Model-driven Evidence-based Privacy Risk Control in Trustworthy Smart IoT Systems

Victor Muntés-Mulero*, Jacek Dominiak†, Elena González‡
Beawre Digital SL
Barcelona, Spain
{*victor.muntes, †jacek.dominiak, ‡elena.gonzalez}@beawre.com

David Sanchez-Charles
Trialog SA
Paris, France
david.sanchez@trialog.com

***Abstract*—Preventing privacy-related risks in the creation of Trustworthy Smart IoT Systems (TSIS) will be essential, not only because of the growing amount of regulations that impose strict mechanisms to control risks and quality, but also to effectively mitigate the effect of potential threats exploiting vulnerabilities that may jeopardize privacy. While privacy-related risks usually consider assets represented by elements of the functional description of a TSIS, most monitoring efforts are focused on monitoring security aspects related to the system architecture components and it is not straightforward to link the evidences collected through this monitoring systems to functional description elements, making it difficult to use this information for privacy-related risk management. In this paper, we propose a methodology for continuous risk management and a model for risks to increase trustworthiness in IoT systems by enabling continuous monitoring of privacy-related risks. Our approach is based in connecting our risk model with a modelling language to describe IoT architectures, such as that proposed in GeneSIS, and Data Flow Diagrams (DFD). With the combination of architecture (technical description) and data flow models (functional description), we enable continuous risk management using monitoring to improve risk assessment related to data protection issues, as required by GDPR.**

***Index Terms*—Risk management, Trust, IoT, Continuous, Privacy, Security**

I. Introduction

Until now, IoT system innovations have been mainly focused on sensors, device management and connectivity, usually aimed at gathering data for processing and analysis in the cloud[1]. The next generation IoT systems need to perform distributed processing and coordinated behaviour across IoT, edge and cloud infrastructures, manage the closed loop from sensing to actuation, and cope with vast heterogeneity, scalability and dynamicity of IoT systems and their environments [9]. To unleash the full potential of IoT, it is essential to facilitate the creation and operation of trustworthy Smart IoT Systems or, for short, TSIS. TSIS typically operate in changing and often unpredictable environments. Thus, the ability of these systems to continuously evolve and adapt to their new environment is essential to ensure and increase their trustworthiness, quality and user experience. Besides, by 2021, the number of connected things will grow to 25 billion, according to Gartner[2]. Thus, processes that were formerly run by humans will be automated, making it much more difficult to control data ownership, privacy and regulatory compliance.

Yan et al. [27] described the different dimensions of trust for IoT systems, concluding that risk management is essential to guarantee trustworthiness. Markets in the need of TSIS, such as eHealth, are just flourishing and businesses will be continuously adapting to new technologies. In this context, poor risk management together with a reactive strategy usually forces companies to continuously re-factor application architectures to improve software quality and security, incurring high re-implementation costs [5]. In general, there is the lack of solutions to support continuous control of risks through evidence collection. Companies have little control on actual effectiveness of the mitigation actions defined during risk management process. Besides, many companies fill this gap by using manual procedures based on storing all the information in spreadsheets, by departments and locally [1]. This approach rapidly turns inefficient as projects or teams grow.

In parallel, GDPR discusses data protection by design and by default, remarking that it is essential to consider privacy from the beginning to address related issues successfully. This is specially true in the IoT arena, where technologies are not consolidated yet and mixing legal requirements with a deep technical understanding is challenging. In particular, including privacy aspects in a continuous risk management process is difficult. Continuous evidence collection to support risk management is usually key, but most monitoring approaches focus on collecting evidences from the TSIS infrastructure or technical architectural components. However, privacy-related risks are usually detected by analyzing functional descriptions of the system [26] (e.g. data flows). Connecting this functional level with the components of the architecture that are being monitored is not trivial. Recognizing the overlap between privacy and security is key to determining when existing security risk models may be applied to address privacy concerns [6].

In this paper, we improve continuous risk management in TSIS development by embedding privacy-related risks explic-

This work is supported by the European Commission through the *ENACT* project under Project ID: 780351 and the *PDP4E* project under Project ID: 787034.

[1]IEC white paper entitled IoT 2020: Smart and secure IoT platform. http://www.iec.ch/whitepaper/pdf/iecWP-loT2020-LR.pdf

[2]https://www.networkworld.com/article/3322517/internet-of-things/a-critical-look-at-gartners-top-10-iot-trends.html

itly through the combined used of models for both the architecture and the data flow implemented on the components of the architecture. We present a new approach that, through linking GeneSIS [8] models and Data Flow Diagrams (DFDs) [7], improves risk assessment process for privacy-related risks. We achieve this by enabling the use of the information that is typically collected from the infrastructure to control security, thanks to the link that LINDDUN [26] establishes between privacy and security threads in STRIDE [22].

This paper is organized as follows. Section II provides an analysis of the state of the art. In Section III, we describe a usual use case for our proposal. Section IV presents a brief analysis of existing approaches and risk guidelines in standards. Section V presents our methodology and a class diagram model to describe the concepts we use in our risk management methodology, and describe the mapping between the architecture model and DFDs as well as our risk model. Finally, in Section VI, we draw some conclusions.

