KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Automated Threat Analysis for Security and Privacy

Laurens Sion

Supervisors:
Prof. dr. ir. W. Joosen
Dr. ir. K. Yskout

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

October 2020

Automated Threat Analysis for Security and Privacy

Laurens SION

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

October 2020

In memory of my father.

Preface

Welcome, dear reader, to the result of a long journey. The past years have been an immensely enriching and rewarding experience. It would not have been possible without the support of many people along the way. To all of them, I want to express my profound gratitude.

First, I would like to thank my supervisor, Wouter Joosen, for giving me the opportunity to pursue a PhD at DistriNet, for the freedom to explore interesting research topics, and for all the guidance and advice.

Second, I would like to thank Koen Yskout, for the continuous guidance and feedback, the lengthy discussions, for always being there when had any questions or problems, often introducing me to new problems I had not yet foreseen. A good meeting always ended with a lot more new and interesting questions to research.

Third, I would also like to thank Dimitri Van Landuyt, for the guidance, advice, pitching paper ideas, and our lengthy discussions. Our joint publication record unequivocally demonstrates the extent of your guidance and support all these years.

Next, I would also like to thank Riccardo Scandariato, for introducing me into this fascinating research field during my master thesis.

I would also like to thank the chair of my jury, prof. Jean Berlamont, and all the jury members: prof. Wouter Joosen, dr. Koen Yskout, prof. Eric Steegmans, prof. Sien Moens, dr. Dimitri Van Landuyt, prof. Riccardo Scandariato, and prof. Awais Rashid for the challenging questions, insightful discussion, and valuable feedback on the text.

Additionally, I would like to thank all the co-authors I had the pleasure of working with over the past years. Special thanks also to the organizers and participants of the Dagstuhl Seminar on Empirical Evaluation of Secure Development Processes, where I had many interesting and though-provoking discussions. I would also like to thank Adam Shostack for generously sharing his feedback and insights on past publications.

Furthermore, I want to thank all of Secdam: Alexander, Dimitri, Kim, Koen, Oleksandr, and Stef, for the supportive environment, the collaborations, and the interesting discussions.

I also want to thank all my current and former office mates: Alexander, Andreas, Arun, Dimitri, Gertjan, Ilias, Kim, Michiel, Nayyab, Shirin, and Vincent, for the daily discussions, the support, and, of course, the coffee breaks. Additionally, a thank you to all the people that joined the daily, although currently suspended, trips to the Alma, for all the stimulating conversations. The global coronavirus pandemic makes it painfully obvious how useful and welcome these breaks can be and how they support the generation of new insights and ideas to work on.

I also want to thank all of DistriNet and especially everyone at the business office: Annelies, Annick, Ghita, Katrien, who make everything run smoothly and are always there to help you out. But also, everyone else at DistriNet, all of you together provide the wonderful, stimulating, and supporting environment that is DistriNet.

Finally, I would like to thank my family and friends for their continuous support all these years.

Laurens Sion
Heverlee
September 2020

This research is partially funded by the Research Fund KU Leuven, IWT, imec, and the Flemish Research Programme Cybersecurity.

Abstract

Security and privacy are long recognized as key concerns in the development of software systems. Despite their recognized importance, much of the attention is focused only on the code-level implementation of these principles. However, the design-level consideration of these principles is instrumental to fully realize secure and privacy-preserving software and avoid costly design flaws. Avoiding these security and privacy flaws requires support for analyzing the software design for security and privacy threats. A common design representation used in this context is the Data Flow Diagram (DFD). Security and privacy threat modeling approaches use this DFD representation to identify security and privacy threats. However, these analyses involve extensive manual effort and quickly lead to an explosion of threats to resolve. This makes the analyses effort-intensive and error-prone. It hinders their reproducibility and overburdens the evaluation of design alternatives.

This dissertation addresses the automation of security and privacy threat analysis and thus supports security and privacy by design. The automated threat analysis is enabled through: (i) an extension to the DFD representation of software systems to support the inclusion of essential information on security and privacy solutions, considering the effects of these solutions in mitigating security and privacy threats, and (ii) extending model-based security and privacy design analysis activities to enable the elicitation of security and privacy threats using model queries and the prioritization of the identified security and privacy threats using risk indicators. These extensions are validated and integrated in the SPARTA tool prototype to realize comprehensive and automated security and privacy threat analyses. The implementation provides a foundation for further exploring and realizing automation opportunities in the construction of system designs, the evaluation of security and privacy design alternatives, and the integration as an automated analysis activity in contemporary continuous integration and deployment practices.

Beknopte samenvatting

Beveiliging en privacy worden reeds lang erkend als essentiële eigenschappen in de ontwikkeling van softwaresystemen. Ondanks hun belang, gaat veel van de aandacht enkel naar de implementatie van deze principes op broncode-niveau. Niettemin is de evaluatie van deze principes op ontwerp-niveau essentieel om veilige en privacybeschermende software te realiseren en kostelijke ontwerpfouten te vermijden. Het vermijden van beveiligings- en privacyproblemen vereist de analyse van het softwareontwerp om beveiligings- en privacy-bedreigingen te identificeren. Een courante voorstelling van het ontwerp in deze context zijn Data Flow Diagramma's (DFDs). Beveiligings- en privacy-analysemethoden gebruiken deze DFD-voorstelling om beveiligings- en privacy-bedreigingen te identificeren. Maar deze analysemethoden vereisten uitgebreide manuele inspanningen en leiden snel tot een explosie van bedreigingen om te verhelpen in het ontwerp. Dit maakt de analyse zeer intensief en foutgevoelig, hetgeen de reproduceerbaarheid belemmert en de evaluatie van ontwerpalternatieven bemoeilijkt.

Dit proefschrift verbetert beveiligings- en privacy-bedreigingsanalyse omdat het automatisatie mogelijk maakt. Deze automatisatie draagt bij aan de realisatie van beveiliging en privacy vanuit het ontwerp. De geautomatiseerde bedreigingsanalyse wordt gerealiseerd door: (i) een uitbreiding van de DFD-voorstelling van softwaresystemen met essentiële informatie over beveiligings- en privacy-oplossingen; en (ii) het uitbreiden van modelgebaseerde beveiligings- en privacy-analyse. De eerste uitbreiding met beveiligings- en privacy-oplossingen

bevat de nodige informatie om de effecten van deze oplossingen in het verhelpen van beveiligings- en privacy-bedreigingen in rekening te brengen. De tweede uitbreiding laat toe om beveiligings- en privacy-bedreigingen te identificeren aan de hand van modelopzoekingen en deze bedreigingen te prioriteren aan de hand van risico-indicators. Deze uitbreidingen zijn gevalideerd en geïmplementeerd in het SPARTA prototype om uitgebreide, geautomatiseerde beveiligings- en privacy-analyse te realiseren. De implementatie vormt een fundering voor verder onderzoek naar automatisatie in de constructie van softwareontwerpen, de evaluatie van beveiligings- en privacy-ontwerpalternatieven, en de integratie met geautomatiseerde analyseactiviteiten in hedendaagse continuous integration en deployment systemen.

List of Abbreviations

AQL Acceleo Query Language. 121

AV Asset Value. 101, 129, 158, 165

BSIMM Building Security in Maturity Model. 8

CAIRIS Computer-Aided Integration of Requirements and Information Security. 144

CAPEC Common Attack Pattern Enumeration and Classification. 61, 64, 111, 112

CAWE Common Architectural Weakness Enumeration. 111

CB Countermeasure Bypassed. 130

CD Countermeasure Defeated. 90, 130, 132

CF Contact Frequency. 95, 130

CORAS Model-Driven Risk Analysis approach (not an acronym [LSS10]). 25, 113, 145, 146

CVE Common Vulnerabilities and Exposures. 18

CWE Common Weakness Enumeration. 61, 84, 111

DFD Data Flow Diagram. iii, 3, 5–8, 10, 11, 14–17, 19, 20, 25, 28, 29, 32–38, 41, 44–53, 56–64, 68, 70–73, 77, 78, 83, 84, 87, 93–96, 99–101, 103, 106, 110, 112–115, 117–119, 121–123, 125, 132, 134–137, 140, 142, 144, 147, 150–152, 154, 157–159, 161–163, 165–169

DPbD Data Protection by Design. 2, 31

DPIA Data Protection Impact Assessment. 3, 39

DREAD Damage Reproducibility Exploitability Affected users Discoverability. 23

DROWN Decrypting RSA using Obsolete and Weakened eNcryption [ASS+16b], a cross-protocol attack on TLS, which uses a flaw in SSLv2. 41

DST Data Subject Type. 129

DTS Data Type Sensitivity. 94, 129

EDFD Extended Data Flow Diagram. 61

EMF Eclipse Modeling Framework. 28, 121, 135

FAIR Factor Analysis of Information Risk. 24, 25, 69, 86, 87, 104, 141, 152

FMEA Failure Mode and Effects Analysis. 145

FTA Fault Tree Analysis. 145

GAPP Generally Accepted Privacy Principles. 2

GDPR General Data Protection Regulation. 2, 31, 39, 109, 152, 153, 158, 159

GPS Global Privacy Standard. 2

HTTP HyperText Transfer Protocol. 37

HTTPS HTTP over TLS. 34–37, 39, 41, 49, 51, 52

IRIS Integrating Requirements and Information Security. 144

LEF Loss Event Frequency. 127, 129

LINDDUN Linkability, Identifiability, Non-repudiation, Disclosure of information, Unawareness, Non-compliance. 20, 28, 59, 63, 68, 70, 71, 73, 77, 87, 93, 97, 105, 107, 110, 114, 153

LM Loss Magnitude. 127, 129

NDS Number of Data Subjects. 129

NR Number of Records. 92, 129

OCL Object Constraint Language. 112

OCTAVE Operationally Critical Threat, Asset, and Vulnerability Evaluation. 25

OpenSAMM Software Assurance Maturity Model. 8

OVVL Open Weakness Vulnerability Modeler, threat modeling tool. 60, 143

OWASP Open Web Application Security Project. 8, 60, 111, 143

PbD Privacy by Design. 31, 64, 68

PA-DFD Privacy-Aware Data Flow Diagram. 62

PASTA Process for Attack Simulation and Threat Analysis. 113

PERT Program (or Project) Evaluation and Review Technique. 127, 128

PET Privacy-Enhancing Technology. 109

PIA Privacy Impact Assessment. 109, 110, 114, 146

PoA Probability of Action. 95, 130

POODLE Padding Oracle On Downgraded Legacy Encryption [MDK14], a TLS downgrade attack. 41

PRIAM Privacy Risk Analysis Methodology. 110, 114, 146

PV Privacy Value. 129

RE Requirements Engineering. 4

RP Retention Period. 91, 129

S Strength. 130

S/MIME Secure/Multipurpose Internet Mail Extensions. 52

SbD Security by Design. 2, 31, 64, 68

SDL Security Development Lifecycle. 23, 24, 109

SDLC Secure Development Life Cycle. 8

SPARTA Security and Privacy Analysis through Risk-driven Threat Assessment. iii, vi, 11, 14, 51, 55, 119–123, 125–127, 132, 135–142, 147, 151–153, 155, 158, 161, 162

SQL Structured Query Language. 16, 62

SQUARE Security Quality Requirements Engineering. 114, 146

STECA Systematic-Theory Early Concept Analysis. 110

STPA System-Theoretic Process Analysis. 110

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of Privilege. 10, 17, 18, 20, 21, 28, 32, 33, 35, 59, 63, 73, 93, 100, 103, 109, 110, 140, 153

TC Threat Capability. 95, 130

TEF Threat Event Frequency. 88, 129, 130

TLS Transport Layer Security. 8, 37, 41, 96

TMT Threat Modeling Tool. 33–37, 39, 41, 42, 60, 143

UML Unified Modeling Language. 5, 26–28, 44, 62, 63, 111, 112, 152

V	Vulnerability. 130, 132
Viatra	VIsual Automated model TRAnsformations. 84, 85, 123–125
Vpn	Virtual Private Network. 51
WebRTC	Real-Time Communication in web browsers. 32–35, 37, 140, 168, 169
Xss	Cross-Site Scripting. 9, 84–86

Contents at a Glance

Contents

List of Figures

List of Tables

List of Snippets

Chapter 1 Outline

1 Introduction

No one writes anything that is worth writing, unless he writes entirely for the sake of his subject.

— Arthur Schopenhauer [Sch91]
The Art of Literature

Security and privacy are important and century old concerns. Long before the existence of computers, such concerns translated to physical security mechanisms such as locks and vaults, to early applications of cryptography [Ker83a, Ker83b] to protect information. Similarly, privacy as a right has already been formulated in the 19th century in the influential article of Warren and Brandeis [WB90], triggered by the advent of photography. With the increasing ubiquity of computers and software, and especially the networking of these systems, security and privacy have become critical, as the operating environment is so different compared to traditional physical security. While a physical lock or safe only needs to defend against attackers present in front of the lock, networked systems can be attacked from all over the globe.

It is exactly the sharing of early computer systems between multiple users and the networking of these systems that introduced such security concerns in the 70s [SS75]. Over the years, great strides have been made in ensuring secure software implementations with improvements such as memory safe languages, embedded systems security [NVBM+17], secure compilation [PAC19], program verification [JSP+11], security testing [FBJ+16], static and dynamic code analysis, etc.

However, practically all these security improvements focus on the implementation-level security aspects of the software systems under consideration. Design-level security does not get nearly as much attention, despite the recognition of the importance of design in traditional software engineering activities [GHJV94, BCK12]. This is unfortunate, given that design errors can be costlier to diagnose and correct [CFH+76]. An example of such a high-impact design flaw is the one in Microsoft Active Directory [Mic15, JAS15b, JAS15a], which took Microsoft over a year to resolve because of the re-engineering effort involved in fixing the problem [JAS15b]. Therefore, it is desirable to build security in from the start [McG06] to detect and avoid these flaws, thereby realizing a Security by Design (SbD) approach.

A *by design* approach is not only important for security, it is also relevant for privacy. In addition to end users' growing awareness of privacy concerns due to increasingly impactful data breaches, the importance of considering privacy during the design is also exemplified by the introduction of legislations and guidelines such as the EU's General Data Protection Regulation (GDPR) [Eur16], the OECD's Privacy Guidelines [OEC80, OEC13], the Generally Accepted Privacy Principles (GAPP) [AC09], and Cavoukian's Global Privacy Standard (GPS) [Cav06]. The GDPR even imposes a *by design* and *by default* approach, formulated as Data Protection by Design (DPbD) [Eur16]. This can be understood as a migrating away from *privacy by policy* approaches to *privacy by architecture* approaches in the terminology of Spiekermann and Cranor [SC09]. The impact of design decisions on privacy can be illustrated with the bitcoin cryptocurrency. As strong anonymity was not a primary design goal, it is possible to associate multiple public keys of a user with each other and link them with external information to identify users [RH13].

A final example to illustrate the usefulness of design representations can be found completely outside of the realm of software. A frequently used, and fruitful (see Alexander *et al.*'s [AIS+77] influence on design patterns [GHJV94]), analogy is that of physical buildings. Construction is a sector with highly standardized materials. One could ask why some constructions fail and others do not, despite making use of the same materials. The answer lies in the fact that not only the materials that are used matter, but also how these materials are composed in the resulting structure. Numerous occurrences on collapsing bridges illustrate the importance of analyzing the entire structure to prevent adverse events such as collapses due to resonance.[1] To complete the analogy, building codes could also be devised for the construction of software systems [Lan13], analogously to physical building codes.

Not only do design-based analyses support the uncovering of security flaws that are not visible in implementation artifacts. Examples of this are the previously mentioned Active Directory design flaw [JAS15b, JAS15a, Mic15] and the design flaw in the Zoom web conferencing software [CVE19b, Lei19] that allows remotely enabling the webcam of a website visitor. It can also be cost effective, as the initial modeling effort can be reused and leveraged in other types of analyses. Consider, for example, how the initial investment in modeling the system for doing a design analysis can also support additional analysis activities such as a Data Protection Impact Assessment (DPIA) in which the system description could be reused for: the compliance assessment, description of processing operations, and the description of the provided security and privacy countermeasures [SDVL+19] for protecting personal data.

Regardless of the type of analysis activity, a design-level representation of the system under consideration is required to support performing design-based security and privacy assessments of the system. One common, high-level, and accessible representation is that of Data Flow Diagrams (DFDs) [DeM79, GS79]. The notation is simple, comprising of only four element types: *processes*, denoting processing operations; *external entities*, denoting parties external to the system being modeled, *data stores*, representing stored information in files and data bases; and *data flows*, representing transfers of information between the above three elements. Additionally, it has already been applied several times in industrial contexts for security analysis [HL06, Sho08, Dhi11].

1 One of the most common examples of this is the widespread video of the Tacoma Narrows Bridge which collapsed in 1940. However, there is some debate on whether it was actually resonance that caused the collapse [OWH15]. Such discussions also result from ambiguous interpretations of *resonance* and the fundamental differences between these interpretations [BS91]. This is an excellent analogy to illustrate the need for precise design representations to reason about the effect(s) of security and privacy countermeasures and avoid terminological ambiguities [And08].

1.1 The Context of Security and Privacy before and by Design

There are strong arguments to address security and privacy requirements, needs, and solutions in an earlier stage of the software development process (before implementing and testing the software). This raises a number of challenging questions, including (i) when to spend efforts on security and privacy by design, (ii) which artifacts to analyze, and (iii) how to incorporate security and privacy solutions.

This work anticipates on the generally and well-known notion of iterative software development processes [LB03]. Many researchers and practitioners have observed the fact that a given refinement or iteration on the design of a software systems can be analyzed or assessed, and subsequently be improved based on the findings of such an analytical effort. Repeating this effort leads in principle to an iterative process, This iterative process applies to security and privacy analysis as well.

One can wonder when the first security and privacy analysis will take place: this is preferably delivered early, in fact when requirements are articulated—during the so called Requirements Engineering (RE) process or RE phase. Additional iterations can subsequently be applied during the definition of software architecture, high-level design and later on, more detailed design. There is no quantitative limitation for the efforts one can spend in delivering the security and privacy analysis, yet the cost-benefit analysis should not be ignored. A key concern in this context is to maximize automation and support traceability.

At all of the stages listed above—and without going into the details of each of these stages—the software, security, and privacy engineers can be expected to have an accurate description of the system being incepted, designed, or built. Indeed, even during the RE process one has to consider (at least) a high-level structural description of the systems. In this respect, our work aligns with the view that considers security requirements to be part of the Twin Peaks [Nus01] that represent an interplay between requirements on the one hand, and architecture and system description on the other hand [Nus01, HYS$^+$11].

In summary, analysis of security and privacy concerns can potentially occur at any early stage of the software development process, from requirements level to relatively detailed design. In all of these stages, we use the term security and privacy analysis as the activity that identifies and models the security and privacy challenges that come with a specific design-level definition of the system created. The second question then is which artifacts to utilize when conducting security and privacy analysis. Multiple options exist and we do not argue in favor of specific models. The starting point of this dissertation is to connect to a relatively popular notation that can indeed span multiple stages of the early half of a software development process, and that has been practically applied in the context of security and privacy engineering. We therefore start from DFDs. This starting point is of a pragmatic nature and as such not the result of a deep and broad study. Yet, the relevance of DFDs is confirmed by significant investments and research efforts in the community [SMC74, YC75, DeM79, YC79, GS79, KG99, SS04, Tor05, HL06, HLOS06, Sho08, DWS+11, Sho14, Wuy15]. While there are numerous notations to expand upon,[2] necessarily an initial choice has to be made. This does not, however, preclude the translation of those improvements to other relevant modeling languages.

