DOI: 10.1561/9781680836837.ch8

Chapter 8

Enabling Continuous Privacy Risk Management in IoT Systems

By Victor Muntés-Mulero, Jacek Dominiak, Elena González and David Sanchez-Charles

8.1 Introduction

The next-generation IoT systems need to perform distributed processing and coordinated behavior 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 (Ferry *et al.*, 2018). 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.[1] 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.* (2014) described the different dimensions of trust for IoT systems, concluding that risk management is essential to guarantee trustworthiness.

1. https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends

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 (Boehm and Turner, 2003). In general, there is a 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 processes. Besides, many companies fill this gap by using manual procedures based on storing all the information in spreadsheets, by departments and locally (Ahmad *et al.*, 2012). 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 especially true in the IoT arena, where technologies are not consolidated yet and mixing legal requirements with a deep technical understanding is challenging. In particular, understanding privacy-related vulnerabilities and including privacy considerations in a continuous risk management process is difficult. On the one hand, knowledge related to vulnerabilities connected to privacy issues is not so commonplace. On the other hand, 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 (Wuyts, 2015) (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 (Brooks *et al.*, 2017).

In this chapter, we establish the basis to create an Automated Vulnerability Detector (AVD) for early detection of privacy issues. We also improve continuous risk management in TSIS development by embedding privacy-related risks explicitly through the combined use 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 (Ferry *et al.*, 2019) models and Data Flow Diagrams (DFDs) (DeMarco, 1979), 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 (Wuyts, 2015) establishes between privacy and security threads in STRIDE (Shostack, 2014).

This chapter is organized as follows. Section 8.2 provides an analysis of the state of the art. Section 8.3 presents our methodology. Then, Section 8.4 describes the basic steps to create an AVD taking LINDDUN as the baseline reference. In Section 8.5, we describe the mapping between the architecture model and DFDs as well as our risk model. Finally, in Section 8.6, we draw some conclusions.

8.2 State of the Art

There exist quantitative risk methodologies and tools, like RiskWatch[2] or ISRAM (Karabacak and Sogukpinar, 2005) and qualitative risk methodologies such as OCTAVE (Alberts and Dorofee, 2002), CORAS (Lund *et al.*, 2010), or STRIDE (Shostack, 2014). 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.

Continuous change management in software development has been studied from different perspectives. As an example, Fitzgerald and Stol (2017) proposed a roadmap and agenda for continuous software engineering. In particular, there is a growing interest on security and privacy, and this has motivated work on continuous security (Merkow and Raghavan, 2011), which was proposed to prioritize security throughout the whole software development life cycle.

Previous work discussed risk management in complex and evolving environments. For instance, Omerovic (2016) and Gupta *et al.* (2015) propose managing risk related to accountability, assurance, agility, or financial aspects in multi-cloud applications. Risks analysis is used to guide the selection of cloud service providers. Shrivastava and Rathod (2017) 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 analyzing 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. Aslam *et al.* (2017) performed a systematic literature review on risks and control mechanisms for distributed software development. Moran (2014) explicitly tackles issues related to risk management for agile software development. Moran proposes a risk modified kanban board and user story map. Finally, Muntés-Mulero *et al.* (2018) propose a framework to manage risks that supports collaboration, agility, and continuous development. In the MUSA Risk Assessment tool,[3] in order to assess the risks in the different components of an application, the tool asks users to choose among predefined 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*). We take this tool as the baseline for the proposal presented in this chapter. Finally, Khan *et al.* (2017) perform a DFD-based analysis of vulnerabilities and threads, using STRIDE as a framework, in Cyber-physical Systems. Casola *et al.* (2019)

2. RiskWatch: https://www.riskwatch.com. Accessed: 2019-07-18

3. https://musa-project.eu/node/326

propose an almost completely automated process for threat modeling and risk assessment.

None of these contributions tackle the automatic detection of privacy-related vulnerabilities or monitor privacy-related risks for continuous risk management. In general, even for security, risks are not analyzed and monitored continuously, and historical data are not taken into account (Ahmad *et al.*, 2012, Baskerville *et al.*, 2014). Besides, security risks are assessed based on speculation rather than evidence (Shameli-Sendi *et al.*, 2016, Webb *et al.*, 2014). 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 (Naseer *et al.*, 2017).

