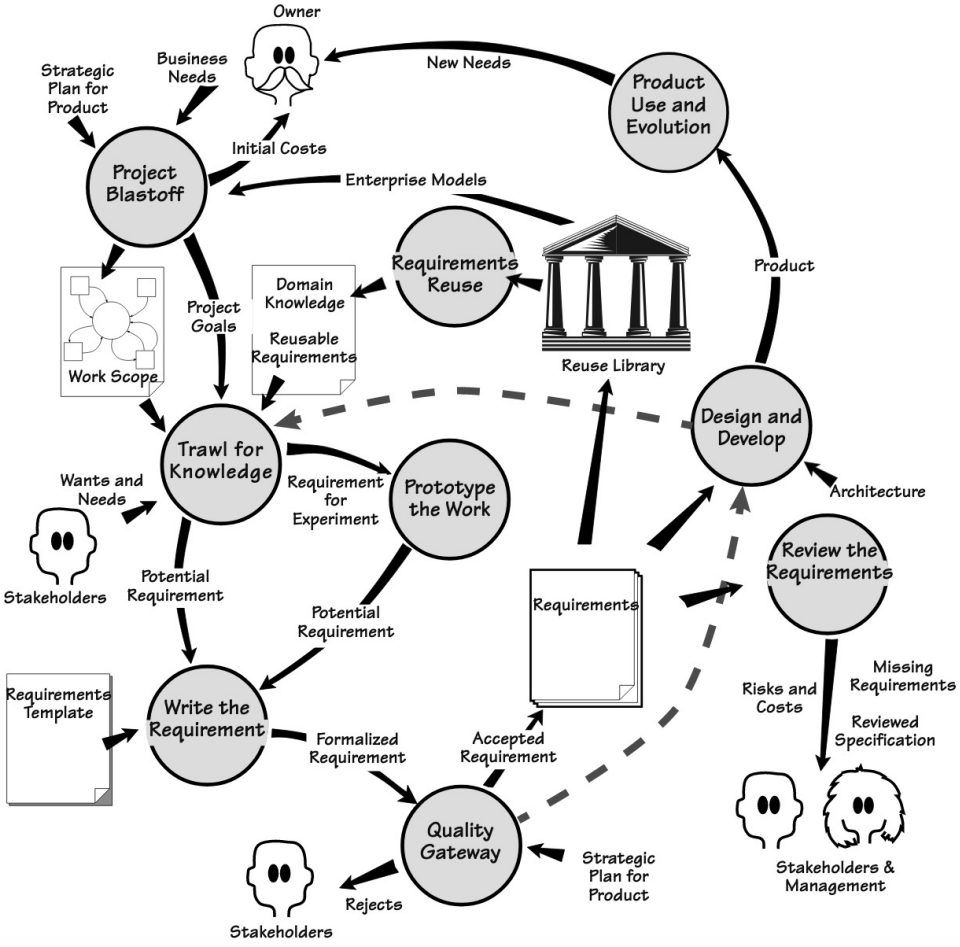
Requirements are the foundation for the IoT development cycle and reflect the intended use of the software and will be translated to specifications that will guide design, development and maintenance/deployment decisions at a later stage.

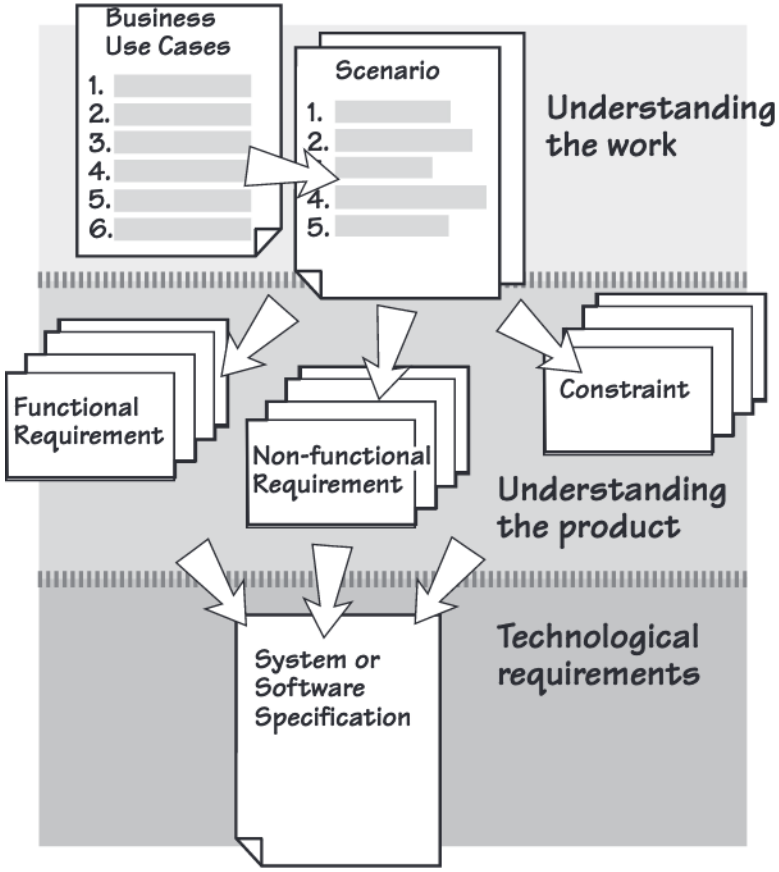
Security has to be considered from this first phase of software development, to ensure that security does not come as an afterthought.

- During the requirements phase, it is essential to conduct a preliminary identification of software security aspects taking into account the aforementioned requirements.
- The security requirements phase 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 offered functionalities (business or use cases).

::.. Security Requirements (2/4)

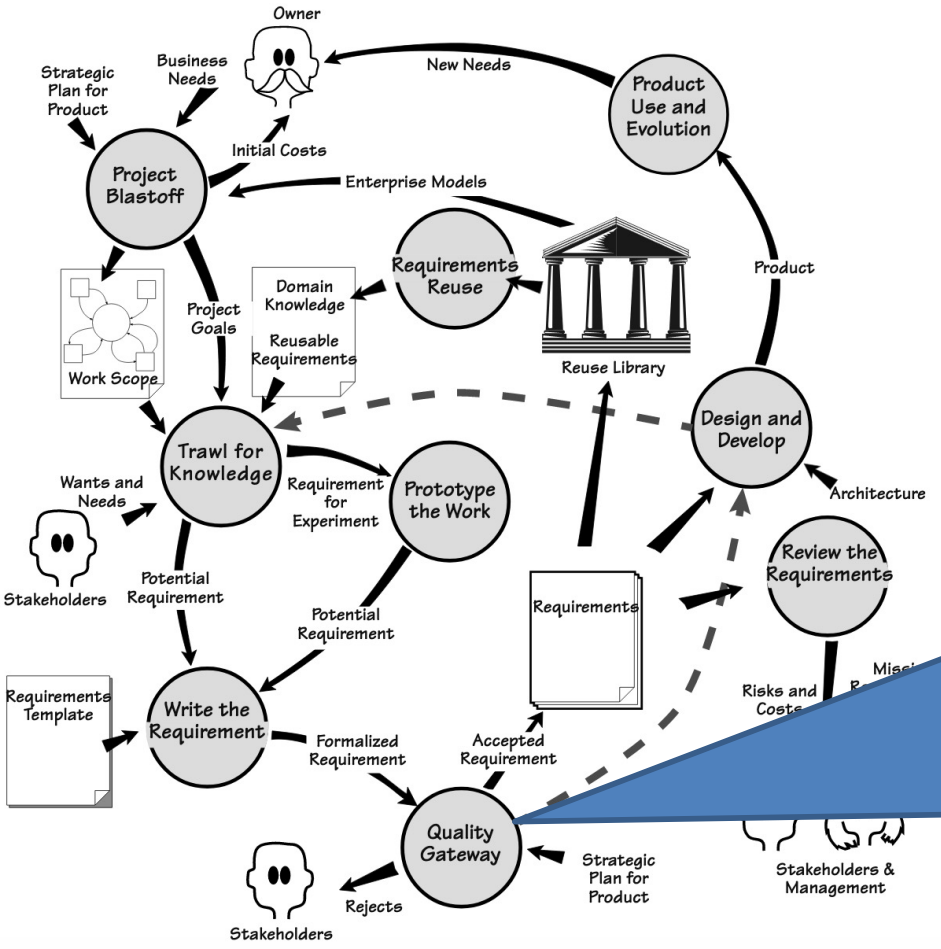
- Since software and hardware are closely related, the security software requirements may have certain implications when selecting the physical media (hardware).



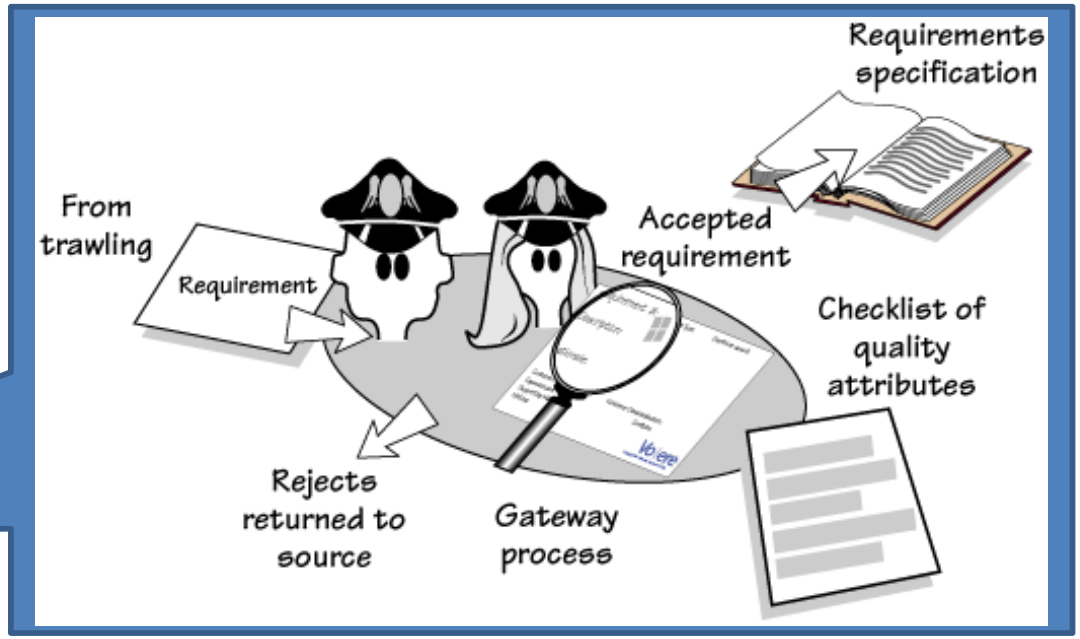


::.. Security Requirements (2/4)

- Since software and hardware are closely related, the security software requirements may have certain implications when selecting the physical media (hardware).

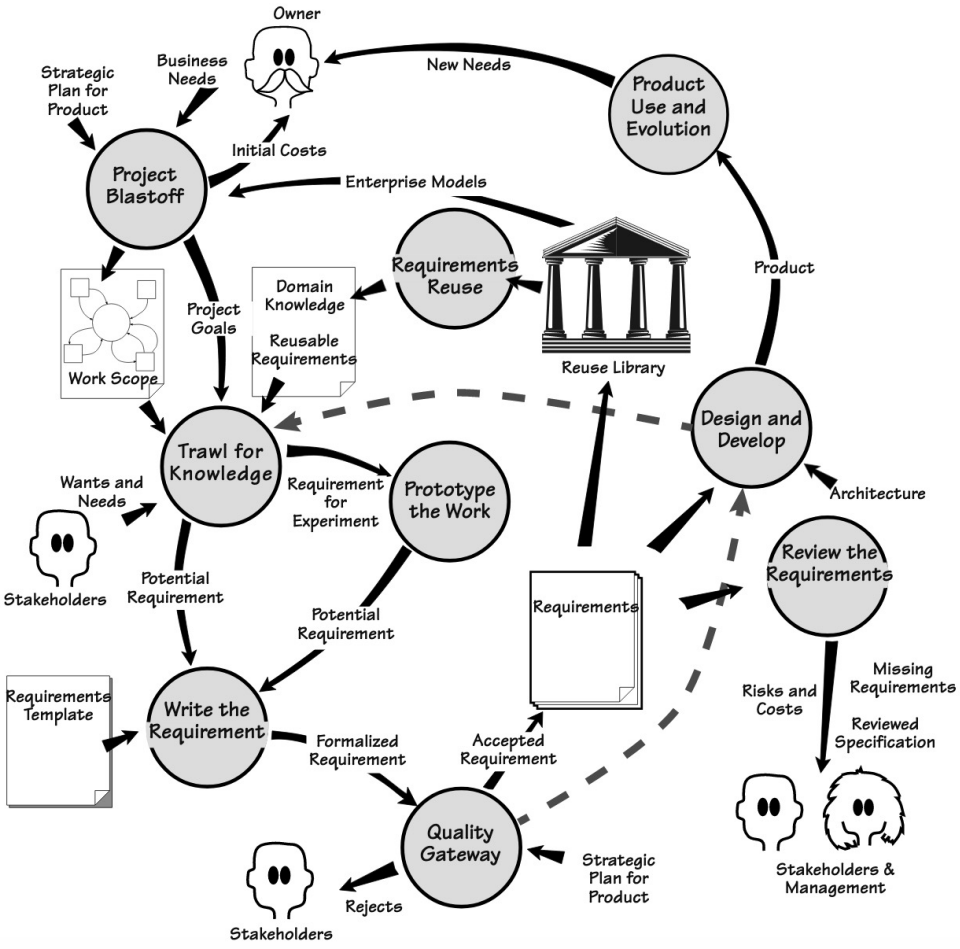


To ensure requirement consistency, quality gateways receive requirements as input and check them.



::.. Security Requirements (2/4)

- Since software and hardware are closely related, the security software requirements may have certain implications when selecting the physical media (hardware).



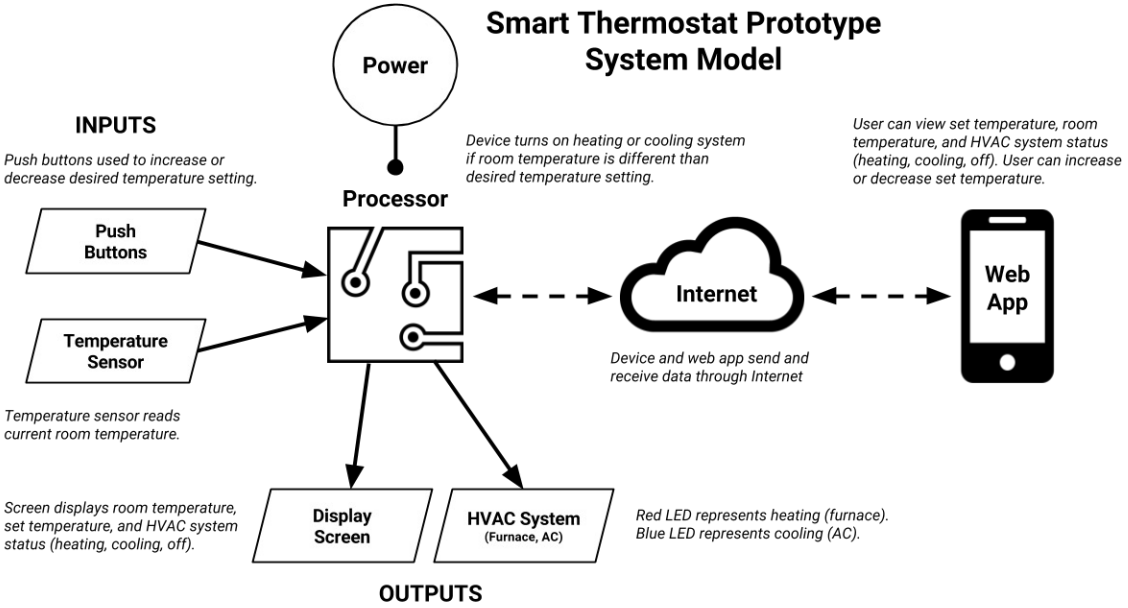
To ensure requirement consistency, quality gateways receive requirements as input and check them.

Risk analysis, as well as making use of best practices documents (e.g. OWASP IoT Top10) and guidelines help to secure software development by means of predefined checklists of most common security risks and pitfalls.

::.. Security Requirements (3/4)

With IoT, the digital and physical worlds are no longer kept apart from one another, and this physical nature introduces additional parameters in the threat modelling equation, and industry-specific threats, such as interoperability with legacy-coded and outdated devices.

- A household thermostat is a good example of a modern IoT device that is connected to the internet and can be controlled from a remote entity.



- Security in this instance is extremely important as the thermostat has access to a private network and control over a potentially dangerous heating system.

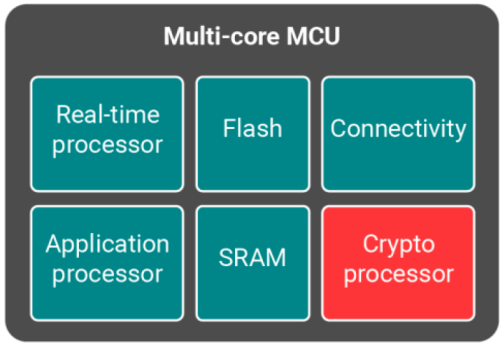
::.. Security Requirements (4/4)

- Secure communication is important for this type of device. In order to implement this, the device must be designed such that it has an identity:
  - It is needed so that the thermostat can be authenticated and can communicate with other systems in a secure manner.
- The thermostat is required to be in service for many years. It is unlikely that no vulnerabilities will be found during the lifecycle of the device. Such a device has to include the capability to allow software updates over the private network:
  - This requires that the device includes a Root of Trust in order to authenticate any software update and ensure that it has originated from a trusted entity.
  - An update policy is also required to protect the device against software rollback attacks, which can occur if older software versions contain a security vulnerability.

::.. IoT Product Architecture (1/2)

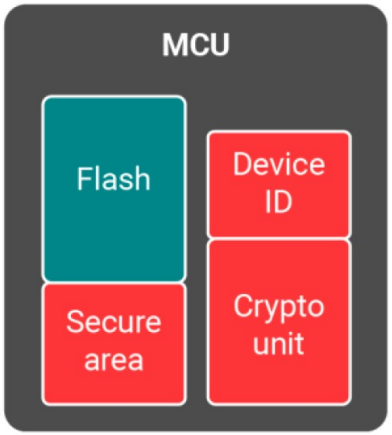
The IoT architecture is dependent on the requirements of the local functions to be processed and the type of connectivity required.

Multi-core Secure MCU



Security subsystemas part of multi-core processor

Fully Integrated Secure MCU

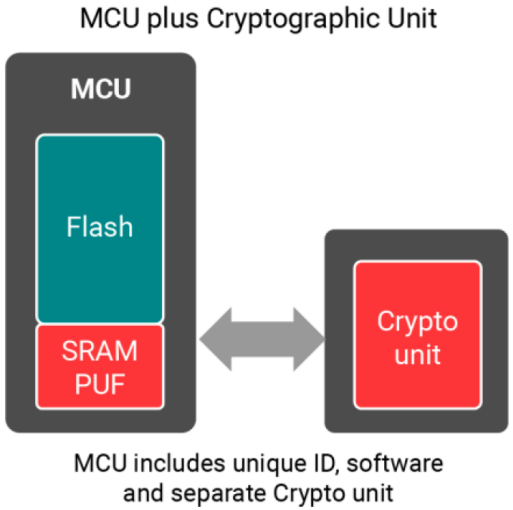


MCU includes hardware Crypto unit and memorey protection features

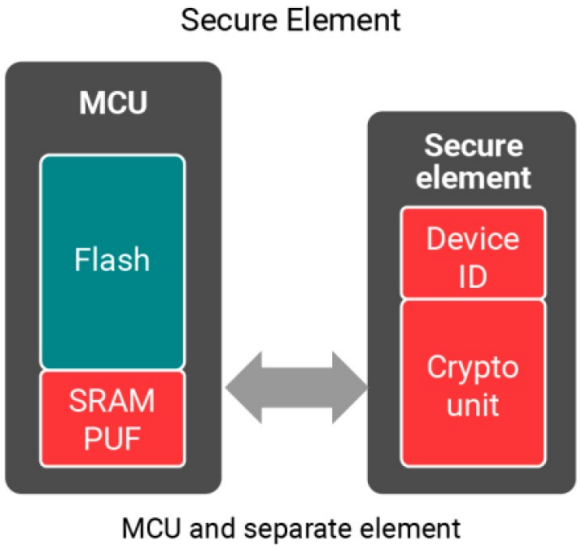
These are high-end processors that are usually associated with a dedicated software tool to allow easy integration with cloud-based applications. A security subsystem as a crypto process may be present.

This architecture is arguably the most common and cost-effective solution, as manufacturers add more security features.

::.. IoT Product Architecture (2/2)



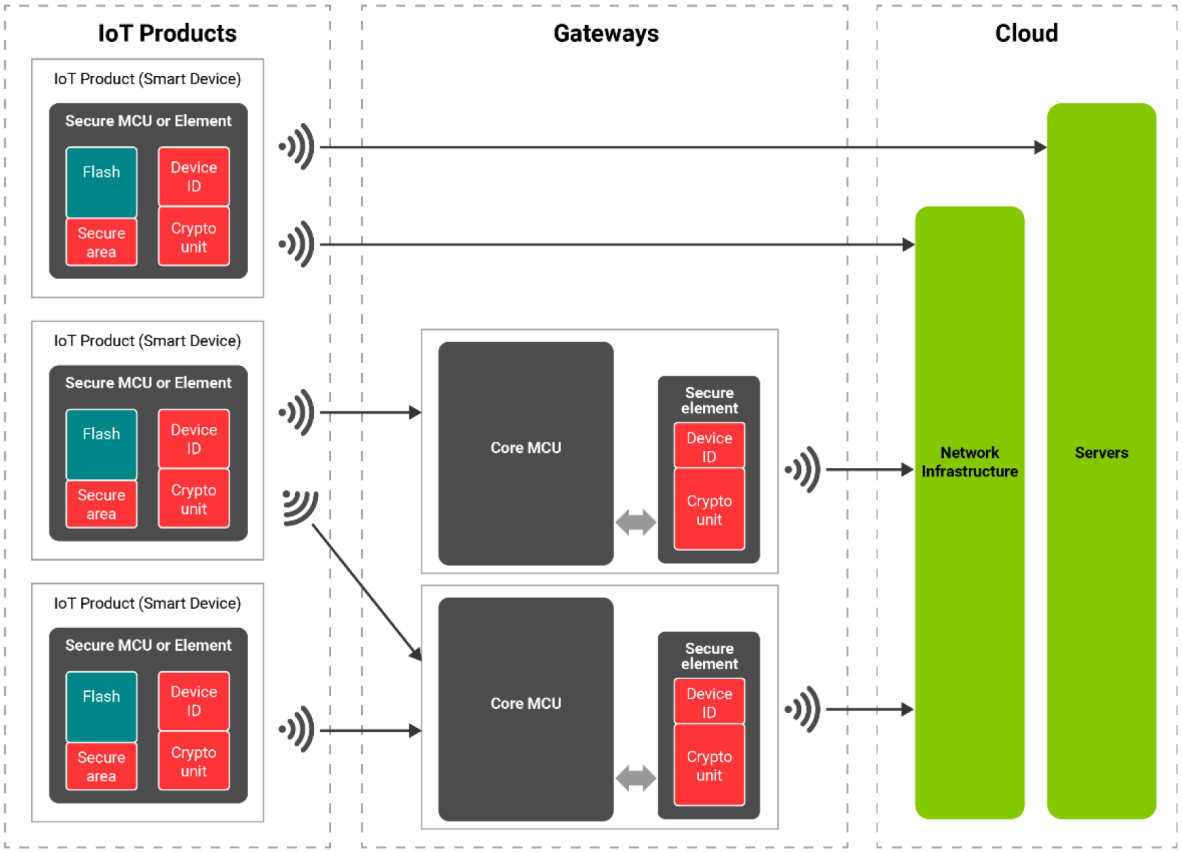
The standard MCU communicates with the dedicated cryptographic unit (CU) via an interface such as SPI or I²C. The exposed serial interface is often secured using the dedicated public key available from the CU. The CUs often include extensive hardware security features which are not currently available in many secure MCU devices.