The outcome of a security and privacy analysis can be twofold: (i) at least a number of security and privacy requirements are articulated, and (ii) solutions (often threat mitigations) that address these requirements are identified. In principle, such solutions can be modelled as extensions and refinements to the existing design artifacts.

The minimal and essential results are the security and privacy requirements (or needs) imposed by the initial or evolved design of the software system under consideration. One important (and widely applied) family of security and privacy requirements is threats [HLOS06, DWS+11, Sho14, Wuy15]. In the remainder of this work, we will focus on security and privacy analysis based on threat elicitation with DFDs. This starting point limits the scope of our work, yet it has maximized the feasibility of the overall research trajectory.

2 Another candidate could be the Unified Modeling Language (UML), on which already some extensions have been built in the past such as, for example, UMLsec [Jür05]. However, it is not at all that frequently used in practice [Pet13]. UML's complexity is a recurring reason for this lack of usage [Pet14].

In other words, while our generic goal is to enable powerful and efficient security and privacy analysis in the early stages of the software engineering process, the initial boundary conditions of our work have been to focus on threat modeling starting from DFDs.

1.2 The Role of Automation

The previous section already touched upon how the cost-benefit balance of performing security and privacy analyses should not be ignored. Indeed, performing these types of analyses relies on security and privacy experts, who possess the relevant background knowledge and expertise. Additionally, the exhaustive nature of threat elicitation implies that it is time-consuming and requires substantial effort. The reliance on these scarce resources further increases the cost for this essential activity in the development lifecycle.

Combining these scarce resources with the application of these analyses in the context of contemporary iterative development practices precludes a single up front investment to adress the security and privacy concerns. Previously made design decisions may need to be revisited and modified as the design further materializes.

Both the required cost and the increasing need for frequent re-assessment are key drivers for increasing automation to support these security and privacy analysis activities. In addition to the cost reduction and reduced reliance on experts,[3] it offers a number of compelling benefits: (i) *reproducibility*: a more automated threat analysis enables more reproducible results regardless of which user performs the analysis; (ii) *explainability*: for example, by being able to retro-actively query a threat model to find out why specific threats were considered relevant (or not); (iii) *evaluation of alternatives*: alternatives can be constructed and efficiently re-evaluated to guide in decision-making; and (iv) *what-if analyses*: in which changes to the properties of security and privacy solutions can be considered to quickly assess the impact of, for example,

3 Reduced reliance as the application of automation does not replace the experts, but aims to employ them more efficiently. They still encode the relevant knowledge in tool support, provide expert estimates to assist in the prioritization, etc.

countermeasures that no longer sufficient protection due to a newly discovered vulnerability.

1.3 Research Goal and Questions

The main goal of this dissertation is to investigate the application of automation to provide support for systematic, security and privacy threat analysis using DFDs. This requires semantic support for representing security and privacy solutions, and analysis activities relying on these extensions to identify security and privacy threats.

Automated security and privacy threat analysis

The manual application of a rigorous and systematic security and privacy analysis of the modeled system is time-consuming and tedious. A manual threat elicitation takes considerable effort. After performing such an analysis, the threat modeler still has to process the resulting list of threats to prioritize them in order to be able to address the most important threats first. Furthermore, the manual steps make it hard to impossible to reuse this effort when modifying the system. This makes it difficult to evaluate the impact of design alternatives. To improve and support these activities, a shorter feedback loop is required which can be realized in tool support by continuously re-assessing the modeled system for security and privacy threats. Such a tool solution requires the integration of both (i) extensive support for modeling security and privacy solutions in the design and (ii) elicitation of security and privacy threats from the design and analysis of these identified threats for prioritizing the threats. The support for modeling solutions is essential for automated analysis activities to be able to take the security and privacy effects of these solutions into account while eliciting the threats. Furthermore, all the information on the threat, the system context, and any potentially involved security or privacy solutions needs to be taken into account to determine to which degree a threat is mitigated. This way, the security and privacy analysis feedback loop during system design and improvement can be drastically shortened, and continuous security and privacy feedback on the system design can be provided.

Semantic support for security and privacy solutions

Despite the use of models such as DFDs in the context of threat modeling [KG99, SS04, HL06, Sho08, Dhi11, Sho14], the representations used here are completely agnostic of security or privacy design information such as the instantiation of security and privacy solutions in the system. For example, the decision to rely on TLS, to protect the confidentiality and integrity of transmitted information and to provide authentication of the server, is not present in these diagrams and, therefore, the effect of this solution cannot be taken into account when analyzing these models later on, which can lead to duplicated or wasted effort.

Not only is this problematic from an analysis perspective, the documentation of this information is important as well and is even required by maturity models such as the Building Security in Maturity Model (BSIMM) [MMW18] and OWASP's Software Assurance Maturity Model (openSAMM) [OWA17b]. These maturity models explicitly require annotating threat models with *compensating controls* for attaining the highest maturity level. They are thus a key activity in Secure Development Life Cycles (SDLCs) [HL06, McG06].

Existing approaches [Mic16, BSK16] to include such information indirectly by recording the effects as properties on the model elements introduce a number of serious drawbacks such as the lack of traceability to the original solutions, the inconsistent recording of the effects, the strong dependency from threat knowledge bases on the properties that they have to whitelist, and the difficulty in capturing complex solutions. Furthermore, the information on security and privacy solutions should be captured in a generic way to ensure an updatable solution catalog. This way, the expert knowledge can be captured, reused, updated, and consistently applied in DFD models. The aforementioned problems trigger the following research question:

RQ1: How can the DFD-based modeling representation be extended to support a first-class representation of security and privacy solutions?

Supporting model-based security and privacy analysis

The modeling of security and privacy solutions as discussed above is not only useful from a knowledge management perspective. Indeed, this information can be leveraged in automated analyses of the security and privacy of a design to ensure a very systematic assessment. There are two key problems in the context of design analysis: the identification and the prioritization of security and privacy threats.

First, current analysis approaches detect security and privacy threats at a low-level of granularity, which lacks a lot of contextual information that cannot be taken into account to determine the applicability of threats. Consider, for example, stored cross site scripting (XSS) attacks. Detecting the presence of this threat requires analyzing a more complex path through the system instead of looking at a single element.

Second, systematically analyzing a design for security and privacy threats leads to a large list of threats. In order to improve the usefulness of the results, support is needed for prioritizing the identified security and privacy threats by taking into account the impact of these threats, as well as the effectiveness of the involved security and privacy solutions that mitigate them (either partially or fully). Such a prioritization assessment requires additional information on the strength of security and privacy countermeasures, as well as an attacker model that captures the capability of the attacker and the projected frequency of attacks. Furthermore, this information is not static in nature. The strengths of countermeasures may change over time as vulnerabilities are discovered, while attackers may increase their frequency of attack attempts as targets become more value. Therefore, this information should be recorded explicitly to enable revisiting these parameters over time.

RQ2: How can security and privacy solution information be leveraged in the design analysis to more precisely identify threats and prioritize them according to their impact and the effectiveness of the solutions?

1.4 Approach

The ultimate goal is to demonstrate the effectiveness of the proposed techniques by delivering tool support that relies on solid automation and that can effectively reuse existing knowledge. The realization of this tool support requires addressing two key challenges: (i) representing security and privacy solutions in DFDs, and (ii) the model-based security and privacy analysis of these DFDs.

The *first* challenge is addressed with the creation of an extended DFD meta-model to address the lack of support for representing security and privacy solutions. Additionally, the meta-model is extended to support a reusable and extensible representation of threat types to incorporate the existing STRIDE knowledge bases [HL06, Sho14, Mic16] in order to make this threat knowledge available and reusable in later analyses.

For addressing the *second* challenge, two analysis activities are provided: the querying of the constructed models using patterns to elicit security and privacy threats, and an automated risk assessment for prioritizing each of the identified threats. These extensions are subsequently evaluated on the real-world whistle-blower submission system SecureDrop [Fre18a, Fre18b] to assess whether the prioritization resulting from the threat analysis is consistent with the security and privacy countermeasures implemented by the developers.

Finally, the solutions to these two challenges are subsequently combined in a tool prototype to support their automated application.

1.5 Contributions

This dissertation provides the following three contributions.

1. We provide a *design-level representation of security and privacy concerns in DFDs for use in threat modeling contexts*, showing positive improvements in terms of semantic quality, traceability, separation of concerns, and dynamism, respectively due to the

instantiation of security and privacy solutions, the traceability of their effects, the independent evolution of security and privacy solution catalogs, and impact analysis of architecture-level security and privacy decision making support.

2. We propose a *pattern-based threat elicitation* approach to support the elicitation of more complex security and privacy threats, combined with a *security and privacy risk model for the risk assessment of these elicited threats* that leverages the provided model extensions to integrate and automate the elicitation and risk-driven threat prioritization in a threat modeling approach.

3. We present SPARTA, *a threat modeling tool that integrates the presented extensions* to: capture security and privacy design decisions in DFDs, provide continuous threat elicitation based on this abstraction, and conduct risk analysis to prioritize the elicited security and privacy threats.

1.6 Overview

This dissertation is structured as follows. First, Chapter 2 provides some background information, consisting of the three dimensions of security needs (design, threats, and goals) [Tür17] in which the DFD representation for threat modeling is introduced and a primer on meta-modeling, and closes with a brief summary.

After that, Chapter 3 presents the *first contribution* on the meta-models for modeling security and privacy solutions in DFDs in support of threat modeling activities. Following the representations, Chapter 4 introduces the *second contribution* on the two analysis aspects: the security and privacy threat elicitation and the risk-driven threat prioritization. Then, Chapter 5 combines the previous two contributions for the *third contribution* in the SPARTA tool for continuous threat modeling.

Finally, Chapter 6 concludes this dissertation and provides a number of directions for future research.

Chapter 2 Outline

2 Background

For Knowledge is grateful to the Understanding, as Light to the Eyes

— John Locke [Loc12]
Some Thoughts Concerning Education

This chapter elaborates on the interplay between security requirements (threats and goals) and security architecture and design, as introduced in the introduction above. Its main purpose is to summarize the essential background research and information to enable a smooth interpretation and reading of Chapters 3 to 5 which present the main contributions. The interplay between requirements and design has been discussed (amongst others) by Türpe [Tür17] and we leverage upon his description (visualized in Figure 2.1 together with the relevant sections) to navigate the background knowledge presented here.

The information presented here constitutes the background knowledge to aid in the understanding of the contributions presented later. It does not cover all the related work. Instead, the related work for the different contributions will be discussed separately in the following chapters together with the contributions they pertain to.

We briefly explain the different dimensions from Türpe [Tür17]: design, threats, and goals, to illustrate the structure of this chapter. Figure 2.1 shows these three dimensions and is followed counterclockwise. The diagram is navigated starting with *design* in the top right for which

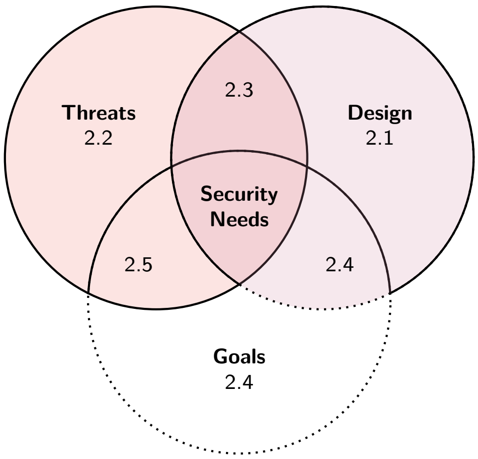


Figure 2.1: Overview of the three dimensions of security needs.
This figure shows Türpe's [Tür17] three dimensions of security needs and indicates which background sections cover the three dimensions and their intersections.

Section 2.1 elaborates on the current modeling support for security analyses. This section introduces the Data Flow Diagram (DFD) design representation for modeling systems which is necessary for explaining the contributions presented in Chapters 3 and 5.

After that, Section 2.2 discusses *threat* knowledge on the different types of security or privacy threats. This dimension focuses on the threat knowledge itself. These are part of the inputs into the analyses presented in Chapter 4 and realized in Chapter 5.

Section 2.3 focuses on the intersection of the previous two dimensions on design and threats and covers the analysis activities that enable the identification of the threats in software designs. This section introduces threat modeling to enable the systematic elicitation of threats based on design descriptions of the system. This is the analysis approach which is extended and presented in the first part of Chapter 4 and realized in the SPARTA tool support presented in Chapter 5.

At the bottom of Figure 2.1 is the goal dimension. This dimension covers security requirements, but phrased as desired properties from a positive perspective. This dimension is not the primary focus of this dissertation, but its intersection with the threat dimension is relevant. Section 2.4 provides some background information on this dimension and also addresses the intersection with design to cover the security and privacy patterns to realize these requirements in design models.

Finally, the intersection of the *threats* and *goals* dimensions covers the prioritization of threats using risk assessment. Section 2.5 provides the relevant background information in support of the risk-driven threat prioritization presented in Chapter 4 and realized in Chapter 5.

Underlying any modeling support for security or privacy is a meta-model that defines the different the concepts that can be used in user models. For the readers interested in how meta-modeling techniques can be leveraged to create more expressive modeling support, Section 2.6 provides a primer on meta-modeling. Meta-models can provide support in all the above dimensions: more expressive design models, structuring threat knowledge, and representing goals.

2.1 Model Representations

Chapter 1 highlighted the importance of performing design-based analysis for assessing the security and privacy properties of software systems. Such analysis activities require model representations of the design to be analyzed. This section provides some background on the DFD model representations that can be used for the security and privacy analysis of software systems.

The DFD modeling notation originates from the structured charts representation for modeling and analyzing programs, resulting from more than ten years of Constantine's research, presented by Stevens *et al.* [SMC74] in 1974. In the following years, multiple authors have further revised and refined the notation [YC75, YC79, DeM79, GS79] resulting in the current notation which has remained stable over a very long time and is still in use today, also because of the design excellence of DeMarco's visual representation [Moo09].

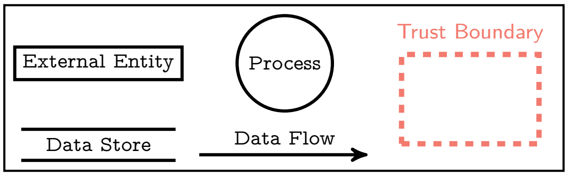


Figure 2.2: Data Flow Diagram (DFD) Legend
This diagram shows the four different DFD element types for modeling systems and how data flows between the elements. The trust boundary at the right-hand side can be drawn around the other DFD elements to delineate different trust zones.

The DFD modeling notation consists of the following five element types, which are also displayed in Figure 2.2: (i) external entity, (ii) data store, (iii) process, (iv) data flow, and (v) trust boundary. The latter element type has been added later when DFDs were used in the context of threat modeling [SS04, HL06]. The different DFD element types have the following meaning:

External Entity represents any entity (both human and code) that is external to the system and outside its control.

Data Store represents any kind of data storage system such as a local file or database. However, it should only represent the storage itself. For a database system, for example, there would be a process in front to perform the translation of SQL queries.

Process represents any kind of processing operation. The processes can describe the system at different abstraction levels depending on the required level of detail for subsequent analyses.

Data Flow represents the communication of data between processes, external entities, and data stores.

Trust Boundary represent differences in trust between the elements in a system and are used to group elements of the same trust.

A key problem with these system representations when using them for security and privacy analyses is their lack of support for the representation of security and privacy solutions. This missing support requires any subsequent analysis to rely on a manual knowledge of the analyst in order to take these into account. There are a number of proposals in the literature that attempt to resolve this knowledge gap by including some support for representing this information.

These are shortly outlined below. Chapter 3 discusses them in more detail while presenting the extensions for a richer, first-class representation of security and privacy solutions in DFD models.

Microsoft Threat Modeling Tool [Mic16] To take into account existing security and countermeasures in the STRIDE security threat elicitation, the Microsoft Threat Modeling Tool extends existing DFD elements with attributes. These attributes can be used to represent the security effects of existing solutions such as, for example, '*provides confidentiality*' for an encryption solution. These element attributes are taken into account during threat elicitation.

Berger *et al.* [BSK16] Berger *et al.* apply a similar approach. They provide a number of property extensions to different DFD elements which again express the effect of security solutions to consider while checking for the applicability of certain security flaws.

2.2 STRIDE & LINDDUN Threat Knowledge

The identification of security and privacy threats requires information on the types of threats to identify. Threat modeling relies on high-level '*categories*' of security [HLOS06] or privacy [DWS+11] threats, which can contain multiple subtypes of threats in a tree structure. The term 'categories' is sub-optimal here as they are merely meant as a mnemonic to assist in recalling the main threat types. The categories are not disjunct groups under which all the different threats types can be unambiguously placed. Instead, there are several overlaps and dependencies between the different threat types. For example, spoofing an administrator can lead to elevation of privilege. Each of the security and privacy threat categories is explained below.

Spoofing Impersonating an entity in interactions with a system.

Tampering Performing unauthorized modifications to data (whether in storage or transferred over the network) or running processes.

Repudiation Denying having taken an action.

Information Disclosure Unauthorized disclosure of information.

Denial of Service Making a service unavailable to legitimate users.

Elevation of Privilege Performing actions for which the entity performing the actions does not have the privileges.

Linkability Being able to link multiple items of interest on a data subject. It can lead to identifiability.

Identifiability Being able to identify a data subject.

Non-repudiation Not being able to deny having taken an action.

Detectability Being able to detect the presence of an item of interest.

(Disclosure of Information) The security category from above.

Unawareness The data subject being unaware of the data processing operations on its personal data.

Non-Compliance Data processing operations that are non-compliant with privacy regulations.

There are a number of more detailed threat type descriptions available as threat trees [HL06, Sho14, DWS+11, Wuy15] and implemented in tool support [Mic16, Mic20]. Besides the threat modeling threat types, there are number of other knowledge sources on security and privacy issues at differing levels of abstraction. These range from low-level vulnerabilities in concrete software products, described in the CVE catalog [CVE19a], to architectural weaknesses [CWE20, STM17], design flaws [ADD+14, MH17, THMS19], and attack patterns [CAP18].

2.3 Threat Modeling

Threat modeling is situated at the intersection of the *Threats* and *Design* dimensions of Türpe. While there are numerous design approaches, the focus here is explicitly on the STRIDE threat modeling approach as outlined by Shostack [Sho14]. The related work in Section 3.6 discusses alternative approaches, while the excellent survey by Tuma et al. [TCS18] provides a very comprehensive overview.

Threat modeling [KG99, SS04, HL06, Sho08, Sho14, DWS+11, Wuy15] is a design-based system analysis activity that starts from DFD description of the system under consideration and has been applied several times since, in real-world industrial contexts [Tor05, Sho08, Dhi11]. It provides a methodology for analyzing a system's DFD design systematically in order to identify security [SS04, HL06, Sho08] or privacy [DWS+11, Wuy15] threats. In subsequent activities these threats can be analyzed to determine the priorities for resolving them.