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 (Martin and Kung, 2018) focuses on the importance to involve engineers in the loop, for Privacy-by-design (PbD) to be viable.

GeneSIS Modeling Language

GeneSIS (Ferry *et al.*, 2019), developed in the ENACT H2020 project (Ferry *et al.*, 2018), facilitates the development and continuous deployment of TSIS, allowing decentralized processing across heterogeneous IoT, edge, and cloud infrastructures. GeneSIS includes a domain-specific modeling language to model the architecture of TSIS as well as the orchestration and deployment. GeneSIS modeling language is an extension of ThingML (Fleurey and Morin, 2017). We will base our Risk Management methodology in the GeneSIS domain-specific modeling.

LINDDUN

LINDDUN (Wuyts, 2015) is a threat modeling methodology to support risk analysts when addressing privacy risks affecting end users in an application. This methodology provides some guidance to identify and categorize threats under a set of general risks (*Linkability, Identifiability, Non-repudiation, Detectability, Disclosure*

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, L119:1–88, May 2016.

of information, Unawareness, and Non-compliance). LINDDUN can be 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 (Shostack, 2014). We take this link between privacy and security as the baseline for our proposal in this chapter. The LINDDUN methodology requires to formalize the functionality of the system and their dependencies with respect to personal data [i.e., the Data Flow Diagrams (DeMarco, 1979)]. 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).

8.3 Model-based Continuous Risk Management Methodology

In this section, 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 (see Figure 8.1) can be summarized in 6 steps:

- **S1: TSIS Assets Definition:** First, we edit or load DFDs representing the functional description of the system. These DFDs are useful to manage risks

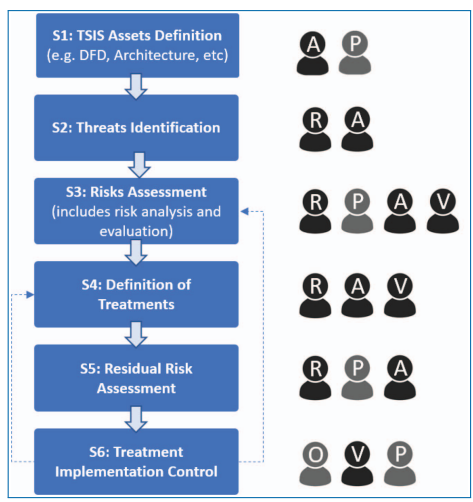

Figure 8.1. Proposed model-based risk management methodology. Roles: (O) Risk Management Owner; (P) Product Owner; (A) Architect; (V) Developer and (R) Risk Analyst.

related to privacy as described by LINDDUN, although they were initially used for security risk control (e.g., in STRIDE). Although DFDs and the elements they consist of 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 architecture 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 assets. Use of the AVD defined later in this section would also be part of this step.

- **S2: Threats Identification:** In this step, users identify threats that may affect the components in the described system.
- **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 analyzed 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.

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. However, the methodology can be easily adapted to other types of systems and used generically.

8.4 Automatic Vulnerability Detection

In order to facilitate an effective identification of privacy-related risks, it is important to make it easy for users to detect those vulnerabilities that expose the system to attacks that may violate data subject's rights. We establish the methodology and bases for the creation of an Automatic Vulnerability Detector (AVD). An AVD uses a set of DFDs to describe an IoT system under development. Based on these DFDs, it is able to detect potential vulnerabilities to kick off the risk analysis process (see step S1 in the previously described methodology).

In order to create the AVD, we use the following methodology: (i) for each DFD component type and for each LINDDUN threat tree, we analyze all the nodes in the tree (containing vulnerabilities) and examine the conditions for those vulnerabilities being relevant in the system; (ii) we create a list of conditions that need to hold for a vulnerability to be effective; (iii) for each instance of each element in every DFD, we collect information about these conditions when defining the system; and (iv) for each component in each DFD related to the system, we filter out vulnerabilities depending on the information collected about these conditions and show those vulnerabilities that are still relevant.