II. State of the art

There exist quantitative risk methodologies and tools, like RiskWatch[3] or ISRAM [13] and qualitative risk methodologies such as OCTAVE [2], CORAS [14] or STRIDE [22]. Traditional risk management methodologies focus on the assessment of risks at a singular stage in time, and do not address the challenge of managing continuously evolving risk profiles.

Managing continuous changes in software development has been studied from different perspectives. The most common practice in industry is continuous software integration [24]. Fitzgerald et al. [10], for instance, published a roadmap and agenda for continuous software engineering. Besides, there is a growing interest on security and privacy and this has motivated work on continuous security and regulatory compliance. Continuous security [16] was proposed to prioritize security throughout the whole software development life cycle.

There has been work on risk management in complex and evolving environments. For instance, different papers [12], [20] propose managing risk related to accountability, assurance, agility or financial aspects in multi-cloud applications. Risks analysis is then used to guide the selection of cloud service providers. Shrivastava et al. [23] present a risk management framework for distributed agile development, studying risks related to software development life cycle, project management, group awareness, etc. However, they focus on analysing risk factors that represent a threat to the successful completion of a software development project, rather than risk related to non-functional requirements such as security or privacy. In [3], the authors performed a systematic literature review on risks and control mechanisms for distributed software development. Moran [17] explicitly tackles issues related to risk management for agile software development. Moran proposes a risk modified kanban board and user story map. Finally, in [18], a framework to manage risks that supports collaboration, agility, and continuous development is proposed. None of these contribution tackles how to monitor privacy-related risks for continuous risk management.

In general, even for security, risks are not analysed and monitored continuously and historical data is not taken into account [1], [4]. Besides, security risks are assessed based on speculation rather than evidence [21], [25]. The interaction between analytics capabilities and information security risk management (ISRM) capabilities can help organizations to perform continuous risk assessments and enable evidence-based decision making [19].

In parallel, Article 25 in GDPR[4] discusses data protection by design and by default, underlining that considering privacy from the beginning is essential to address privacy successfully. GDPR establishes binding data protection principles, individuals rights, and legal obligations to ensure the protection of personal data of EU citizens. However, legal measures need to come along with technical measures to protect privacy and personal data in practice. PDP4E H2020 project [15] focuses on the importance to involve engineers in the loop, for Privacy-by-Design (PbD) to be viable.

A. *MUSA Risk Assessment tool*

The MUSA Risk Assessment tool[5] is a reference implementation of the continuous risk management framework proposed by [18]. The tool uses a pull system in the style of Kanban, where the status of each asset with respect to a risk analysis methodology is expressed through the different columns in the Kanban board. Albeit this approach makes the tool agnostic to any specific risk analysis methodology, the tool showcases a methodology that is tailored to multi-cloud environments.

In order to assess the risks in the different components of an application, the tool asks users to provide an informal description of the system and its components. Then, users are free to choose among pre-defined threats that may potentially affect each individual component. Once threats are selected, they are automatically classified in the STRIDE security-oriented framework (*Spoofing identity, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege*). Then the user can continue with the assessment, evaluation and posterior mitigation of the risk. We take this tool as the baseline for the proposal presented in this paper.

B. *GeneSIS Modelling Language*

Recent work in the ENACT H2020 project [9] defined TSIS as the next generation IoT systems, capable of performing distributed processing and coordinated behaviour across IoT, edge and cloud infrastructures. GeneSIS [8] facilitates the development and continuous deployment of TSIS, allowing decentralized processing across heterogeneous IoT, edge, and cloud infrastructures. GeneSIS includes a domain-specific modelling language to model the architecture of TSIS as well

[3]RiskWatch: https://www.riskwatch.com . Accessed: 2019-07-18

[4]Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, L119:188, May 2016.

[5]https://musa-project.eu/node/326

as the orchestration and deployment. In this paper, we will base our Risk Management methodology in the GeneSIS domain-specific modelling. Figure 1 shows a high level description of the generic elements modelled through GeneSIS.

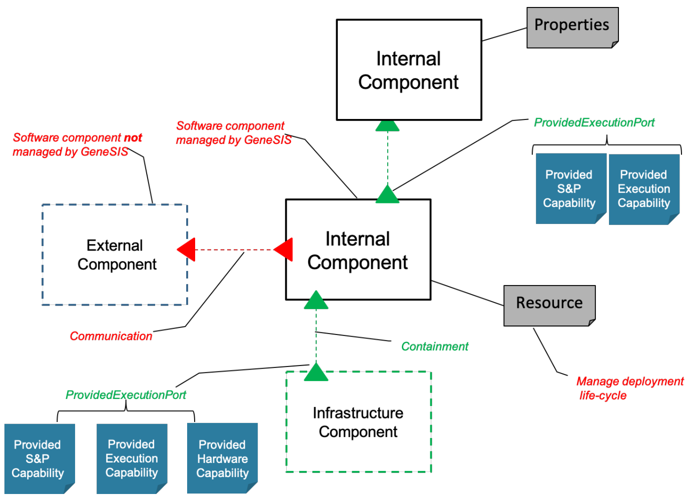

Fig. 1. GeneSIS concepts [8].