Shostack succinctly summarizes the threat modeling approach as a combination of the following four key questions [Sho14]: *1. What are you building? 2. What can go wrong? 3. What should you do about those things that can go wrong? 4. Did you do a decent job of analysis?*

This section follows that same structure by elaborating on each of first three steps by addressing: 1. modeling, 2. eliciting, 3. mitigating.

2.3.1 Modeling the System

The first step in threat modeling involves the creation of a DFD-based representation of the system under consideration, using the concepts offered by the DFD notation and explained in Section 2.1.

This system description can be modeled at various levels of abstraction. The threat modeling approaches do not impose a specific level of detail. An analysis can start from a DFD context diagram, after which the system processes can be further decomposed as desired in order to include additional details in the model description. Depending on the support of the used modeling framework, details on already present security or privacy solutions could also be added so that the effect of these solutions can be incorporated in the subsequent analysis activities.

Constraints for Sound DFD Models Listed below are common criteria for valid DFD models. Not all these criteria need to be applied strictly, although the lack of, for example, outgoing data flows from a data store may be an indication that the system functionality that uses this data is not modeled and, hence, will not be considered in the analysis.

- *No direct data flows between two data stores [Sho14].* Data stores are passive elements. They cannot directly communicate with one another. The data is moved by a process.
- *No direct data flows between external entities.* Communication between external entities is out of scope.
- *No direct data flows between external entities and data stores.* There needs to be a process in between to handle the storage of the data in a data store.
- *No data stores with only outgoing data flows (i.e. no data sources).* Data has to originate from somewhere. There had to be some process that put the data in the data store in the first place.
- *No data stores with only incoming data flows (i.e. no data sinks) [Sho14].* Data is stored for a reason. There has to be a process that reads and uses the data being stored.
- *No DFD elements without incoming or outgoing flows.* Completely disconnected elements cannot perform any function.

2.3.2 Eliciting Security and Privacy Threats

This threat modeling step involves a systematic analysis of the previously created DFD model of the system to identify security and privacy threats. There are two main ways of iterating over the system design to elicit threats: element-based threat elicitation, in which every element of the system is considered; and interaction-based threat elicitation, in which every interaction (i.e. data flow) is considered.

The element-based elicitation offers the benefit of simplicity. It only requires considering the concrete element types to determine whether a threat is applicable (see Table 2.1). The interaction-based elicitation is more complex, as an interaction also involves the sending and receiving element types, but it provides additional context information for eliciting threats that are more specific.

Every threat is subsequently documented to enable further analyses such as a risk analysis or prioritization of the identified threats. The documentation of the threat instances consists of the threat category (from STRIDE or LINDDUN), the specific threat type, the context (element

Element Type	S	T	R	I	D	E	L	I	N	D	U	N
External Entity	X		X				X	X			X	
Process	X	X	X	X	X	X	X	X	X	X		X
Data Flow		X		X	X		X	X	X	X		X
Data Store		X	?	X	X		X	X	X	X		X

Table 2.1: STRIDE-per-element and LINDDUN-per-element.
The STRIDE-per-element table from Shostack [Sho14] combined with the LINDDUN-per-element table from Deng et al. [DWS+11]. It shows which of the STRIDE and LINDDUN threat types are applicable for each DFD element type. Note that the second D (Disclosure of Information) from LINDDUN is omitted here because it equals the I (Information Disclosure) from STRIDE.

or interaction), a concrete description of the threat, and any relevant assumptions used when considering the threat's applicability [WJ15].

2.3.3 Mitigating the Elicited Threats

The third step involves addressing the identified security and privacy threats by: (i) implementing security or privacy countermeasures, (ii) changing the system or some of its functionality, (iii) transferring them by relying on someone or something else (e.g., user decision or operating system controls), and (iv) accepting the risk. [Sho14]

2.4 A Note on Security Goals

The third main dimension from Türpe's diagram (Figure 2.1) of security needs are the security goals which express the desired protection of a system in a specific environment [Tür17]. These goals can be expressed independently as protection or compliance conditions, regardless of the design of the system or its threats. However, these goals are closely related to the security threats discussed in Section 2.2. Indeed, Hernan et al. [HLOS06] have provided a mapping of the STRIDE threat types to the security goals (properties in their terminology) they threaten. This overview is displayed in Table 2.2. Similar as

Threat	**Goal**
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Table 2.2: Mapping of STRIDE threats to security goals
This table from Hernan et al. [HLOS06] shows the mapping from the STRIDE security threat types (left-hand side) to the security goals (right-hand side) they threaten.

with the security threats, more extensive and detailed taxonomies on security requirements exist [Fir04, ALRL04, VM02, McG06, PP03].

Finally, the intersection of the goals and the design dimension involves the design process of realizing the relevant security goals through multiple design decisions such as the instantiation of security countermeasures, patterns [YHSJ06, Sch03, SFBH+06, FB13, SNL05], and principles [SS75].

Security requirements frameworks such as that of Haley et al. [HLMN08] can assist in the realizing the security goals in an iterative fashion, in line with the twin peaks model [Nus01, HYS+11].

2.5 Risk Analysis and Threat Prioritization

After identifying the relevant security or privacy threats in the system under consideration, these threats still have to be triaged and prioritized. None of the existing threat modeling approaches provides an explicit prioritization scheme, other than explicitly mentioning this as an additional activity in the threat modeling process that has to be performed after the elicitation of the threats.

The purpose of this section is to explore the application of risk analysis in support of prioritizing the elicited security and privacy threats. It is not the intent to attain full coverage of the complete risk analysis domain, which is out of scope and for which we refer the reader to the literature on the topic [Vos08, Hai05, NIS12, GJF06, NIS19b, FJ14].

In its essence, risk is defined as a function of (i) the adverse impact if an event occurs and (ii) the likelihood of occurrence [NIS12, GJF06]:

$$\text{Risk} = \text{impact} \times \text{likelihood}$$

Each of the following threat prioritization approaches try, in some shape or form, to assist in estimating one or both of these factors by breaking them down in components that are easier to estimate or rank.

2.5.1 DREAD

The DREAD [LeB07] approach for assessing the risk of identified threats uses the following subcomponents:

Damage provides an indication of the impact or size of the problem
Reliability specifies how reliably a threat can be realized by an adversary
Exploitability expresses the difficulty of exploiting the attack ranging from simple scripts to requiring high-level access to a system
Affected users to specify how many users are affected
Discoverability to indicate whether the attack is publicly known or requires intimate knowledge of the system

The *damage* and *affected users* focus on assisting in estimating the impact, while the other DREAD components focus on the estimation of the likelihood. Each of these categories requires a rating between 1 and 10. These ratings are then combined to obtain an overall risk score for the threat. However, "*DREAD is fairly subjective and leads to odd results in many circumstances. Therefore, as of 2010, DREAD is no longer recommended for use by the Microsoft SDL team.*" [Sho14]

2.5.2 Bug bar

The bug bar [Mic18] is used quite extensively at Microsoft [HL06, Sho14] to prioritize identified threats. The bug bar provides, for each

Server		
Critical	Elevation of Privilege:	*description of criteria*
Important	Denial of service: Elevation of Privilege: ...	... ...
Moderate	...	...
Low	...	...

Client		
Critical	...	...
...	...	...

Table 2.3: Illustration of the Structure of the Bug Bar
This table illustrates the structure of the bug bar [Mic18]. For every criticality level it provides the criteria for the STRIDE threat types to be assigned that level.

level of criticality, a detailed description of the criteria for a threat type to be assigned that level of criticality. For example, when discovering an *Elevation of Privilege* threat in a server software product, one can look up whether it meets the criteria for: a critical threat; if not critical, an important threat; if not important, a moderate threat; and so on. Table 2.3 provides a high-level overview of the structure of such a bug bar, based on the SDL example provided by Microsoft [Mic18].

2.5.3 Factor Analysis for Information Risk (FAIR)

The final risk analysis approach is the FAIR method from Freund and Jones [FJ14]. The FAIR method provides a detailed decomposition of risk into its underlying factors. Figure 2.3 provides an overview of the FAIR risk decomposition. Given the finer granularity of these underlying risk components, they can be easier to provide an expert estimate for instead of providing just the likelihood × impact assessment for risk. Furthermore, FAIR implies a numerical approach instead of low/medium/high risk categories which enables automating the risk assessment in tool support.

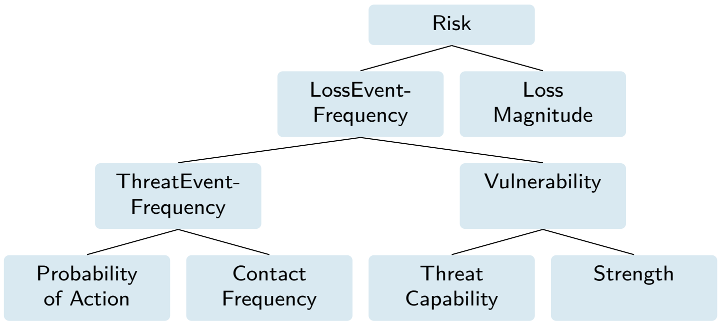


Figure 2.3: Overview of the FAIR Risk Components
The above figure illustrates how the risk is decomposed into sub-components.

The FAIR components are independent from the methodology used to determine the threats requiring the risk calculation, they do not impose a specific methodology such as threat modeling. While the FAIR model does not depend on such a specific methodology, the created DFD models can be used as a source of information for assessing the different risk components. The reliance on threat modeling artifacts to provide inputs into the risk assessment is covered in Chapter 4.

2.5.4 Other Risk Approaches

The focus of this section has been on the most relevant risk assessment methods in the context of threat modeling. There is, however, a large range of other security risk assessment methods such as, for example, CORAS [LSS10], OCTAVE [ABPW99, ADSW03], fault tree analysis [IEC06], and failure modes and effects analysis [IEC08].

2.6 Meta-Models

This section discusses the meta-modeling support separate from the three dimensions from Türpe, because meta-modeling represents more fundamental and cross-cutting background knowledge in support of the other dimensions. Indeed, meta-modeling support can actually be leveraged in each of these dimensions by: modeling the system itself (*design* dimension), modeling the knowledge on the different threat types and how they relate to one another (*threats* dimension), and modeling different security goals and the relations between them (*goals* dimension). Furthermore, the meta-modeling support is also relevant for the intersections of these dimensions (e.g., risk models).

This section starts with a generic discussion on meta-models to illustrate how they support the creation of modeling languages. Next, some more background is provided on the meta-models of the Unified Modeling Language (UML).

2.6.1 Modeling Supporting using Meta-Models

A meta-model is a model that describes other models. Every object in these other models is an instance of an object in the meta-model. Figure 2.4 shows a simple example of a meta-model to support modeling books and their authors. The meta-model provides: two concepts *Book* and *Person*, two attributes *title* and *name*, and two relations *author* and *cites*. With these elements provided by the meta-model, concrete models such as the one at the bottom of Figure 2.4 can be created. This example models two books and three authors. The meta-model also imposes some constraints for valid models. In this case, a book needs to have at least one author. Any model not meeting these constraints is not a valid model. Since the meta-model does not provide additional attributes such as *subtitle* or *year* of publication, it is not possible to represent this information in the model.

The above example illustrated how models contain instances of objects in the meta-model. This relation also applies to the meta-models. A meta-model has a meta-meta-model which describes the concepts that are available for creating meta-models and every object in the meta-model is an instance of an object in the meta-meta-model.

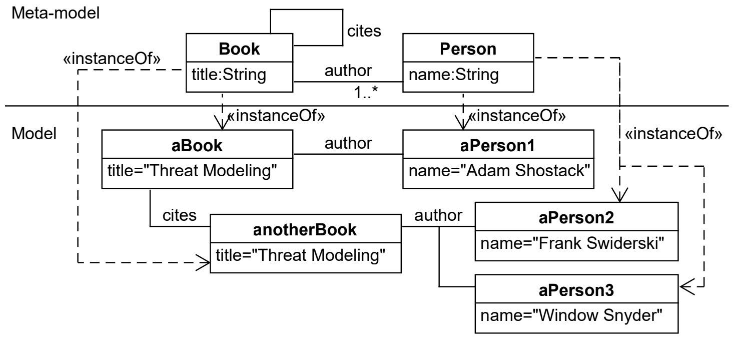


Figure 2.4: Simple Meta-Modeling Example
This figure shows a simple example meta-model for modeling books and their authors (top) and illustrates the use for creating a concrete model (bottom).

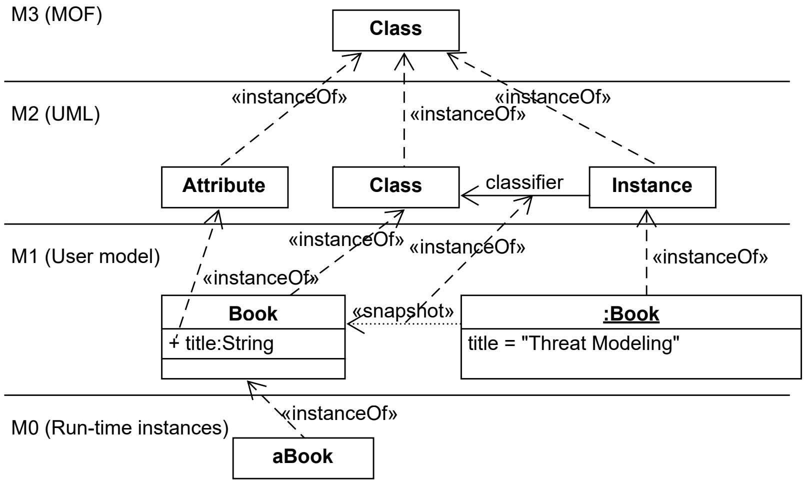


Figure 2.5: Four-layer example (UML Infrastructure Specification)
This diagram shows a modified example in the four-layer meta-model hierarchy from Figure 7.8 in the UML Infrastructure Specification [ISO12a].

2.6.2 UML Meta-Model Hierarchy

The UML also has such a meta-model hierarchy, in which the highest layer is reflective; meaning it describes itself and, thus, does not require any additional layers [ISO12a].

Figure 2.5 contains the example from the UML infrastructure specification [ISO12a] and illustrates the UML meta-model hierarchy. The top layer (*M3*) provides the meta-meta-model which provides the class concept that is used to specify the UML concepts on layer *M2* to support modeling classes, their attributes, and instances. Layer *M1* shows a concrete UML user model which uses the UML concepts to specify a *Book* class with an attribute and a *:Book* instance. This mechanism of providing an *Instance* concept in the meta-model enables the end-user to also model instances of the other created concepts (such as *Book* in the example), instead of having to move down another layer in order to create instances of the modeled concepts. For a full in-depth discussion, we refer to reader to the UML infrastructure specification [ISO12a], the Eclipse Modeling Framework (EMF) [SBMP08] and the literature [AK03].

Chapter 3 provides the full details on how these mechanisms are leveraged to enable the representation of security and privacy solutions in DFD models to support analyses incorporating this information.

2.7 Summary

This chapter provided some contextual background information to help understand and situate the contributions in the next chapters. Figure 2.6 shows a schematic overview of the following chapters together with the relevant background sections presented above.

In support of the modeling contributions presented in Chapter 3, we introduced the DFD design representation with the *process*, *data flow*, *data store*, and *external entity* element types in Section 2.1.

To support the threat elicitation and prioritization in Chapter 4,

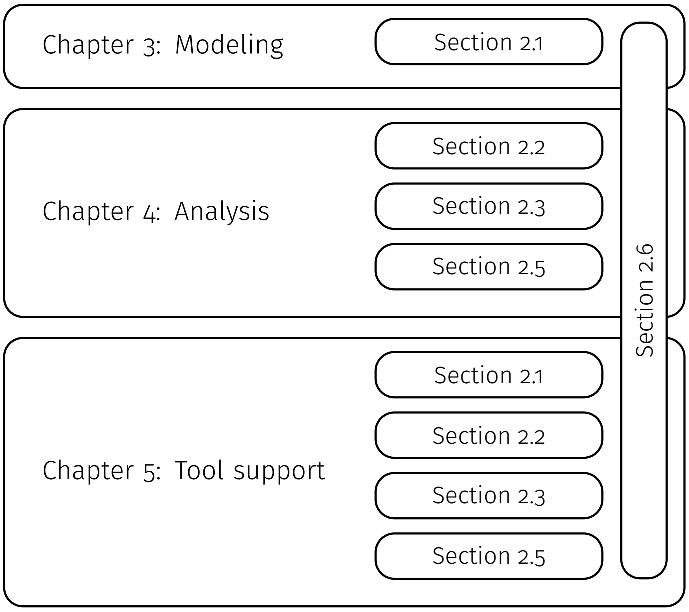


Figure 2.6: Mapping of Background Sections to Chapters
This diagram provides an overview of how the different background sections map to the following three contribution chapters.

we introduced the relevant background knowledge on the STRIDE and LINDDUN security and privacy threat types (Section 2.2), threat elicitation (Section 2.3), and threat prioritization (Section 2.5).

For the tool support chapter, the background on the DFD representation (Section 2.1), threat types (Section 2.2), threat elicitation (Section 2.3), and threat prioritization (Section 2.5).

Finally, meta-modeling (presented in Section 2.6) underpins the contributions presented in all the chapters as the modeling of solutions, threat elicitation and analysis, and implementation in tool support rely on extensive modeling support provided by a rich meta-model.

Chapter 3 Outline

3 Modeling Security and Privacy Concerns

All models are wrong
but some are useful.

— George Box [Box79]
Robustness in Statistics

This chapter is an extended version of the originally published article:
Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Solution-aware Data Flow Diagrams for Security Threat Modelling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1425–1432, New York, NY, USA, 2018. Association for Computing Machinery

Security by Design (SbD) and Privacy by Design (PbD) are principles that are increasingly recognized as essential to deal pro-actively and effectively with design flaws that can compromise security or privacy [AS16]. Their application is even required by the EU-wide General Data Protection Regulation (GDPR) [Eur16], as it imposes an approach embodying Data Protection by Design (DPbD) and *by default* for all systems that involve the processing of personal data. Systematic security [HL06, Sho14] and privacy [DWS+11, Wuy15] threat modeling approaches strongly contribute to the implementation of these *by design* principles because of their methodical and rigorous nature.

Data Flow Diagrams (DFDs) [DeM79] are the core artifacts used to support these design-based analysis activities. Since they are system-level abstractions that represent the system using four simple element types (see Section 2.1), they are in fact architectural views [BCK12, ISO11] that are relatively easy to create and comprehend. Furthermore, a DFD-based view of the system under design is strongly suited for focusing on the security- and privacy-relevant parts of the system (which are often centered around data items, and the locations where these are processed or stored). DFDs are used so frequently for this purpose that they are sometimes called '*threat model diagrams*' [Sho14].

However, threat modeling based on DFDs is not an activity that happens in total isolation. Indeed, the system under analysis could be an already-existing system embedded in a context in which certain security decisions and assumptions were already made. DFDs do not provide a way to capture this information as they do not support expressing any security- or privacy-related architectural decisions, i.e. the applied security and privacy solutions and assumptions, in a structured way.