As an example, in Figure 8.2 we show the threat tree related to detectability threats of a data flow in a DFD.[5] A part from the upper node of the tree, which represents the threat under analysis (i.e., Detectability in a data flow), the rest of the nodes refer to vulnerabilities related to this threat. In the figure, we identify several areas from A to E that we will use to define conditions that must hold for the vulnerabilities in those areas to be relevant. Table 8.1 provides an example of some

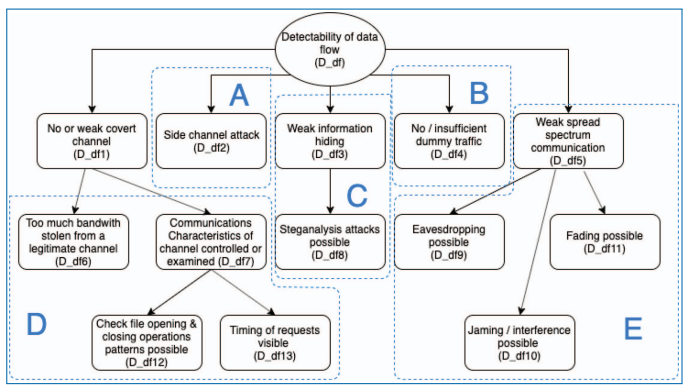

Figure 8.2. Analysis of vulnerability scope in the LINDDUN threat tree for detectability in a data flow.

5. LINDDUN Detectability threat tree catalog: www.linddun.org/detectability. Last accessed in Jan 18th 2020.

conditions that must hold for a subset of vulnerabilities extracted from LINDDUN threat trees, related to those areas. For instance, in area C, vulnerabilities related to steganalysis become relevant if the channel is not encrypted. As a second example, in area B, depending on the volume of traffic in the channel, "insufficient dummy traffic" may or may not actually be a vulnerability. In the latter, note also that this condition may change along time, increasing or decreasing the relevance

Table 8.1. Example of conditions for vulnerabilities related to detectability in a data flow (as defined in LINDDUN) to be relevant.

	Vulnerability/ies (Wuyts, 2015)	**Conditions for relevance**	**Rationale**
A	Side channel analysis (D_df2) is based on timing information, power consumption, electromagnetic leaks, etc. It can be used as a source of information which can be exploited to detect the communication.	Do any actions lead to generate footprints in the communication channel? (e.g., Timing information, power consumption, electromagnetic leaks)	If there are no actions generating footprints in the communication channel, the vulnerability is not relevant.
B	Transmitted data can become detectable when there is no or insufficient dummy traffic (D_DF4) sent at some lower layer of the communication network, such that messages fail to appear random for all parties except the sender and the recipient(s).	Is data traffic in the channel very low?	If the traffic is not low, this vulnerability may not be relevant.
C	When weak information hiding techniques (D_df3) are used, steganalysis attacks (D_df8) are possible (detecting messages hidden using steganography).	Channel not encrypted?	If channel is not encrypted, low entropy of unencrypted data facilitates steganography attacks.
D	Detectability of a data flow may happen if the system uses a covert channel in the wrong way (D_df6, D_df7, D_df12, D_df13).	Covert channel used to avoid detectability?	If covert channel is not used these vulnerabilities are not relevant.

(*Continued*)

Table 8.1. Example of conditions for vulnerabilities related to detectability in a data flow (as defined in LINDDUN) to be relevant (continued).

	Vulnerability/ies (Wuyts, 2015)	Conditions for relevance	Rationale
E	The detectability threat can occur because of a weak spread spectrum communication (D_df5), resulting in deficiencies in the establishment of secure communications (allowing eavesdropping (D_df9)), insufficient resistance to natural interference and jamming (D_df10), and insufficient resistance to fading (D_df11).	Is the communication channel wireless?	These vulnerabilities are relevant if the communication is performed on a wireless channel.

of this vulnerability. Therefore, continuous risk management may also include the continuous monitoring of metrics that allow measuring the level of relevance of vulnerabilities or the likelihood of threats to occur.