GeneSIS modelling language is an extension of ThingML [11]. ThingML is a domain specific language for modelling distributed IoT systems including the behaviour of the distributed components in a platform-specific or -independent way. Currently, GeneSIS engine maintains a list of the generic component types that are available in the registry, namely FIWARE Orion, Arduino, RFXCom, and ThingML. For instance, a component written in ThingML will use the ThingML type. However, GeneSIS is very flexible and allows any user to create new component types at any level of abstraction (e.g. *"SQL database"* or even *"Oracle SQL database"*). Any of these types can be modelled as a subtype of an *Internal*, an *External* or an *Infrastructure Component* (see Figure 1). Communication channels can be established between internal and external components and also between internal components through execution ports.

C. LINDDUN

LINDDUN [26] is a threat modelling methodology that encourages risk analysts to address privacy risks affecting end-users of the application or system. This methodology provides some guidance to identify and categorize threats under a set of general risks (*Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance*). LINDDUN is sometimes considered the privacy-oriented alternative to the STRIDE framework. In fact, LINDDUN threats are described in the so-called LINDDUN trees which are explicitly connected to STRIDE threats trees [22]. We take this link between privacy and security as the baseline for our proposal in this paper.

Unlike the approach proposed by MUSA Risk Assessment tool, which grounds its threat modelling on descriptions of individual components, the LINDDUN methodology requires to formalize the functionality of the system and their dependencies with respect to personal data. In such sense, LINDDUN proposed the usage of the Data Flow Diagrams [7]. The notation of a DFD is based upon 4 distinct element types: (i) an external entity (i.e., end-users or third party services that are external to the system), (ii) a data flow (explains data propagation and dependencies between all the functional components), (iii) a data store (i.e., a passive container of information) and (iv) a process (i.e., a computation unit).

III. Use Case Motivation

We devote this subsection to explain the usual scenario we may find in a software company that offers Trustworthy IoT solutions. The aim of this section is to provide some context for the paper rather than characterizing the generic organization of TSIS companies, as this would require conducting an exhaustive and more accurate study.

The scenario discussed is particularly relevant in highly regulated markets where a company needs to prove they follow a risk-based approach as required by GDPR or comply with existing standards such as ISO 27001 for security for instance. This company needs to gain trust because it is dealing with data that may be crucial about users (e.g. health data about remotely monitored patients at home). Content of this section is based on our experience with our customers.

Companies in highly regulated markets usually have risk management owners in the organization. The exact role they play may depend on the type of company and the type of requirements this company may have. For instance, the company may have a Chief Security Officer (CSO), a Data Protection Officer (DPO) as requested in some situations by GDPR, etc. In large organizations, it is common to have risk analysts to support the risk management process, while in SMEs this is typically a role played by individuals whose background and knowledge is not on risk management. Companies that are truly following DevOps principles, tend to make decisions around risk in a collegial manner, involving product owners, risk analysts, developers, operations, etc.

In this context, companies need to face many challenges, including coping with rapid software evolution, the need to obtain and keep relevant certificates to gain the trust of their customers, the need to prepare for any unwanted incident that may negatively impact their businesses, etc. For this, they need to prepare internal policies to describe their procedures to handle risk management. Among these procedures, frequent meetings are usual where they need to rapidly understand the status of risks in their projects, as well as, to reshape or re-prioritize risk plans to better accommodate the current situation and market. When these needs are crossed with IoT, proper risk management procedures become even more significant. Requirements change frequently, technology evolves rapidly and, consequently, companies adopt agile software development processes. Therefore, continuous risk management is probably the only effective strategy to ensure that risks are properly mitigated in long-term development processes.

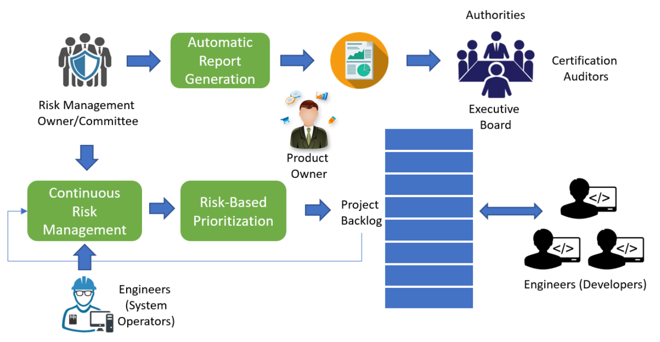
Authorities
Automatic Report Generation
Certification Auditors
Risk Management Owner/Committee
Executive Board
Product Owner
Continuous Risk Management
Risk-Based Prioritization
Project Backlog
Engineers (Developers)
Engineers (System Operators)

Fig. 2. High-level overview of the tasks around risk management that are relevant in the creation of TSIS.