This is problematic, because the lack of security- and privacy-relevant information can lead to the elicitation of threats that are (i) already resolved in the system but for which the corresponding countermeasures are not reflected in the DFD model or (ii) left undiscovered because the countermeasures that would trigger them (e.g., the confidentiality of cryptographic keys or the privacy impact of certificates containing personal information as attributes) are not considered in the analysis. To address the lack of solutions, existing approaches [Mic16, BSK16] add properties to DFD elements to capture the *effects* of these solutions.

In this chapter, we argue why properties are insufficient and present improved modeling support for capturing security and privacy solutions in DFDs by (i) identifying and discussing *four desired qualities* that emerge from the ability to express security and privacy solutions, related to semantics, traceability, separation of concerns, and dynamic and continuous threat assessment; (ii) presenting a meta-model that supports these qualities by enriching DFDs with security and privacy solutions to take into account during threat modeling and to enable exploratory change impact analysis; and (iii) *validate* the framework in the context of a STRIDE analysis of a WebRTC reference architecture.

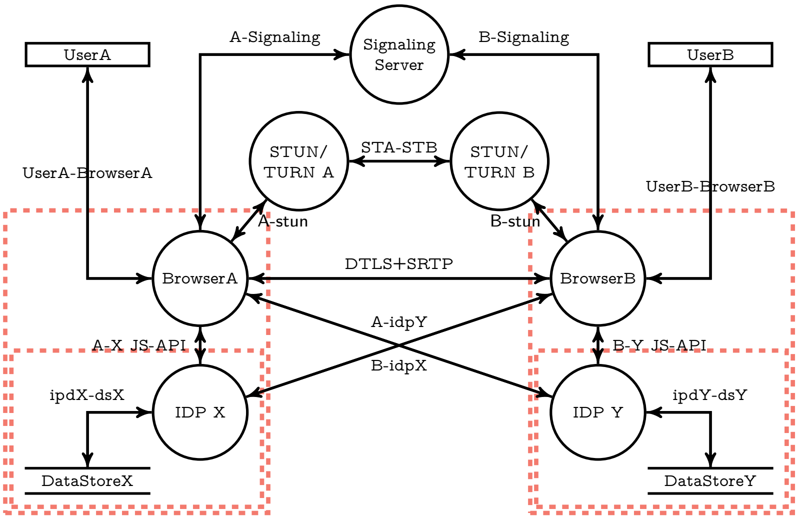
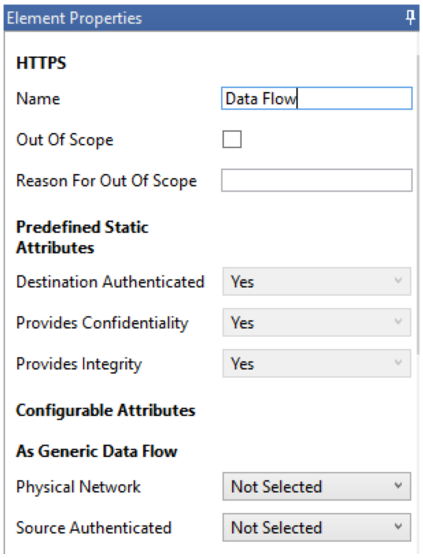


Figure 3.1: WebRTC Data Flow Diagram
Simplified DFD representation of the WebRTC reference architecture [Goo17]. Bidirectional flows are modeled as two separate flows in opposite directions.

3.1 Running Example

The concepts introduced in this chapter are illustrated using the WebRTC-based collaboration system [Goo17]. A systematic threat analysis with STRIDE [Sho14] is used to assess the security of this architecture. Such a threat analysis starts with the construction of a DFD model of the system (Figure 3.1). This WebRTC DFD contains 7 processes (circles), 2 external entities (rectangles), 2 data stores (two parallel lines), 4 trust border boundaries (the red dashed lines) and 28 data flows (bidirectional flows are modeled internally as two separate flows).

The Microsoft Threat Modeling Tool (TMT) [Mic16] is the most well-known and readily available tool for STRIDE threat elicitation and provides a good illustration of the properties-based approach for including security information in DFD models (also employed by other approaches in the literature such as, for example, Berger et al. [BSK16]).


Element Properties

HTTPS

Name	Data Flow
Out Of Scope	☐
Reason For Out Of Scope	

Predefined Static Attributes

Destination Authenticated	Yes
Provides Confidentiality	Yes
Provides Integrity	Yes

Configurable Attributes

As Generic Data Flow

Physical Network	Not Selected
Source Authenticated	Not Selected

Figure 3.2: Microsoft Threat Modeling Tool Data Flow Properties
This screenshot of the Microsoft Threat Modeling Tool shows the attributes that can be set for a data flow and how the HTTPS *subtype of a data flow already fixes a number of these attributes up-front to capture the effects of this solution.*

To support the augmentation of DFD models with security-related information, the TMT provides a set of customizable enum-like properties to are attached to DFD elements. For example, a data flow has the property '*provides confidentiality*', which can take the value 'Yes' or 'No'. Figure 3.2 shows a set of properties for a data flow in the TMT. In addition to the element properties, the TMT also subtypes several DFD element types. For example, the data flow type has an HTTPS subtype which constrains the '*confidentiality*', '*integrity*', and '*destination authentication*' properties to 'Yes'. Figure 3.2 illustrates the three predefined properties for an HTTPS data flow. This can only approximate solutions that affect a single element, since subtypes can only constrain the properties of the element they replace. Setting these properties, either manually or through subtyping, leads to an initial elimination of approximately 20% of the threats in the WebRTC case, reducing the amount of threats that require further examination.

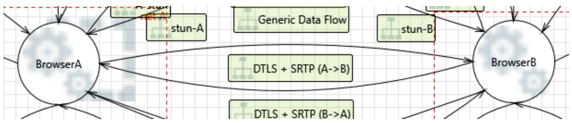


Figure 3.3: Example Threat Modeling Tool Flaw
This DFD *shows the part of the diagram for which the* TMT *matches the threat:* "Potential Lack of Input Validation for [destination] "*, with* [destination] = *BrowserB.*

To analyze the system for threats, the TMT comes with a catalog of 41 threat templates, which will later be analyzed in detail in Section 3.2. These templates contain a number of parameters (e.g., *source*, *target*, and *flow*) which are filled in with concrete elements during the threat elicitation phase. In the WebRTC example, STRIDE threat elicitation yields 236 threats. An example of such a threat is: "Potential Lack of Input Validation for *BrowserB*", which applies for the flow "*DTLS+SRTP*" from *BrowserA* to *BrowserB* (see Figure 3.3).

However, there are four main problems with this approach for representing security and privacy solutions. First, a single decision of using HTTPS for both sending and receiving data between two processes leads to, in total, six property constraints on two distinct data flows. Hence, the decision is scattered over multiple elements which makes it harder to assess its impact. Second, because of the nature of security as a moving target, earlier solutions may need to be revised in light of new information on their effectiveness (e.g., newly discovered vulnerabilities). Re-evaluating solutions becomes a complex endeavor when they do not comprise of a single element to replace or modify but, instead, consist of a combination of multiple element subtypes with property constraints and manually set properties scattered over multiple elements in the DFD while avoiding interference with other solutions. Third, the lack of a single representation may hamper auditing or compliance checking activities as solutions will have to be reconstructed from the effects in the diagram or by relying on separate documentation describing them but potentially inconsistent with the design. Fourth and finally, security and privacy expertise are scarce resources that rely on an extensive body of knowledge. Security and privacy solutions are prime candidates to provide in a reusable and extendable catalog to support consistency and reuse of these solutions across different models.

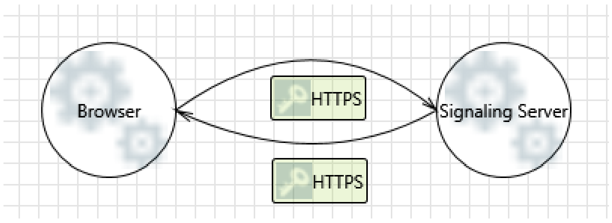


Figure 3.4: Threat Modeling Tool HTTPS data flow
This example illustrates the use of the HTTPS data flow subtype in the Microsoft TMT [Mic16] to model a data flow which provides confidentiality, integrity, and destination authentication. Because of the asymmetry of the HTTPS data flow, using this subtype causes the browser to become authenticated in the return flow.

3.2 Quality analysis

This section presents four desired qualities of security and privacy solution representations to support threat modeling activities. For each quality, it provides a discussion of the degree to which the quality is supported in state-of-practice tools or state-of-the-art approaches.

3.2.1 Expressiveness

A DFD model should support a first-class representation of security and privacy solutions, to enable verifying the correct application of these solutions and prevent ambiguity in interpreting their effects.

Security and privacy are text-book examples of cross-cutting quality concerns. The instantiation of security and privacy solution to address such cross-cutting concerns can impact multiple elements. This has the several implications when representing these solutions in a model: (i) because of the cross-cutting nature of security and privacy, their solution representations should capture which elements are affected by them, (ii) the representations should also capture in what capacity DFD elements take part in a solution, and finally (iii) the representations should facilitate the correct instantiation of these security and privacy solutions in a concrete design.

Example

The WebRTC architecture is used to analyze this quality in the context of a concrete example: the communications between the *Browser* and the *SignalingServer*, which are protected by HTTP over TLS, a common solution for the protection of web traffic.

State-of-the-art

The TMT offers the HTTPS data flow subtype, to represent this security solution (Figure 3.4). This subtype is part of the built-in set of types available in the TMT. The HTTPS data flow element constrains the following data flow properties: {*provides confidentiality*=true, *provides integrity*=true, *destination authentication*=true}.

Berger *et al.* [BSK16] use a similar approach with a hierarchy of types that specify annotations with the security effects, such as an HTTP connection as a subtype of a data flow. These subtypes can contain the necessary annotations such as *isEncrypted*.

Analysis

While the *destination authentication* property is a logical effect when transmitting data to an HTTPS server, this does not hold for data that is received from the server as clients are usually not authenticated in this context. Indeed, it more often not the case as illustrated by the large variety in application-level authentication mechanisms (e.g., username/password). However, the careless instantiation of the HTTPS data flow for both the sending and receiving flows implies that the client is also authenticated, causing threats for the client to be missed. For example, the threat of spoofing the client may be falsely eliminated based on the *destination authentication* attribute on the return flow. This illustrates how the capturing of a security or privacy solution in a DFD element subtype to constrain some of its properties is insufficient to fully capture asymmetric solutions.

Furthermore, the representation of security and privacy solutions as properties on DFD elements raises a number of interpretation difficulties because of semantic ambiguities about: (i) the target element of the property (the data flow itself or the connected element); (ii) whether a property on an element holds for: all the data flows connected to that element, only the data flows with the extra property set, or all flows except those with a property set to exclude them; (iii) whether a property on a data flow is implemented by the sending element or receiving element (or both); (iv) keeping the previous interpretations about the meaning, scope, and interpretation of these properties consistent over time and over multiple different properties; and (v) whether all the elements involved in a certain security or privacy solution have their properties correctly set.

3.2.2 Traceability

Security and privacy effects should be traceable to the security and privacy solutions that caused them. Similarly, solutions should be traceable to the specific threat type(s) they aim to prevent.

Bosch and Jansen [JB05] described how software architectures can be considered as a set of architectural design decisions. Managing these decisions then becomes an architectural knowledge management [BDLvV09] problem. This problem is also relevant for security and privacy decisions. Indeed, keeping track of such information is an essential requirement of the higher levels in security maturity models [OWA17b, MMW18]. These decisions have to be captured to understand the resulting system with its security solutions and the threats those solutions are intended to resolve, also described by Ven *et al.* [VJNB06] as bridging the gap between architecture and rationale.

By preserving the links between security and privacy effects (i.e. preventing specific threat types) on the system elements and the security and privacy solutions that provide these effects, these original solutions remain present in the model and prevent later modification from introducing conflicts because the knowledge on the original solutions was lost. Furthermore, the link from the solutions to the specific threat types they resolve should also be preserved as to record why (i.e. for which threats) such a solution was introduced in the design. Capturing this decision information with both the effects of security solutions and the threats those solutions address, previous decisions can be revisited later on and the resulting documentation can be used for compliance assessment and auditing. For example, the model documented with security and privacy solutions is a useful resource to collect all the applicable solutions involved in the protection of the personal data. This information is instrumental for assessing whether appropriate countermeasures are applied to protect personal data (and meet the obligations of the GDPR). Another example is the need to perform a Data Protection Impact Assessment (DPIA), which considers factors such as the type of processing and the sensitivity of the personal data. The model with security and privacy solutions can assist in performing such an assessment. Several of the DPIA checks can even be automated by relying on the system model and its solutions combined with a representation of the data processing activities [SDVL+19].

Example

Revisiting the example from Section 3.2.1, the model should capture that HTTPS is used to achieve confidentiality and integrity of the data flows between the *Browser* and the *SignalingServer*, and to authenticate the *SignalingServer*. Hence, the solution is introduced to prevent: (i) *information disclosure* and *tampering* of the data sent over these flows, and (ii) *spoofing* of the *SignalingServer*.

State-of-the-art

The property-based approaches (TMT and Berger *et al.* [BSK16]) do not have a first-class representation for including security and privacy solutions in the model. They did not introduce the security and privacy solutions as new elements in the model. Instead, these approaches capture the *effects* of the solutions as properties on the model elements. In the example from Figure 3.4, the decision can be reconstructed by combining the two HTTPS flows, but there is no guarantee that the correct elements are combined when revisiting security and privacy decisions. These reconstruction problems can only be avoided when the solution can be represented with a single element.

Analysis

Without a first-class representation for security and privacy solutions, the effects on the model elements can be linked back to the solutions that provide these effects. Similarly, the motivation as to why a solution was introduced is also missing. Reconstructing this information from the model requires, in the worst case, a review of the complete threat template catalog to find out any threat type(s) that are actually affected by the properties set in the model which is a very cumbersome and labor-intensive effort.

3.2.3 Separation of concerns

The threat type catalogs and security and privacy solution catalogs should be structured in a way to facilitate their independent evolution, in order to limit the impact of adding, updating, or removing threat types or security and privacy solutions.

The threat type and security and privacy solution catalogs address separate concerns that should be able to evolve independently. These catalogs should be structured to enable the introduction of modifications in one catalog without requiring additional changes in the other catalogs or artifacts.

Complete isolation of these artifacts is, however, not possible because of the dependencies between threat types and the solutions trying to prevent these threats. Because of the dynamic nature of security and privacy, new flaws in existing, previously thought secure, solutions will be discovered, requiring replacement of or modifications to these solutions. Therefore, the solution catalogs should support this dynamic aspect inherent to security and privacy, allowing changes to the security and privacy solution catalogs without impact other artifacts involved in the threat modeling process, to ensure that the solution catalog can remain up-to-date and evolve independently.

Example

Revisiting the HTTPS example from Figure 3.3, consider the case where a new TLS vulnerability is discovered (e.g., Heartbleed [CVE14], POODLE [MDK14], DROWN [ASS$^+$16b]). To be able to assess the impact of such a vulnerability, a re-assessment is required that takes this new information into account. This requires modifications to the solution in the catalog to update its effectiveness value. Such an update should only modify the effectiveness of the affected solution. Other solutions that also provide confidentiality through encryption should not necessarily be affected as they may rely on different mechanisms. Furthermore, this update should not require changes to the threat type catalog.

State-of-the-art

In the Microsoft TMT [Mic16] and the DFD extension presented by Berger *et al.* [BSK16], security solution information is embedded in the threat types as exclusion conditions. The example tampering threat in Figure 3.5 and information disclosure threat type in Figure 3.6 illustrate how, respectively, the '*Exclude*' or 'AND NOT' expressions refer to properties set on model elements in order to determine whether or not the threat is applicable for a specific DFD element. This is solution-specific information that is specified in the threat catalog.

Analysis

Having the threat descriptions rely on information stored in the element properties creates a strong dependency from the threat type catalog to the effects of the security solutions. This dependency hinders the independent evolution of the security and privacy solutions catalog, as every newly introduced solution requires adding the appropriate exceptions for its effects to all the relevant threat types. The overhead of

Title:
Potential Lack of Input Validation for {*target*.[Name]}
Include:

```
(source is [Generic Process] or source is [Generic
External Interactor]) and target is [Generic Process]
and (flow crosses [Generic Trust Line Boundary] or
flow crosses [Generic Trust Border Boundary])
```

Exclude:

```
flow.[Provides Confidentiality] is 'Yes' and
flow.[Provides Integrity] is 'Yes'
```

Figure 3.5: Threat Modeling Tool Tampering Threat Template
Tampering threat type from the Microsoft TMT *[Mic16] which shows the inclusion criteria for determine the threats applicability, and the exclusion criteria which rely on the element properties to determine whether the threat type is not applicable.*

```
MATCH //ommitted
WHERE //ommitted
AND ANY (d IN flow.data WHERE d.IsConfidential)
AND NOT flow.IsEncrypted
```

Figure 3.6: Threat Pattern from Berger *et al.*
This threat pattern from Berger et al. *[BSK16] illustrates how the pattern relies on the* IsEncrypted *property of a data flow to determine the threat's applicability.*

these operations becomes even larger when modifying existing solutions as now all threat types have to be checked for the exclusion criteria to verify they are up-to-date and consistent with the modified solutions.

3.2.4 Support for dynamism

Threat modeling activities should support partial and lightweight architectural design efforts, continuous evolution, and frequent architectural refactoring. Embedding threat modeling activities into agile development *processes requires novel approaches of dynamic and continuous threat modeling, and co-evolution with the system under design.*

Traditional threat modeling approaches assume a single-shot threat modeling exercise, conducted in the early stages of development. This is a consequence of the manual nature of the embodied threat modeling process which involves considerable effort by the threat modeler. Such a view, however, is in stark contrast with contemporary development practices such as agile development and continuous integration/delivery.

Example

As this quality pertains to tooling infrastructure to support continuously re-assessing a design, the example consists of a use case. The threat modeling approach and tooling should support continuous assessment to support the threat modeler in evaluating multiple design alternatives to assess which of these has the best impact on the resulting threats.

State-of-the-art

Current approaches in the state-of-the-art do not provide support for continuous assessment or trade-off analyses. The manual approach embodied in these approaches and limited expressiveness in the modeling reduces solutions to their effects on the model elements. The currently available threat modeling tools [Mic16, OWA18, Con18, Res19, SR19], while being able to automatically elicit threats, require the end-user to re-asses the list of results from the analysis to evaluate the impact from instantiating countermeasures. To our knowledge, no tools currently exist in the state-of-the-art or -practice that support the lightweight continuous assessment of design alternatives.