Secure elements (SE) have hardware security measures integrated into the semiconductor devices providing features such as tamper detection, internal metal shields, encrypted memories and fully secure production test methodologies.

::.. IoT Connectivity

Connectivity requirements also influence the architecture. Communication protocols such as DTLS can be supported by some SEs allowing offloading of the MCU software.



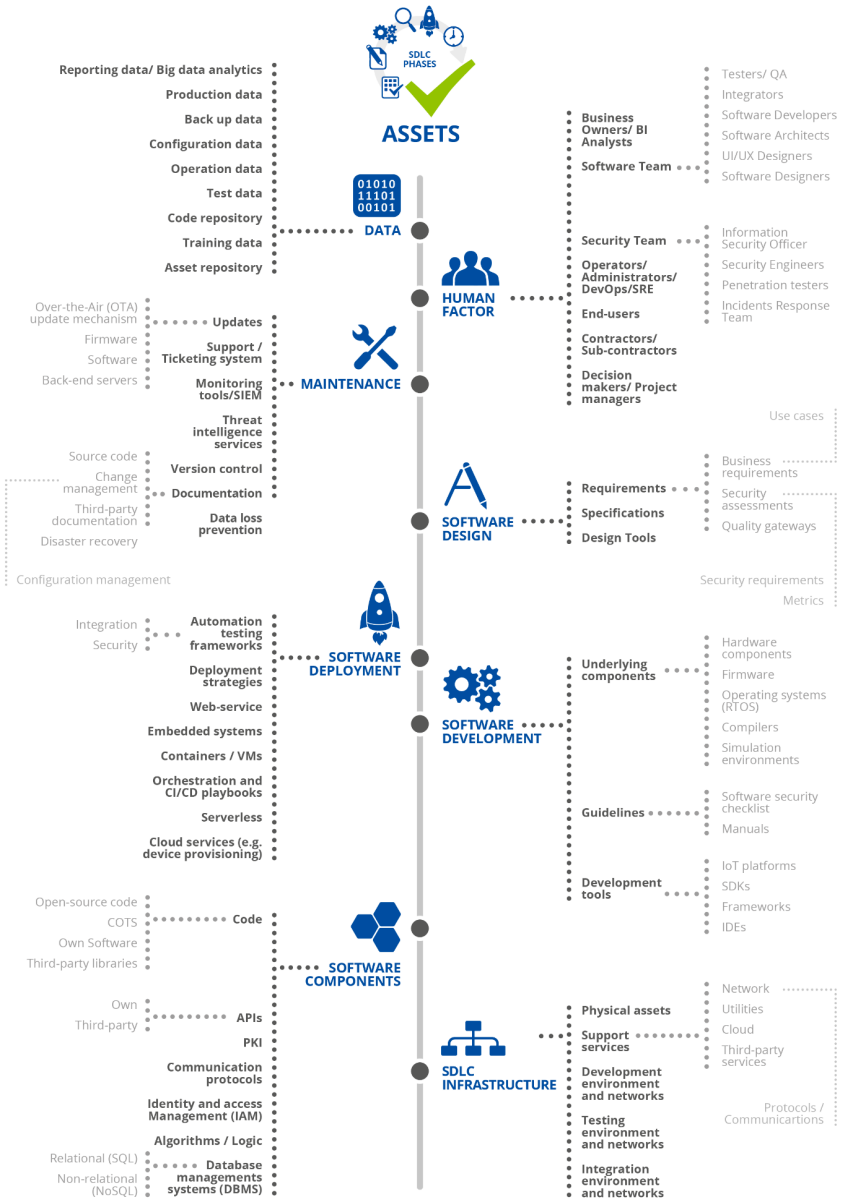
::.. Security Threat Analysis (1/6)

Once the IoT product architecture has been considered with regards to connectivity and hardware, it is important to then consider the possible threats to the IoT product.

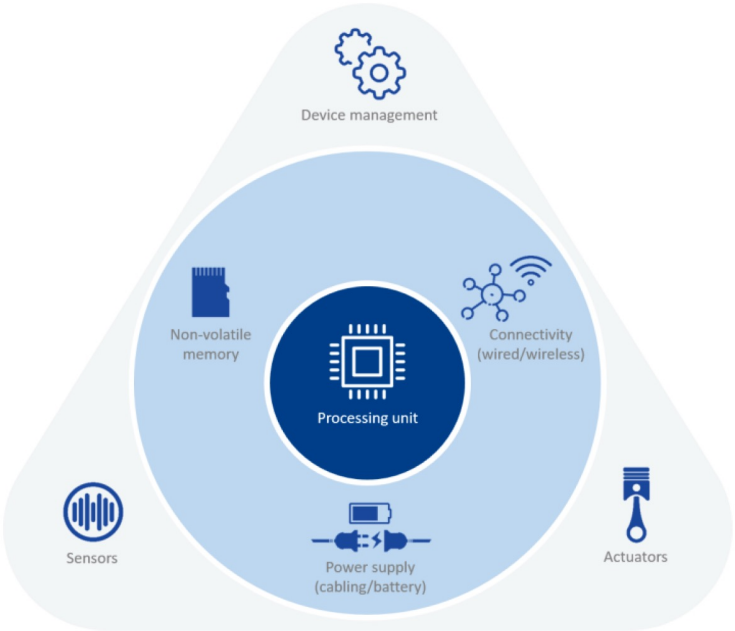
- A threat analysis is part of the development process and considers the attack vectors that may be available to bad actors who are intent on intellectual property theft.
- The threat analysis and remaining development process will assist with the choice of hardware elements to be used in the design.
- Also, it is important to understand the support from tools and infrastructure available which will allow the IoT product to be securely manufactured.

- Location: Is the IoT product easily accessible?
- Access (physical): Is the IoT product circuit board exposed?
- Access (electrical): Does the product have multiple communications ports?

::.. Security Threat Analysis (2/6)

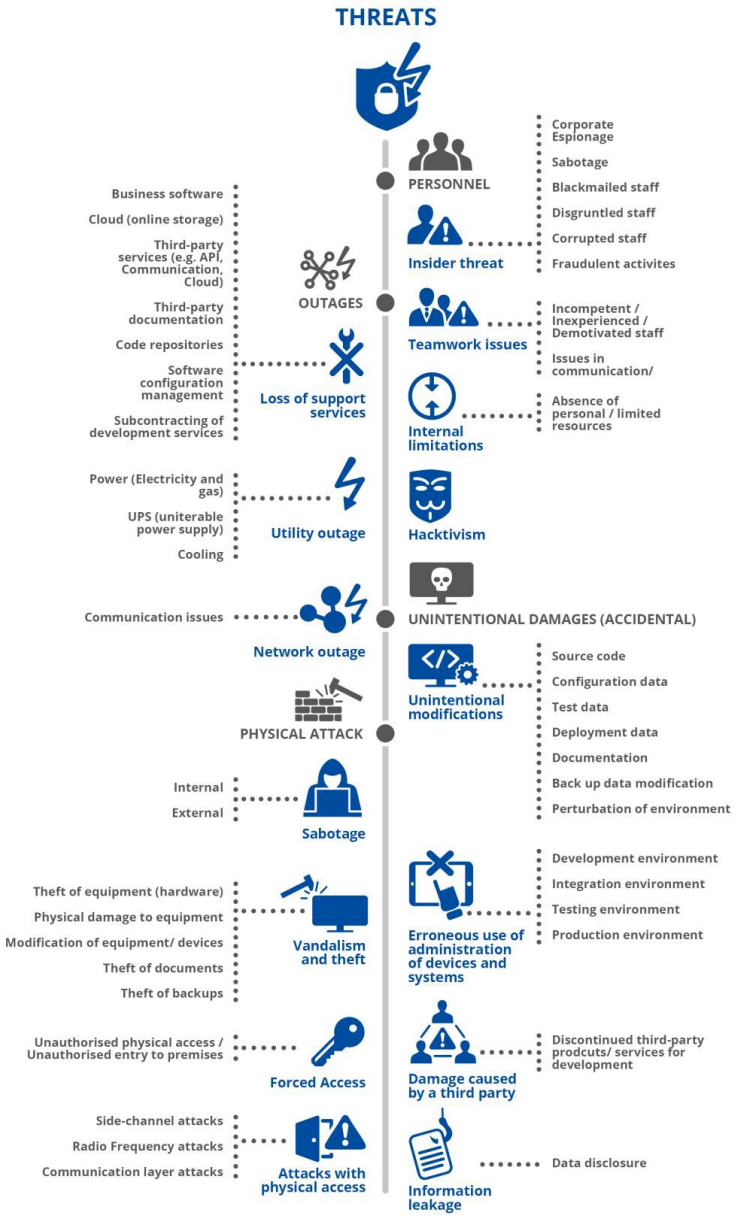


It is essential to start from identification and decomposition of assets of such vast and complex environments focusing on software development.



SESSION		AMQP, CoAP, DDS, MQTT, XMPP
NETWORK	ENCAPSULATION	6LowPAN, Thread
	ROUTING	CARP, RPL
DATALINK		Bluetooth / BLE, Wi-Fi / Wi-Fi HaLow, LoRaWAN, Neul, SigFox, Z-Wave, ZigBee, USB

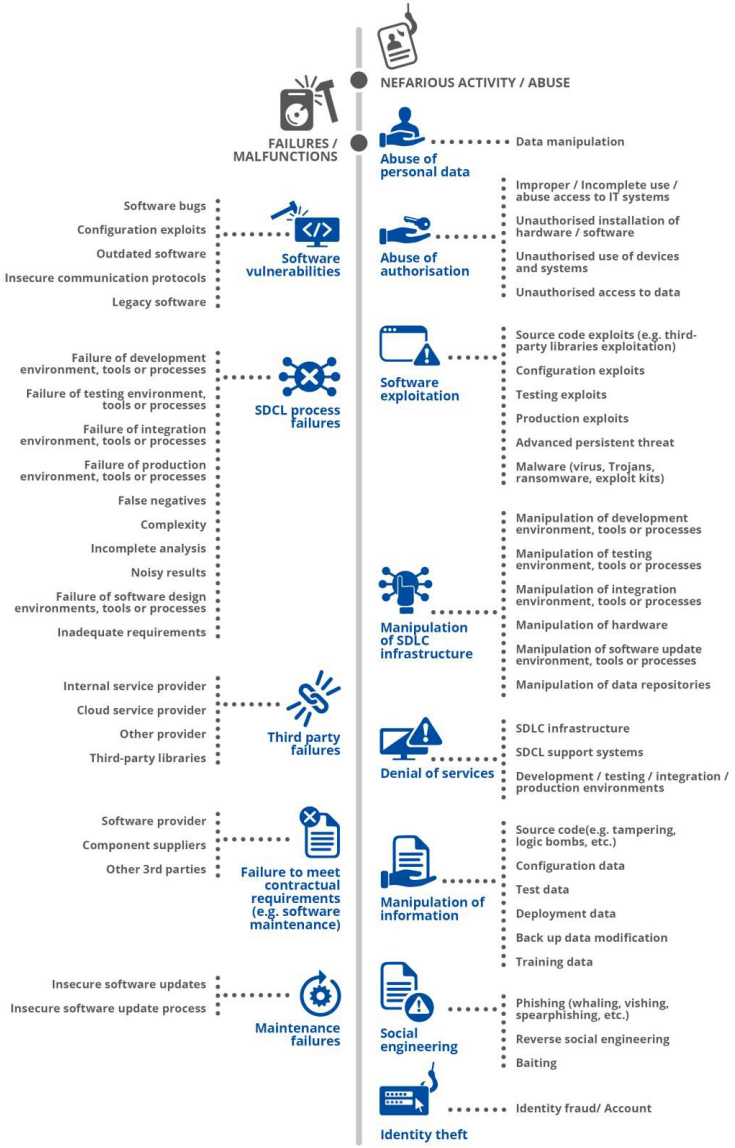
::.. Security Threat Analysis (2/6)



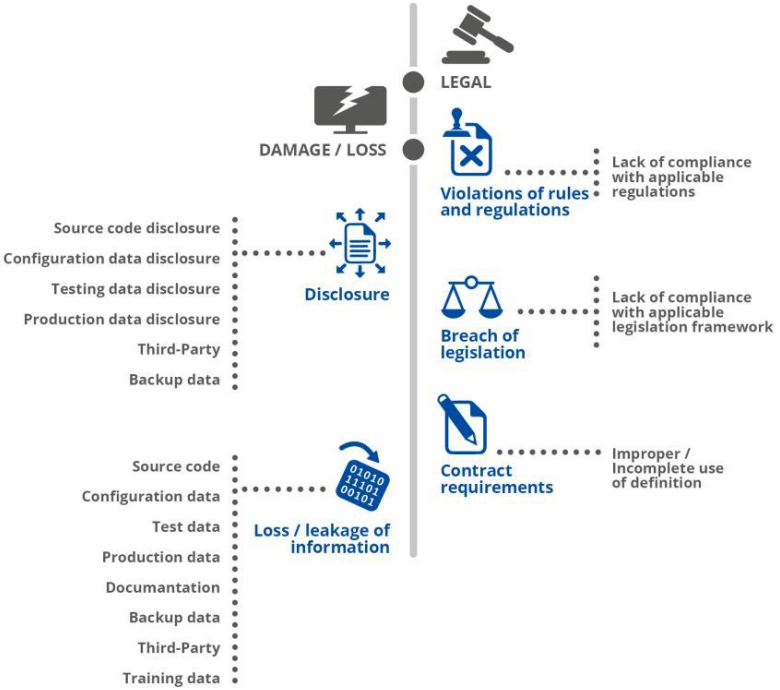
It is essential to start from identification and decomposition of assets of such vast and complex environments focusing on software development.

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.

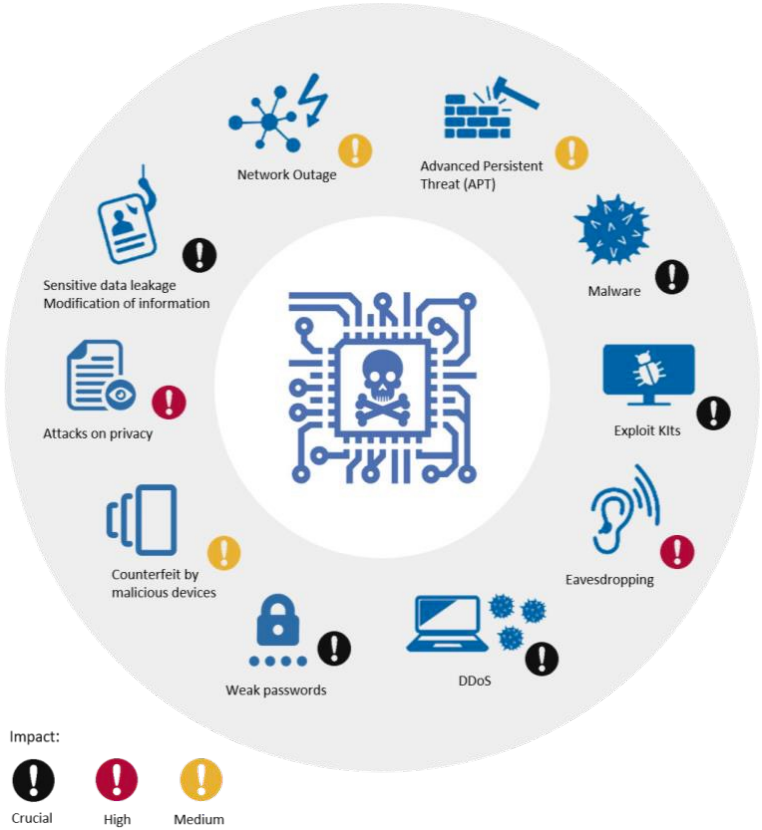
::.. Security Threat Analysis (2/6)



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.



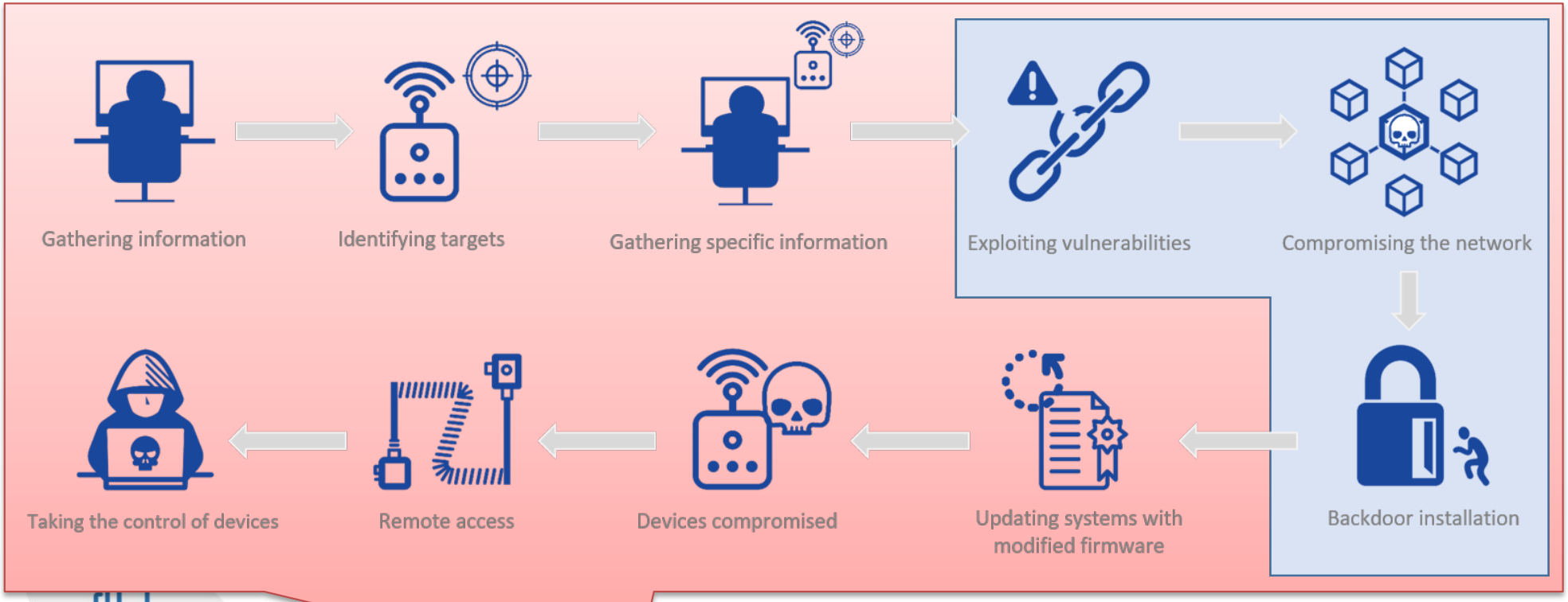
::.. Security Threat Analysis (3/6)

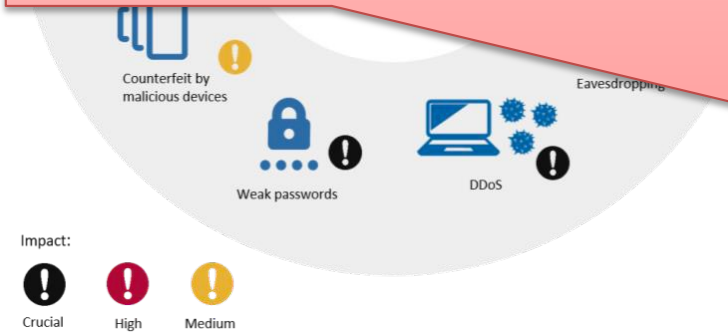


The different threats have different potential impacts, since they vary according to the use case scenarios. In the interviews, the IoT experts provided insight into the varying impact of the threats.

The threats and risks could be used by attackers to cause cascade effects and further damages at different levels in the infrastructures.