Figure 2 shows the high-level description of a typical risk management processes implemented in software companies. A company may be developing software in the form of a product or SaaS (e.g. connected to a trustworthy IoT system). Each product will have a person playing the role of the product owner, who will supervise the whole development of that product. Apart from this role, the company will typically have a risk owner, who owns the risk management process and strategy. This role may also take the form of a committee, composed of different people playing different roles, and bringing different perspectives, including Chief Product Officers, Chief Technology Officers or Chief Operation Officers. Risk owner will typically start and contribute and monitor the risk management process. Engineers may also be involved in this process including developers and operators, to empower a DevOps approach. Architects and Product Owners will also be involved in the risk management process. An efficient risk management process will help the organization to understand risk level associated to detected risks and prioritize the implementation of mitigation actions (or treatments or controls) in the form of new product features. Finally, risk management owner(s) needs to report the status of risk management.

In this context, companies lack mechanisms to continuously monitoring processes. In particular, there is a lack of mechanisms to monitor privacy-related issues. They require mechanisms to monitor the implementation of related mitigation actions and the effectiveness of the treatments.

IV. Alignment with Standards and Best Practices

While most risk management methodologies presented in the literature and accepted by the international community through standards, scientific work or other best practices are similar, they also differ in some aspects. Following, we analyze some well-known risk management methodologies or risk management guidelines proposed in standards, focusing in particular in those related to privacy issues. In particular, we explore the following best practices in industry and some previous related FP7 and H2020 projects:

- Risk management methodologies used in MODAClouds[6] and MUSA[7] (and CORAS methodology implicitly): MODAClouds risk management methodology has a strong influence from CORAS, simplified to favour usability. It was later on inherited and refined in MUSA. We use it as the baseline of our methodology.
- ISO/IEC 29134:2017 : it gives guidelines for: (i) a process on privacy impact assessments, and (ii) a structure and content of a Privacy Impact Assessment (PIA) report. PIAs include a methodology for risk management.
- ISO/IEC 27001:2013 : it specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization. It also includes requirements for managing information security risks.
- ISO/IEC 27550 (to be published in 2019) : this standard complements ISO/IEC 27001:2013 by providing an engineering, privacy-oriented perspective. LINDDUN is mentioned as a methodology in this risk-oriented standard.
- ISO 31000:2018 : it provides guidelines on managing risk. It can be used throughout the life of the any organization and can be applied to any activity, including decision-making at all levels. Since it is the most generic standard to describe risk management activities and it is agnostic to a particular context, we take it as a general reference for our methodology.

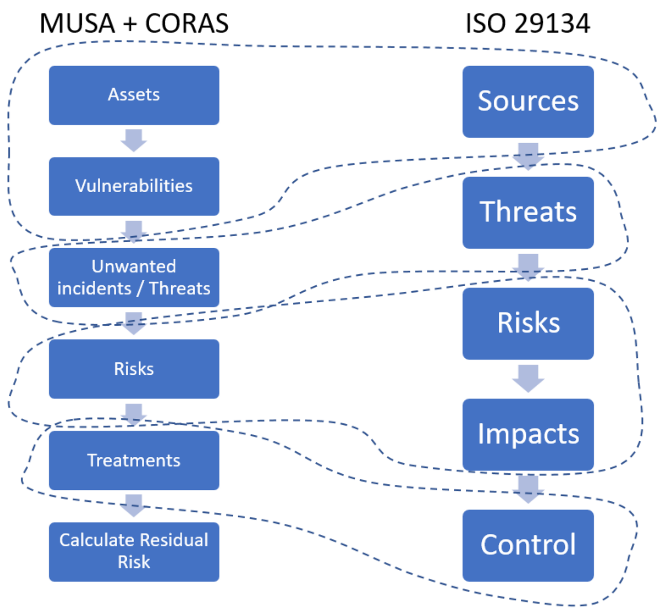

Fig. 3. Comparison between the methodology followed by the Risk Assessment tool of MUSA (inspired by CORAS) and ISO/IEC 29134:2017.

Figure 3 shows a visual summary of the main steps followed by the risk management methodology in MUSA and the steps suggested in ISO/IEC 29134:2017. While the vocabulary is not identical, the processes are similar, and it is possible to establish reasonable mappings among them. For instance, in MUSA

[6]MODAClouds FP7 (Project id: 318484) www.multiclouddevops.com

[7]MUSA H2020 (Project id: 644429) www.musa-project.eu

assets and vulnerabilities are defined and threats are identified with respect to those. In ISO/IEC 29134:2017, the definition of assets and vulnerabilities is quite ambiguous, but they put the emphasis in the description of risk sources. Both define threats (i.e. unwanted incidents in CORAS) and then risks. In general, a risk is an unwanted incident which likelihood and impact have been analyzed. Some methodologies talk about treatments, while others talk about controls. In general, these are all different terms to refer to mitigation actions.

V. Model-based Continuous Risk Management

In this section, we will present our risk management methodology, focusing in particular on capturing those aspects that make the risk management process suitable to enable continuous risk management on privacy-related issues.