Analysis

To support this requirement, threat modeling tools should support embedding in lightweight software development activities by providing

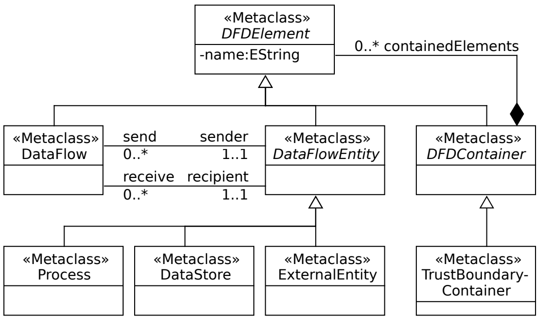


Figure 3.7: DFD Meta-Model
Unified Modeling Language (UML) diagram of the DFD meta-model. It shows the four main DFD elements: data flow*,* process*,* data store*, and* external entity*. Elements can further be grouped in* Trust Boundaries*. The meta-model enforces constraints such as requiring every flow to have a single sender and recipient.*

security and privacy design assistance such as: (i) raising awareness of security and privacy threats as the architecture evolves by frequently or continuously re-assessing the model while changes are being made, (ii) providing suggestions of security and privacy solutions to the architect to counter threats together with guidance to evaluate alternative solutions, (iii) providing an architectural impact assessment of these solutions (i.e. change impact analysis) by applying and evaluating alternatives, and (iv) integrating with existing security and privacy knowledge bases to incorporate changes in the effectiveness of the security and privacy solutions by reflecting changes to solutions such as the discovery of vulnerabilities back into the catalog of solutions.

3.3 Meta-Model

To improve the state-of-the-art in the four qualities defined above in Section 3.2, we propose a new meta-model as a foundation for threat modeling approaches. This meta-model consists of two parts. The first part provides a meta-model for DFDs, as it is previously largely used as a visual and informal notation [TK91] with multiple graphical variants [YC75, YC79, DeM79, GS79]. The second part extends upon the first one and introduces the necessary concepts for modeling security and privacy solutions and using this information in threat modeling contexts. It provides modeling support for: (i) threat types and their catalog, (ii) security and privacy solutions and their catalog, and (iii) instances of these security and privacy solutions to insert in a concrete model.

3.3.1 DFD Meta-Model

Before information about security and privacy solutions can be captured in a model, the underlying system representation needs to be specified. The DFD representation used in threat modeling [HL06, Sho14, DWS+11, Wuy15] is mainly a visual notation that does not precisely specify the elements and constraints. Tool support requires a meta-model to be able to analyze these models as illustrated in Sections 3.2.1 and 3.2.2. Such a meta-model can also assist in ensuring that the constructed DFD models are sound. Figure 3.7 depicts the meta-model that define the DFD element types and the relations between these elements. It also provides support for trust boundaries (with the *TrustBoundaryContainer* class), which are often used in the context of threat modeling to group elements together with the same '*trust level*',[1] to allow the exclusion of certain threat types between elements within the same trust boundary. For example, by assuming that machines on an internal network do not tamper with each other's traffic. When using trust boundaries, threats are often elicited only on data flows that actually cross over these trust boundaries.

Extensions to the DFD Meta-Model

Two major extensions are provided to the generic DFD formalism as already used in existing security and privacy threat modeling context: support for decompositions and support for modeling data types.

1 However, the meaning and implications of such a trust boundary are often implicit and left open to interpretation with multiple different meanings [SYvdB+20] such as: delineating trust or privilege, attacker assumptions, delineating between machines, or the presence of countermeasures (e.g., enforcement of access control) [Sho20].

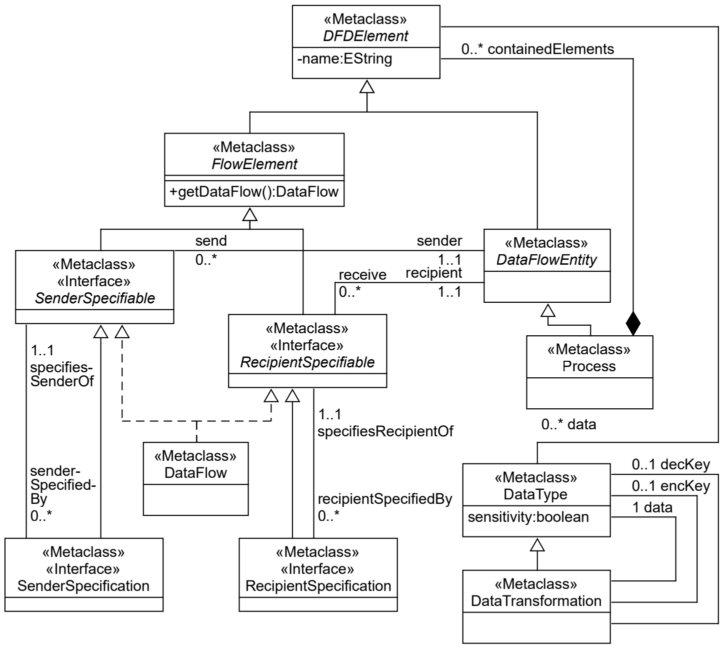


Figure 3.8: Dfd Extensions Meta-Model
The extensions make the dfd *modeling formalism more useful by allowing the decomposition of* processes*, while ensuring data flow consistency with the parent. The extension also supports modeling how data flows through the system.*

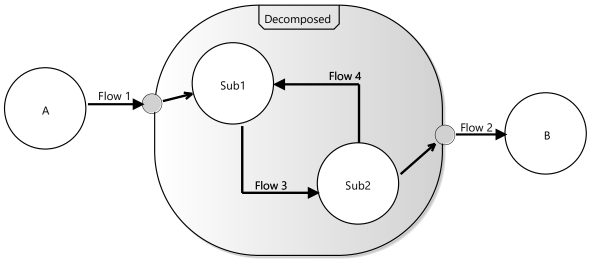


Figure 3.9: Dfd Decomposition Example
This example dfd *decomposition contains a* RecipientSpecification *(circle on the border) which specifies that the actual recipient of* Flow 1 *is Process* Sub1*, and a* SenderSpecification *which specifies that the sender of* Flow 2 *is Process* Sub2*.*

Decompositions By supporting decompositions, larger systems can be modeled at various abstraction levels, keeping the resulting DFD models still manageable and comprehensible. Decomposing DFDs is supported in the original notation [DeM79]. However, formalizing the decomposition support in the meta-model ensures the construction of consistent decompositions. The extension for decomposition changes the process element into a container. This way, a process can contain any number of sub-elements as required for modeling more complex systems. The *data flow* mechanism is also extended with *SenderSpecifiable* and *RecipientSpecifiable* classes. These classes specify that there is a more specifc sub-*process* at the sender or receiver that actually sends or receives the data. By convention, a decomposition is complete in the sense that all data flows at the parent element need to have a subprocess as sender or recipient. I.e. there should be no hidden functionality remaining in the parent. Figure 3.9 illustrates the decomposition with a concrete example, in which the *SenderSpecification* and the *RecipientSpecification* are visualized on the border of the decomposed process. These specifications point to the sub-processes that are the actual recipients or senders from the data flows to ensure all data flows have a final source or destination process.

Data Types The second major extension to the DFD meta-model involves the support for modeling data in the system [TSSY20]. The extension supports the modeling of different data types, and where these data types are transferred, processed, or stored in the system by explicitly linking from a DFD element to the data type. For example, a data flow can specify the different data types it transfers. Furthermore, the relations between the different data types are modeled as well. This enables the modeling of plain text and encryption pairs, and which data serves as a key for encrypted data. These extensions enable the detection of data-related security threats such as unprotected sensitive data and improper key management [STY+19, TSSY20]. This extension supports tracking how sensitive data can be transformed (e.g., encrypted) and further processed. Other DFD data extensions in the literature [TSWS17] support different scenarios such as tracking the sources and destinations of data elements.

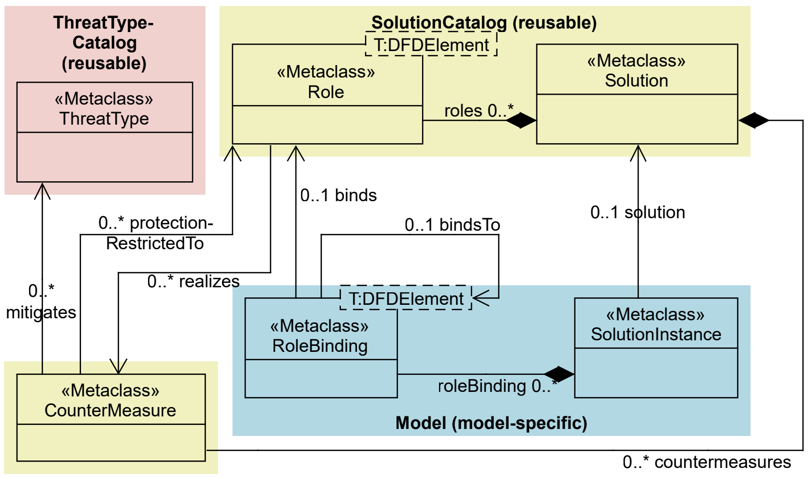


Figure 3.10: Security and Privacy Solution Meta-Model
This figure shows the meta-model classes for representing security and privacy solutions. The colors indicate the reusable solution catalog (green), the DFD model-specific (blue), and the threat type catalog (red) elements.

3.3.2 Security and Privacy Solution Meta-Model

Generic DFDs do not support the representation of security and privacy solutions in the model. While pragmatic extensions with properties attached to DFD model elements can include some security or privacy information, such solutions suffer from several problems as discussed in Section 3.2. We therefore present a more extensive meta-model to capture all the relevant details of security and privacy solutions.

Figure 3.10 depicts the meta-model for representing security and privacy solutions in DFDs. The generic security and privacy solutions are represented as *Solutions* in a catalog to enable sharing and reuse of this knowledge across multiple DFD models. The catalog can be used to capture existing sets of security and privacy solutions [YHSJ06, FB13, SFBH$^+$06]. These solutions can be subsequently instantiated in a concrete DFD model using *SolutionInstance*s.

A security or privacy solution is specified using the meta-model as follows. A new *Solution* is created which contains a list of *Role*s that are generic descriptions for the DFD elements involved in the solution. These *Role*s are parameterized according to the type of the DFD elements that can fulfill the role (e.g., *Process* or *Data Flow*). An example of this is a Secure Pipe [YHSJ06] *Solution* which contains four *Roles*, two *Process Role*s for the client and server, and two *Data Flow Role*s for the sending and receiving flows between them.

A *Role* can realize a number of *Countermeasure*s. Each *Countermeasure* specifies against which threat types it protects and to which other *Role*s the protection is limited (i.e. the scope of the countermeasure). Contrary to the property, this provides additional information on which solution provides the *Countermeasure*, what the scope of its protection is, and it avoids conflicts when multiple solutions influence the same properties. In the case of the Secure Pipe, the two data flow *Role*s have *Countermeasure*s against tampering and information disclosure, while the server *Role* has an authentication *Countermeasure* to protect against spoofing. The protection against spoofing is limited to the two *Data Flow*s in the solution. This is modeled by having the authentication *Countermeasure* explicitly link to the roles that are in scope (using the *protectionRestrictedTo* relation). Therefore, other clients that communicate with the server outside of the Secure Pipe are not protected against spoofing of the server. The generically modeled Secure Pipe pattern illustrates how common solutions such as HTTPS to protect web traffic can be modeled using the solution meta-model.

Finally, to be able to use these security and privacy solutions in concrete DFD models, they have to be instantiated in these models. To instantiate the solutions, the meta-model offers the *SolutionInstance* class, this class represents a concrete instantiation of a *Solution* in a specific DFD model. Analogously to the *Solution*'s *Role*s, the *SolutionInstance*s contain a list of *RoleBinding*s. A *RoleBinding* links a *Role* from the generic *Solution* to a concrete DFD element to specify which concrete element in a DFD model fulfils that *Role*.

Figure 3.11 provides a schematic representation of how solutions can be instantiated in a concrete DFD model. This is separate from any visualization that may be implemented on top of the meta-model. The right-hand side of Figure 3.11 shows the threat type catalog on top

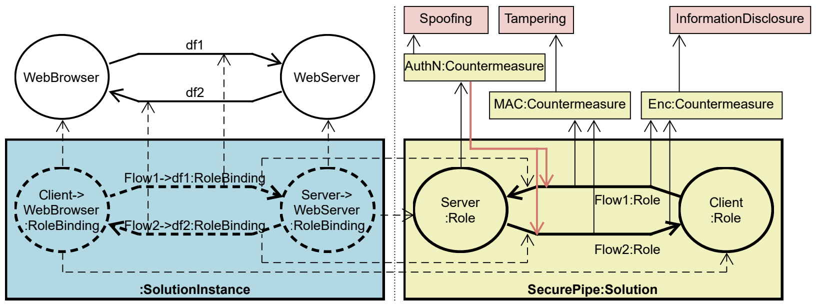


Figure 3.11: Example instantiation of the secure pipe solution
The right-hand side shows (from top to bottom) the ThreatType*s,* CounterMeasure*s, and* SecuritySolution *containing 4* Role*s. The left-hand side shows the* DFD *model and (below that) the* SolutionInstance *containing the 4* RoleBinding*s, linking* DFD *model elements to the solution's roles. The same color-coding is used as in Figure 3.10. Note that the focus is on the model semantics, not the visualization.*

and a single solution, *SecurePipe* at the bottom with four roles which realize three countermeasures. The left-hand side of Figure 3.11 shows a concrete DFD model, in which a single instance of the *SecurePipe* solution is instantiated using four *RoleBinding*s, which link each of the *Solution*'s *Role*s on the right-hand side to concrete DFD elements.

3.3.3 Impact on Threat Elicitation Process

The usage of the solution-enriched DFDs is expected to have minimal impact on existing threat elicitation activities. During threat elicitation, the elicitation engine (or threat modeler in case of a manual threat elicitation) iterates over the elements [HL06, DWS⁺11, Sho14] or interactions [Sho14, SWY⁺18] in the DFD and for each of these an iteration over the *ThreatType* catalog is performed to verify whether the considered *ThreatType* is applicable in that context. This verification step involves checking whether the DFD element is not linked to a *Role* that provides a *Countermeasure* against the considered *ThreatType* (also taking into account the scope of the *Countermeasure*). It is not necessary to know the details of the specific security or privacy solution in question.[2] For example, the presence of an encryption countermeasure can prevent an information disclosure threat on a data flow. The threat elicitation engine does not need rely on *ThreatTypes* specifying all potential countermeasures to know whether the threat is prevented (due to, for example, a VPN or HTTPS) as it can extract the necessary information from the solution catalog. This is an advantageous property, because it ensures that the threat type catalog remains independent from the solutions catalog.

3.4 Evaluation

This section provides a qualitative evaluation of the presented meta-model using the four qualities presented in Section 3.2. The functional validation of the meta-model is deferred to Chapter 5, which provides a complete discussion of the implemented prototype SPARTA.

3.4.1 Expressiveness

The meta-model supports a first-class representation of security and privacy solutions in DFD models by creating solution instances that bind concrete DFD model elements to the solution's roles. It is the binding to the role with the countermeasure that determines whether a specific threat type applies to an element. This generic representation of security and privacy solutions in DFDs has the following advantages: (i) asymmetric security and privacy solutions (recall, for example, the HTTPS solution) can be expressed, as the effects are realized in different roles; (ii) it avoids any ambiguity in interpreting the effect of a countermeasure as it explicitly supports scoping the effect of a countermeasure; and (iii) it does not lead to conflicts when multiple security or privacy solutions affect the same element, as this would just require multiple role bindings to the different security or privacy solutions (while the specialization approach to DFD elements would

2 The expert encoding the security and privacy solutons in the catalog will need to be aware of these particulars to appropriately capture the effects and scope of the solutions and update their effectiveness as new vulnerabilities are discovered.

require the same DFD element to be replaced with multiple different elements at the same time).

The following reasoning provides a qualitative argument as to why the presented meta-model provides a strictly positive improvement with regard to the semantic quality. The argument is two-fold: (**A**) the presented meta-model is equally expressive as the property-based approach, and (**B**) the presented meta-model can express solutions that cannot be properly represented with the property-based approach.

A The meta-model be shown to be equally expressive as the property-based approach by simulating it. Every possible property on a DFD element can be represented as a very simple solution with a single role that captures the effect of the property in the attached countermeasure.

B The example in Section 3.2 for Section 3.2.1 presents the instantiation of HTTPS for the protection of the communication between a client and server. This security solution is asymmetric in that it provides authentication of the server to the client, but not the other way around. So far, this solution can be expressed as a specialized data flow element. However, consider that this flow is now used to communicate signed and encrypted email messages (e.g., using S/MIME), which also provides authentication of the sender. In the property-based approach such a solution cannot be added, as there is already another specialization of the data flow to represent the HTTPS solution. In order to still capture the effects, a new data flow specialization would have to be created to represent the combination of HTTPS and S/MIME, or the effects would have to manually set via the properties (losing the first-class representation, traceability, and allowing for inconsistencies to be introduced). Either of these options is sub-optimal and does not scale well when new or combinations of solutions are introduced.

3.4.2 Traceability

Traceability is a relevant quality when revisiting and evaluating previously made security and privacy decisions. The property-based approach can only offer traceability when the DFD element specializations are used extensively. However, as the evaluation of the semantics in Section 3.4.1 explained, this can lead to conflicting element specializations to be required at the same time, otherwise the designer will have to revert to a scattered set of effects on the different elements. The explicit first-class representation for security and privacy solutions allows for the explicit manifestation of every decision as a solution in the DFD model. This makes the effects of these decisions traceable to the security and privacy solutions that caused them.

Besides the traceability from security and privacy effects to the solutions causing these effects, the security and privacy solutions expressed in the meta-model also link to the corresponding threat types in the catalog. This also enables traceability from the security and privacy solutions to the different threat types they are intended to counter. In the property-based approaches, the threat elicitation activity will need to check for the presence of certain properties on the DFD elements under consideration. To navigate from a property to the threat types it is supposed to counter would then require checking all the defined threat type conditions for mentions of that specific property.

The traceable link between security and privacy solutions and the threat types they counter has a number of additional benefits: (i) threats are not eliminated (included or excluded) but mitigated by one or more security or privacy solution instance which allows looking up which solutions mitigate a certain threat; (ii) the explicit link supports additional analyses of a threat's impact and likelihood, taking the precise system context and relevant security and privacy solutions into account (see Chapter 4 for more information on this); (iii) changes in a solution's effectiveness or countermeasures can easily be evaluated over all instances of that solution (for example, when a solutions countermeasure against information disclosure fails, modifying the generic solution updates the effect of all its instances in a concrete DFD model), thereby supporting the realization of Section 3.2.4.

3.4.3 Separation of Concerns

The separation of concerns quality is evaluated by comparing the impact of change in: (i) changing security and privacy solutions (add, modify, or remove), and (ii) changing threat types (add, modify, or remove) in both the property-based and meta-model representations.

Changing Security and Privacy Solutions

Property-based Adding a new security or privacy solution requires: new properties to be defined to capture the effects of the new solution, optionally new element specializations to constrain these properties, and updating the threat types that are prevented by the solution to add the properties to the exclusion criteria.
Modifying or removing a solution requires the following changes: all the threat type exclusion criteria have to be checked and updated to modify or remove the corresponding properties, any specialization using the modified properties has to be updated, and finally any existing models that have manually set these properties have to be updated as well. Changing security or privacy solutions thus introduces a ripple effect, especially in the exclusion criteria and any manually set properties.

Meta-model Introducing a new security or privacy solution only requires a change in the solution catalog to add: the solution, its roles, and the countermeasures which link to the threat types they counter. Modifying or removing a solution only impacts existing models when the solution or any of its roles are removed. Other changes remain limited to the catalog and will automatically be reflected in the models using the solutions from that catalog.