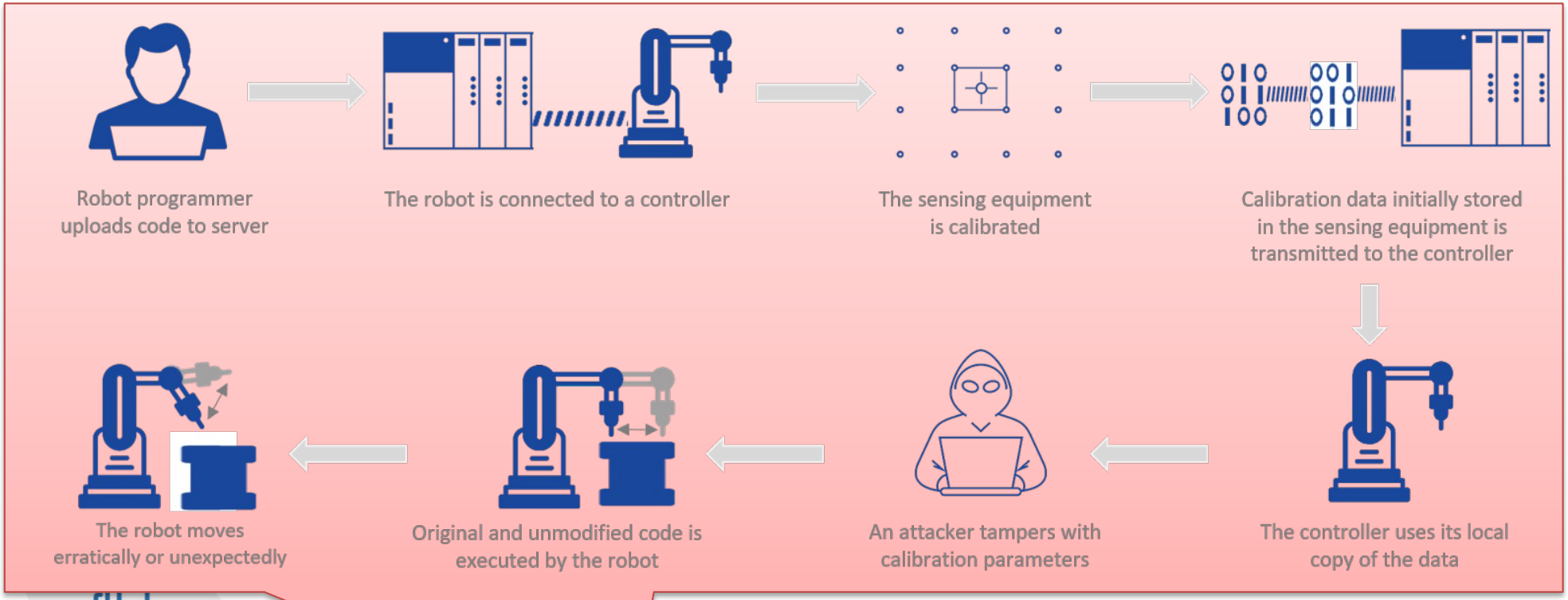
ATTACK SCENARIOS	IMPORTANCE LEVEL
1. Against the network link between controller(s) and actuators	High – Crucial
2. Against sensors, modifying the values read by them or their threshold values and settings	High – Crucial
3. Against actuators, modifying or sabotaging their normal settings	High – Crucial
4. Against the administration systems of IoT	High – Crucial
5. Exploiting protocol vulnerabilities	High
6. Against devices, injecting commands into the system console	High – Crucial
7. Stepping stones attacks	Medium – High
8. DDoS using an IoT botnet	Crucial
9. Power source manipulation and exploitation of vulnerabilities in data readings	Medium – High
10. Ransomware	Medium – Crucial[70]

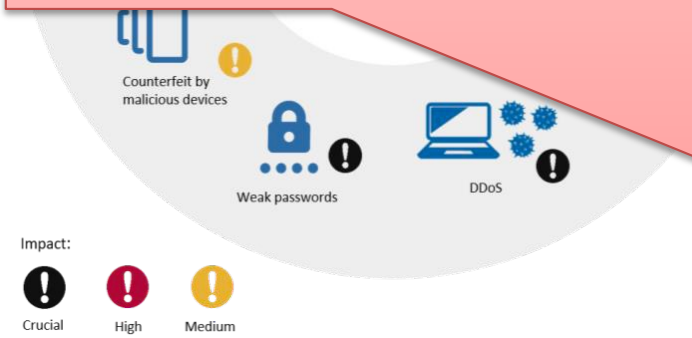




The threats and risks could be used by attackers to cause cascade effects and further damages at different levels in the infrastructures.

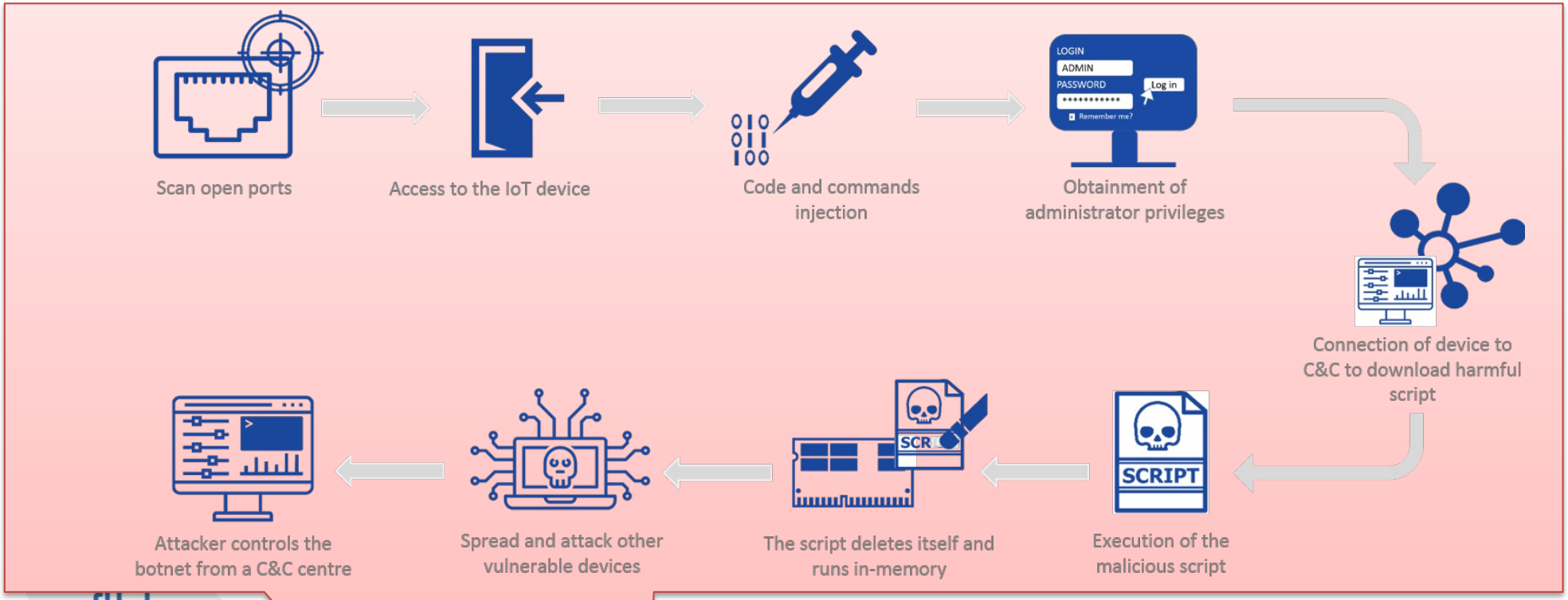
ATTACK SCENARIOS	IMPORTANCE LEVEL
1. Against the network link between controller(s) and actuators	High – Crucial
2. Against sensors, modifying the values read by them or their threshold values and settings	High – Crucial
3. Against actuators, modifying or sabotaging their normal settings	High – Crucial
4. Against the administration systems of IoT	High – Crucial
5. Exploiting protocol vulnerabilities	High
6. Against devices, injecting commands into the system console	High – Crucial
7. Stepping stones attacks	Medium – High
8. DDoS using an IoT botnet	Crucial
9. Power source manipulation and exploitation of vulnerabilities in data readings	Medium – High
10. Ransomware	Medium – Crucial[70]

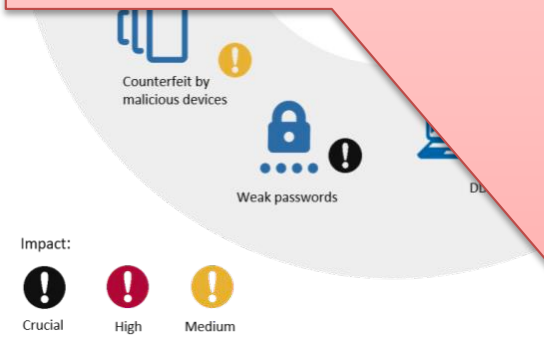




The threats and risks could be used by attackers to cause cascade effects and further damages at different levels in the infrastructures.

TTACK SCENARIOS	IMPORTANCE LEVEL
L. Against the network link between controller(s) and actuators	High – Crucial
2. Against sensors, modifying the values read by them or their threshold values and settings	High – Crucial
3. Against actuators, modifying or sabotaging their normal settings	High – Crucial
4. Against the administration systems of IoT	High – Crucial
5. Exploiting protocol vulnerabilities	High
6. Against devices, injecting commands into the system console	High – Crucial
7. Stepping stones attacks	Medium – High
8. DDoS using an IoT botnet	Crucial
9. Power source manipulation and exploitation of vulnerabilities in data readings	Medium – High
10. Ransomware	Medium – Crucial[70]





The threats and risks co… be used by attackers to… cause cascade effects and further damages at different levels in the infrastructures.

TACK SCENARIOS	IMPORTANCE LEVEL
Against the network link between controller(s) and actuators	High – Crucial
Against sensors, modifying the values read by them or their threshold values and settings	High – Crucial
. Against actuators, modifying or sabotaging their normal settings	High – Crucial
. Against the administration systems of IoT	High – Crucial
5. Exploiting protocol vulnerabilities	High
6. Against devices, injecting commands into the system console	High – Crucial
7. Stepping stones attacks	Medium – High
8. DDoS using an IoT botnet	Crucial
9. Power source manipulation and exploitation of vulnerabilities in data readings	Medium – High
10. Ransomware	Medium – Crucial[70]

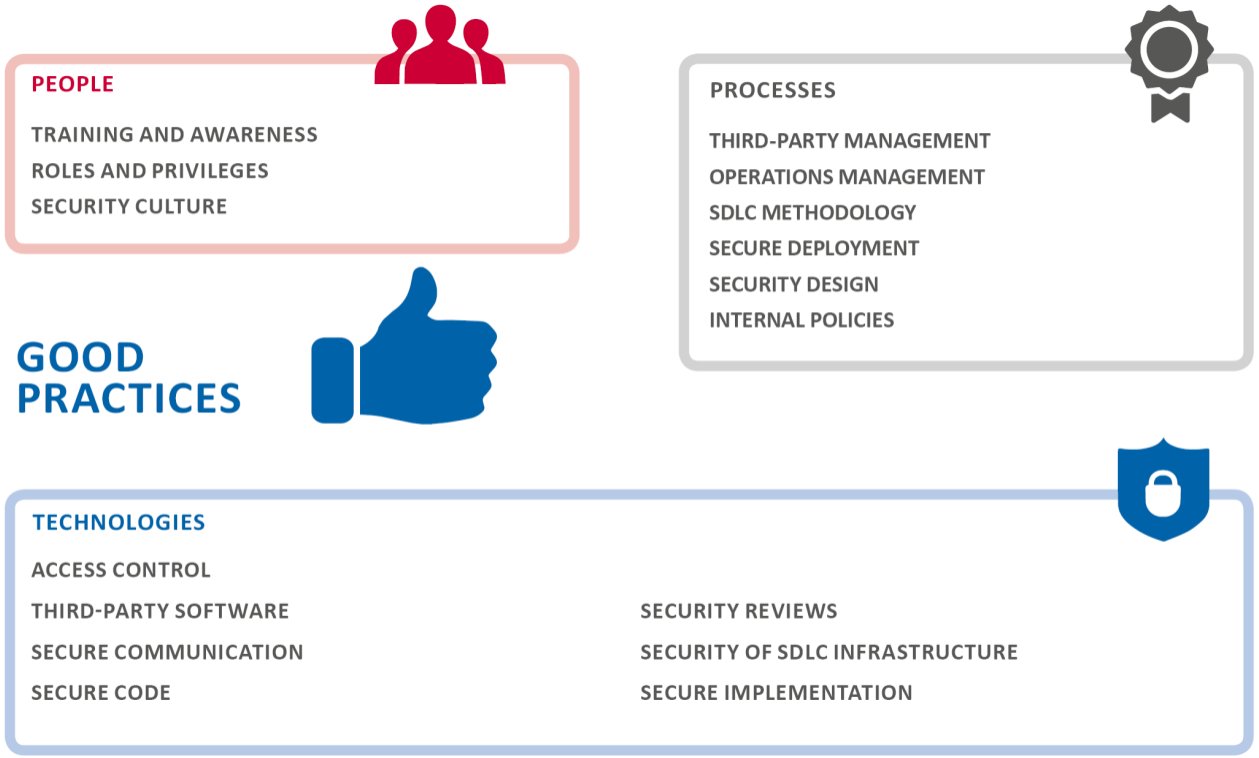
::.. Security Threat Analysis (4/6)

A guide developed at Microsoft for threat analysis uses the acronym ‘STRIDE’ to describe six categories of threats to software.

- Spoofing: using someone else’s credentials (forged or stolen) to gain access to otherwise inaccessible assets.
- Tampering: is changing data to mount an attack. In this situation, a malicious user could alter the files, thus breaching system security.
- Repudiation: occurs when a user denies performing an action, but the target of the action has no way to prove otherwise.
- Information disclosure: the disclosure of information to a user who does not have permission to see it.
- Denial of service: attacks threaten the ability of valid users to access resources
- Elevation of privilege: an attack that can occur if an unprivileged user gains privileged status.

::.. Security Threat Analysis (5/6)

- 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.

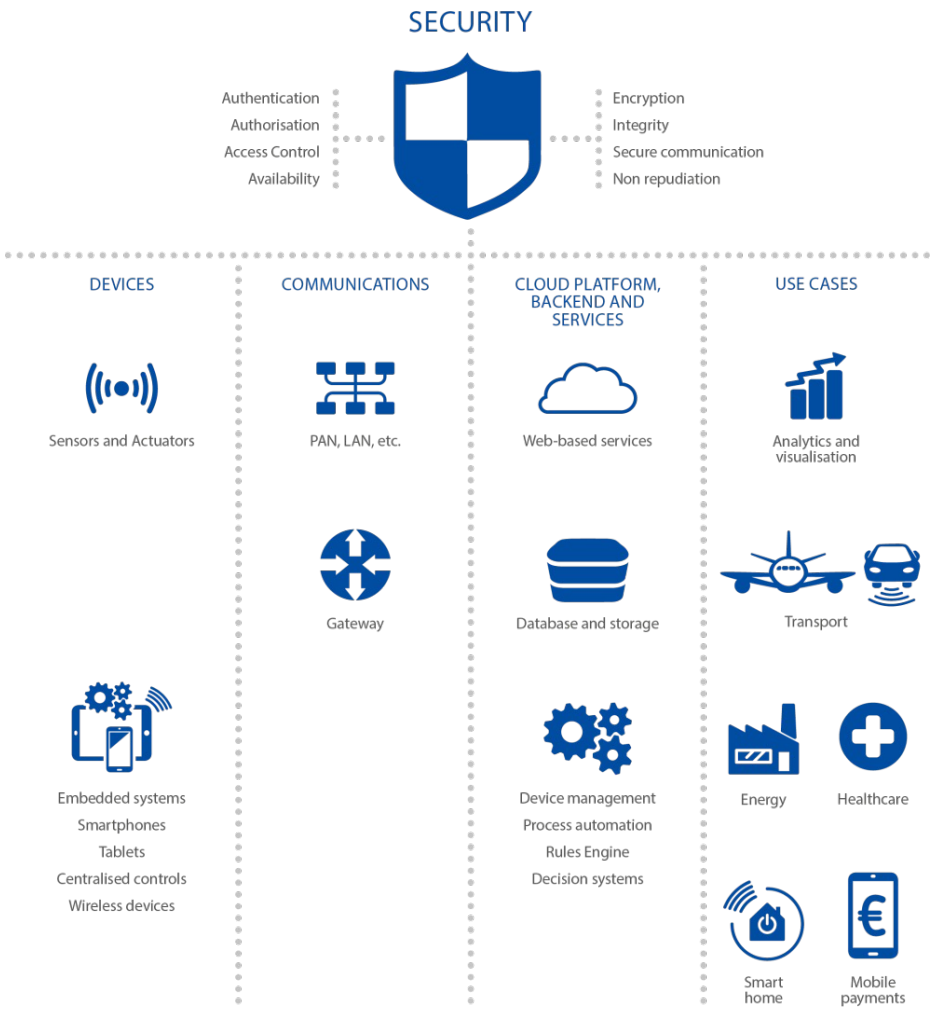


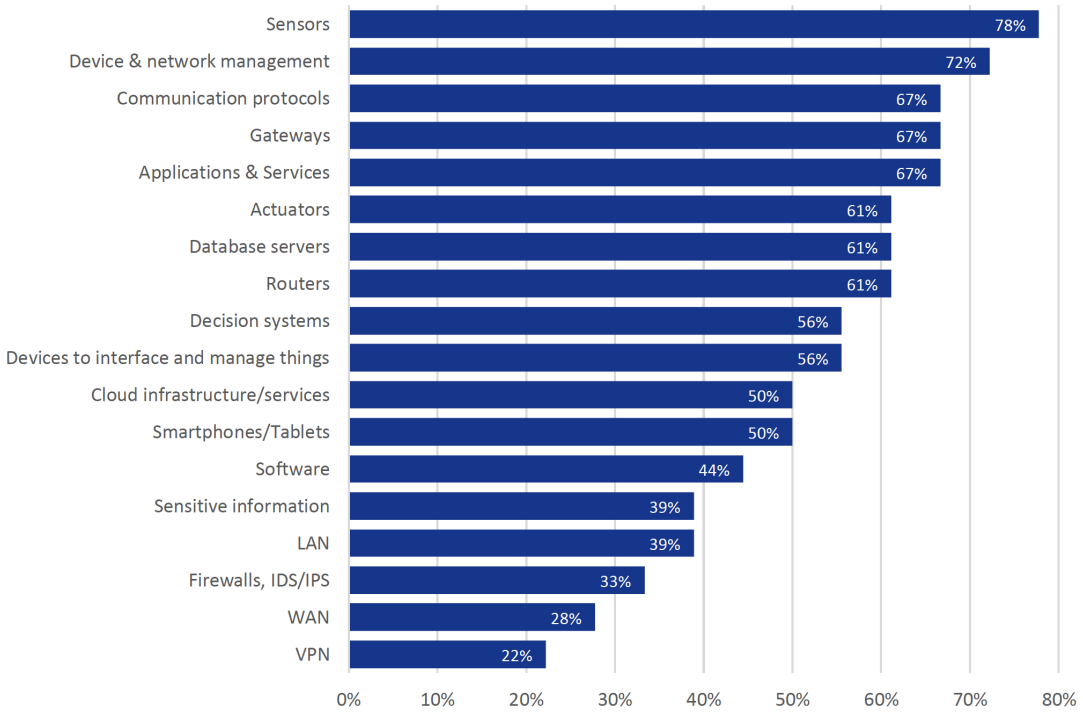
- Technologies: technical measures and elements used in order to reduce vulnerabilities and flaws during the software development process.

::.. Security Threat Analysis (6/6)

The different elements that compose the IoT high-level reference model needs to be equipped with some of the security mechanisms .

Each element is characterised by a different criticality degree.





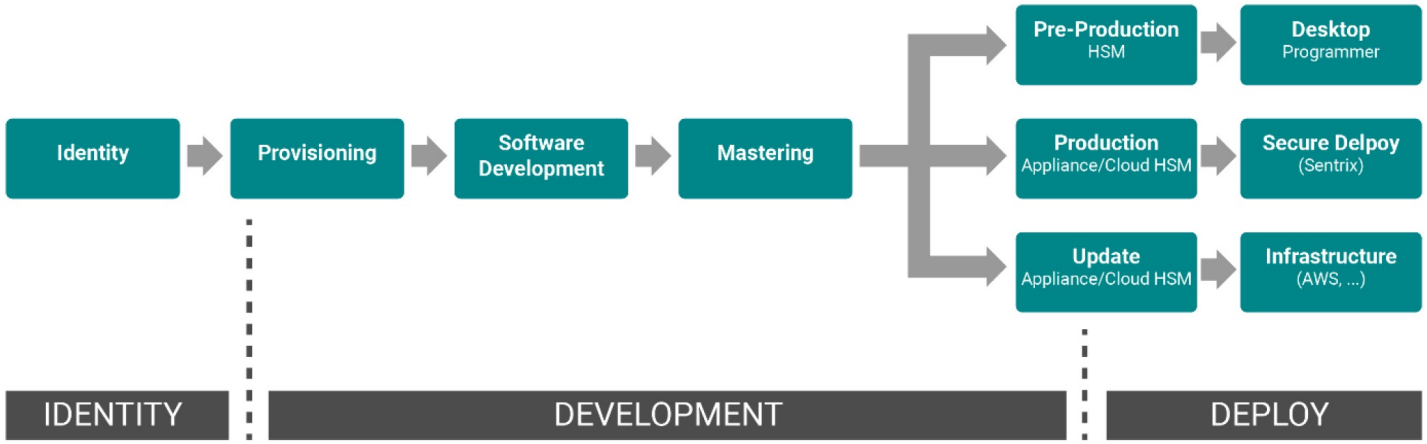
::.. Development Strategy (1/2)

Once security requirements are defined and security threats have been identified, it is important that the IoT product developer creates a security implementation plan that will identify the specific critical security credentials for the IoT product as well as other critical security items like the development process, manufacturing process and life cycle process.

There are three critical areas that must be understood :

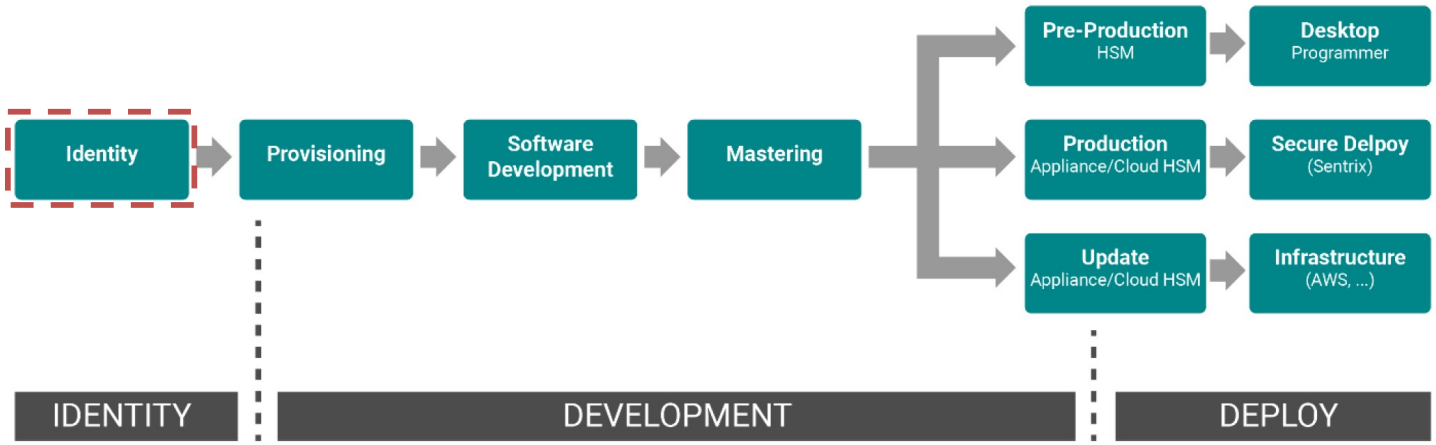
1. What are the trust anchors during software development?
   - How is the product identity formed? How is the Root of Trust established? How are certificates created?
2. How is a secure manufacturing / programming process to be realised?
   - Security compromised during provisioning? Keys leaked? Product identities cloned? Software IP theft?
3. How will security continue to be enforced after the product has been manufactured?
   - Secure software updates? System compromised? Customer data protected?