Figure 4 presents a class diagram to model risk management concepts to allow to control privacy risks using DFDs. In particular, we consider each element of a DFD (Entity, DataFlow, DataStore and Process) a specific asset, following LINDDUN methodology. This diagram is inspired by the UML diagram presented by Gupta et al. [12]. We generalize that UML diagram eliminating the part of the model that was specific to cloud aspects. We also introduce the idea of a vulnerability associated to a combination of different assets in the system. In order to consider vulnerabilities and unwanted incidents on multiple assets, we propose a new AssetSet class and link vulnerabilities to this class (in addition to the link to Asset class). Finally, note that when simplifying the risk management process is important, the explicit description of vulnerabilities may be taken into consideration implicitly. Therefore, an alternative UML diagram could be considered that connects Asset class with Unwanted Incident class.

In Figure 5, we propose a methodology for risk management for the creation of TSIS and we indicate the actors involved in each of those steps. Our methodology, inspired by the previous analysis, can be summarized in 6 main steps:

- **S1: TSIS Assets Definition**: first, we edit or load DFDs representing the functional description of the system. This DFDs are useful to manage risks related to privacy as described by LINDDUN, although they were initially used for security risk control (e.g. in STRIDE). Although

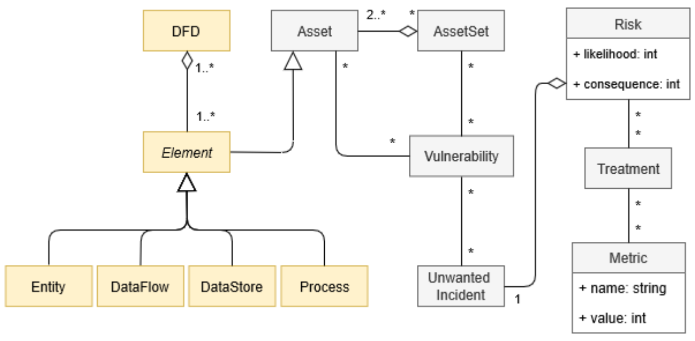

Fig. 4. Class diagram to model risk management concepts using DFDs to describe system functionality.

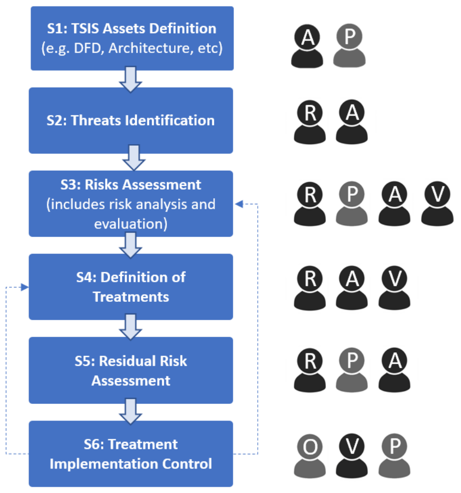
S1: TSIS Assets Definition (e.g. DFD, Architecture, etc)
S2: Threats Identification
S3: Risks Assessment (includes risk analysis and evaluation)
S4: Definition of Treatments
S5: Residual Risk Assessment
S6: Treatment Implementation Control
A P
R A
R P A V
R A V
R P A
O V P

Fig. 5. Proposed Model-based Risk Management methodology. Roles: (O) Risk Management Owner; (P) Product Owner; (A) Architect; (V) Developer and (R) Risk Analist

DFDs represent our primary assets in the risk management process, our proposal entails the use of a second layer of supporting assets in the form of TSIS archictecture description. Therefore, in S1, GeneSIS Models (or equivalent architectural models) are also traversed and components are pre-loaded. This step also includes the definition of the vulnerabilities related to a component of an architecture or a subset of components.
- **S2: Threats Identification**: in this step, users identify threats that may affect the components in the described system. These threats are captured as instances of the Unwanted Incident class in the UML diagram in Figure 4.
- **S3: Risk Assessment**: risk assessment is composed of two different steps: risk analysis, where risks are evaluated in terms of likelihood and consequence, and risk evaluation, where risks are accepted, or they are classified as risks that need to be mitigated.
- **S4: Definition of Treatments**: mitigation actions are defined in the form of treatments.
- **S5: Residual Risk Assessment**: once the mitigation controls are defined, the residual risks need to be reassessed. This involves again two steps: risk analysis, where likelihood and consequence are updated after the application of the control(s), and risk re-evaluation, where risks are analysed again, and they are classified as accepted or further mitigation actions required.
- **S6: Treatment Implementation Control**: finally, we add a last step in the methodology that goes beyond other previous methodologies. In particular, it involves the monitoring of the effectiveness of the mitigation actions proposed in the previous step. This step requires the connection to data collectors or agents that collect evidences from a monitoring system in order to match them to treatments and risks. Mapping of DFDs and architectural models will be the key for this to be possible.

TABLE I
CONNECTION BETWEEN LINDDUN THREAT CATEGORIES (PRIVACY) AND STRIDE THREAT CATEGORIES (SECURITY).