As illustrated with the numerous vulnerabilities collected in vulnerability databases [CVE19a, NIS19a], the effectiveness of existing solutions can quickly change. The modeling approach should support the modification of these solutions to reflect this reality without cascading the required changes to other elements, in order to make the solution catalog easier to maintain. Changing the solutions is quite problematic for the property-based approaches as they require revisiting the threat elicitation criteria. The meta-model-based approach is better equipped to cope with such changes as the impact remains limited to the solution catalog and, more specifically, the affected solutions.

Changing Threat Types

Property-based Adding, modifying, or removing threat types only requires changes to the threat types themselves. No other changes to the properties or element specializations are required, unless new properties are introduced, or older unused properties are removed. Otherwise, these remaining unused properties may cause confusion as they imply a certain security or privacy effect, but the elicited threats will remain unchanged.

Meta-model Adding or modifying threat types only requires changes to the threat type catalog. Removing threat types, however, can impact the solution catalog. Some solutions may prevent these threat types that no longer exist and will have to be updated.

Both the property and meta-model approaches can limit the impact of changing threat types to their respective catalogs, although the property-based approaches can introduce some ripple effects as older unused properties remain in the models and can introduce confusion. When reusing existing properties, the property-based approach can already take existing solutions into account. However, while this does prevent the re-assessment of existing solutions, it can have unintended consequences as the interpretation of the effect of a property may not be unambiguous which can cause the unintentional omission of threats that are not mitigated. By design, the meta-model solutions explicitly refer to threat types to avoid these problems.

3.4.4 Support for dynamism

The final quality, support for dynamism, strongly depends on the availability of appropriate tool support. Therefore, in this context, the evaluation of this quality is limited to an assessment on how, and to what extent, the presented meta-model provides the foundation for enabling this quality in tool support. Later on, Chapter 5 will discuss the realization of this quality in the SPARTA threat modeling tool. The following four dimensions are considered in the assessment of the dynamism quality: (i) threat evolution, (ii) suggesting solutions, (iii) impact analysis, and (iv) knowledge base integration.

Threat Evolution Any approach relying on a system model that can be dynamically re-queried for eliciting security and privacy threats can support the threat evolution dimension. Given a model-based representation, both property-based and model-based approaches support this dimension of dynamism.

Solution Suggestion In order to support suggesting security and privacy solutions for the elicited threats in a concrete DFD model, traceability of the threat types and solutions is required. The realization of this aspect requires the relations between the threat types and corresponding solutions to be traversable. The relation is not directly captured in the property-based approach, as every solution that influences the properties that determine the applicability of a threat type has to be considered. The solutions directly refer to mitigated threat types and, therefore, make solution suggestions relatively straightforward.

Impact Analysis A straightforward way to analyze the impact of introducing new security or privacy solutions is to repeat the threat elicitation and compare the results. Any approach that handles threat evolution well can assess the solution impact this way. More advanced solution impact analyses, such as the number of involved elements or interactions with other solutions require first-class solution representations that are lacking in property-based approaches.

Knowledge Base Updates The consolidation of security and privacy solutions in a catalog forms a good foundation for a reusable design security and privacy knowledge base. To further improve the relevance of this resource, updating this resource with recent vulnerability and flaw information from online resources can assist in dynamically re-assessing the impact on a system. This requires a flexible and updatable specification of solutions. Specializing DFD element types may prove to be too inflexible to cope with frequent changes.

3.4.5 Summary

The four previous sections illustrated how the first-class representation of security and privacy solutions in DFD models provides a number of advantages in: (i) semantics, by increasing the expressiveness to support representing complex solutions involving multiple elements; (ii) traceability, by enabling bidirectionally tracing between solutions and the threat types they counter; (iii) separation of concerns, by avoiding changes outside the modified solutions and minimizing the impact of changes to threat types; (iv) dynamism, by supporting multiple dynamic use cases when relying on the first-class solution representation in tool support.

3.5 Discussion

The discussion below considers some of the implications of using the presented meta-model in threat modeling contexts and reviews some additional extensions to further improve the representation of security and privacy solutions along the aforementioned qualities.

Semantics of Trust Boundaries

The presented meta-model supports trust boundaries as a container element that can contain other DFD elements. All the elements inside the same container are within the same trust boundary. While this does support nesting multiple levels of trust boundaries, it is not possible to intersect trust boundaries so that they share only some elements but not others. Supporting these models would require a different kind of representation for trust boundaries. In current threat modeling approaches [SS04, HL06, Sho14], trust boundaries influence the applicability of a threat type only when there is a data flow that crosses a trust boundary. When using rectangular (i.e. closed) trust boundaries, data flows between elements in different intersecting trust boundaries will always have a data flow crossing the trust boundary, so this is not a severe limitation as it does not influence the threat elicitation results. Only with trust boundaries consisting of a single line (i.e. open trust boundaries) can the results differ, but these lines can introduce inconsistencies in which some data flows between the same DFD elements cross the trust boundary while others do not.

Introducing new Model Elements as Part of Solutions

In the current meta-model's representation of security and privacy solutions, a solution instance's bindings require existing DFD elements in the model to be bound to the solution's roles. However, some solutions may want to introduce new elements in the system that would not exist separately outside of the solution they are a part of. Currently such elements would have to be introduced manually and would be indistinguishable from other system elements in the DFD. Consider, for example, an encryption solution which could introduce a local key store (as data store) or rely on an existing one in the system. Hence, it is still possible to model and use these solutions with the added effort of creating additional, for example, processes to fulfil the extra roles of the solution. The main problem with the lack of support for solution-specific elements is when they are replaced or removed. Then, these additional elements introduced for specific solutions may no longer be needed and have to be removed as well.

Decoupling Solutions and Threat Types

While the current representation of security solutions is already largely decoupled from the threat types, there still is a reference from a security solutions countermeasure to the threat type it prevents. When threat types would be removed, these solutions would have to be updated. If necessary, it is possible to further decouple them by introducing an intermediary concept of a security or privacy effect. Both security and privacy solutions and the threat types could refer to these intermediary effects, allowing both of them to be changed independently. This intermediary concept is currently not implemented in the model because changes to the threat type catalog (which reflects the existing STRIDE [HLOS06] and LINDDUN [DWS+11] threat types) were deemed to be less frequent compared to the changes to security and privacy solutions [YHSJ06, Sch03, SFBH+06, FB13].

Usability of the bindings

The richer expressiveness for including security and privacy solutions in DFD models introduces some complexity when instantiating solutions in concrete models. The solution representation provides some support to ensure a correct instantiation of solution instances by: (i) providing the complete set of roles that have to be fulfilled by elements in the model and (ii) restricting those roles to specific types of DFD elements. The user still has to bind the roles to the correct model elements. Tool support can provide assistance by automatically instantiating solutions to avoid user errors in the role assignments.

3.6 Related Work

This section discusses the related work. First, a number of existing DFD extensions is covered together with threat modeling tool support in which the discussion is primarily focused on the underlying modeling support for security and privacy solutions. This discussion covers both security and privacy threat modeling approaches and model extensions. Next, a number of non-DFD-based modeling representations are discussed to provide some insight into the support in other modeling languages. Following that is a discussion on security and privacy requirements elicitation approaches and to which degree those approaches support relying on previously made security and privacy design decisions. Finally, we end the discussion with other security and privacy design analysis approaches that do not fit into the aforementioned categories but that do systematically analyze a system to identify security or privacy threats.

3.6.1 Security and Privacy Threat Modeling

A number of tools are available to support threat modeling activities. The Microsoft Threat Modeling Tool [Mic16, Mic20] is the most common and readily available tool support and provides properties on the DFD elements to record the effects of security solutions in combination with DFD element subtypes that force some of these properties to appropriate values. Another publicly available tool is OWASP's ThreatDragon [OWA18]. ThreatDragon also relies on properties (e.g., *IsEncrypted*) to capture the effects of security. There is, however, no automatic threat elicitation that takes these into account. The Irius Risk tool from Continuum Security [Con18] uses a different approach than the above tools. The DFD in Irius Risk is an auxiliary view generated from a list of elements. The next version of Irius Risk will provide direct modeling support in a graphical editor. The primary content driving the elicitation of threats is driven by a list of components that is constructed by asking the user a number of questions on the used technologies. Later on, different types of countermeasures can be specified to mitigate the identified threats. There is thus no separate first-class representation of the solutions in the design, nor is it possible to capture a single solution affecting multiple elements. The next threat modeling tool is OVVL [Res19, SR19], which similar to the Microsoft TMT and OWASP's ThreatDragon provides support for creating DFD models where properties on the DFD elements can be set (e.g., *Authenticates itself*) to indicate the effects of security solutions. These can subsequently be considered during the elicitation. Finally, there is the SecuriCAD tool from Foreseeti [For20]. SecuriCAD does not rely on a DFD but instead uses a custom model which is more focused towards infrastructure with support for elements such as hosts, networks, routers, protocols. Models in SecuriCAD are analyzed with attack simulations where a certain attacker profile traverses through the system by exploiting weaknesses in intermediary hops. These attack paths are very similar to attack tree analyses [Sch99].

With the increasing importance of privacy, threat modeling approaches have been extended to elicit privacy threats [DWS+11, Wuy15] in support of privacy by design [AS16]. While privacy threat modeling approaches are sufficiently similar for existing threat modeling tool support to elicit privacy threats, there are current no threat modeling tools available that do so. Incorporating such functionality in existing tool support would rely on similar property-based mechanisms to incorporate the effects of privacy solutions to prevent privacy threats.

3.6.2 DFD Extensions

In addition to threat modeling tool support, there are some DFD extensions that incorporate additional information in the models to take into account in subsequent analysis activities.

Dhillon [Dhi11] describes the use of annotations on DFD model elements (e.g., privilege levels, programming languages, authenticated data flows, encrypted data flows). Developers receive guidance but the annotations do not have a specific structure but are applied in an ad-hoc fashion, which limits the use to manual threat elicitation activities.

Berger *et al.* [BSK13] present a meta-model for DFDs in which the generic *Element* type contains a list of annotations to express information on different security measure. Contrary to Dhillon, these annotations are specified systematically in the model and refer to the threat types they prevent in order to support automated analysis. While this does not pose a problem in their envisioned use case of extracting the security architecture of existing applications, the individual notations do not support expressing and reasoning about more complex security solutions that affect multiple elements, nor do they assist designers in ensuring that solutions are correctly applied.

In later work, the same authors [BSK16] introduce Extended Data Flow Diagrams (EDFDs), which abstract away the DFD element types to *elements*, *channels* for data flows, *trust areas* (i.e. trust boundaries), and introduce *data.* Their approaches supports the creation of different subtypes for these element types in order to set some of the properties (e.g., *Java Process* implying a *Process* with the *Java* annotation, or *Credentials* implying a *Data* type with the *IsConfidential* annotation). These EDFDs are translated graphs for analysis on which patterns created for different CWEs or CAPECs are matched (as illustrated in Figure 3.6). While they briefly mention support for solutions, they seem to be limited to subtypes (e.g., an SQL-based database) or reusable model fragments to model common technical solutions.

A final DFD extension is the extended DFD notation of Tuma *et al.* [TSWS17] which introduces a number of extensions for marking assets. More specifically, the asset *sources*, *targets*, and their security objectives, to support the end-to-end tracing of these assets through the system. Their approach to asset-centric threat modeling has been applied to automotive domain. Our extension, presented in Section 3.3.1, focuses more on tracking data types and their transformations (such as the ciphertext after encryption) in the model.

There are also DFD extensions specifically focused on privacy. Antignac et al. [ASS16a] presented Privacy-Aware Data Flow Diagrams (PA-DFDs) which contain a number of privacy-aware annotations to support privacy concepts: *data subject*, *data controller*, *data processor*, and *purpose*. Our presented meta-model does not introduce privacy-specific annotations (e.g., data purpose), but, instead, relies on links to privacy threats in the catalog to represent privacy solutions. This is focuses on technical privacy threats, while the PA-DFD extension is very suitable for compliance assessment exercises and can assist in bridging the gap between software engineering views and legal views on the personal data processing operations [SDVL+19].

Finally, there are some efforts towards more formal underpinnings for DFD for functional correctness [TK91, Fra92], these do not specifically target security analysis activities. For a more extensive discussion on threat modeling and DFD-based analysis techniques, we refer the reader to literature reviews on the topic [TCS18, XL19].

3.6.3 Non-DFD-Based Modeling Approaches

Moving away from DFD-based modeling notations, there are many different security modeling notations, many of them based on UML. Given our focus on DFD-based extensions, we briefly mention the most important security modeling notations.

The UML-based modeling notations such as UMLsec [Jür05], SecureUML [BDL06], and Secure Object Flows [HSS12] rely on the existing UML [ISO12a, ISO12b] extension mechanisms such as stereotypes to include additional security-relevant information in the models. The same approach is also applied for including privacy-relevant information in UML models [ASRJ18]. These stereotype-based approaches are quite similar to the annotation-based approaches [Dhi11, BSK13] for including security information in DFDs. Instead of threat modeling the resulting diagrams, these approaches rely on their own UML-based tool support to assess or prove whether certain security properties can be met in the resulting system model [Jür05] or to apply model transformations to enforce the modeled access control policy [BDL06]. For a full overview of other security modeling notations, we refer the reader to the literature [vSYJ17, UFF12]

3.6.4 Solution and Threat Knowledge

In addition to the semantic support for representing security and privacy solutions, there is also a need for knowledge bases containing security and privacy solutions to be modeled and used in the resulting DFD models. For this the existing literature on security and privacy pattern can be consulted for security [Sch03, DGFRLP04, SNL05, YHSJ06, SFBH$^+$06, FB13] and privacy [CHH16, Pri20] patterns. The existing pattern catalogs confirm the need for a first-class representation for security and privacy solutions, as they have to rely on ad-hoc and improvised notations to represent the solutions. Regardless of the pattern knowledge source, a translation step will always be required to convert the pattern information into a DFD representation.

In the current threat modeling approaches, the main source of the threat knowledge information are the STRIDE [HLOS06, HL06, Sho14] and LINDDUN [DWS$^+$11, Wuy15] threat types and threat trees. The knowledge sources can be expanded by relying on other collections of knowledge on security and privacy threats [HLOS06, HL06, Sho14, DWS$^+$11, Wuy15] vulnerabilities [CVE19a] and their impact on the effectiveness of security and privacy solutions, weaknesses [CWE20, STM17, ADD$^+$14, OWA17a], and attack patterns [CAP18].

3.6.5 Other Security and Privacy Analysis Techniques

We close the discussion on the related work with some additional security and privacy analysis approaches that do not operate from a design description, such as DFD, from the system. One approach closely related to threat modeling are attack trees [Sch99]. They start from the perspective of an attacker that wants to realize a certain attack on an system asset, and from there on explore the attacker's possibilities to achieve that goal and what possible countermeasures can prevent it. Such an approach is more perpendicular, as it enables an in-depth analysis of a particular threat and provides guidance in choosing how to protect against that threat. Related to that are abuse or misuse cases [MF99, SO05] which formulate undesired behavior from a misuser that has to be prevented by the system. Other types of anti-requirements are Abuse Frames [LNI+03]. To bring some systematicity in the attack analyses, the existing attack pattern repositories such as CAPEC [CAP18] can be leveraged [LPM+16].

3.7 Conclusion

Integrating threat modeling activities in the software development life cycle is a promising and effective way to implement the SbD and PbD principles. Unfortunately, existing threat modeling approaches that rely on plain DFDs lead to a combinatorial explosion of threats to consider. To keep the resulting lists of threats manageable, they require extensions to exclude certain threats from being raised.

However, the careless application of these extensions can lead important security and privacy threats to be omitted because of the ambiguities in properly capturing the effect of security and privacy solutions. In this chapter, we introduced modeling support for capturing this security and privacy information to limit and scope the threat elicitation space.

We have shown positive improvements in terms of semantic quality, traceability, separation of concerns, and dynamism, respectively due to: (i) the first-class representation of security and privacy solutions in the system under design, (ii) traceability of security and privacy effects to the solutions causing these effects, and thereby ensuring that the decisions to apply these security and privacy solutions remain documented, (iii) independent evolution of the security and privacy solution catalog to stay on par with changes in the field, and (iv) dynamic and continuous threat assessment by providing impact analysis and architectural security and privacy decision making support.

We argue that the additional complexity introduced for representing security and privacy solutions is manageable in practice with appropriate tool support and provides a number of compelling benefits in expressiveness, traceability, separation of concerns, and support for dynamism. With tool support that better leverages the available models, more advanced threat analysis activities are possible that move away from a binary threat classification (i.e. (not) applicable or (not) mitigated) to more probabilistic risk-based assessments that better characterize to which degree a certain threat may be mitigated by the presence of one or more security or privacy solutions. Such a probabilistic assessment also enables more precise and continuous re-assessments of threats and the effectiveness of the provided countermeasures by incorporating information an new vulnerabilities or changes in the effectiveness of the used security and privacy solution. The next chapter will go into detail on these analyses.

Chapter 4 Outline

4

Design-Level Analysis for Security and Privacy Threats

We must become more comfortable with probability and uncertainty.

— Nate Silver [Sil12]
The Signal and the Noise

This chapter builds on content from the following previously published articles:

Laurens Sion, Kim Wuyts, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Interaction-based privacy threat elicitation. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 79–86. IEEE, 2018

Laurens Sion, Katja Tuma, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. Towards automated security design flaw detection. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, SEAD '19, pages 49–56, 2019

Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Risk-based design security analysis. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*, SEAD '18, page 11–18, New York, NY, USA, 2018. Association for Computing Machinery

Laurens Sion, Dimitri Van Landuyt, Kim Wuyts, and Wouter Joosen. Privacy risk assessment for data subject-aware threat modeling. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 64–71. IEEE, 2019

As discussed in the previous chapters, the realization of the Security by Design (SbD) and Privacy by Design (PbD) principles requires analyzing the design for its security and privacy properties. This dissertation focuses on the application of security and privacy threat modeling approaches [HL06, Sho14, DWS+11, Wuy15] to perform such a systematic assessment. Chapter 3 presented and extended the Data Flow Diagram (DFD) model of a system to represent the design and its security and privacy solutions.

This chapter considers the next step in the approach: the analysis of these design models in order to identify and prioritize security and privacy threats. The analysis in this chapter is divided into two main parts: (i) the eliciting security and privacy threats using DFD models; and (ii) the prioritization of these previously elicited security and privacy threats by applying risk analysis.

Identifying security and privacy threats. There is no clear consensus on the most appropriate abstraction level for threat elicitation. Originally, threat modeling approaches performed an *element-based* threat elicitation by exhaustively iterating over the individual DFD elements [SS04, HL06, DWS+11, Wuy15]. More recent approaches, however, perform *interaction-based* threat elicitation [Sho14, Mic20, SWY+18], in which a systematic iteration over the interactions in the model (i.e. 'sender-flow-recipient'-combinations) is performed. There are other more formal approaches [TSB19] which rely on some additional extensions to the DFD model (discussed in Section 4.4).