::.. Development Strategy (2/2)



The key stages of the product development flow are

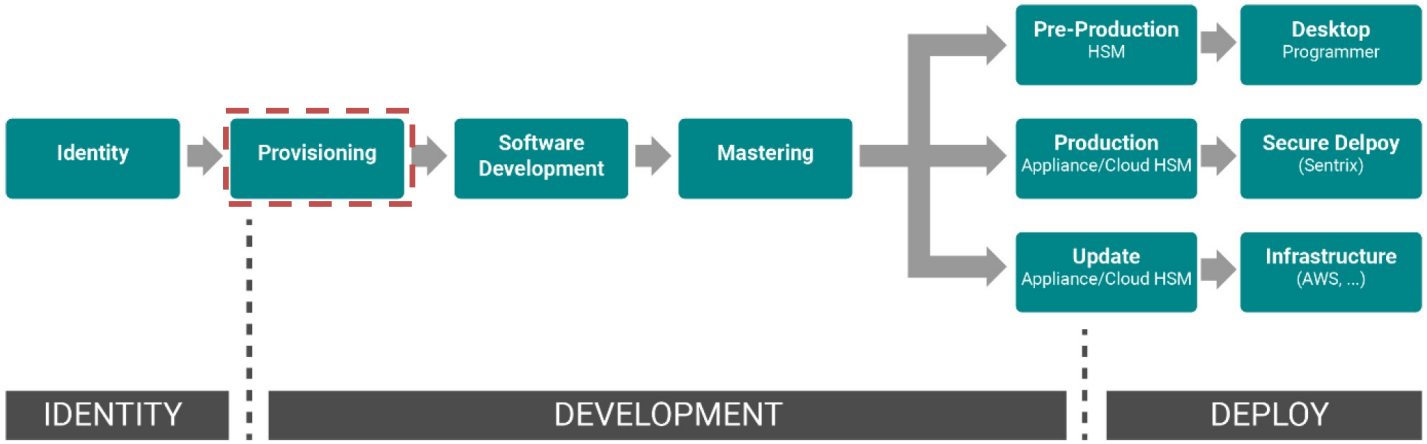
::.. Development Strategy (2/2)



The key stages of the product development flow are

- Identity – with a unique strong device identity, an IoT product can be authenticated when it tries to connect and come online, and it can ensure secure and encrypted communication between other devices, services and users. The identity (device private key, device certificate, silicon ID etc) is the critical element in ensuring that a Public Key Infrastructure (PKI) can be implemented.

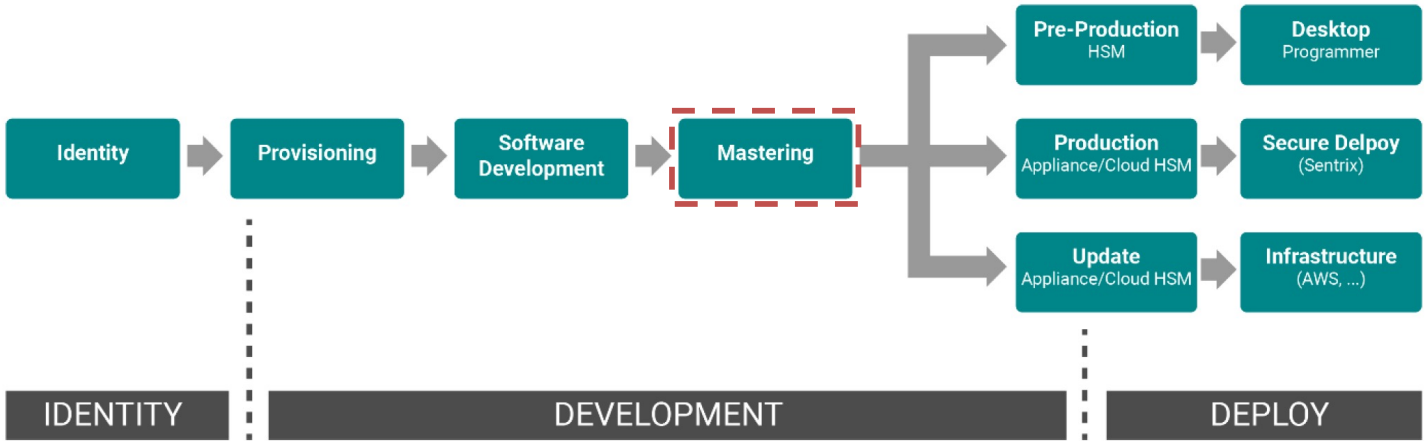
::.. Development Strategy (2/2)



The key stages of the product development flow are

- Provisioning. In the context of a SE or secure MCU, it means to program a device with the identity/root of trust anchors so that it can be handed-off to an end user or application, for a specific use in a functional manner. More specifically, for an IoT product, the device is linked to a certificate and any boot manager firmware is programmed into secure memory areas.

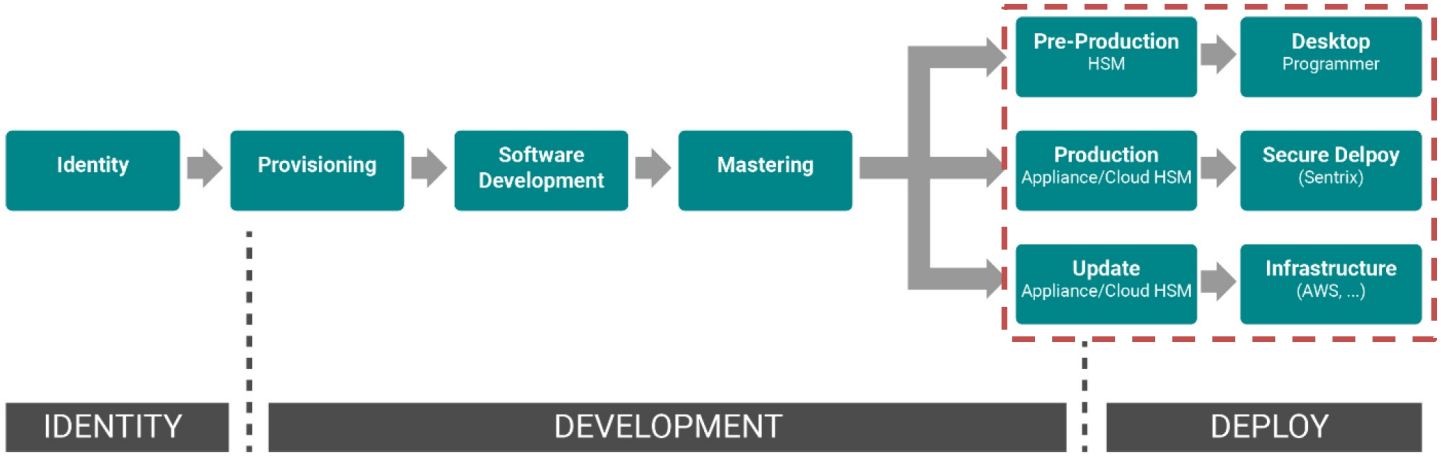
::.. Development Strategy (2/2)



The key stages of the product development flow are

- Mastering –The final software image is released, then packaged (signed and encrypted), ready to be sent to a secure programming facility. The output of this process is a mastered package which is typically comprises an encrypted/signed software image and a separate update protection package which is signed by the Original equipment manufacturer (OEM) private key. The files either form part of the initial product or as part of an in-the-field post-production update. This process is carried out at a secure facility.

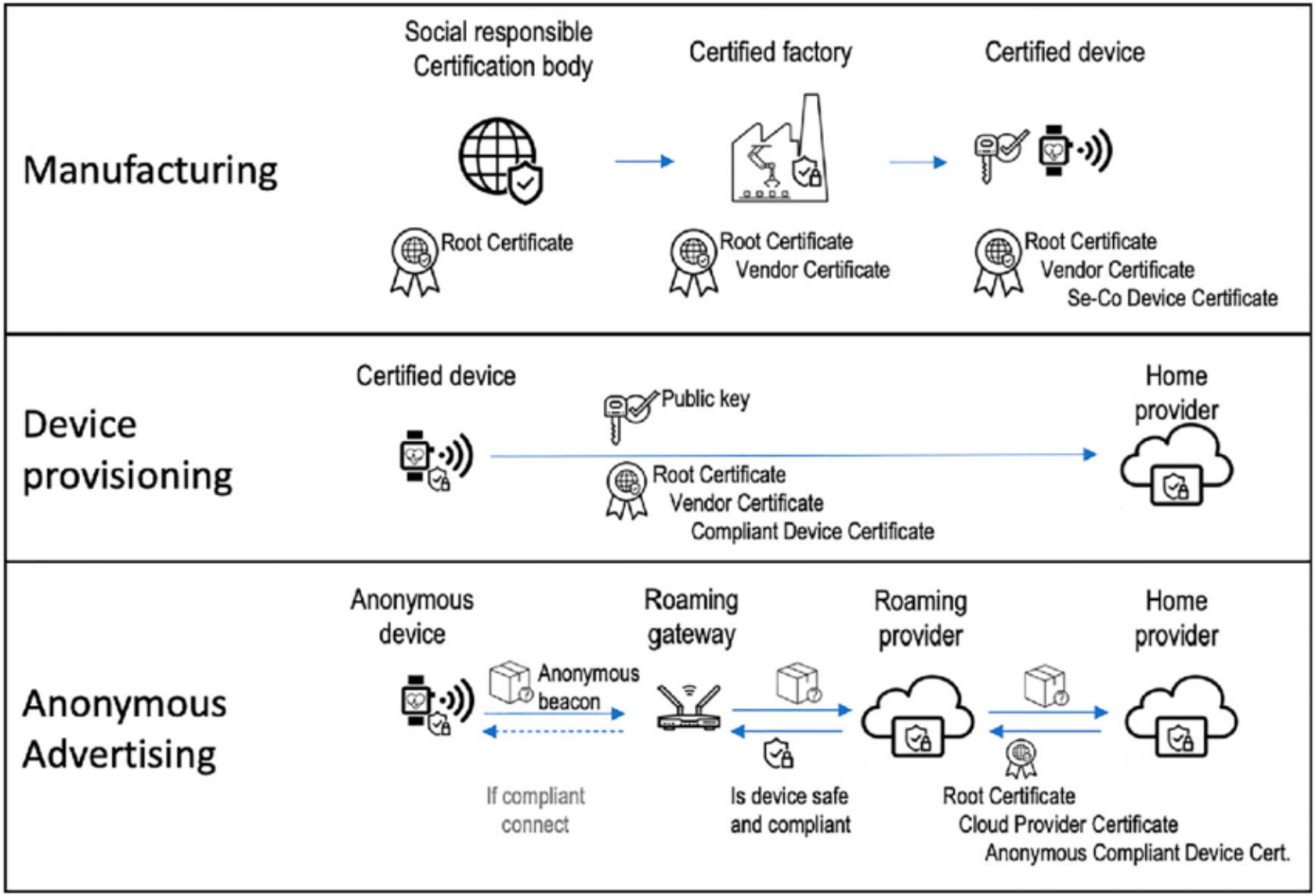
::.. Development Strategy (2/2)



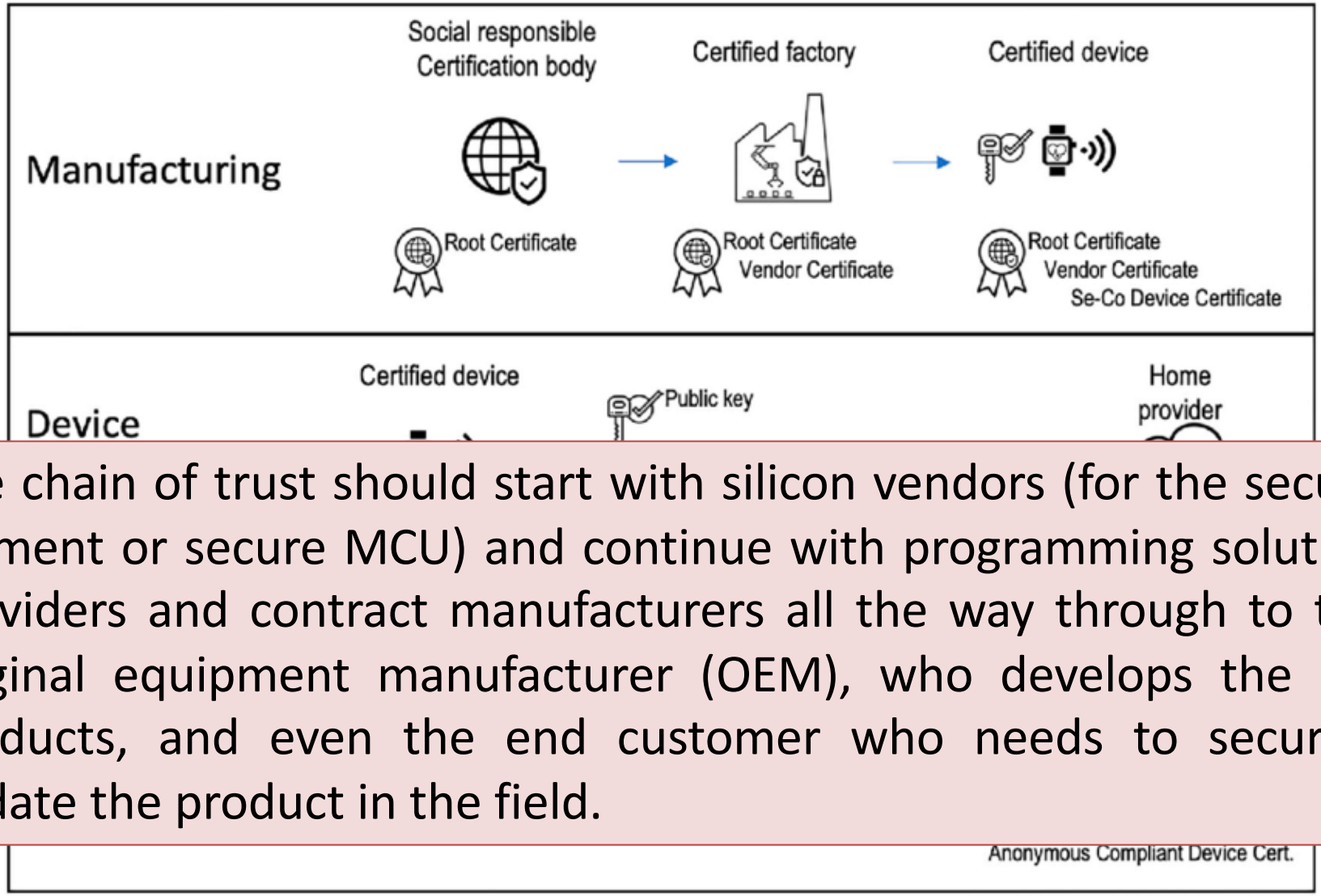
The key stages of the product development flow are

- Deployment – this is the process whereby the mastered software package is delivered to the IoT product secure device.
  - In the case of production, the mastered package is securely delivered to the programmer which injects the software into the IoT device.
  - In the case of a secure update, the mastered software package is delivered directly to the IoT device that is in the field (e.g. over-the-air or OTA update).

::.. Identity, Root of Trust, Chain of Trust



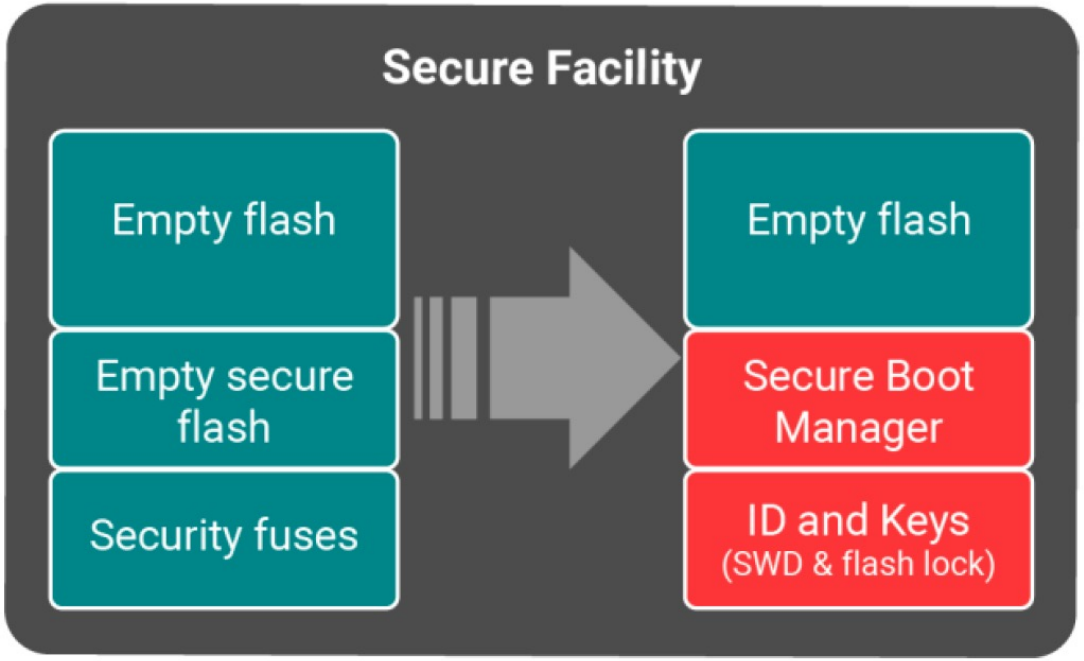
::.. Identity, Root of Trust, Chain of Trust



The chain of trust should start with silicon vendors (for the secure element or secure MCU) and continue with programming solution providers and contract manufacturers all the way through to the original equipment manufacturer (OEM), who develops the IoT products, and even the end customer who needs to securely update the product in the field.

::.. Provisioning

Provisioning is critical to creating the RoT in an IoT product: it is when the bare metal secure MCU or secure element is programmed with the required certificates / keys / data and configuration that defines the IoT product and prepares it for operation in the field. The provisioning step can also include the programming of the customer's/OEM's firmware (IP). The configuration ensures that the secure device is locked and that secrets within the device are only accessible by secure boot managers.

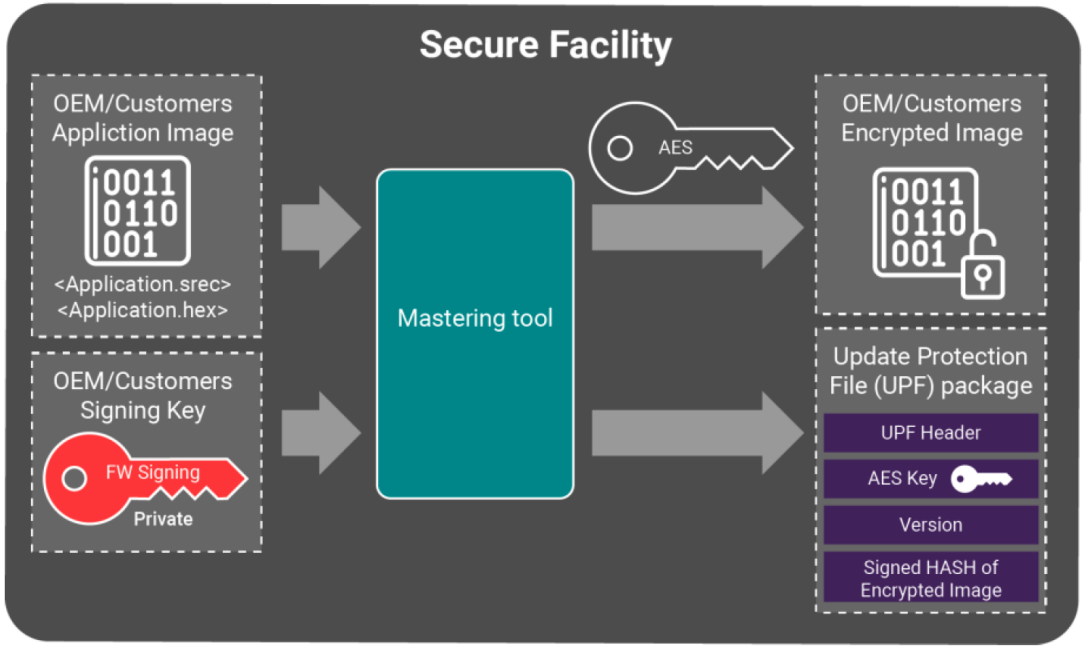


This step must be absolutely secure as provisioning generally takes place at programming centres and contract manufacturing facilities which are outside of the control of the OEM. A policy of "zero trust" is imperative to ensure the supply chain of trust

::.. Mastering

Mastering ensure that products can be securely produced and updated in the field. Ideally, the target device has been provisioned with a secure boot manager along with the OEM/customers' secret keys.

Any firmware or software to be loaded into the MCU must go through a "mastering" process where the image is signed and encrypted. This inhibits intellectual property theft and undermines counterfeiting.



The output of the mastering process is a pair of encrypted files which are loaded into a secure manufacturing process. These secure files can only be decrypted and verified by the secure boot manager that was provisioned with corresponding master keys and product keys.

::.. Secure Deployment (1/2)

Ultimately, once the development and testing of the application have been completed, it must be delivered to either:

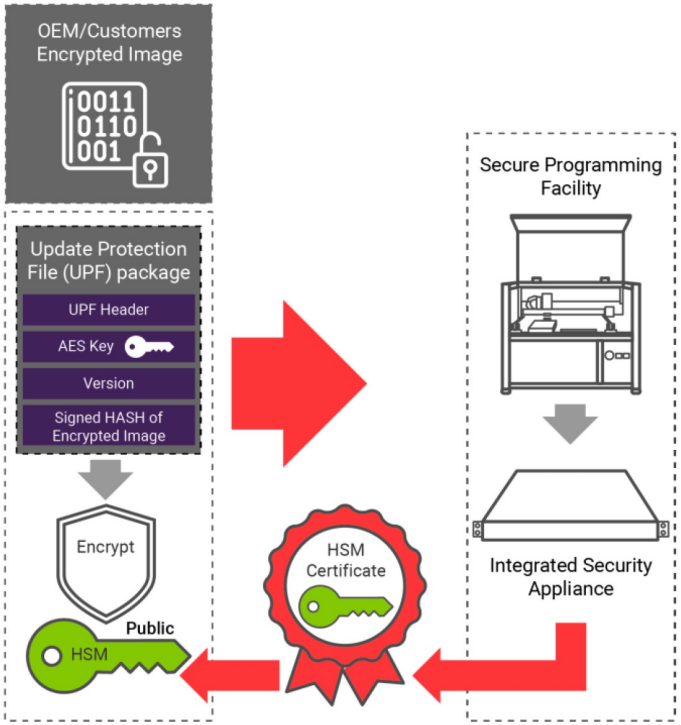
1. a programming facility to allow high volume manufacturing of the IoT product, or to
2. the IoT product that is deployed in the field and forms part of a secure software update.