	L	I	N	D	D	U	N
Entity	Information Disclosure at data flow (between user and service)		▨	▨	Information Disclosure	▨	▨
Data flow	Information Disclosure at data flow		▨	▨	Information Disclosure	▨	▨
Data store	Information Disclosure at data store Spoofing External Entities Tampering against authorization process				Information Disclosure	▨	Tampering with policy data store
Process	Information Disclosure of a process Spoofing External Entities Tampering against authorization process Tampering threats against persistent storage/process blocks				Information Disclosure	▨	▨

Note that this methodology is continuous in the sense that new evidence is continuously collected to challenge previous knowledge upon which risk-related decisions were made in the past. While immediate triggering of mitigation actions is not our main focus, conditional options based on the status of the system could be included in S4 to strengthen the continuous approach. Also, the methodology is particularly customized for TSIS, specifically in S1, where GeneSIS models are used as the baseline, and S6, where evidences are collected to consider privacy-related issues, as we show later on in this section. However, the methodology can be easily adapted to other types of systems and used generically. This will mainly depend on the models and the risks and mitigation actions defined.

Model-based Risk Management approach

GDPR establishes a set of duties imposed on the data processors, controllers and third parties which are aimed at honoring the corresponding data subjects rights. In GDPR, risk is explicitly scoped (Rec. 76) with regards to the *rights and freedoms of the data subject*. In order to understand whether these data subject rights and principles (described in GDPR) are being effectively protected by mitigating existing risks, current approaches tend to analyze the data flow in the system (e.g. LINDDUN and DFDs). However, elements in a DFD tend to be defined at a functional level (e.g. *process to collect profile data from a user* or *process to merge data from two existing data sources*). In this situation, collecting system data to monitor a threat related to the *identifiability of an entity* as defined by LINDDUN, just to take an example, is not obvious, if we do not understand the relationship of a data flow in the application with the architecture of the infrastructure that we monitor. Without a proper connection of these elements to architectural levels that can be monitored, continuous risk management for privacy becomes much more difficult. Most monitoring tools monitor the infrastructure level. These tools may monitor system aspects in a data center or a particular server, tools for network monitoring, etc, and collect metrics.

The main contribution of the approach proposed in this section is the mapping of TSIS architecture models, such as the one proposed by GeneSIS, with data flows in the application under analysis (e.g. DFDs). In S6, we may assume that the elements from both models have been mapped establishing a link between the two types of models. Details on this mapping are provided below. For instance, a NoSQL data store in our system architecture may be mapped to a data store component in a DFD. As another example, a particular process element in a DFD may be mapped to the server the process is running on. As an alternative, it is possible that the mapping is not pre-established before starting the risk analysis process. In this situation, we consider that phase S1 can be extended to allow the user to establish this mapping manually.

1) Model mappings: in order to drive the mapping between the architecture model and DFDs, we take as the baseline the connection between privacy-related threats studied in LINDDUN and the vulnerabilities that are related to security aspects of the architecture, as captured by STRIDE. Please note that LINDDUN defines a connection between threats in LINDDUN categories and STRIDE threats through their threat trees catalogs [26]. Table I is a summary of the analysis of the connection of these LINDDUN and STRIDE threat catalogs.

In Table II, we show a sample of vulnerabilities extracted from [22] related to the STRIDE threats analyzed in Table I. The table shows a brief description and some usual mitigation actions. In the last two columns, we show examples of metrics that could be used for continuous security monitoring and what type of architectural components are being monitored. We would like to remark that this table is not meant to be an exhaustive list of all the potential threats, but some examples for illustrative purposes that serve the purpose of analyzing which DFD components are to be mapped to which architectural components.

Note that these vulnerabilities are specially relevant in an IoT context. For instance, let us take the example in Table II related to Information Disclosure of Data Flow. There may be different vulnerabilities associated in particular to TSIS. For instance, if we have a device IoT hub, eavesdropping or interfering the communication between the device and the gateway would be a potential threat. You may also find similar threats in the communication between devices, where data may be read in transit, tampering with the data or overloading the device with new connections. Or even with the cloud gateways, where eavesdropping or communication interference between devices and gateways may occur.

In general, mappings between architectural models and DFD will link:

- entities in DFDs with the associated network channels in

TABLE II
SAMPLE OF THE MAPPING BETWEEN STRIDE THREATS (RELATED TO LINDDUN THREATS IN TABLE I) AND RELATED METRICS THAT CAN BE DETECTED FROM MONITORING ARCHITECTURAL COMPONENTS.