Note though that Figure 8.2 shows an example for a simple case where the LINDDUN threat tree is not related to any other threat tree. However, in most cases vulnerabilities related to a particular LINDDUN threat trees are related to vulnerabilities detected in other LINDDUN or STRIDE threat trees. Figure 8.3 describes the detail of this connection in the form of a graph, where every node is one of the LINDDUN threat trees (blue nodes) or one of the STRIDE threat trees related to LINDDUN trees (red nodes). As it can be observed, Detectability of a data flow (D_df) is one of the rare cases where the vulnerabilities in the threat tree are not related to those in other trees. In general, most of the trees are related to others. In order to understand the vulnerabilities of a particular component in a DFD, it is important to navigate through these connections. For instance, the analysis of vulnerabilities related to the Identifiability of an entity (I_e) generates a cascade analysis of vulnerabilities that may include I_ds, I_df, ID_df, ID_ds, S_e, and T_p, following directed edges in the graph. Note that edges are colored in gray if threat trees refer to the same DFD element (e.g., $I\_ds \rightarrow ID\_ds$), and they are colored in red if they refer to different types of DFD elements (e.g., $I\_e \rightarrow I\_ds$). For those relationships represented in gray, we assume that we refer to the same element in the same DFDs. For those relationships represented in red, we have explored them one by one and established a rule to propagate the analysis from one

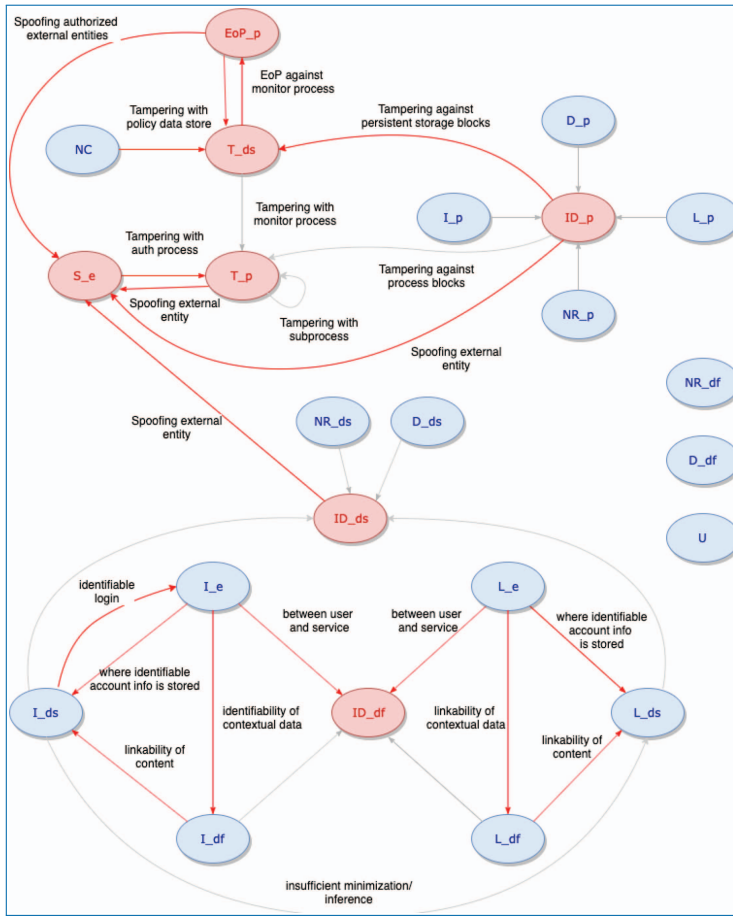

Figure 8.3. Graph describing the relationship between LINDDUN (blue) and STRIDE (red) threat trees. The first part of the ids stand for each of the categories of LINDDUN (Linkability, Identifiability, Non-repudiation, Detectability, Unawareness, and Non-compliance – Disclosure of information is not represented as it is understood as the Information disclosure already captured by STRIDE) and STRIDE (only for those related to LINDDUN: Spoofing identity, Tampering, Information disclosure and Elevation of privilege). The second part after the "_" represents the type of DFD component (entity, data flow, data store, process). Labels in the edges represent indications extracted from LINDDUN descriptions in the corresponding threat trees.

tree to another. For instance, given an entity e in a DFD, for $I\_e \rightarrow I\_ds$, we refer to the data store where the entity credentials of e or other identifiable account information are stored. This means that we will need to ask for the data store names where identifiable account information is stored for each entity. As another example, for $I\_e \rightarrow ID\_df$, we will need to examine all the potential vulnerabilities for all the data flows in the DFD where the origin of the data flow is e. We repeat this analysis for all red arrows in the graph.