It cannot be assumed that the transportation channel is secure and therefore a secure means of delivery of the software application image must be utilised as part of the deployment process by incorporating the “zero-trust” philosophy.

::.. Secure Deployment (1/2)

Ultimately, once the development and testing of the application have been completed, it must be delivered to either:

1. **a programming facility to allow high volume manufacturing of the IoT product**, or to
2. the IoT product that is deployed in the field and forms part of a secure software update.

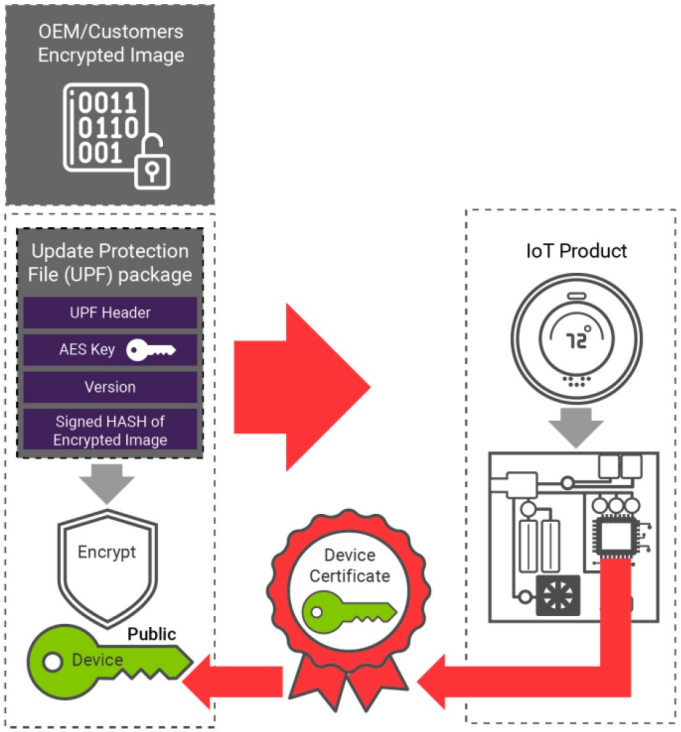


The output from the mastering process (encrypted image and UPF) is passed to the secure production facility for high volume manufacturing, by utilising a hardware security module (HSM). To transport over insecure channels, the certificate of the target should extract the public key of the HSM. The public key to encrypt the UPF. Both encrypted image and encrypted UPF can now be delivered.

::.. Secure Deployment (1/2)

Ultimately, once the development and testing of the application have been completed, it must be delivered to either:

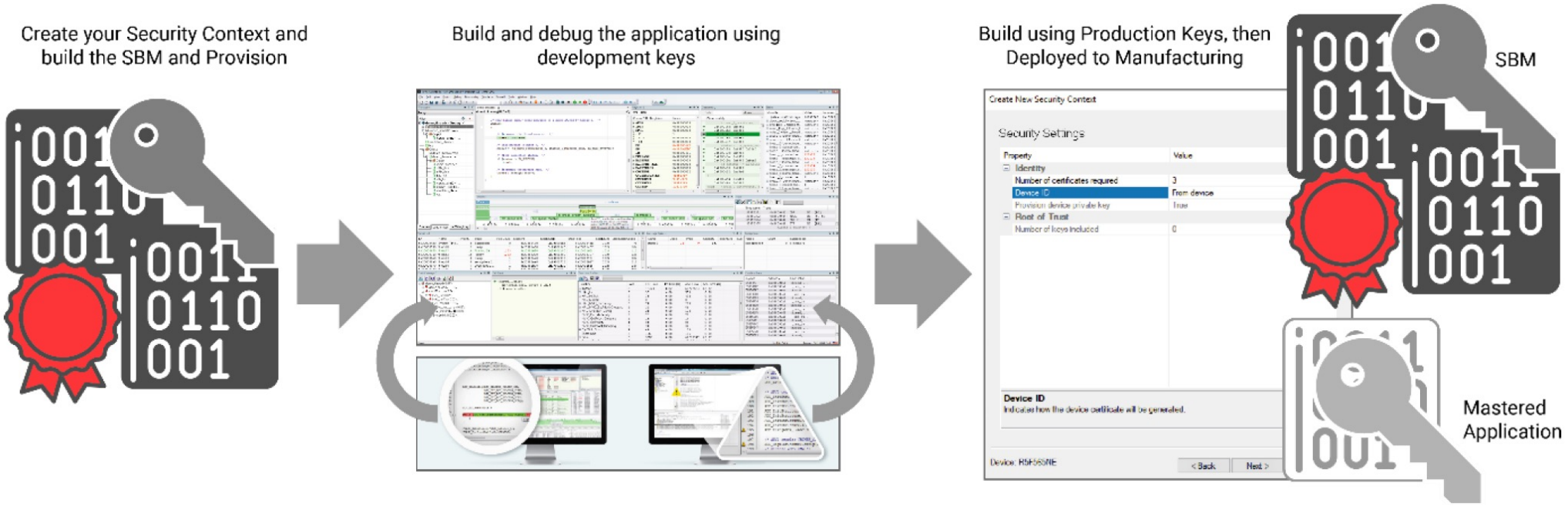
1. a programming facility to allow high volume manufacturing of the IoT product, or to
2. **the IoT product that is deployed in the field and forms part of a secure software update.**



The output from the mastering process is directly passed to an IoT product deployed in the field. The secure microcontroller's certificate can be used to extract the public device key. The UPF and software application image are both encrypted and can only be unwrapped by the secure microcontroller within the IoT product.

::.. Secure Deployment (1/2)

Ultimately, once the development and testing of the application have been completed, it must be delivered to either:



There are tools available to development engineers to simplify the entire process, and to help leverage the secure hardware built into next-generation microcontrollers to provide the low-level trust anchors and secure services needed for trustworthy IoT solutions.

::.. Secure Deployment (2/2)

The steps that define the framework for the secure development environment are

::.. Secure Deployment (2/2)

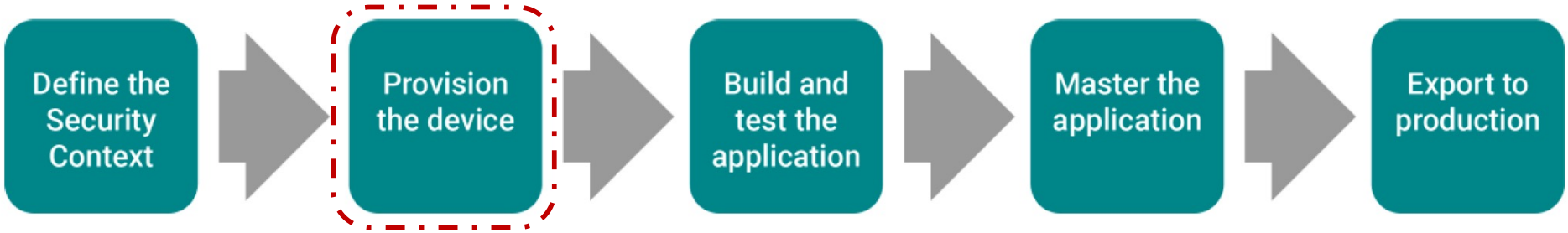
The steps that define the framework for the secure development environment are

Identity creation (security context)

This involves defining the identity of the target device (microcontroller) which forms the core of the IoT product. This identity is created as part of the secure context definition which is the security environment that will protect the IoT product application. The secure context definition process includes specifying security properties like the public keys and certificates, the secure boot manager that manages on the device an application update process, and the device memory layout.

::.. Secure Deployment (2/2)

The steps that define the framework for the secure development environment are

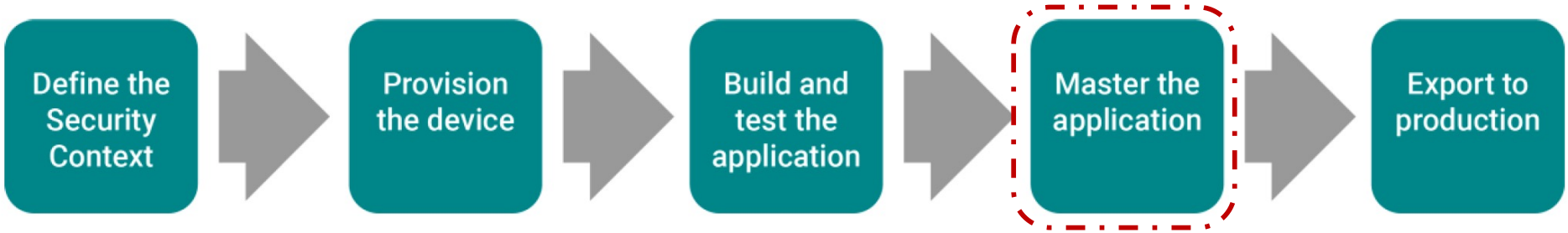


Provisioning

This process includes programming the identity, certificates, keys, and security lock bits into the microcontroller, and ìthe programming of the secure boot manager. After this stage, the secure boot manager can no longer be changed and all subsequent “application programming” for the processor must be delivered via the secure process that uses the boot manager to ensure that the application code is signed and encrypted correctly.

::.. Secure Deployment (2/2)

The steps that define the framework for the secure development environment are

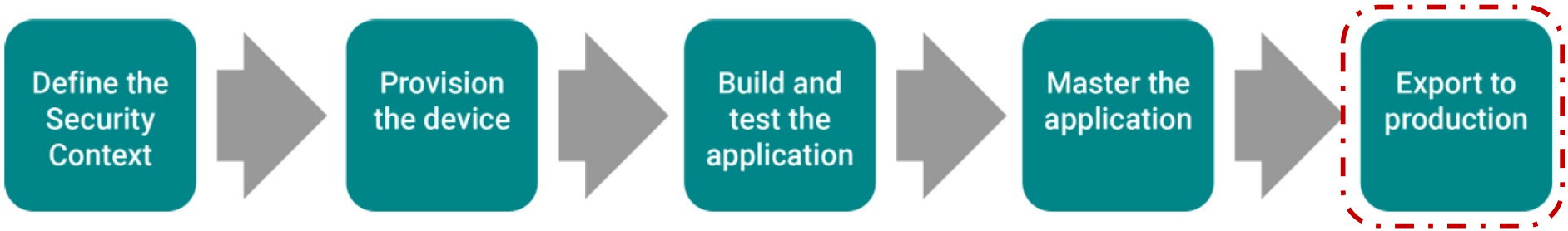


Mastering

This process generates a secure software update file by using an integrated security development environment suitable for programming into a device with a provisioned secure boot manager. The image should contain the encrypted application and an update header; hashed and signed using a specified private mastering key, whose public part has been provisioned on the device with the secure boot manager. It can only be installed on a device that has been provisioned with a secure boot manager generated with the same security context.

::.. Secure Deployment (2/2)

The steps that define the framework for the secure development environment are



Secure deployment

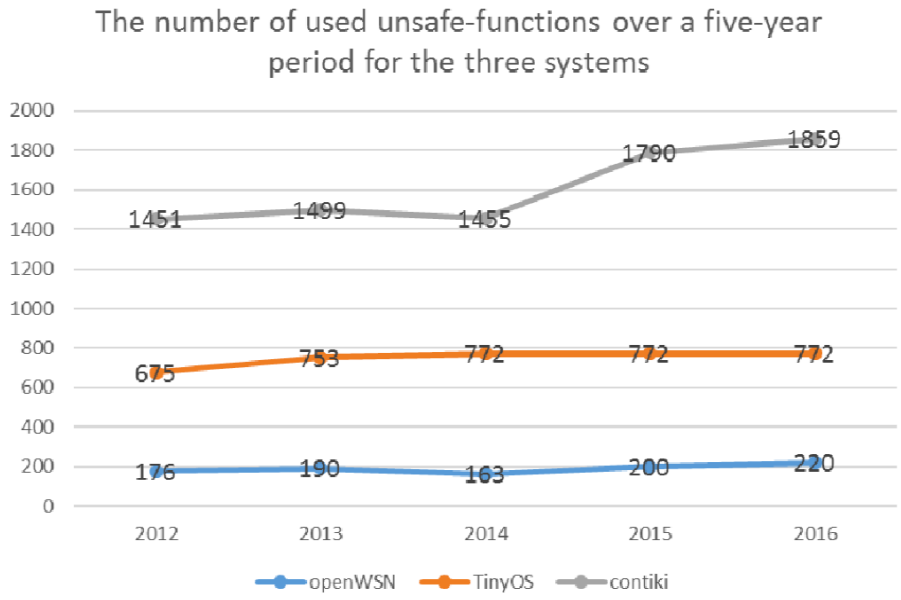
This process prepares the secure software update file, using the public key of the hardware security module (HSM) or target device, to ensure secure delivery of the software application to the target whether that be the IoT product or the secure programming facility. A security development environment would simplify the preparation of the secure package ready for delivery into a ‘production export’ function for secure deployment. This could then have the capability for an OEM to select its chosen programming service provider.

Secure Coding

::.. Introduction (1/2)

Unsafe Function	TinyOS	openWSN	Contiki
puts	0	0	92
strcpy	11	0	91
strcat	10	0	13
fgets	1	0	3
sprintf	5	5	79
strchr	3	0	33
strlen	94	2	343
sscanf, scanf	3	0	40
strcmp	133	0	90
malloc	162	1	7
free	69	1	21
fopen	7	0	8
Localtime, system	5	0	22
memcpy	236	211	712
alloca, CopyMemory	0	0	11
Vsprintf, snprintf, vsnprintf, wsprintf	32	0	280
strtok	0	0	14
Total	772	220	1,859

Most of the detected security threats are due to vulnerabilities in the code. Thus, minimizing usage of insecure commands plays a big role in protecting the systems from any potential attacks.



::.. Introduction (2/2)

Developers need to be aware of those unsafe commands and following good programming practices can minimize the time and effort spent on finding and fixing them in later stages.

C/C++ programming languages are preferred when it comes to write code for IoT devices, but their security vulnerabilities, such as integer vulnerability, buffer overflow, and string vulnerability, need to be addressed and avoided from the beginning when writing the source code.

The process of protecting software systems from those vulnerability issues is done by either completely removing known unsafe commands and functions, or replacing them with what some call safe replacements.

::.. Validation & Sanitization (1/2)

- **Validation** checks if the input meets a set of criteria (such as a string contains no standalone single quotation marks).
- **Sanitization** modifies the input to ensure that it is valid (such as doubling single quotes), by removing any illegal character.

Developers would normally combine these two techniques to provide in-depth defence to applications.

Validation strategies:

- **Accept known good**, also known as "whitelist" or "positive" validation. The idea is that you should check that the data is one of a set of tightly constrained known good values. Any data that doesn't match should be rejected.

::.. Validation & Sanitization (1/2)

- **Validation** checks if the input meets a set of criteria (such as a string contains no standalone single quotation marks).
- **Sanitization** modifies the input to ensure that it is valid (such as doubling single quotes), by removing any illegal character.

Developers would normally combine these two techniques to provide in-depth defence to applications.

Validation strategies:

- **Reject known bad**, also known as "negative" or "blacklist" validation, consists in rejecting strings containing illegal characters or patterns. Maintaining the list of "known bad" characters and patterns implies by definition to have incomplete protection.

::.. Validation & Sanitization (2/2)

Sanitization strategies:

- **Sanitize with Whitelist** - Any characters which are not part of an approved list can be removed, encoded or replaced. Note that it is needed to validate the resulting data after sanitization as well. This is not only beneficial for security, but it also allows using a wider range of valid user input.
- **Sanitize with Blacklist** - Eliminate or translate characters (such as to HTML entities or to remove quotes) in an effort to make the input "safe". Like blacklists, this approach requires maintenance and is usually incomplete. As most fields have a particular grammar, it is simpler, faster, and more secure to simply validate a single correct positive test than to try to include complex and slow sanitization routines for all current and future attacks.

::.. Buffer Overflow (1/9)

```
int check_pwd() {
   char pwd[12];
   gets(pwd);
   return (strcmp(pwd, "p4ssw0rd") == 0);
}
int main() {
   int pwd_ok;
   puts("Enter password:");
   pwd_ok = check_pwd();
   if (!pwd_ok) {
      puts("Access denied");
      exit(-1);
   }
   puts("Access granted");
   // ...
}
```

::.. Buffer Overflow (1/9)

```
int check_pwd() {
   char pwd[12];
   gets(pwd);
   return (strcmp(pwd, "p4ssw0rd") == 0);
}
int main() {
   int pwd_ok;
   puts("Enter pass
   pwd_ok = check_
   if (!pwd_ok) {
      puts("Access denied
      exit(-1);
   }
   puts("Access granted");
   // ...
}
```

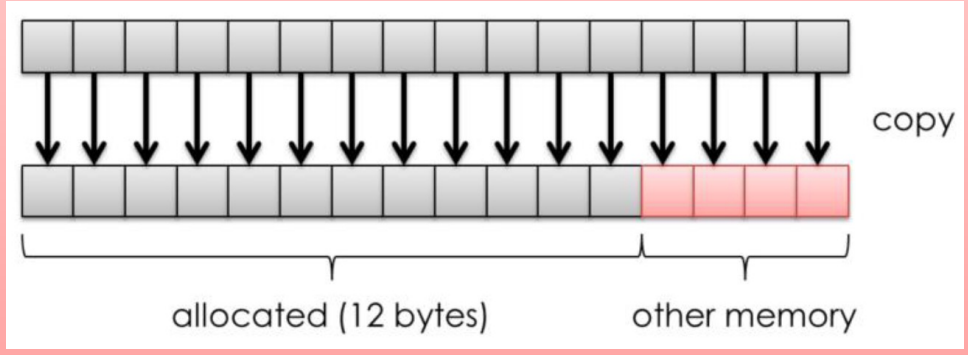
If the user inserts a password of 12 characters or more, the gets() function will write beyond the last character of the variable “pwd”. This programming anomaly is called buffer overflow.

- Managed languages such as Java and C# could check for out-of-bound write and raise a exception.
- The majority of C/C++ compilers does not have out-of-bound check to be performed on buffer accesses.

::.. Buffer Overflow (1/9)

```
int check_pwd() {
   char pwd[12];
   gets(pwd);
   return (strcmp(pwd, "p4ssw0rd") == 0);
}
int main() {
   int pwd_ok;
   puts("Enter pass
   pwd_ok = check_
   if (!pwd_ok) {
      puts("Access denied
      exit(-1);
   }
   puts("Access granted");
   // ...
}
```

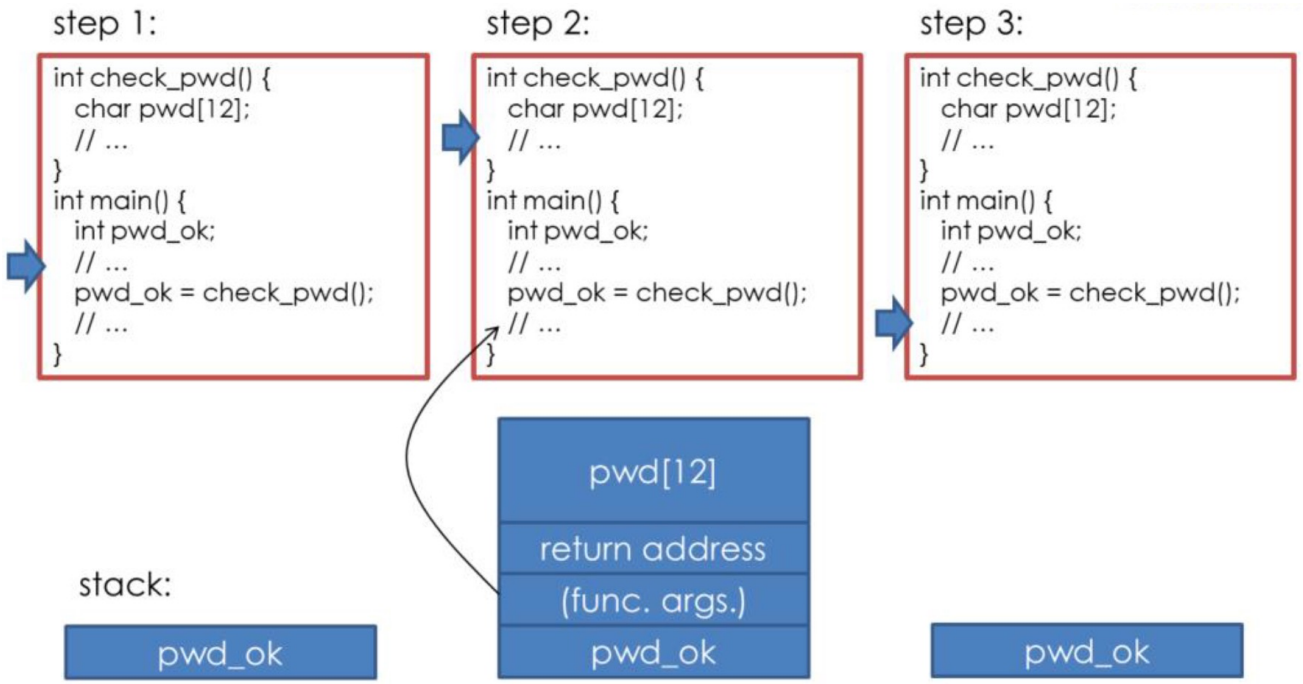
The data that overflows the buffer is written in the memory space that happens to be contiguous to the overflowed buffer, containing other data (variables, etc.). Therefore, other data is over written.



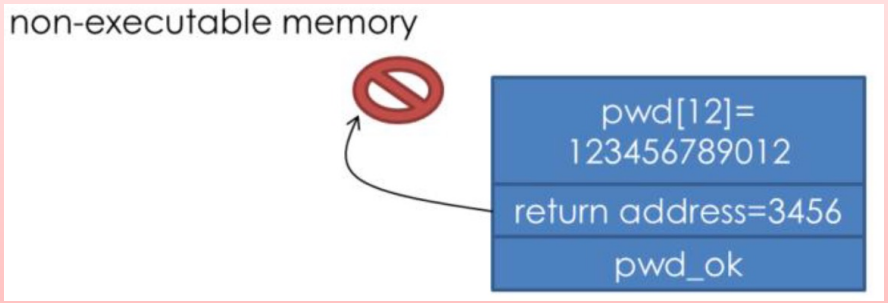
::.. Buffer Overflow (2/9)

A buffer overflow in the heap is called heap overflow, while a an overflow in the stack is called stack overflow. Both heap and stack overflows can read/change the value of other variables.