STRIDE Threat	STRIDE tree node [22]	Description	Key mitigation actions	Metrics for continuous monitoring	Monitored Components
Spoofing External Entities	Transit	An attacker copies an authenticator from a non-encrypted channel or tamper with the connection.	Use standard authentication protocols, instead of your own. Use strong encryption and authentication.	Continuous authenticator detector warnings (comparing data in the channel with user database)	Network / Communication Channels
	Obtain credential from storage at a 3rd party	Reuse of passwords is common. 3rd parties may leak passwords your customers use.	Avoid static passwords. Avoid using e-mail addresses as usernames. Detect brute-foce attempts, or increase in successful logins from new locations.	Monitoring of number of brute-force attempts or successful logins from a new location or IP. Continuous user behavior analysis.	Network / Communication Channels
	Predictable credentials	Predictable usernames or a poor random generator for passwords.	If assigning passwords, use strong randomness.	Detect usernames sent matching actual user names in your internal databases.	Database / Data store
Information Disclosure at Data Flow	No or Weak Confidentiality (Observing a Message Structure)	Content of a message not protected or weakly protected.	Cryptographic. Use available message protection add-ons or tunneling.	Continuous monitoring message content readability. Can we detect some content structure or detect words with meaning?	Network / Communication Channels
	No or Weak Confidentiality (Observing a Channel)	No or weak protection of channel contents. Data about the messages can be revealing.	Encrypt the channel. Tunneling.	Continuous channel monitoring content readability. Can we get some structure of the content? Or detect words with meaning?	Network / Communication Channels
Tampering at Data Store	Bypassing protection rules because of no or weak protection	ACLs, permissions, policies, etc allow people to alter data without a clear justification.	Ensure data is created with appropriate permissions. Change permissions.	Random data editor reporting the number of successful attempts to change data with no or low privileges.	Database / Data store
Information Disclosure of a Process	Timing	Code time execution can reveal confidential information.	Design for cryptography to take constant time.	Monitoring execution time and control variability.	Software components to solve tasks requiring secrecy.

which entity information is transmitted or data stores in which the information is stored;

- data flows in DFDs with the corresponding network channels;
- data stores in DFDs with the corresponding databases or data stores used in the system architecture; and
- processes in DFDs with the corresponding software components executing tasks related to those processes, specially when these tasks entail some level of secrecy.

More formally, we define a set of DFDs, which represent a functional description of the system, $D = \{D_1, D_2, \dots, D_n\}$, as a set of graphs $D_i = (DV_i, DE_i)$, where $DV_i = \{e_1, \dots, e_m\}$ are the elements in the DFD representing elements of the system architecture at the functional level. Each element e_i can be an *external entity*, *data store* or a *process*. $DE_i = \{f_1, \dots, f_k\}$ represents the *data flows* connecting elements in DV_i. We define a graph $A = \{V_A, E_A\}$ where $V_A = \{c_1, \dots, c_m\}$ represent the architectural components at the technical level represented in a model such as those created by GeneSIS and $E_A = \{l_1, \dots, l_k\}$ represents the connections between any two components (c_i, c_j) in the system architecture. Note that components can be both software or hardware components. We define a mapping between the two models D_i and A as a function $\mathcal{M} : DV_i \cup DE_i \rightarrow V_A \cup E_A$.

2) Exploiting model mappings: once the mapping is established, then a final step is necessary to link metrics, such as those presented as examples in Table II, to the related risks or mitigation actions defined in S3 and S4. Following the risk methodology presented above, we will obtain instances for the rest of classes in the diagram in Figure 4, including risks and mitigation actions. Once the risk model is developed for each DFD, we propose two ways to exploit the established mappings to enable continuous risk management:

- *Automatic likelihood recalculation*: during risk assessment in S3, an initial likelihood and consequence values are determined for each risk. However, a basic principle for continuous risk management is that conditions in our system may change as time goes by. Let e be an element in a DFD D_i, $e \in DV_i \cup DE_i$, considered an asset in the risk management process. In S3, we define risks $r_1, \dots, r_j$ associated to e. Given a system architecture defined through a model $A = \{V_A, E_A\}$, we define a set of metrics $m_1^c, \dots, m_h^c$ for each component $c \in V_A \cup E_A$. Automatic likelihood recalculation involves defining a function f for each risk r related to each asset e such that $c = \mathcal{M}(e)$ and $f(m_1^c, \dots, m_h^c)$ provides a new value for the likelihood or risk r. With this, we connect the metrics defined for the architectural components and use them to calculate the likelihood of risks connected to elements of the functional description of a system, in the corresponding DFD.
- *Treatment effectiveness control*: a parallel approach to

enable continuous risk management, involves the definition of a function g for each r related to an asset $e \in DV_i \cup DE_i$ such that $c = \mathcal{M}(e)$ and $g(m_1^c, \ldots, m_h^c)$ represents a KPI that needs to be satisfied for a risk to be considered effectively mitigated. If g exceeds a particular threshold, then a warning needs to be issued for the risk plan including mitigation actions to be reviewed.

VI. Conclusions

Handling privacy-related risks is not well understood in the IoT context. There is a lack of mechanism to collect meaningful evidences and continuously monitor these risks. We need to find the intersection between data protection challenges and the enabling of TSIS. We have presented a first step towards the enactment of continuous privacy-related risk control for TSIS, leveraging the link offered by LINDDUN between privacy and security threat categories and combining functional and architectural models. As future work, we need to establish a stronger connection between privacy threat models like LINDDUN and the actual requirements imposed by GDPR, as current threat models were devised before GDPR and they may not fully cover all derived requirements. In particular, the relationship between data subject rights and GDPR principles and LINDDUN categories is unclear. Besides, we need to understand how other privacy-related evidences can be collected beyond those collected for security purposes. In particular in TSIS, where complex and distributed systems increase system weaknesses and the attack surface.