8.5 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 subject's rights. In GDPR, risk is explicitly scoped (Rec. 76) with regard 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.

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. In Table 8.2, we show a sample of vulnerabilities extracted from (Shostack, 2014) related to the STRIDE threats in Figure 8.3. 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

Table 8.2. Sample of the mapping between STRIDE threats (related to LINDDUN threats in Figure 8.2) and related metrics that can be detected from monitoring architectural components. STRIDE tree node extracted from Shostack (2014).

STRIDE Threat	STRIDE tree node	Description	Key mitigation actions	Metrics for continuous monitoring	Monitored components
Spoofing External Entities (*S_e*)	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-force 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

(*Continued*)

Table 8.2. Sample of the mapping between STRIDE threats (related to LINDDUN threats in Figure 8.2) and related metrics that can be detected from monitoring architectural components. STRIDE tree node extracted from Shostack (2014) (continued).

STRIDE Threat	STRIDE tree node	Description	Key mitigation actions	Metrics for continuous monitoring	Monitored components
Information Disclosure at Data Flow (ID_df)	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 (T_ds)	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 (ID_p)	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.

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 8.2 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 link:

- Entities in DFDs with the associated network channels in 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, especially when these tasks entail some level of secrecy.

More formally, we define a set of DFDs, which represent a functional description of the system, $D = \{D_1, D_2, \ldots, D_n\}$, as a set of graphs $D_i = (DV_i, DE_i)$, where $DV_i = \{e_1, \ldots, e_m\}$ 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*. $DE_i = \{f_1, \ldots, f_k\}$ represent the *data flows* connecting elements in DV_i. We define a graph $A = \{V_A, E_A\}$ where $V_A = \{c_1, \ldots, c_m\}$ represent the architectural components at the technical level represented in a model such as those created by GeneSIS and $E_A = \{l_1, \ldots, l_k\}$ represent 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 $\mathcal{M} : DV_i \cup DE_i \rightarrow V_A \cup E_A$.

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 8.2, to the related risks or mitigation

actions defined in S3 and S4. 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 $r_1, \ldots, r_j$ associated to e. Given a system architecture defined through a model $A = \{V_A, E_A\}$, we define a set of metrics $m_1^c, \ldots, m_h^c$ 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 $f(m_1^c, \ldots, m_h^c)$ provide 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.

8.6 Conclusions

Handling privacy-related risks is not well understood in the IoT context. There is a lack of mechanism to effectively detect vulnerabilities related to privacy and to collect meaningful evidences and continuously monitor 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.

Acknowledgments

This work is supported by the European Commission through the *ENACT* project under Project ID: 780351 and the *PDP4E* project under Project ID: 787034. David Sanchez-Charles was part of the staff of Trialog SA by the time this article was written. We would like to thank A. Kung and the rest of Trialog's team for their continuous support in this research.

References

Ahmad, A., J. Hadgkiss, and A. B. Ruighaver. 2012. "Incident response teams–Challenges in supporting the organisational security function." *Computers & Security*. 31(5): 643–652.

Alberts, C. J. and A. Dorofee. 2002. *Managing Information Security Risks: The Octave Approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321118863.

Aslam, A., N. Ahmad, T. Saba, A. S. Almazyad, A. Rehman, A. Anjum, and A. Khan. 2017. "Decision Support System for Risk Assessment and Management Strategies in Distributed Software Development." *IEEE Access*. 5: 20349–20373. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2757605.

Baskerville, R., P. Spagnoletti, and J. Kim. 2014. "Incident-centered information security: Managing a strategic balance between prevention and response." *Information & Management*. 51(1): 138–151.