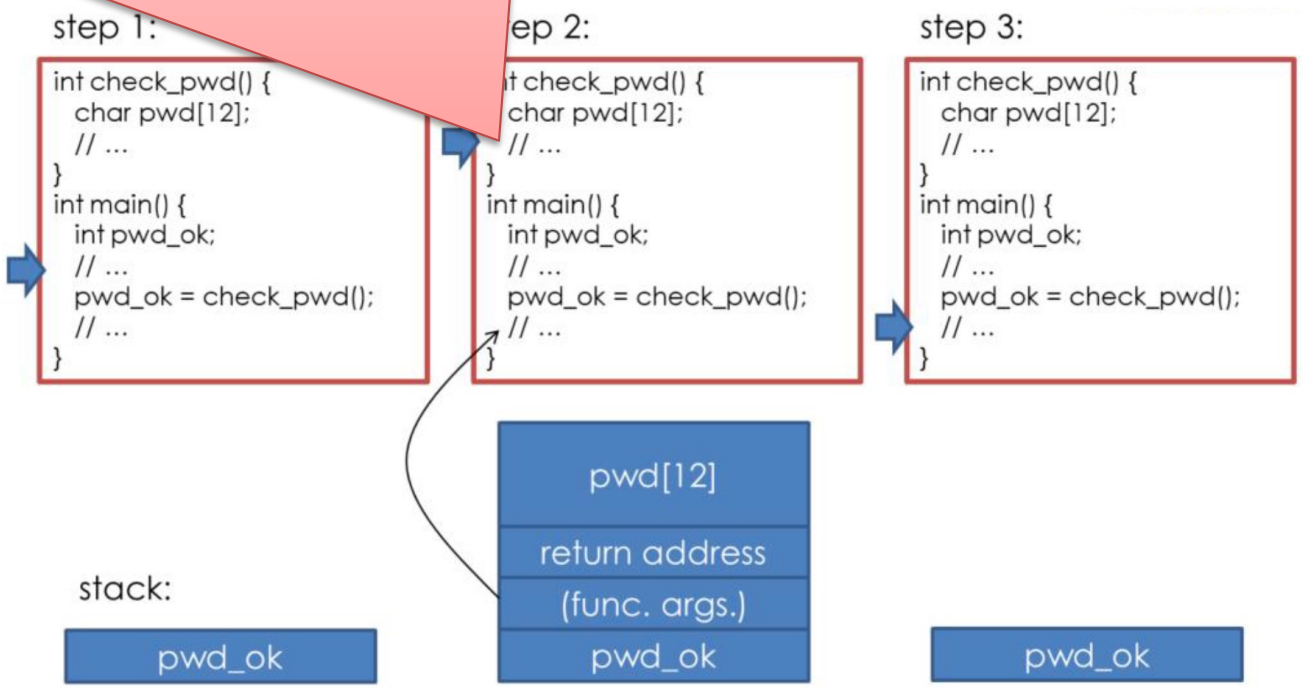
- Stack overflow is generally more dangerous, because it can directly change the execution sequence by changing the return address of the subroutines.



::.. Buffer Overflow (2/9)



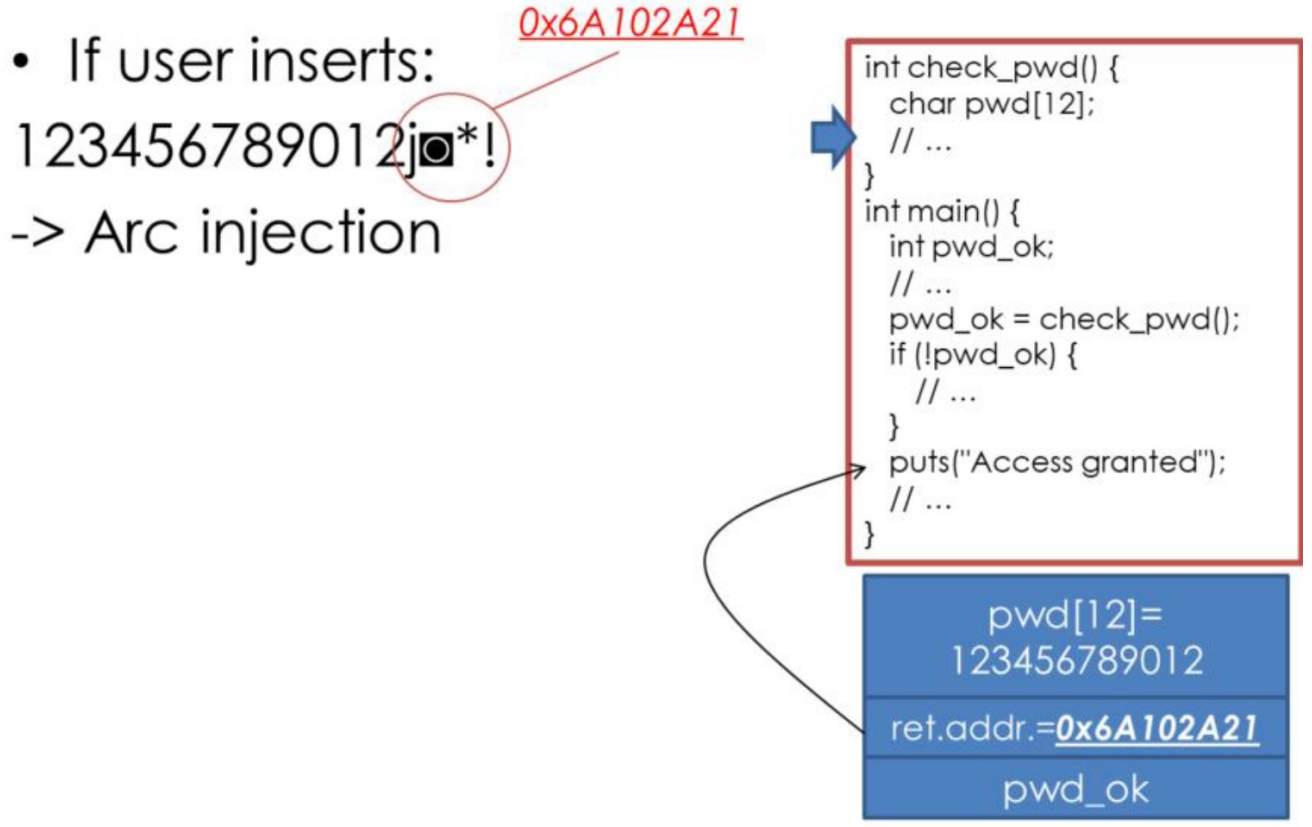
If the user insert a password with a length of 12 characters or more, then the return address will be over written. In this case, the overflowed data is inter-preted as a return address.



::.. Buffer Overflow (3/9)

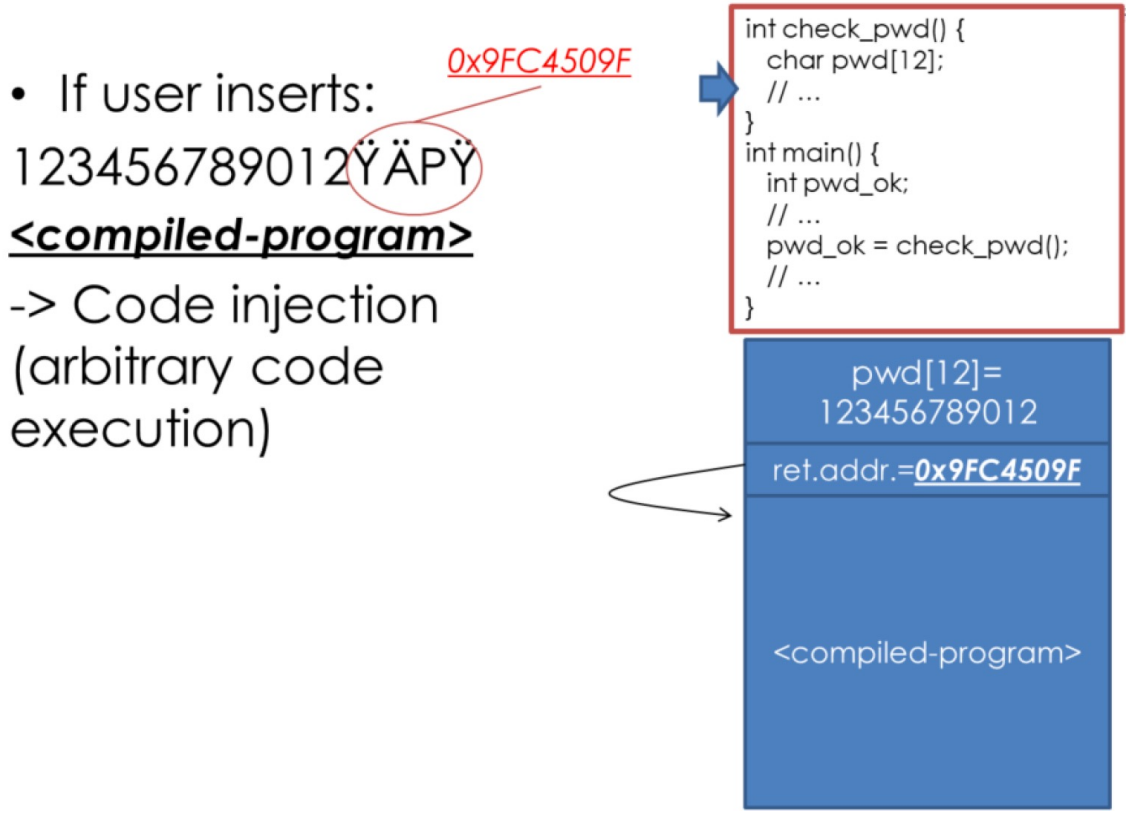
If the malicious user knows the allocation of a given line of code of the program, the password check can be bypassed.

Such an attack is commonly known as arc injection , because the attacker injects a malicious «arc» in the program execution flow.



::.. Buffer Overflow (4/9)

If the overwriting characters ŸÄPŸ correspond to the address 0x9FC4509F in hexadecimal digits pointing to the first memory location of the injected malware, we have the so-called code injection , and it results in an arbitrary code execution.



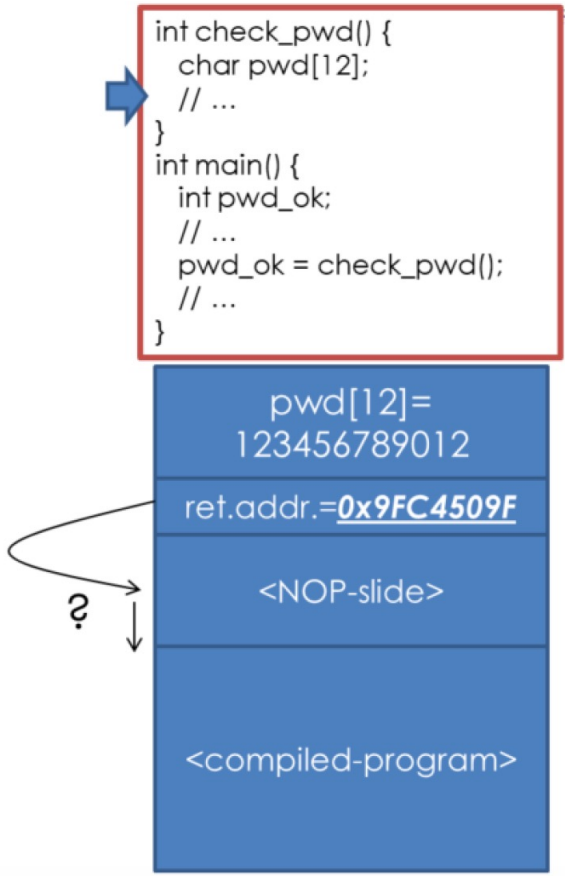
::.. Buffer Overflow (5/9)

To mount a code injection, the attacker must know the exact address of the stack, which is not always predictable. To overcome this, the attacker can prepend the malware with a large number of NOP instructions (called NOP slide in the stack).

- If malware's address is known only approximately, the user can insert:

123456789012ŸÄPŸ

<NOP-slide><compiled-program>



The overwritten return address needs only to jump in the middle of the NOP slide, so it can be approximated.

::.. Buffer Overflow (6/9)

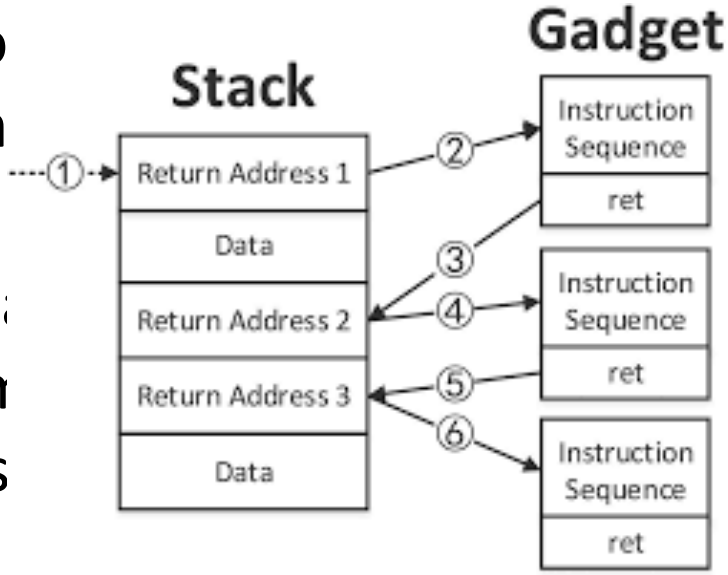
The major operating systems and compilers adopted a series of stack overflow protections. They do not make the attack impossible, but only more technically challenging.

- **Data Execution Prevention** works by marking each part of the memory as executable or non executable. The stack is marked as non executable, so when the victim process tries to execute the malware in the stack, a fault is raised and the process abnormally terminates.

::.. Buffer Overflow (6/9)

The major operating systems and compilers adopted a series of stack overflow protections. They do impossible, but only more technically ch

- **Data Execution Prevention** works by memory as executable or non execut as non executable, so when the victim the malware in the stack, a fault is abnormally terminates.



A violation of this protection is the **Return-oriented programming** (ROP) where an attacker executes carefully chosen machine instruction sequences called "gadgets". Each gadget typically ends in a return instruction. Chained together, these gadgets allow an attacker to perform arbitrary operations on a machine employing defences that thwart simpler attacks.

::.. Buffer Overflow (7/9)

Performing ROP is simple when the adversary knows deterministically all the addresses of the victim code.

With **Address Space Layout Randomization** (ASLR), the operating system decides randomically where to load the program to be executed. All the absolute addresses of the programcode are relocated by the same random offset.

::.. Buffer Overflow (7/9)

Performing ROP is simple when the adversary knows deterministically all the addresses of the victim code.

With **Address Space Layout Randomization** (ASLR), the operating system decides randomically where to load the program to be executed. All the absolute addresses of the programcode are relocated by the same random offset.

ASLR is an additional source of uncertainty for the attacker, but it does not make the attack impossible, rather probabilistic.

- The possible relocation addresses are few. The attacker can leverage pointer leaks , that are program bugs that expose the value of pointers. If such pointers point to the code, they reveal the relocation address.
- The attacker could do ROP with gadgets from non randomized memory, as the one containing the code of libraries shared between different processes.

::.. Buffer Overflow (8/9)

```
int check_pwd() {
   char pwd[12];
   // ...
}
int main() {
   int pwd_ok;
   // ...
   pwd_ok = check_pwd();
   // ...
}
```

In the **stack canaries** technique, the prologue of every function pushes a value canary in the stack between the local variables and the return address.

If an attacker wants to overwrite the return address with a buffer over-flow, then he must corrupt also the canary.

The function epilogue checks if the canary (unpredictable by the attacker) still has the correct value, and it raises an error if it has not.

::.. Buffer Overflow (9/9)

Stack canaries are currently the most effective defence against stack overflow, because they are hard to bypass or to guess. Stack canaries are typically 4 byte long, so the attacker has a chance over 2^{32} to guess it.

Stack canaries slightly slow down the program performance since they introduce some operations at the prologue and at the epilogue of every function. So, some compilers optimize the program by avoiding stack canaries in those function that they consider « safe » from stack overflow vulnerabilities.

They cannot defend against stack overflows that do not overwrite the return address. It is still possible to use a stack overflow to simply produce an abnormal program termination, thus causing a denial of service. Moreover, stack canaries cannot prevent buffer over-reads: overrunning a buffer’s boundary in a read operation, which leads to information leaks.

::.. Secure Coding (1/7)

Secure coding studies those programming errors which have caused the most common, dangerous, and disruptive software vulnerabilities in the past.

Secure coding is strongly language dependent. As a general rule , the lower the programming language level is, the more error prone, and thus the more susceptible to vulnerabilities.

C and C++ languages are particularly error prone, because they are intended to be lightweight and to produce a small codeprint . The programmer, especially if coming from higher level languages like Java, may assume that the C language implicitly perform checks when it does not.

Despite their poor security characteristics , C and C++ are widely used still today, in particular when performances are a requirement.

::.. Secure Coding (2/7)

- An **undefined behavior** is a behavior on which the C/C++ standard poses no requirements, for example in case of out of bound buffer access, null pointer dereferencing, or signed integer overflow.
- An **unspecified behavior** is a behavior on which the C/C++ standard gives two or more possibilities, for example the order of argument evaluation in a function call.
- An **unexpected behavior** is a well defined behavior, yet unexpected by the programmer, due to incorrect programming assumptions.

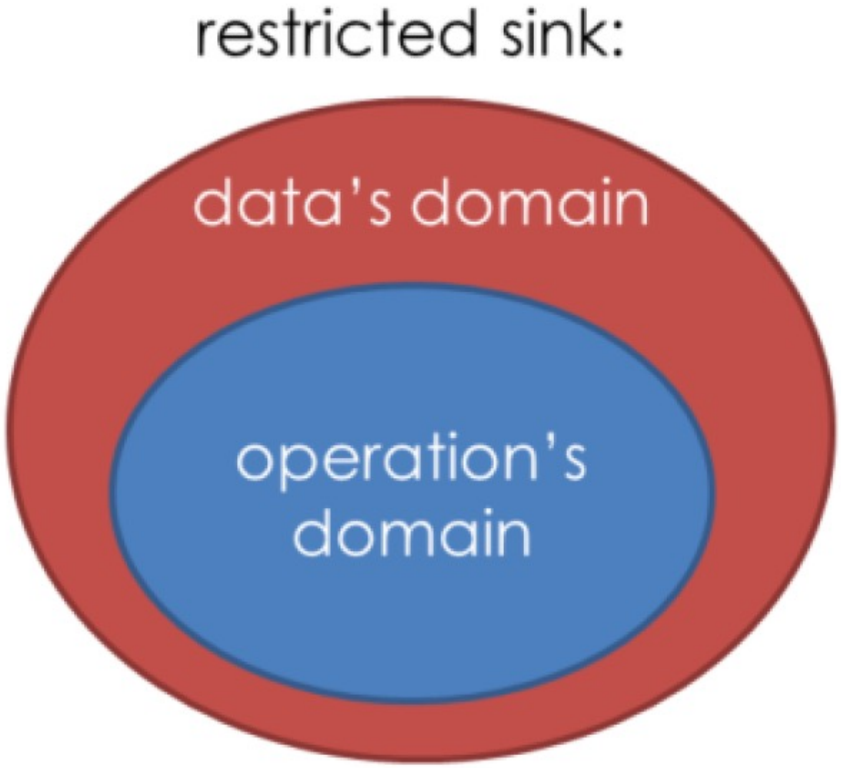
The majority of software vulnerabilities is based on undefined behaviors. The attacker tries to induce the program to perform undefined behaviors by means of special crafted inputs. Then, the attacker tries to damage somehow the system.

::.. Secure Coding (2/7)

- An **undefined behavior** is a behavior on which the C/C++ standard poses no requirements, for example in case of out of bound buffer access, null pointer dereferencing, or signed integer overflow.
- An **unspecified behavior** is a behavior on which the C/C++ standard gives two or more possibilities, for example the order of argument evaluation in a function call.
- An **unexpected behavior** is a well defined behavior, yet unexpected by the programmer, due to incorrect programming assumptions.

- Undefined behaviors must be avoided.
- Unspecified behavior must be known.
- Unexpected behaviors must be expected.

::.. Secure Coding (3/7)



Data that comes from a source external to the program and that has not been sanitized yet is called **tainted data**. The result of an operation over tainted data is tainted data too.

An operand or a function argument whose domain is a subset of the domain of its type is called **restricted sink**.

Undefined/unspecified/unexpected behaviors happen when tainted data is given as input to a restricted sink.

::.. Secure Coding (4/7)

Most vulnerabilities in C are related to buffer overflows string manipulation. In most cases, this would result in a segmentation fault, but specially crafted malicious input values, adapted to the architecture and environment could yield to arbitrary code execution.

- The stdio gets() function does not check for buffer length and always results in a vulnerability.

```
char username[8];
int allow = 0;
printf ("Enter your username, please: ");
gets(username); // user inputs "malicious"
```

::.. Secure Coding (4/7)

Most vulnerabilities in C are related to buffer overflows and string manipulation. In most cases, this would result in a segmentation fault, but specially crafted malicious input values, adapted to the architecture and environment could yield to arbitrary code execution.

- The stdio gets() function does not check for buffer length and always results in a vulnerability. Prefer using fgets (and dynamically allocated memory).

```
char username[8];
int allow = 0;
printf ("Enter your username, please: ");
gets(username); // user input
```

```
char* username, *nlptr;
int allow = 0;
 username = malloc(LENGTH * sizeof(*username));
if (!username)
    return EXIT_FAILURE;
printf ("Enter your username, please: ");
fgets(username,LENGTH, stdin);
```

::.. Secure Coding (5/7)

- The strcpy built-in function does not check buffer lengths and may very well overwrite memory zone contiguous to the intended destination. In fact, the whole family of functions is similarly vulnerable: strcpy, strcat and strcmp.

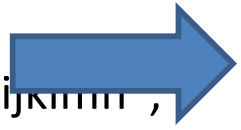
```
char str1[10];
char str2[]="abcdefghijklmn";
strcpy(str1,str2);
```

::.. Secure Coding (5/7)

- The strcpy built-in function does not check buffer lengths and may very well overwrite memory zone contiguous to the intended destination. In fact, the whole family of functions is similarly vulnerable: strcpy, strcat and strcmp. A remediation is to use strncpy, which prevents buffer overflows, but does not guarantee '\0'-termination.