Acknowledgment

The authors would like to thank anonymous referees for their valuable comments and suggestions. We would also like to thank SINTEF and the GeneSIS team for allowing us to use some figures generated by them.

References

[1] A. Ahmad, J. Hadgkiss, and A. B. Ruighaver, “Incident response teams–challenges in supporting the organisational security function,” *Computers & Security*, vol. 31, no. 5, pp. 643–652, 2012.

[2] C. J. Alberts and A. Dorofee, *Managing Information Security Risks: The Octave Approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[3] A. Aslam, N. Ahmad, T. Saba, A. S. Almazyad, A. Rehman, A. Anjum, and A. Khan, “Decision support system for risk assessment and management strategies in distributed software development,” *IEEE Access*, vol. 5, pp. 20 349–20 373, 2017.

[4] R. Baskerville, P. Spagnoletti, and J. Kim, “Incident-centered information security: Managing a strategic balance between prevention and response,” *Information & management*, vol. 51, no. 1, pp. 138–151, 2014.

[5] B. Boehm and R. Turner, *Balancing agility and discipline: A guide for the perplexed, portable documents*. Addison-Wesley Professional, 2003.

[6] S. Brooks, S. Brooks, M. Garcia, N. Lefkovitz, S. Lightman, and E. Nadeau, *An introduction to privacy engineering and risk management in federal systems*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[7] T. DeMarco, “Structure analysis and system specification,” in *Pioneers and Their Contributions to Software Engineering*. Springer, 1979, pp. 255–288.

[8] N. Ferry, P. H. Nguyen, H. Song, P.-E. Novac, S. Lavirotte, J.-Y. Tigli, and A. Solberg, “Genesis: Continuous orchestration and deployment of smart iot systems.” In the proceedings of the IEEE COMPSAC conference, Milwaukee, USA, July 15-19. Springer, 2019.

[9] N. Ferry, A. Solberg, H. Song, S. Lavirotte, J.-Y. Tigli, T. Winter, V. Muntés-Mulero, A. Metzger, E. R. Velasco, and A. C. Aguirre, “Enact: Development, operation, and quality assurance of trustworthy smart iot systems,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 112–127.

[10] B. Fitzgerald and K. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2015.06.063

[11] F. Fleurey and B. Morin, “Thingml: A generative approach to engineer heterogeneous and distributed systems,” in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 2017, pp. 185–188.

[12] S. Gupta, V. Muntés-Mulero, P. Matthews, J. Dominiak, A. Omerovic, J. Aranda, and S. Seycek, “Risk-driven framework for decision support in cloud service selection,” in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 545–554.

[13] B. Karabacak and I. Sogukpinar, “Isram: Information security risk analysis method,” *Comput. Secur.*, vol. 24, no. 2, pp. 147–159, Mar. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.cose.2004.07.004

[14] M. S. Lund, B. Solhaug, and K. Stølen, *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[15] Y.-S. Martin and A. Kung, “Methods and tools for gdpr compliance through privacy and data protection engineering,” in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2018, pp. 108–111.

[16] M. Merkow and L. Raghavan, “An ecosystem for continuously secure application software,” RUGGED Software, CrossTalk March/April, 2011.

[17] A. Moran, *Agile Risk Management*. Springer International Publishing, 2014.

[18] V. Muntés-Mulero, O. Ripolles, S. Gupta, J. Dominiak, E. Willeke, P. Matthews, and B. Somosköi, “Agile risk management for multi-cloud software development,” *IET Software*, vol. 13, no. 3, pp. 172–181, 2018.

[19] H. Naseer, G. Shanks, A. Ahmad, and S. Maynard, “Towards an analytics-driven information security risk management: A contingent resource based perspective,” *Procs of ECIS 2017*, pp. 2645–2655, 2017.

[20] A. Omerovic, “Supporting cloud service selection with a risk-driven cost-benefit analysis,” in *Advances in Service-Oriented and Cloud Computing*, A. Celesti and P. Leitner, Eds. Cham: Springer International Publishing, 2016, pp. 166–174.

[21] A. Shameli-Sendi, R. Aghababaei-Barzegar, and M. Cheriet, “Taxonomy of information security risk assessment (isra),” *Computers & Security*, vol. 57, pp. 14–30, 2016.

[22] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[23] S. V. Shrivastava and U. Rathod, “A risk management framework for distributed agile projects,” *Information and Software Technology*, vol. 85, pp. 1 – 15, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584916304815

[24] D. Ståhl and J. Bosch, “Modeling continuous integration practice differences in industry software development,” *J. Syst. Softw.*, vol. 87, pp. 48–59, Jan. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2013.08.032

[25] J. Webb, A. Ahmad, S. B. Maynard, and G. Shanks, “A situation awareness model for information security risk management,” *Computers & security*, vol. 44, pp. 1–15, 2014.

[26] K. Wuyts, “Privacy threats in software architectures,” 2015.

[27] Z. Yan, P. Zhang, and A. V. Vasilakos, “A survey on trust management for internet of things,” *Journal of network and computer applications*, vol. 42, pp. 120–134, 2014.