In this chapter, we provide a detailed and in-depth analysis of the shortcomings in element-based threat elicitation and argue how interaction-based elicitation resolves these shortcomings by including more system context information (i.e. not just a single element, but a data flow and the element types it connects) into account when considering its applicability. Next, the interaction-based variant of LINDDUN is presented as to enable interaction-based privacy threat elicitation. Afterwards, we discuss how evolving beyond interactions allows even more advanced threat applicability assessments that take more system context into account beyond the narrow scope a single interaction between two DFD elements. While the additional available information and increased elicitation complexity may not help the user [TS18], automated elicitation does not suffer from these problems. Indeed, it can very systematically and consistently apply the criteria regardless of the size of the system.

Threat prioritization. Even with more detailed threat applicability conditions that take more context into account (e.g., the interaction and its connected element types), the systematic elicitation of security and privacy threats leads to the problem of *threat explosion* [SYVJ18c, WVHJ18] in which too many (less relevant) threats are raised and negatively impact the overall effectiveness of the threat modeling activities. Such large lists of security and privacy threats also complicate the assessment to determine their relevance (e.g., based on likelihood and impact). Existing threat modeling methodologies assume a very coarse-grained and manual classification of the threats (i.e. applicable/not applicable or high/medium/low). Such an assessment does not take into account: (i) which threats have a critical impact on the system or the assets it processes or stores, (ii) the types of attackers and how capable they are, and (iii) how well existing security and privacy countermeasures protect against those types of attackers. While the application of risk analysis approaches [LSS10, FJ14] in conjunction with threat modeling can ameliorate some of these disadvantages, their application would require a substantial manual effort for each elicited threat, which would not scale well and worsen the impact of the threat explosion problem.

To address these problems in a threat modeling context, we present a security and privacy risk assessment approach for estimating a threat's risk to enable subsequent prioritization of the elicited threats. This approach is based on the FAIR [FJ14] risk model and explicitly supports uncertainty in the inputs to the risk analysis. Integrating risk analysis into the threat modeling context enables a more nuanced assessment of a threat's relevance. The integration of the approaches in a single design analysis activity achieves the following benefits: (i) it provides guidance in triaging threats and focusing on the most important, high-risk threats first, (ii) it supports the consideration of the strength of existing security and privacy measures in protecting against threats, (iii) it replaces the binary or categorical (high/medium/low) classification of threats with a more nuanced view based on the calculated risk, and (iv) it allows for measuring risk reduction progress over multiple design iterations.

4.1 Automating Threat Elicitation

This section elaborates on extending the threat elicitation. First, an analysis of element-based threat elicitation discusses the problems with this approach because of the limited information on the system context (i.e. only a single element) that is considered during the elicitation. These problems are partially resolved with interaction-based security threat elicitation [Sho14, Mic20], but not for privacy threat elicitation. Hence, the second part will introduce the interaction-based elicitation for the LINDDUN privacy threat types in order to support considering a more extensive system context (i.e. the data flow and the sending and receiving elements) when eliciting privacy threats. Finally, the last part moves beyond the interaction-based elicitation with an approach to incorporate an even broader system context by looking at more elaborate configurations of DFD elements to identify design flaws.

4.1.1 Analysis of Element-Based Elicitation

This section discusses the advantages and disadvantages of the application of element-based threat elicitation. The element-based LINDDUN threat type mapping template (Table 4.1) is used throughout this section to illustrate the limitations of the element-based elicitation.

The main advantage of the element-based threat elicitation approach is its simplicity. A single, small mapping table template (Table 4.1) suffices for guiding the elicitation and determining a threat type's applicability. This mapping table template details for each DFD element of a specific type (listed in the rows), which corresponding LINDDUN categories (in the columns) have to be considered ($\times$) or can be omitted ($-$). Applying the element-based elicitation in practice therefore simply involves iterating over every DFD element and eliciting the applicable

Table 4.1: Element-based LINDDUN Mapping Table Template

Element Type	**L**	**I**	**N**	**D**	**D**	**U**	**N**
Process	×	×	×	×	×	–	×
Data Store	×	×	×	×	×	–	×
External Entity	×	×	–	–	–	×	–
Data Flow	×	×	×	×	×	–	×

LINDDUN threats following the template. This makes the element-based elicitation particularly well-suited for manual execution.

However, the main disadvantage of such an element-based approach is the fact that it is completely agnostic to the local DFD context of the element. The element-based approach does not take into account contextual information such as whether the element sends/receives or the type of the other element it communicates with (e.g., an external entity). The lack of this contextual information during threat elicitation may lead to: (i) the omission of relevant threats (false negatives), (ii) the elicitation of threats that are not applicable (false positives), and (iii) redundancies in the documented threats. Each of these problems are discussed in more detail below.

Omitting Threats (False Negatives)

Element-based threat elicitation can lead to threats being omitted when a single element is involved in multiple interactions with other elements. When there are multiple incoming or outgoing data flows connected to an element, that element may be threatened in different ways over each of those flows. If those different types of interactions are not taken into account by instead focusing only on the element itself, some threats may remain undiscovered as the context of the different types of interactions is unknown by the threat modeler.

For example, the insufficient minimization of results from a data store (e.g., a database) can cause *identifiability* threats because it could be possible to read or derive the identity of a data subject in the data store. However, queries to a database can also reveal information about the user performing those queries. Therefore, it is necessary to consider the different roles the database plays in all the other interactions it is involved in. Another example involves the application of data minimization techniques (to prevent *identifiability*) when storing data in a data store. These protections would not automatically hold for any other process that saves data to that data store. Instead, this depends on how and where the countermeasure is applied. If this is only considered on the element-level at the data store, *identifiability* threats resulting from other processes storing data would be incorrectly omitted because these other processes are not considered.

Eliciting Inapplicable Threats (False Positives)

As the —'s in Table 4.1 illustrate, not all threat types always apply. However, determining whether a threat type applies depends on more than the DFD element type. It can also depend on the direction of the data flow. For example, a *detectability* threat at the level of the process is only applicable when the process is sending data, not when receiving. This cannot be captured in the element-based threat template (Table 4.1), as there has to be an × to ensure *detectability* threats are elicited for the sending case, which can lead to wasted analysis effort in trying to elicit a threat in the receiving case when it was not even applicable. Therefore, more fine-grained applicability conditions support the elimination of these inapplicable cases based on the context information.

Consider the following example. A data store in a system may be susceptible to a *detectability* threat for user records when their existence is acknowledged with, for example, an access denied error. However, such a threat would only be applicable when there is a returning data flow from the data store to the requesting process. If the data store does not respond to any request (not even reporting on success or failure), then it cannot reveal the presence of any existing records and the *detectability* threat is not applicable. Eliciting threats without taking a broader context into account can lead to threats that will have to be removed later on.

Eliciting Redundant Threats

Observations on the application of STRIDE element-based threat elicitation in an industrial context have indicated that the element-based threat elicitation is more time-consuming and redundant [Dhi11, p. 43]. Without taking into account the interactions and their directions, element-based threat elicitation can lead to threat duplication (at the source, the data flow itself, and the destination). This duplication is caused by the lack of a clear distinction between the different roles of the DFD elements involved in single interaction.

4.1.2 Interaction-Based LINDDUN

This section first introduces the construction and resulting mapping table of the interaction-based privacy threat elicitation variant of LINDDUN. Hereafter, the differences in the application of element-based and interaction-based threat elicitation are explained in further detail. Finally, the section concludes with a discussion on how the context information can be further expanded with pattern-based elicitation.

Construction of the Interaction-Based Mapping

The creation of the interaction-based mapping template starts with listing all the DFD element interaction combinations, taking into account the three different elements that are part of an interaction (i.e. source, data flow, and destination), leading to a table with 105 combinations that have to be considered. To avoid interpretation ambiguities because the interpretation of the threat types differs slightly from the element-based approach, a completely empty mapping is used as the starting point. This table is subsequently evaluated for each privacy threat type to determine the applicability of the threat type given the involved DFD element types. This initial table was created at the basis of expert knowledge (with the involvement of the creators of LINDDUN) and guided by example threats for each of the combinations [SWY+18]. Table 4.2 shows the resulting interaction-based mapping table.

Table 4.2: Interaction-based LINDDUN Threat Type Mapping

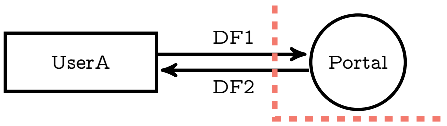
Source		**Destination**	**L**	**I**	**N**	**D**	**D**	**U**	**N**
Process		Process	×	×	×	×	×	–	×
Process		DataStore	×	×	×	×	×	–	×
Process	flow	ExternalEntity	×	×	×	×	×	–	×
DataStore		Process	×	×	×	×	×	–	×
ExternalEntity		Process	×	×	×	×	×	×	–
Process		***Process***	×	×	×	–	–	–	×
DataStore		***Process***	×	×	×	–	–	–	×
ExternalEntity	flow	***Process***	×	×	×	–	–	–	×
Process		***DataStore***	×	×	×	–	×	–	×
Process		***ExternalEntity***	×	×	×	–	–	×	×
Process		Process	×	×	×	×	×	–	–
Process		DataStore	×	×	×	×	×	–	–
Process	***flow***	ExternalEntity	×	×	×	×	×	–	–
DataStore		Process	×	×	×	×	×	–	–
ExternalEntity		Process	×	×	×	×	×	–	–

The elements in the first three columns highlight the element to which the privacy threat is associated (using a *colored and emphasized* notation). Note that invalid DFD element combinations (such as *DataStore-flow-DataStore* or *ExternalEntity-flow-ExternalEntity*) are not included in this table.

Table 4.3: Interaction-based LINDDUN examples

Source		Destination	Linkability	Identifiability	Non-Repudiation	Detectability	Information Disclosure	Unawareness	Non-Compliance
P		P	Cookies sent by the browser (S) can link a data-subject's activities.	A browser's (S) eID login to a website (D) leaks identity info	Sender (S) has no plausible deniability without pre- cautions (e.g., OTR chat).	Website (S) showing incorrect password error reveals account existence.	Process (S) reveals data to unauthorized recipient (D).	–	Processing (S) user data without lawful basis.
P		DS	Process (S) uses identifiers (e.g., certificate) or predictable timing.	Identifiable information in source's (S) eID certificate (i.e., full name, etc.).	Sender (S) has no plausible deniability without pre- cautions (e.g., OTR chat).	Website (S) showing incorrect password error reveals account existence.	Writing (S) data to DB (D) which shouldn't be there (e.g., passwords).	–	Processing (S) user data without lawful basis.
P	– flow –	EE	Sharing (S) email addresses can allow a third party (D) to link it with own data.	Sharing (S) identifiable info enables third party (D) data-subject identification.	Sender (S) has no plausible deniability without pre- cautions (e.g., OTR chat).	Communicating (S) PII to third parties (D), enables detection of your users.	third party (D) receives data from process (S) it's not authorized to receive.	–	Sharing (S) with a third party (D) without lawful basis.
DS		P	Insufficiently anonymized data in DB (S) can be linked (D).	Insufficiently anonymized data in DB (S) could be de-anonymized (D).	Sender (S) has no plausible deniability without pre- cautions (e.g., OTR chat).	Write errors from DB (S) can reveal that a record already exists to (D).	A database (S) without access control can disclose information to (D).	–	DB (S) Stores or sends data without lawful basis.
EE		P	Sending (S) pseudonymous identifiers can lead to linkability (D).	Sending (S) identifiable data to (D).	Sender (S) has no plausible deniability without pre- cautions (e.g., OTR chat).	Context data from sensors (e.g., accelerometer) at (S) can be used for detection (D).	Process (D) is not authorized to receive the data from (S).	Data-subject (S) is unaware of which or how much data is being shared with (D).	–
P		***P***	An SSO service provider (D) can link users' (S) movements across services.	Sign-up via SSO (D) can lead to identifiability if the SSO sends user data.	A process (D) adding digital signatures.	–	–	–	Processing (D) user data without lawful basis.
DS		***P***	A browser process (D) links all history (S) in single user session.	Retrieval (D) of identifiable information from DB (S).	A process (D) adding digital signatures.	–	–	–	Processing (D) user data without lawful basis.
EE	– flow –	***P***	An SSO service provider (D) can link users' (S) movements across services.	Sign-up via SSO (D) can lead to identifiability if the SSO sends user data.	A process (D) adding digital signatures.	–	–	–	Processing (D) user data without lawful basis.
P		***DS***	Insufficient data minimization or anonymization (D).	Queries to a DB (D) can reveal info about the user (S) performing them.	Logging (D) user (S) actions prevents plausible deniability.	–	Insecure storage (e.g., unencrypted) (D).	–	Insufficient minimization/ anonymization (D)
P		***EE***	Sending (S) identifiers to third party (D) allows it to combine it with its own info.	Sending (S) identifiable info to a third party (D) enables identification.	Sending (S) user interaction records to a third party (D) prevents plausible deniability.	–	–	Data-subject is unaware of sharing of data with third parties (D).	Sharing data with a third party which is non-compliant.
Any	***– flow –***	Any	Linking flows using the meta-data.	Identifiable data in an unprotected data flow.	Capture data flow to collect evidence of communication.	Detection of data flows (e.g., wifi traffic).	Data-flow contents and meta-data are sniffed on the wire.	–	–

The elements in the first three columns highlight the element to which the privacy threat is associated (using a ***colored and emphasized*** notation), using the definitions from Section 4.1.2. The (S) and (D) annotations in the examples refer to the source and destination elements respectively.



Element-based	**Interaction-based**
Iteration 1: Data Flow DF1	
→ DF1 threat(s)	→ DF1 threat(s)
—	→ UserA threat(s) (*source*)
—	→ Portal threat(s) (*destination*)
Iteration 2: Data Flow DF2	
→ DF2 threat(s)	→ DF2 threat(s)
—	→ UserA threat(s) (*destination*)
—	→ Portal threat(s) (*source*)
Iteration 3: External Entity UserA	
→ UserA threat(s)	—
Iteration 4: Process Portal	
→ Portal threat(s)	—

Figure 4.1: Element- and Interaction-Based Elicitation Differences
This diagram and table illustrate the differences in elicitation iterations between the element-based and interaction-based elicitation approaches. It shows how the interaction-based approach has fewer but more complex iterations that involve all the roles in an interaction, while the element-based has more but simpler iterations.

After construction, the interaction-based table was compared with the original element-based LINDDUN table to perform an in-depth analysis of the differences. This analysis highlighted multiple differences between the two tables (5 additions and 12 eliminations of ×'s, or 16% of the entries). These discrepancies mainly result from the lack of semantics in the element-based approach to associate a threat type to a specific participating element in an interaction. Therefore, the following interpretations were used to unambiguously specify whether the threat should be located at the *source*, the *data flow*, or the *destination*:

Source The threat arises at the level of the element that shares or communicates data where the sharing of the data can cause a privacy threat. An example of this is a browser that retransmits cookies or other linkable identifiers to each recipient.

Data Flow The threat arises at the level of the data flow, i.e., when the data (both contextual, i.e., meta-data, and transactional, i.e., the content itself) are in transit. For example, contextual data about the source and destination can be used to link multiple data flows, or to identify the parties involved in the communication.

Destination The threat arises at the level of the element that receives the data where the data can be processed or stored in a way that causes a privacy threat. An example of this is insecure storage or insufficient minimization of the data upon storing.

Applying Interaction-Based Elicitation

Figure 4.1 illustrates the differences between the interaction-based and element-based elicitation approaches with a trace of the elicitation iterations in the two approaches on a small DFD fragment. Instead of iterating over each DFD element, the interaction-based elicitation approach iterates over the data flows in the model (in which threats are elicited for the source, the flow itself, and the destination of that data flow). Because of this, different threat types have to be specified for these three roles of an interaction. Each of these correspond with the outer rows in Table 4.2, from top to bottom respectively source, destination, and flow.

Table 4.3 provides an overview of privacy threat examples for each of the ×'s in the interaction-based threat table (Table 4.2). This table can assist the human threat modeler in understanding and interpreting the different privacy threat types and serves as a key knowledge asset during threat modeling brainstorming sessions. Obviously, such a table of examples is always incomplete, as there are multiple privacy threat types possible for each individual cell. Nonetheless, such a table is invaluable for concretization and brainstorming purposes. Existing privacy threat knowledge sources such as privacy threat trees [WJ15] can be used in conjunction with this table to complement this knowledge.

4.1.3 Beyond Interaction-Based Elicitation

Traditional threat modeling approaches support a rigorous and systematic assessment of the system under consideration. The threats that can be discovered using their element-based or interaction-based elicitation approaches remain limited to those that can be identified with the system context as it is available during threat elicitation. Some security or privacy design flaws, however, are more elaborate and involve multiple elements and interactions. Therefore, these flaws are harder to find when focusing on a single element or interaction.

As the complexity of the design flaws to detect increases, a systematic assessment of every possible combination of larger sets of DFD elements and their interconnections is no longer tractable. Instead, approaches can rely on catalogs of security and privacy design flaws in order to uncover these flaws in concrete DFD models. Numerous different catalogs of design flaws have been published [CVE19a, CWE20, CAP18, ADD+14, SPM+17, OWA17a, MH17, THMS19] that provide extensive sets of flaws for which systems can be analyzed. However, the application of this knowledge to find these flaws in DFDs remains

Table 4.4: Flaw catalog of Malamas and Hosseini [MH17, THMS19]

	Flaw Name
1	Missing authentication
2	Authentication bypass
3	Relying on single factor authentication
4	Insufficient session management
5	Downgrade authentication
6	Insufficient crypto key management
7	Missing authorization
8	Missing access control
9	No re-authentication
10	Unmonitored execution
11	No context when authorizing
12	Not revoking authorization
13	Insecure data storage
14	Insufficient credentials management
15	Insecure data exposure
16	Use of custom/weak encryption
17	Not validating input/data
18	Insufficient auditing
19	Uncontrolled resource consumption

a difficult challenge, as the catalogs only contain textual descriptions of these flaws which need to be correctly recognized in a concrete design.

Security & Privacy Design Flaws

To determine the modeling requirements for detecting design flaws, we start from an existing catalog of security design flaws [MH17, THMS19]. This catalog contains 19 common architectural security design flaws concerning authentication, access control, authorization, resource availability, integrity, confidentiality. To illustrate the application of this knowledge for detecting flaws in concrete systems, the *"Insecure Data Exposure"* flaw is used as an example. Snippet 4.1 shows the

Security Design Flaw 15: Insecure Data Exposure

Description Data is not transferred in a secure way. For example a web application uses the HTTP instead of HTTPS. This leaves the channel vulnerable to eavesdropping, Man In The Middle (MITM) attacks etc.

Detection

- Locate the valuable information in the model.
- Track them through the architecture to determine where and how they are transferred.
- At each step examine the following:
  - Is the reuse of packets prevented (Replay attacks)?
  - Is there any form of timestamping, message sequencing or checksum in the exchanged packages?
  - Is the traffic over an encrypted channel (SSL/TLS)?

Snippet 4.1: Design Flaw 15: Insecure Data Exposure
Description and inspection rules of security design flaw 15: insecure data exposure, from the security design flaw catalog of Malamas and Hosseini [MH17, THMS19].

catalog entry which consists of: (i) a *name*, (ii) a brief *description*, and (iii) a series of *detection rules*, i.e. closed questions to determine the flaw's applicability.