For the other functions in the family, the *n* variants exist as well: strncpm and strncat.

```
char str1[10];
char str2[]="abcdefghijklmn";
strcpy(str1,str2);
```

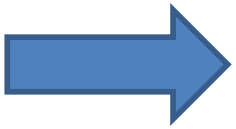


```
enum { BUFFER_SIZE = 10 };
char str1[BUFFER_SIZE];
char str2[]="abcdefghijklmn";

strncpy(str1,str2, BUFFER_SIZE);
/if (str1[BUFFER_SIZE-1] != '\0') {
   /* buffer was truncated, handle error? */
}
```

::.. Secure Coding (6/7)

- Also sprintf does not check the buffer boundaries and is vulnerable to overflows. Prefer using snprintf, which prevents buffers overflows and returns the minimal size of buffer needed to fit the whole formatted string.
- One other vulnerability category is concerned with string formatting attacks, exploitable in any of the following functions: printf, fprintf, sprintf and snprintf, i.e. all functions that take a “format string” as argument.

```
#FormatString.c
#include <stdio.h>

int main(int argc, char **argv) {
   char *secret = "This is a secret!\n";
   printf (argv[1]);
   return 0;
}
```



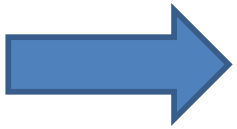
```
$ gcc -mpreferred-stack-boundary=2
FormatString.c -o FormatString $
./FormatString %s This is a secret! $
```

::.. Secure Coding (6/7)

- Also sprintf does not check the buffer boundaries and is vulnerable to overflows. Prefer using snprintf, which prevents

If an attacker gives a message msg ” containing some “%x” wildchars, printf () will try to retrieve and print (in hexadecimal digits) some non existent arguments, leading to an undefined behavior.
Usually, the actual arguments of a variadic function are stored in the stack, so printf () will write on the screen the current content of the stack, which constitutes an serious information leakage.

```
#FormatString.
#include <stdi

int main(int
   char *sec
   printf (argv[1]);
   return 0;
}
```



```
$ gcc -mpreferred-stack-boundary=2
FormatString.c -o FormatString $
./FormatString %s This is a secret! $
```

::.. Secure Coding (6/7)

- Also sprintf does not check the buffer boundaries and is vulnerable to overflows. Prefer using snprintf, which prevents buffers overflows and returns the minimal size of buffer needed to fit the whole formatted string.
- One other vulnerability category is concerned with string formatting attacks, exploitable in any of the following functions: printf, fprint
  that take a "format strir

Always hardcode the format string. At least, never let it come directly from any user's input.

```
#FormatString.c
#include <stdio.h>

int main(int argc, char **argv) {
   char *secret = "This is a secret!\n";
   printf ("%s", argv[1]);
   return 0;
}
```

```
$ gcc -mpreferred-stack-boundary=2
FormatString.c -o FormatString $
./FormatString %s This is a secret! $
```

::.. Secure Coding (7/7)

- The C++ object library addresses buffer overflows. The library std::string combined with stream operators (>> and <<) reduces the likelihood of overflows. However, these can be compromised when a program still uses C API functions.
- Also Java does not make internal buffer overflow attacks, as well as other C vulnerabilities being easy, thanks to its Security Architecture.

Both are not completely free from unsafe coding styles and functions (type confusion attack in Java or shared variables can cause race conditions in C+).

Common Weakness Enumeration (CWE) is a community-developed list of common software security weaknesses, and is regarded as an universal online dictionary of weaknesses that have been found in computer software.

::.. Secure Coding (7/7)

OWASP Dependency-Check can be easily integrated in your build process for example by using the dependency-check-maven plugin.

pom.xml

```xml
<project>
    ...
    <reporting>
        <plugins>
            <plugin>
                <groupId>org.owasp</groupId>
                <artifactId>dependency-check-maven</artifactId>
                <version>1.3.4</version>
            </plugin>
        </plugins>
    </reporting>
    ...
<project>
```

Common Weakness Enumeration (CWE) is a community-developed list of common software security weaknesses, and is regarded as an universal online dictionary of weaknesses that have been found in computer software.

GDPR and IoT Privacy

::.. Introduction (1/7)

The IoT is defined by connections and linked services, tailored to the specific requirements of users. Without repeated and consistent identification of users, linked up, seamless services would not be possible. However, the pursuit of identification and personalisation of users poses a risk to privacy.

Users can easily perceive this insight as invasive, unexpected, and unwelcome. The impossibility of anonymising data and weak cybersecurity standards can exacerbate privacy risks.

The GDPR creates governing principles of data processing and establishes new data protection means relating to informed consent, notification duties, privacy by design and privacy by default, data protection impact assessment, algorithmic transparency, automated decision-making, and profiling.

::.. Introduction (1/7)

The IoT is defined by connections and linked services, tailored to the specific requirements of users. Without repeated and consistent identification of users, linked up, seamless services would not be possible. However, the pursuit of identification and personalisation of users poses a risk to privacy.

Users can easily perce… insight as invasive, unexpected, and unwel… k cybers…

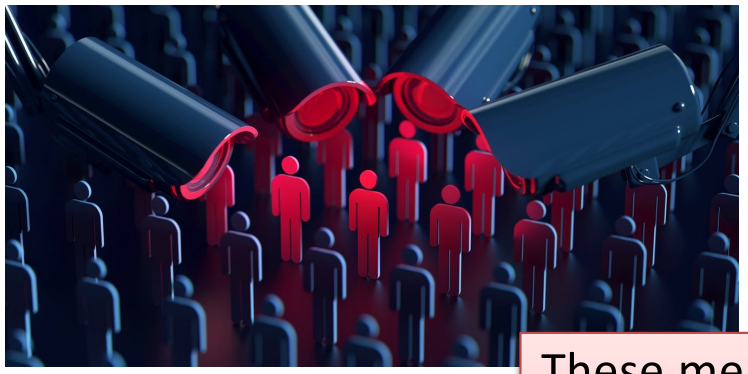
The G… d establis… t, notifica… a protect… d decisio… making, and profiling.

It emerged that

- 59% of devices do not offer adequate information on how the personal data of data subjects is collected, used and communicated to third parties;
- 68% do not provide appropriate information on how to keep data;
- 72% do not explain to users how to erase data from the device;
- 38% do not guarantee simple contact methods for customers who want clarifications regarding respect for their privacy.

::.. Introduction (1/7)

The IoT is defined by connections and linked services, tailored to the specific requirements of users. Without repeated and consistent identification of users, linked up, seamless services would not be possible. However, the pursuit of identification and personalisation of users



These means will be undermined by the tendency of IoT devices and services to collect, share, and store large and varied types of personal data, to operate seamlessly and covertly, and to personalise functions based on prior behaviour.

Users can easily d
unwelcome. The k
cybersecurity standards can e bate privacy risks.

The GDPR creates governing principles of data processing and establishes new data protection means relating to informed consent, notification duties, privacy by design and privacy by default, data protection impact assessment, algorithmic transparency, automated decision-making, and profiling.

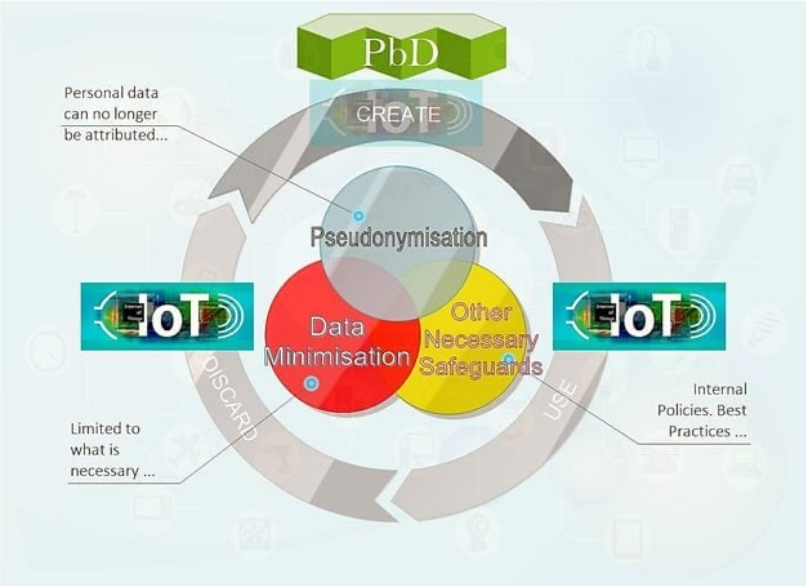
::.. Introduction (2/7)

Securing the Internet of Things (IoT) and protecting the privacy of its users do not require a radically new, complex set of ideas and principles. What it needs is an evolution of best practices that have been built up over many years in all areas of IT security/privacy.

The starting point for securing an IoT deployment and preserving privacy should be the principles of security/privacy by design as suggested by GDPR. Security/privacy considerations were often included late in the design and prototyping phase. By prioritizing speed to market or other design considerations, security/privacy requirements can end up being added on. This approach has led to serious security breaches and privacy violations in the past.

Art. 25 of the GDPR makes mention of which methods data controllers/processors may choose to use in order to apply the Privacy by Design (PbD) : “pseudonymization,” “data minimization” and other “necessary safeguards [that] protect the rights of data subjects.”

::.. Introduction (3/7)



- Pseudonymization - if all identifiers are completely remove from personal data, it will not be deemed personal any more within the meaning of the EU data protection law.

Art. 4(5) of the GDPR

"'pseudonymisation' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person;"

- Services/applications in the realm of the IoT technology to process only the minimum amount of information necessary for the fulfilment of the particular service/function/transaction.

Art. 5(1) (c) of the GDPR

"Personal data shall be adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation');"

::.. Introduction (4/7)

According to the Working Party 29's (WP29) opinion on IoT (n. 8/2014):

- Device manufacturers qualify as Controllers for the personal data generated by the device.
- Third party app developers that organise interfaces to allow individuals to access their data stored by the device manufacturer will be considered Controllers.
- Other third parties are Controllers when using IoT devices to collect and process information about individuals. These third parties usually use the data collected through the device for other purposes that are different from the device manufacturer.
- Other stakeholders such as IoT data platforms and social platforms may also be considered as Controllers for the processing activities, for which they determine purposes and means. On the contrary, they may be considered as Processors where they process data on behalf of another IoT stakeholder that acts as a controller.

::.. Introduction (5/7)

The GDPR is not the only relevant legislation for IoT. Another important piece of legislation is the upcoming ePrivacy Regulation, which will replace the existing ePrivacy Directive and will be directly applicable throughout the EU (and likely beyond).

- Article 4(3)(aa) of the Parliament version of the proposal explicitly provides that 'machine-to-machine communications' are covered by the Regulation. The communication of non-personal data between machines may be covered by the finalised ePrivacy Regulation but not by the GDPR. For example, end-users' consent may still be required for processing (non-personal) communication data (Article 8 of the draft ePrivacy Regulation).

Recital 12 of the Council version of the proposal clarifies that the material scope is limited to transmission services and excludes the application layer (e.g., a mobile app). Therefore, as the application layer does not represent an electronic communications service, it would not fall within the scope of the proposed ePrivacy Regulation, but rather the GDPR (insofar as it contains personal data).

::.. Introduction (6/7)

- The privacy must be embraced right from the beginning of implementing a solution and considered at all times if their products process personal data. Usually, it is difficult and expensive to add privacy to a product at a later stage or, even worse, reengineer it following a failure.

PbD embedded into connected objects also includes the presence of cyber-hygiene best practices, such as:

- Security transmission protocols and encryption techniques for data in transit and at rest. Unfortunately, the majority of IoT objects today can’t use these features due to various inherent technical constraints.
- Proper authentication controls, limiting permissions (assigned on a need-to-know basis).
- Accounting operations for post-mortem analysis.

::.. Introduction (7/7)

- For critical communications, authentication and encryption are imperative for optimal protection, but providing means to allow the data integrity to be verified can be a recommended best practice with additional value to privacy and security.
- Options to allow privacy/security default settings to be changeable, as well as services with a proven track record of creating vulnerable environments.
- Training company staff in privacy and data security best practices.
- Application containerization, where apps are installed in a contained environment.

Some practical approaches that may facilitate the implementation of the PbD idea into a real and workable privacy shield are backend isolation, data separation, segregation, redaction and data transform techniques that remove personally identifiable information.

::.. User Identity (1/5)

Identification technologies are a crucial component of trusted communication in the IoT:

- To be trustworthy, communication should occur only between intended nodes in the network. Consistent and unique identification of objects and users is required to avoid to be exposed to third party attacks, data leakage, or communication with unintended devices and users.
- As a basis for trusted communication, identity management is often coupled with access control to authenticate and authorise nodes for operations, transmission, and access to data. Biometric authentication has grown in recent years due to be potentially more secure than traditional username/password combinations, these data sources introduce new risks of invasive inferences due to the inherent sensitivity of health data. Similar concerns arise for authentication based on behavioural modelling.

::.. User Identity (2/5)

- The existing diversity of possible systems for identity management and access control can be beneficial for cybersecurity purposes (by avoiding a single potential point of failure or attack), it can simultaneously increase the burden on users seeking to verify that the standards governing a particular device or service are fit-for-purpose, and thus do not pose undue risks to their privacy.

Privacy is inherently connected to the integrity of the information constituting one's identity. Information and communication technologies, such as IoT create new flows of information describing users, which impact, on how the user self-identifies, and how others perceive and understand them.

Such external management of identity applied by IoT or similar infrastructures, seen for example in the creation of knowledge about an individual through inferential profiling, constitutes an attack on the integrity of the individual's identity.

::.. User Identity (3/5)

Behavioural profiling creates and modifies information about the user, which becomes part of the individual's identity and shapes his/her future treatment within IoT systems. When a user is unaware that his/her IoT devices and services generate information about him/her, he/she lacks the ability to incorporate this information into his/her self-constructed identity (or narrative), and to view himself/herself as others view him/her (with the inaccessible information).

The user may remain unaware of the full extent and nature of his/her identity, as the information constituting this identity can be externally constructed (i.e. through inferential analytics).

The user may lack control or oversight of the content of his/her identity, how it changes over time and shapes his/her experiences when the identity is known to other users. Further, users are often unable to assess the validity and quality of inferenced data. As a result, the externally constructed identity often remains unclear to the user.

::.. User Identity (4/5)

A user's digital identity can be segmented for various purposes. Only portions of the user's identity or profile are typically available to specific entities. These segments of a user's overall identity or profile can be referred to as 'virtual identities'.

By sharing a virtual identity with a device, the user is able to influence and constrain its operation. As the identity is passed across and between networks, it constrains actions by other nodes on behalf of the user (mainly referred as the user's 'digital shadow'). However, the virtual identity only needs to fulfil a context-specific function, which does not normally require disclosure of a user's full identity or profile. Digital shadows only implicitly indicate their owner's identity.

Well-curated virtual identities, used as a digital shadow was originally proposed as a privacy-enhancing mechanism for identity management in the IoT.

::.. User Identity (5/5)

As the IoT requires linkage between devices and services for personalisation, a fundamental tension exists between linkage of IoT nodes and privacy by controlling personal data. In fact, in order to provide personalised services, IoT controllers and third parties will need to link data about specific users from disparate sources, with or without the consent of the user.

- This neglects the possibility of using digital shadows and makes virtual identifies richer of details on the users.

::.. Risks from User Identification (1/2)

The main privacy risks and violations of the GDPR obligations made by the main IoT systems are the following ones:

- IoT devices collect vast amounts of data often without the awareness of the data subjects.
- Users often have very little control over their data, and lack sufficient knowledge to give free and informed consent.
- Pervasive data collection, AI-based analyses, and the combination of datasets pose serious risks for privacy as data controllers can draw invasive inferences about the user.

There are at least four senses in which identification technologies pose a risk to user privacy:

1. By enabling linkage of user identities and records generated from IoT devices, which can lead to potentially invasive profiling, inferences, and discrimination;

::.. Risks from User Identification (2/2)

2. by revealing sensitive information to other IoT users that the user would otherwise prefer to keep confidential, and inhibiting user's control over such disclosures;
3. by generating information or inferences about the user, which could not have been predicted when setting access policies, or consented to at the point of adoption;
4. by limiting user oversight and transparency in management of identity and profiling, which can facilitate breaches of privacy and undermine trust among IoT users, objects, and device or service providers.

::.. Profiling (1/4)

There are at least three possible ways of monitoring and profiling that offer grounds for discrimination in IoT systems:

- data collection that leads to inferences about the person;
- profiling at large through linking IoT datasets;
- profiling that occurs when data is shared with third parties that combine data with other datasets.

Unpredictable, invasive profiling and inferential analytics can result from data sharing, in particular when multiple IoT devices provide data that is linked to a single user (virtual) identity.

While some inferences and profiling drawn from IoT can be benign, they can also lead to unfair discrimination. The potential for discrimination holds true even when using non-sensitive data categories, from which sensitive information can still be inferred. Third parties with access to IoT data linked to an identified target can use this data for purposes with which the user would not agree if asked.

::.. Profiling (2/4)

- Fitbit data could, for example, be relevant to prospective employers, who could make inferences about “impulsivity and the inability to delay gratification - both of which might be inferred from one’s exercise habits - correlate with alcohol and drug abuse, disordered eating behaviour, cigarette smoking, higher credit-card debt, and lower credit scores.
- Lack of sleep - which a Fitbit tracks-has been linked to poor psychological well-being, health problems, poor cognitive performance, and negative emotions such as anger, depression, sadness, and fear.
- Data controllers increasingly use the IoT for “monitoring people’s online behaviour and using the information collected to show people individually targeted advertisements”.

A lack of awareness of such methods can be unfair to users, hence an increased level of transparency is required.

::.. Profiling (3/4)

Weaknesses of anonymisation to prevent profiling and resulting discrimination cause further problems, such as used to tracking specific individuals, linked to information in other databases, and possibly exploited to predict future behaviour.

These problems will be exacerbated by the proliferation of machine learning in the IoT. In fact, machine learning will lead to even less predictable inferences, while the complexity and opaqueness of machine learning algorithms can inadvertently hide discriminatory treatment from users.

Data cannot be permanently anonymised without destroying its analytical value, non-technical methods may be necessary to prevent profiling and discrimination in the IoT.

- Article 21 in GDPR introduces the right of data subjects to object to data processing, including profiling, at any time. However, the technical feasibility of ceasing data collection is challenging.

::.. Profiling (4/4)

- Article 22 introduces additional safeguards against automated decision-making, including profiling, but only when data processing is solely automated and has legal or similar significant effects. In cases where such decision-making and profiling are necessary for entering or performing a contract between data subject and the data controller, or based on explicit consent (Article 22(2)(a) and (c)), data subjects are granted rights to obtain human intervention on the part of the controller, to express a point of view and to contest the decision (Article 22(3)). If automated decision-making, including profiling, has significant effects on a data subject, individuals will possess a legal remedy if they disagree with the outcome.

::.. Sharing of Data/Identities (1/3)

Users are often powerless to prevent potentially discriminatory profiling due to a lack of control over disclosures of personal data and identity in the IoT. Mechanisms to support targeted disclosure of identity may thus facilitate protection of privacy. Granting disclosure control to users who do not understand the potential risks and benefits of revealing different aspects of their identity can actually expose users to greater privacy risks. Generic or provider-defined disclosure policies may thus be preferable in some cases.

Multi-user and multi-controller models of ownership of objects also challenge control of identity and personal information disclosure in the IoT. Segmenting control have implications for privacy and confidentiality of the data of individuals and groups of users.

The challenges for individuals to have meaningful control over personal data are echoed in the GDPR.

::.. Sharing of Data/Identities (2/3)

- The right of access (Article 15) grants data subjects the right to request at any time information about the types of personal data that a data controller processes, and to receive a copy of the processed data (Article 15(3)). However, rights and freedoms of others (e.g. privacy, trade secrets, Article 15(4), Recital 63) can trump the latter.
- The right to rectification (Article 16) grants users to rectify inaccurate data and complete incomplete data. Finally, to mitigate undesired data usage, data subjects will – under certain circumstances (Article 17(1)) – be able to request erasure of their data, for example, when the data is no longer essential for purposes for which they were collected.

These rights foster the implementations of the GDPR's guiding principle of transparency (Article 5). Clarity is lacking on how the interests of data controllers and users will be balanced.

::.. Sharing of Data/Identities (3/3)

- Users have a clear interest in disclosure control to prevent privacy violations and discriminatory treatment, which must be balanced with the rights and freedoms of others (including data controllers).
- These rights also do not require data controllers to indicate precisely which data was (most) relevant to a particular decision or effect of identification or profiling.
- Individuals may have to comb through thousands of entries to identify potentially inaccurate, incomplete, or misleading data.

This creates an additional barrier to meaningful control of informational identity.

::.. Consent and Uncertainty (1/3)

Privacy-and trust-enhancing identity management and access control systems frequently use a user-centric definition of privacy and trust:

- Accordingly, a system is considered privacy- and trust-enhancing if it grants the users oversight and choice over the way in which their IoT devices communicate and take actions on their behalf, and thus how their IoT-generated data is shared with IoT devices, other users, and data controllers.

The actual protection offered by such schemes depends upon the quality of a user's choice in setting high- or low-level permission preferences. Similar to informed consent, user-centric models are only effective to that extent that users are able to express their interests related to privacy and trust, are informed of the potential risks of permitting communication or data sharing between IoT nodes, and are made aware of downstream risks resulting from these actions.

::.. Consent and Uncertainty (2/3)

GDPR's article 25 creates a general duty for data controllers to implement privacy by default and privacy by design mechanisms to resolve the uncertainty of privacy invasive analytics and thus offer a better basis for informed consent.

- If assured that privacy will be protected by default, the user can make an informed choice as the potential privacy consequences become foreseeable.

Article 25 states that data controllers need to take technical and organisational measures, such as data minimisation, storage and purpose limitations, but each measure faces implementation barriers.

- The 'data maximalism' reflected in the IoT's built-in reliance on vast data collection and sharing for personalised services suggests minimalism will be unlikely, or at least require significant trade-offs in terms of functionality by radically rethinking IoT applications and services.

::.. Consent and Uncertainty (3/3)

To inform about the potential risks of data collection, the GDPR has created higher standards relating to informed consent (Article 7) and notification duties (Articles 13 and 14).

- Article 7 states the need to properly communicate the possible risks of data processing, and the possibility of excessive data collection in making consent invalid.
- Articles 13 and 14 aim to make users immediately aware of the intended data collection without expecting users to read elaborate privacy notices.

::.. Honesty, trust, and transparency (1/3)

Trust relationships for sharing data can be defined at a device-to-device, device-to-user, and device-to-controller level, with permissions attached to the identity of specific devices, users, and controllers. Prevention of unauthorised objects and users from accessing a system can enhance confidentiality, and thus increase user trust.

- Trust between objects refers to authentication prior to communication and data access. In contrast, trust between users and objects addresses user perceptions and impressions of control.