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

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

Casola, V., A. De Benedictis, M. Rak, and U. Villano. 2019. "Toward the automation of threat modeling and risk assessment in IoT systems." *Internet of Things* 7: 100056.

DeMarco, T. 1979. "Structure analysis and system specification." In: *Pioneers and Their Contributions to Software Engineering*. Springer, pp. 255–288.

Ferry, N., P. H. Nguyen, H. Song, P.-E. Novac, S. Lavirotte, J.-Y. Tigli, and A. Solberg. 2019. "GeneSIS: Continuous Orchestration and Deployment of Smart IoT Systems." In: the proceedings of the IEEE COMPSAC conference, Milwaukee, USA, July 15–19. Springer.

Ferry, N., A. Solberg, H. Song, S. Lavirotte, J.-Y. Tigli, T. Winter, V. Muntés-Mulero, A. Metzger, E. R. Velasco, and A. C. Aguirre. 2018), '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*. pp. 112–127.

Fitzgerald, B. and K. Stol. 2017. "Continuous software engineering: A roadmap and agenda." *Journal of Systems and Software*. 123: 176–189. DOI: 10.1016/j.jss.2015.06.063. URL: http://dx.doi.org/10.1016/j.jss.2015.06.063.

Fleurey, F. and B. Morin. 2017. "ThingML: A generative approach to engineer heterogeneous and distributed systems." In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. pp. 185–188.

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

Karabacak, B. and I. Sogukpinar. 2005. "ISRAM: Information Security Risk Analysis Method." *Comput. Secur.* 24(2), 147–159. issn: 0167-4048. DOI: 10.1016/j.cose.2004.07.004. URL: http://dx.doi.org/10.1016/j.cose.2004.07.004.

Khan, R., K. McLaughlin, D. Laverty, and S. Sezer. 2017. "STRIDE-based threat modeling for cyber-physical systems." In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. 1–6. DOI: 10.1109/ISGTEurope.2017.8260283.

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

Martin, Y.-S. and A. Kung. 2018. "Methods and tools for GDPR compliance through privacy and data protection engineering." In: *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 108–111.

Merkow, M. and L. Raghavan. 2011. "An Ecosystem for Continuously Secure Application Software." RUGGED Software, CrossTalk March/April.

Moran, A.. 2014. *Agile Risk Management*. Springer International Publishing. isbn: 978-3-319-05007-2.

Muntés-Mulero, V., O. Ripolles, S. Gupta, J. Dominiak, E. Willeke, P. Matthews, and B. Somosköi. 2018. "Agile risk management for multi-cloud software development." *IET Software*. 13(3): 172–181.

Naseer, H., G. Shanks, A. Ahmad, and S. Maynard. 2017. "Towards an Analytics-Driven Information Security Risk Management: A Contingent Resource Based Perspective." *Procs of ECIS* 2017: 2645–2655.

Omerovic, A. 2016. "Supporting Cloud Service Selection with a Risk-Driven Cost-Benefit Analysis." In: *Advances in Service-Oriented and Cloud Computing*. Ed. by A. Celesti and P. Leitner. Cham: Springer International Publishing. 166–174. isbn: 978-3-319-33313-7.

Shameli-Sendi, A., R. Aghababaei-Barzegar, and M. Cheriet. 2016. "Taxonomy of information security risk assessment (ISRA)." *Computers & Security*. 57: 14–30.

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

Shrivastava, S. V. and U. Rathod. 2017. "A risk management framework for distributed agile projects." *Information and Software Technology*. 85: 1–15. issn: 0950-5849. DOI: https://doi.org/10.1016/j.infsof. 2016.12.005. URL: http://www.sciencedirect.com/science/article/pii/S0950584916304815.

Webb, J., A. Ahmad, S. B. Maynard, and G. Shanks. 2014. "A situation awareness model for information security risk management." *Computers & Security*. 44: 1–15.

Wuyts, K. 2015. "Privacy Threats in Software Architectures."

Yan, Z., P. Zhang, and A. V. Vasilakos. 2014. "A survey on trust management for Internet of Things." *Journal of Network and Computer Applications*. 42: 120–134.