“trust is a complicated concept with regard to the confidence, belief, and expectation on the reliability, integrity, security, dependability, ability, and other characters of an entity.” To achieve trust in an IoT system, developers must demonstrate how they plan to mitigate the significant legal and ethical risks associated with their devices. Developers often currently fail to demonstrate compliance with data protection rules, for example through accreditation programmes.

::.. Honesty, trust, and transparency (2/3)

The GDPR has several provisions that enhance trust and transparency.

- Data controllers will need to carry out a data protection impact assessment (DPIA) (Article 35), which can be useful to increase user trust, especially if published publicly, as recommended by Article 29.
- Article 13(2)(f) and 14(2)(g) aim to inform data subjects about "the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged con-sequences of such processing for the data subject." These provisions are meant to empower data subjects to know the risks when algorithmic methods are used prior to data processing (Article 12(7)).
- The guidelines suggest that data subjects should be made aware of the risks of profiling as soon as possible to enable them to exercise their rights. However, it is uncertain whether notification alone can achieve this goal as behaviour of advanced data protection means can be difficult to understand for lay people as well as experts.

::.. Honesty, trust, and transparency (3/3)

- The GDPR also relates trust to cybersecurity standards as having a secure system meeting at least recognised minimal standards can increase user trust.
- Articles 33 and 34 describe how data controllers must react to data breaches by informing the Supervisory Authority within 72 hours. In cases of breaches posing a high risk data controllers have to inform the data subjects. Each of these provisions can contribute to transparency and a trusting relationship between users and IoT device and service providers. However, as notification is only required in the case of ‘high risk’ breaches, how this thresh-old is defined generally and in specific contexts or use sectors will have significant impact on the actual protection offered to IoT users.

::.. Privacy-Preserving Methods (1/2)

Privacy issues in traditional Internet mostly impact connected users. However, in IoT scenarios, privacy concerns may affect people who are not even using any IoT service but happen to be in the environment.

- In traditional Internet services, the W3C group has defined the Platform for Privacy Preferences (P3P), which provides a standard language for the description of privacy preferences and policies.
- In IoT settings it is far complex to precisely capture privacy violations due to the lack of well-defined control boundaries.

Solution	Summary
Authentication and Authorization	(i) Lightweight authentication and key establishment mechanisms (ii) Frameworks based on device fingerprinting techniques (iii) Context-aware access control models and enforcing mechanisms
Edge Computing and plug in architecture	(i) Software modules on the edge to overcome privacy concerns (ii) Privacy aware systems to allow user control over data (iii) Decentralized architectures based on Personal-Cloud Butlers
Data Anonymizing and denaturing	(i) Data brokers and separation algorithms to offer flexibility to service providers, yet respect user-predefined access rules (ii) Generalization to mask personal data (iii) Frameworks that provide emotion analytics lifecycle to allow denaturing
Digital Forgetting and Data Summarization	(i) Delete encrypted data when decryption key is deleted (ii) Acquire only the strictly needed data rather than all data (iii) Apply knowledge discovery in databases and data mining technologies

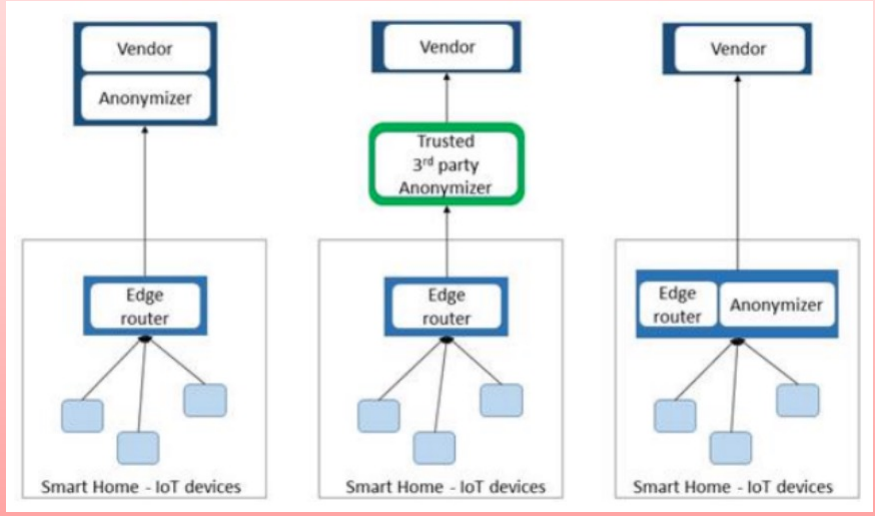
::.. Privacy-Preserving Methods (1/2)

Privacy issues in traditional Internet mostly impact connected users. However, in IoT scenarios, privacy concerns may affect people who are not even using any IoT service but happen to be in the environment.

- In traditional Internet services, the W3C group has defined the Platform for Privacy Preferences (P3P), which provides a standard language for the description of privacy preferences and policies.
- In IoT settin… …y violations due…

Solution	Summary
Authentication and Authorization	(i) Lightweight a… (ii) Frameworks … (iii) Context-awa…
Edge Computing and plug in architecture	(i) Software mod…
Data Anonymizing and denaturing	(i) … service … (ii) Generaliz… (iii) Frameworks … denaturing
Digital Forgetting and Data Summarization	(i) Delete encryp… (ii) Acquire only the strictly needed data rather than all data (iii) Apply knowledge discovery in databases and data mining technologies

A plug-in mediator solution to overcome the privacy concerns stemming from overcentralization of IoT systems.



Privacy mediators (i.e., trusted software modules) can be deployed into the data distribution pipeline to enforce the privacy policy specified by sensor/user.

::.. Privacy-Preserving Methods (1/2)

Privacy issues in traditional Internet mostly impact connected users. However, in IoT scenarios, privacy concerns may affect people who are not even using any IoT service but happen to be in the environment.

- In traditional Internet services, the W3C group has defined the Platform for Privacy Preferences (P3P), which provides a standard language for the description of privacy preferences and policies.
- In IoT settings it is far complex to precisely capture privacy violations due to the lack of well-defined control boundaries.

Solution	
Authentication and Authorization	
Edge Computing and plug in architecture	
Data Anonymizing and denaturing	(iii) Frameworks that provide emotion analytics lifecycle to allow denaturing
Digital Forgetting and Data Summarization	(i) Delete encrypted data when decryption key is deleted (ii) Acquire only the strictly needed data rather than all data (iii) Apply knowledge discovery in databases and data mining technologies

Data anonymization is the process of removing identifiable information that may lead to personal identification so that people/objects described by such data remain anonymous, to protect user privacy. Denaturing is the process of using image processing techniques to blur or altera specific part of the image to preserve personal privacy.

::.. Privacy-Preserving Methods (1/2)

Privacy issues in traditional Internet mostly impact connected users. However, in IoT scenarios, privacy concerns may affect people who are not even using any IoT service but happen to be in the environment.

- In traditional Internet services, the W3C group has defined the Platform for Privacy Preferences (P3P), which provides a standard language for the description of privacy preferences and policies.
- In IoT settings it is far complex to precisely capture privacy violations due to the lack of well-defined control boundaries.

Digital forgetting is the process of provably deleting all copies of adataset, while data summarization provides a high-level data abstraction to hide details or reduce granularity.

Solution	
Authentication and Authorization	
Edge Computing and plug in architecture	(iii) Decentralized … -Cloud Butlers
Data Anonymizing and denaturing	(i) Data brok… …ms to offer flexibility to service pr… …ser-predefined access rules (ii) C… …o mask personal data (… …eworks that provide emotion analytics lifecycle to allow …enaturing
Digital Forgetting and Data Summarization	(i) Delete encrypted data when decryption key is deleted (ii) Acquire only the strictly needed data rather than all data (iii) Apply knowledge discovery in databases and data mining technologies

::.. Privacy-Preserving Methods (1/2)

Privacy issues in traditional Internet mostly impact connected users. However, in IoT scenarios, privacy concerns may affect people who are not even using any IoT service but happen to be in the environment.

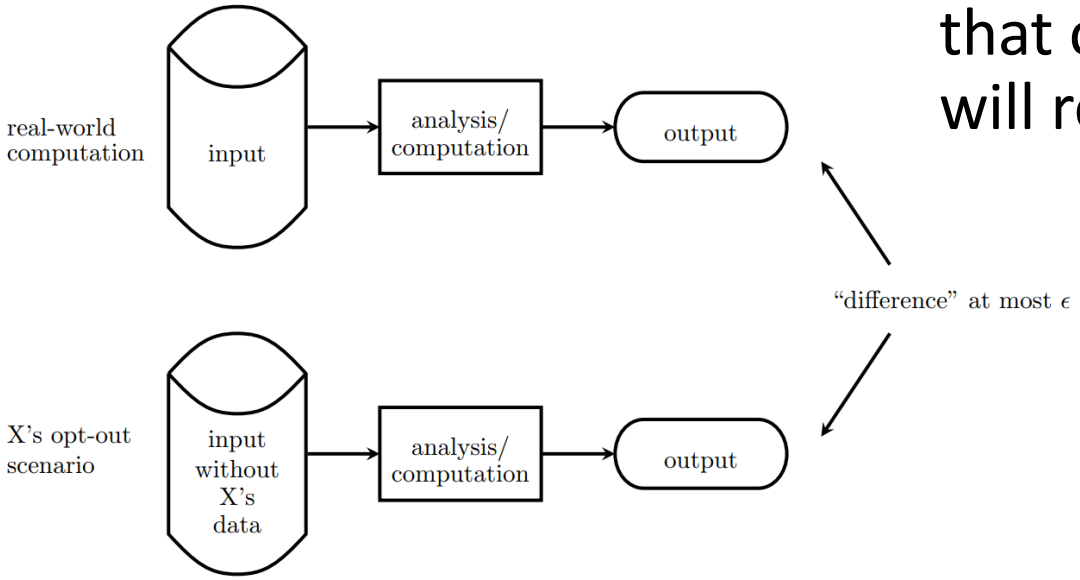
- In traditional Internet services, the W3C group has defined the Platform for Privacy Preferences (P3P), which provides a standard language for the description of privacy preferences and policies.
- In IoT settings it is far complex to precisely capture privacy violations due to the lack of well-defined control boundaries.

Solution	Summary
Authentication and Authorization	(i) Lightweight authentication and key establishment mechanisms (ii) Frameworks based on device fingerprinting techniques (iii) Context-aware access control models and enforcing mechanisms
Edge Computing and plug in architecture	(i) Software modules on the edge to overcome privacy concerns (ii) Privacy aware systems to allow user control over data (iii) Decentralized architectures based on Personal-Cloud Butlers
Data Anonymizing and denaturing	(i) Data brokers and separation algorithms to offer flexibility to service providers, yet respect user-predefined access rules (ii) Generalization to mask personal data (iii) Frameworks that provide emotion analytics lifecycle to allow denaturing
Digital Forgetting and Data Summarization	(i) Delete encrypted data when decryption key is deleted (ii) Acquire only the strictly needed data rather than all data (iii) Apply knowledge discovery in databases and data mining technologies

Differential privacy is a rigorous mathematical definition of privacy: by looking at the output of a data processing means, one cannot tell whether any individual's data was included in the original dataset or not.

::.. Privacy-Preserving Methods (2/2)

In other words, the guarantee of a differentially private algorithm is that its behavior hardly changes when a single individual joins or leaves the input data -- anything the algorithm might output on a database containing some individual's information is almost as likely to have come from a database without that individual's information. This gives a formal guarantee that individual-level information about participants in the database is not leaked.



This method does not guarantee that one believes to be one's secrets will remain secret.

The main idea of differential privacy is to perform random perturbations or adding random noice on the analysis data, such that any individual's presence in the data has negligible impact.

::.. Forgetting & Summarization

Digital forgetting and data summarization are important concepts to relieve people's anxiety around data collection. Moreover, as the cost of storage decreases, the ability to store a large amount of data at low cost dramatically increases. This makes easy collecting data for longer time. Hence, the need for efficient periodic information erasure arises.

Data summarization is classified into the following:

- Temporal summarization: collected data is a function of time;
- Spatial summarization: collected data is a function of location.

Several data forgetting techniques have been proposed using classical cryptography, which seams to be effective in the case of distributed data storage nodes as in the case of the IoT:

- Most of these techniques assume that encrypted data is deleted when the required decryption key is not available anymore.

::.. Sanitization Algorithms (1/4)

To ensure the balance between preserving privacy and guaranteeing data utility, there are sanitization algorithms with individual data or analysis results as inputs, privacy definitions as properties, and metrics for utility.

These algorithms sanitize to decrease the precision of data, within a certain acceptable limit so that the produced outputs remain useful, so that the a-priori knowledge of an adversary cannot increase after the observation of public data.

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

::.. Sanitization Algorithms (1/4)

To ensure the balance between preserving privacy and guaranteeing data utility, there are sanitization algorithms with individual data or analysis results as inputs, privacy definitions as properties, and metrics for utility.

The generalization algorithm groups data into buckets wherein all values of one attribute are replaced by a generalized or coarsened value for each bucket. The same attribute value is generalized differently when appearing in different buckets.

the observation of public data

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

Age	Sex	Zip code	Disease
20-29	Any	4300**	Heart disease
20-29	Any	4300**	Heart disease
30-49	Any	4300**	Flu
30-49	Any	4300**	Flu
50-69	Any	4300**	Heart disease
50-69	Any	4300**	Cancer
50-69	Any	4300**	Flu
50-69	Any	4300**	Cancer

::.. Sanitization Algorithms (1/4)

The suppression algorithm classifies records into private buckets and public buckets and then replaces some attribute values or parts of an attribute value with special symbols for each private bucket, indicating that the value has been suppressed.
For a partial suppression, an exact attribute value can be replaced with a less informative value by rounding, top coding, generalization, intervals, and so forth. In the latter two cases of generalization and intervals, the suppression algorithm is equivalent to the generalization algorithm.
The probabilistic suppression algorithm another technique that suppresses attributes with high posteriori probability to be sensitive after their processing.

the observation of public data

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

Age	Sex	Zip code	Disease
*	M	******	Heart disease
*	F	******	Heart disease
*	M	******	Flu
*	F	******	Flu
*	F	******	Heart disease
*	F	******	Cancer
*	M	******	Flu
*	M	******	Cancer

::.. Sanitization Algorithms (1/4)

To ensure the balance between preserving privacy and guaranteeing data utility, there are sanitization algorithms with individual data or analysis results as inputs, privacy definitions as properties, and metrics for utility.

These algorithms sanitize to decrease the precision of data within

The swapping algorithm swaps the values of sensitive attributes among records while ensuring that certain types of aggregate computations can be exactly performed without violating privacy.

the observation of public data.

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
57	M	430072	Flu
35	M	430071	Flu
59	F	430073	Cancer
54	F	430070	Heart disease
47	F	430079	Flu
23	F	430079	Heart disease
64	M	430072	Cancer

::.. Sanitization Algorithms (1/4)

The bucketization algorithm generally partitions the original data table into non-overlapping buckets; one bucket contains only sensitive attributes, and another bucket contains all general data, thereby de-associating the relationship between sensitive attributes and other attributes. Hence, the bucketization algorithm makes the critical attributes' value of an individual indistinguishable from that of any other individual in the same bucket. Slicing partitions attributes into several columns and allows a column to consist of general information attributes and sensitive attributes, thereby preserving the correlations between them, and preventing membership disclosure.

the observation of public data.

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

Age	Sex	Zip c	Bucket ID	Bucket ID	Disease
24	M	430072	1	1	Heart disease
23	F	430079	1	1	Heart disease
35	M	430071	1	1	Flu
47	F	430079	1	1	Flu
54	F	430070	2	2	Heart disease
59	F	430073	2	2	Cancer
57	M	430072	2	2	Flu
64	M	430072	2	2	Cancer

::.. Sanitization Algorithms (1/4)

To ensure the balance between preserving privacy and guaranteeing data utility, there are sanitization algorithms with individual data or analysis results as inputs, privacy definitions as

The randomization algorithm perturbs the attribute values of original records by adding noise. The noise added is sufficient to prevent the recovery of individual record values, and the statistical property of the sanitized data is indistinguishable with that of the original data. Therefore, randomization algorithms are designed to derive aggregate distributions from perturbed records.

the observation of public data

Age	Sex	Zip code	Disease
24	M	430072	Heart disease
23	F	430079	Heart disease
35	M	430071	Flu
47	F	430079	Flu
54	F	430070	Heart disease
59	F	430073	Cancer
57	M	430072	Flu
64	M	430072	Cancer

Sex	Disease	Count
M	Heart disease	1
M	Heart disease	1
M	Cancer	2
M	Flu	1
F	Flu	2
F	Heart disease	1
F	Cancer	

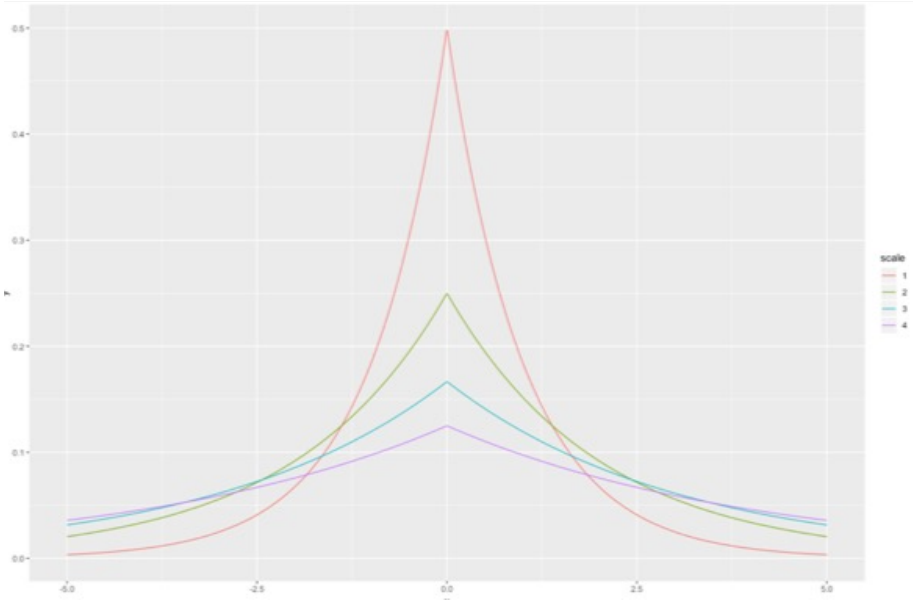
::.. Sanitization Algorithms (2/4)

To better understand the capacity and limits of noice adding, let's consider a concrete example of a IoT solution that collects the user ratings for a given business service.

Rating	Count
Bad	3
Normal	1510
Good	200

Let's assume that the adversary wants to know the number of people who have a bad service rating (which is 3 in the example).

The data is sensitive, so we cannot reveal the ground truth. Instead an algorithm that returns the ground truth, N = 3, plus some random noise will be used.



A random number L is choosen from a zero-centered Laplace distribution with standard deviation of 2, so that the sanitization algorithm returns N+L.

::.. Sanitization Algorithms (3/4)

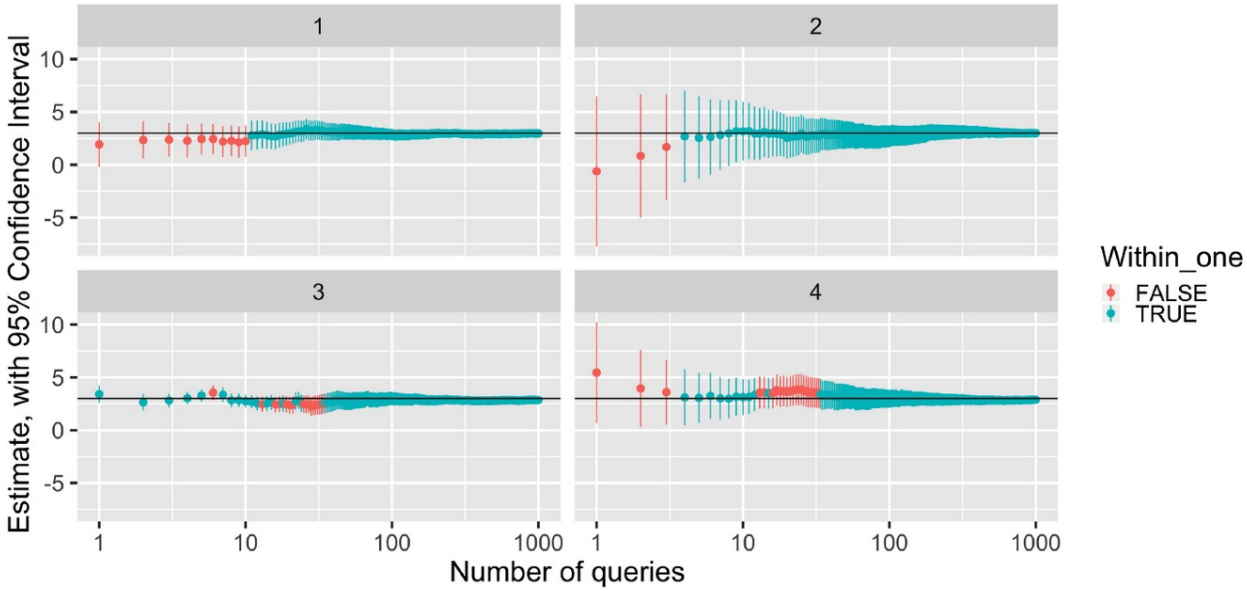
Reporting a nonsensical, noisy answer allows us to strike a balance between utility and privacy. The general public has to gain some utility from IoT database, without exposing any sensitive data to a possible adversary.

Query Number	Response
1	2.915
2	1.882
3	1.292
4	4.026
5	5.346
Average	3.090
90% confidence interval	1.696 to 4.484

What would the adversary actually receive? The first answer the adversary receives is close to, but not equal to, the ground truth, so it has been fooled.

Utility is provided, and privacy is protected; but after multiple attempts, the adversary will be able to estimate the ground truth. This “estimation from repeated queries” is one of the fundamental limitations of this sanitization algorithm. If the adversary can make enough queries to a differentially-private database, he/she can still estimate the sensitive data. The average is very close to the ground truth, but the 90% confidence interval from these 5 queries is quite wide, so that the estimate is not very accurate yet.

::.. Sanitization Algorithms (4/4)



Each vertical line illustrates the width of a 95% confidence interval, and the points indicate the mean of the adversary's sampled data. The noise is drawn from independent and identically distributed Laplace random variables.

Overall, the mean is noisier with fewer queries (since the sample sizes are small). As expected, the confidence intervals become narrower as the adversary makes more queries (since the sample size increases, and the noise averages to zero). Just from inspecting the graph, we can see that it takes about 50 queries to make a decent estimate.