Manual detection. Considering the *"Insecure Data Exposure"* security design flaw, listed in Snippet 4.1, as an example, this flaw is present when data is not transferred securely. This flaw's description illustrates that the manual assessment for the presence of this flaw is non-trivial. It involves: (i) systematically exploring the model to find and track the flow of valuable information through the system, and (ii) evaluating several criteria at every point where this information is present. To support such analysis activities, the automated detection of these flaws can ensure a systematic and comprehensive assessment of the system.

Detection concepts. For the automated detection, the concepts used in the flaws' detection rules need to be identified in the architectural design models. The following five concept types were derived from

Insecure Data Exposure Design Flaw

Detection

- *locate* valuable information *in the model* → [action: traverse model for [data: sensitive]]
- *track them* → [action: track all elements [data: sensitive]]
- At each step examine the following:
  - *. . . prevented* → [countermeasure: tampering]
  - *any form of . . .* → [countermeasure: tampering]
  - *encrypted channel* → [countermeasure: information disclosure]

Snippet 4.2: Insecure Data Exposure Design Flaw
Mapping of the textual description of the security design flaw to model elements. The snippet shows how the textual elements from the Insecure Data Exposure design flaw (shown in Snippet 4.1) can be mapped to model elements from a DFD model.

Insecure Data Exposure Pseudocode

```
for (DFDElement e : DFDModel.elements) {
 for (DataType d : e.dataTypes) {
  if (d.isSensitive()) {
   if (!e.solution.protects(d,tampering)
       || !e.solution.protects(d,infoDiscl)) {
    triggerDesignFlaw15();
   }}}}
```

Snippet 4.3: Insecure Data Exposure Pattern Pseudocode
Conversion of the model element criteria to pattern pseudo-code. The pseudocode shows how the iteration over the model elements and the data types they process to verify whether an appropriate solution is present. The pseudocode here assumes a single solution for simplicity. In practice however, all solutions affecting the considered element should be checked for protection against the specified threat type.

a study of the complete catalog of design flaws [MH17, THMS19]: *information*, *operations*, *countermeasures*, *attacks*, and *actions*. Some of these concepts refer to information that needs to be present in the model (e.g., information), others refer to the type analyses that have to be performed on the model (e.g., actions).

Information Some flaws require the presence of certain types of information: sensitive information, encrypted data, credentials, or cryptographic keys. These information types should be retrievable from the model. The example in Snippet 4.2 relies on model elements which have data attached to them with the *sensitivity* property set.

Operations Some flaws rely on additional information on the types of operations that processes perform. Flaw 9 (“No re-authentication”), for example, relies on this information to detect whether a user is re-authenticated before a process performs security-critical operations.

Countermeasures Some flaws depend on the presence or absence of countermeasures. In order to detect these flaws, the system models should support the representation of security and privacy solutions. The encrypted channel criterion in Snippet 4.1 corresponds with the presence of a security solution that protects against information disclosure threats on a specific communication channel.

Attacks Instead of relying on countermeasures, some flaw descriptions refer to whether or not certain types of attacks are possible. These types of criteria can be resolved analogously to the countermeasure case by relying on the presence of countermeasures to defend against the attacks specified in the flaw descriptions.

Actions Finally, the flaws’ detection rules specify different types of actions (e.g., locating, tracking, finding). Contrary to the previous concepts, actions do not represent new types of information that have to be captured in the model, but instead refer to types of analyses that have to be performed on the model. For the example in Snippet 4.1, the action "track (valuable information)" requires retrieving all the elements that handle (i.e. process, transfer, or store) sensitive data.

Modeling Support

This section considers how the existing modeling support for threat elicitation can be leveraged for design flaw analysis. To optimize the efficiency of security and privacy analyses, additional complementary analysis activities should reuse existing modeling artifacts as much as possible. The conceptual requirements for detecting the design flaws outline above, show that this is possible to a large extent. For each of the concepts introduced above, we outline how the information can be extracted from system models expressed using the meta-model presented earlier in Chapter 3.

Information and Operations Representing information is directly supported by the meta-model, which allows the representation of different data types and which DFD elements transfer or process information of those types. Security-related information can also be derived by relying on naming conventions for identifying *key*s or *credential*s. There is no explicit concept identifying security-critical operations. While this could be modeled using an attribute of a process, currently the easiest way to identify these types of processes is by deriving a process as critical when *sensitive* data is being processed. This approach for detecting security-critical operations requires no additional extensions for detecting them in existing DFDs.

Countermeasures and Attacks These concepts can be grouped together as the existing security and privacy solution modeling support can be used for extracting this information from the models. The solutions provide the relevant information on the presence of countermeasures. Furthermore, these solutions refer to the specific threat types they prevent and, hence, also provide the necessary information on the attacks that are rendered impossible.

Actions Finally, actions should be supported in the analysis. This concept is not represented in the models but should be supported in the analysis of the model. The interaction-based analysis is insufficient for assessments such as tracking sensitive information across multiple model elements. The next section will go into detail on how, by expanding the system context from interactions to generic patterns, the analysis of the DFD models can support the detection of complex configurations of DFD elements, countermeasures, and data types.

Pattern-Based Elicitation

First, we explore how the existing element- or interaction-based threat elicitation can support the detection of design flaws. Considering the pseudo-code in Snippet 4.3, the example shows the localized flaws can easily be detected by iterating over the elements in the DFD model and verifying the necessary conditions for the flaw.

However, when considering a more complex flaw such as stored XSS, described in CWE-79 [CWE19] from the CWE catalog [CWE20] and ranked second in the top 25 most dangerous software errors [MIT19], it becomes clear that the interaction-based elicitation is insufficient as this flaw involves data being stored in a data store (with potentially multiple processes in between) and later retrieved again when constructing a web page for a user. Figure 4.2 shows a graphical representation of this flaw in a DFD, with the relevant data flows marked in color.

Eliciting this flaw by focusing on only one of its interactions could generate multiple false positives, as it is not always possible to determine which interaction of the flaw is matched if there is a chain of multiple linked data flows is used to detect the flaw. To precisely detect such flaws, a more extensive system context needs to be taken into account. Figure 4.2 can also be expressed with the following regex-like pattern:[1]

$$▭ \rightarrow (\bigcirc \rightarrow)^{+} = \rightarrow (\bigcirc \rightarrow)^{+} ▭$$

1 Legend of the pattern elements: ▭: External Entity, ○: Process, ═: Data Store, →: Data Flow, and $(x)^{+}$: at least one x.

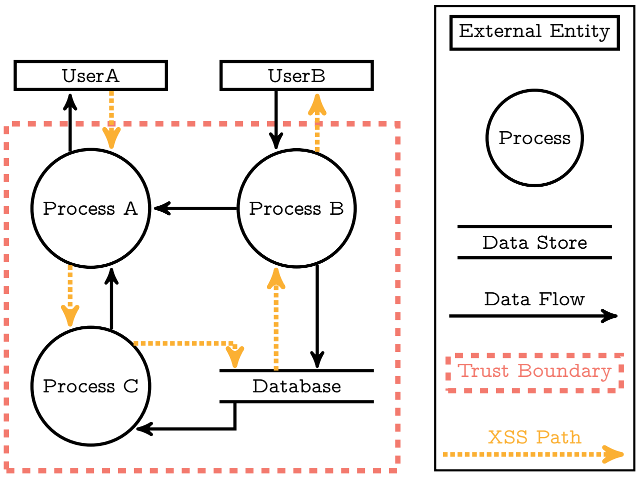


Figure 4.2: Illustration of a DFD with XSS
This Data Flow Diagram shows the path of a stored XSS attack through the system. This illustrates how the interaction-based approach is sub-optimal to detect this flaw as there are several data flows involved which can lead to multiple false positives.

```
pattern xssflaw() {
  ExternalEntity(e1);
  ExternalEntity(e2);
  DataStore(ds);
  Process(p1);

  find communication(e1,df,p1);
  neg find mitigation(p1,df,"Tampering");
  find FlowTo+(p1,ds);
  find FlowTo+(ds,e2);
}
```

Snippet 4.4: Xss Pattern Description
This snippet shows the VIATRA pattern description for the XSS flaw. It constrains the types of the elements, their connections via one or more data flows, and whether a mitigation against tampering is present on the incoming flow from e1*.*

In order to support more complex configurations and to verify the presence of security and privacy solutions, a pattern query language, such as VIATRA,[2] can be used to efficiently query models. Snippet 4.4 shows the pattern for the XSS flaw, which also checks for the presence of a countermeasure against tampering on the incoming data flow. This pattern can be further extended with more checks for additional countermeasures to enable a comprehensive assessment of the flaw's applicability. We have provided a complete description of the approach to translate design flaws to model queries [STY+19].

4.2 Prioritizing Threats through Risk Indicators

Complementary to the security and privacy threat elicitation discussed above, this section covers the prioritization of these threats through the use of risk indicators. The prioritization of security and privacy threats consists of two main parts: (i) the extension of the FAIR [FJ14] risk model to enable its application in a threat modeling context to enable the prioritization of the elicited threats, and (ii) the impact of this additional analysis step in the threat modeling process itself, as multiple parameters of the risk model need to be considered to calculate the risk of each elicited security and privacy threat. The prioritization of threats is applied in the context of interaction-based threat elicitation as described above in Section 4.1.2, as it provides a consistent system context (two elements and the connecting data flow) to determine the impact as part of the risk assessment. The prioritization can also be applied on patterns described in Section 4.1.3, but, given the wide range of possible patterns and elements involved in them, this requires additional information. For example, which of the elements involved in the pattern contain or process the threatened assets for determining the impact in the risk analysis.

2 https://www.eclipse.org/viatra/

4.2.1 Risk Assessment Model

The section presents a security and privacy risk assessment model to support the prioritization of security and privacy threats. This model is inspired by the FAIR [FJ14] risk assessment model, but extends it to support its application in a threat modeling context and expands it with privacy and data subject risk factors in order to enable the risk assessment of the LINDDUN privacy threats. Figure 4.3 shows the complete decomposition of risk into its underlying components.

The FAIR approach (explained in Section 2.5) was chosen as the base risk model because: (i) it provides more granular assessment than high-level categorization (e.g., high/medium/low); (ii) the numerical inputs supports automation for prioritizing elicited threats; (iii) the information for the risk components are providable by the threat modeling inputs (e.g., DFD model, threat catalog); and (iv) it supports expressing uncertainty in its inputs to cope with the difficulty of assigning precise values for risk inputs.

This risk model unifies: (i) technical risk originating from the system context and its security and privacy countermeasures, (ii) data subject risk incorporating data subject types and their data types, and (iii) organizational risk both by scaling the impact according to the number of data subjects and data records involved and by incorporating the impact of security threats on assets being processed or stored.

Below, each of the risk sub-components are explained, following the structure of the risk decomposition as presented in Figure 4.3.

Risk

The risk is comprised of two parts: (i) the *Loss Event Frequency*, which represents the estimated frequency of successful attacks or privacy incidents; and (ii) the *Loss Magnitude*, which represents the damage to assets for security threats and the impact on the data subject(s) for privacy threats. These two factors are multiplied for the overall risk.

$$\text{Risk} = \text{LM} \times \text{LEF}$$

Loss Event Frequency (LEF)

The first risk component is the *Loss Event Frequency*. It represents the total frequency of *successful* attacks or privacy incidents. It is obtained by combining the frequency of the attempts (TEF) with the probability of a successful attack or incident (V). For privacy threats, the attempt frequency is combined with the retention period (RP) which represents how long the data is processed or stored on the same time scale as the frequencies (LEF and TEF). The LEF is calculated as follows:

$$\mathrm{LEF} = \mathrm{TEF} \times \mathrm{V} \times \mathrm{RP}$$

Threat Event Frequency (TEF) The *Threat Event Frequency* represents the frequency of threat *attempts*. These attempts are not necessarily successful. The Threat Event Frequency (TEF) is further decomposed into: (i) the *probability of action (PoA)*, indicating the likelihood of an attempt, and (ii) the *contact frequency (CF)*, representing how frequently the threat actor[3] comes into contact with the system. The TEF is obtained by multiplying these factors:

$$\mathrm{TEF} = \mathrm{PoA} \times \mathrm{CF}$$

Probability of Action (PoA) The *Probability of Action* specifies the likelihood that a threat actor will perform an attempt. This probability varies between different types of adversaries (and is influenced by their incentives, capabilities, and opportunities). For example, an external attacker in a different legal jurisdiction may be very likely to attempt an attack when coming into contact with the system as any repercussions are unlikely. An insider such as an employee, however, may be less likely to act if there are strict monitoring controls imposed and repercussions on discovery; while a system administrator may be able to bypass all these controls to evade detection and, hence, be more likely to attempt something compared to the regular employee. Multiple attacker profiles are needed to capture these differences in the risk analysis.

3 This threat actor can be an external attacker (in the context of security threat modeling), but it can also be the organization for the privacy threats. For example, an organization can collect and process data of which the user is unaware.

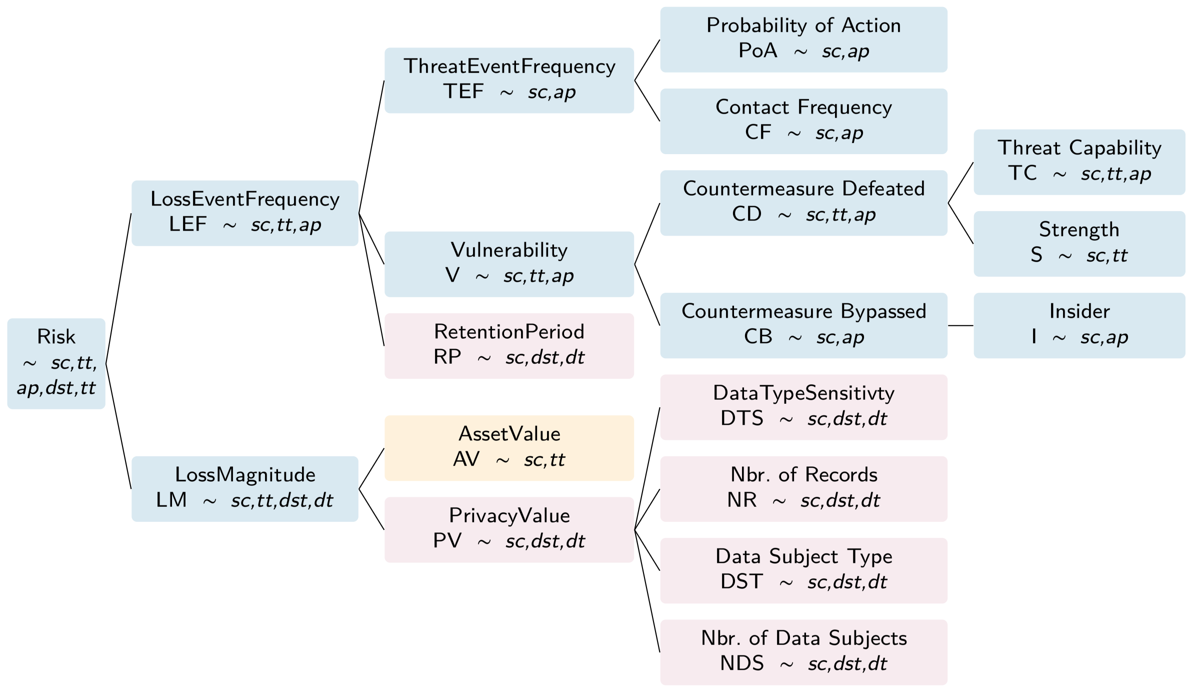


Figure 4.3: Overview of the Risk Assessment Model
The above figure illustrates how the risk is decomposed into sub-components. Components that are specific to security or privacy threats are marked in color. These components are not taken into account when assessing the opposite type's risk.

Contact Frequency (CF) The *Contact Frequency* specifies how frequently a threat actor interacts with the system. This factor again varies tween different attacker profiles in order to support distinguishing between, for example, external adversaries, insiders, or regular users.

Vulnerability (V) The *Vulnerability* factor specifies the probability of a successful manifestation of a security or privacy threat. This takes into account the type of the attacker, the system, and the countermeasures protecting against the considered threat. The vulnerability is calculated as the maximum of: (i) the countermeasure being defeated (*CD*), and (ii) the countermeasure being bypassed (*CB*).

$$\mathrm{V} = \max(\mathrm{CD}, \mathrm{CB})$$

Countermeasure Defeated (CD) The *Countermeasure Defeated* indicates whether a certain security or privacy countermeasure can be defeated by an adversary. This factor is calculated by comparing:

Threat Capability (TC) expressing the 'strength' of an adversary in being able to defeat certain security or privacy countermeasures; with the

Strength (S) indicating the strength of the countermeasure in being able to resist the adversary.

The Countermeasure Defeated (CD) value is calculated as follows:

$$\mathrm{CD} = f(\mathrm{TC}, \mathrm{S}) \text{ with } f(x, y) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases}$$

The result will be binary for a single calculation. However, when doing multiple calculations, the results can be averaged to obtain a fraction to represent the probability of the countermeasure being defeated.

Countermeasure Bypassed (CB) The *countermeasure Bypassed* value indicates whether a certain adversary is able to bypass a security or privacy countermeasure, regardless of the capability of the adversary or the strength of the countermeasure. This factor supports the incorporation of insiders in the risk assessment without having to grant these insiders extraordinary capabilities to model their ability to bypass, for example, strong cryptography countermeasures aimed at external users of a system. While the most straightforward representation of this value is binary, a fraction could also be used to specify the likelihood of the adversary in being able to bypass a countermeasure.

Retention Period (RP) The *Retention Period* represents how long personal data is stored or processed. This factor indicates the time framing during which privacy threats can occur, as personal data must be present for an adversary to be able to exploit it. The Retention Period (RP) supports making the distinction between long-running data processing operations that pose a higher risk to data subjects compared to short-lived transactions which do not retain the data longer than required by the operation. Such short operations can limit the time window during which a successful threat can occur.

Loss Magnitude (LM)

The *Loss Magnitude* represents the impact of a security or privacy threat. It is comprised of two parts: (i) the *asset value* for the organization to indicate the damage of a security threat, and (ii) the *privacy value* for the data subjects to indicate the damage of a privacy threat. Since for most threats the security and privacy categories are mutually exclusive, only one of them needs to be taken into account. For threats that are part of both categorizations (e.g., *information disclosure*), both values need to be added together.

$$\mathrm{LM} = \mathrm{AV} + \mathrm{PV}$$

Asset Value (AV) The *asset value* is used to specify the loss of or damage to the organization in a case of successful attack. While such numbers are most naturally expressed as a currency in a risk context, it is also possible to assign numbers, for example, on an ordinal scale in order to enable a relative ranking of the resulting risk values.

Privacy Value (PV) The *privacy value* represents the impact on the data subject type and is composed of the following parts: (i) *Data Type Sensitivity*, (ii) *Nbr. of Records*, (iii) *Data Subject Type*, and (iv) *Nbr. of Data Subjects*. By combining these parts, both the sensitivity of the data and the type of data subject (e.g., minors) are taken into account. The privacy value can be obtained by multiplying them all together:

$$\mathrm{PV} = \mathrm{DTS} \times \mathrm{NR} \times \mathrm{DST} \times \mathrm{NDS}$$

Data Type Sensitivity (DTS) The *Data Type Sensitivity* component incorporates the sensitivity of the data types being processed into the impact of privacy threat types. The sensitivity values can be assigned on a numerical scale to make the distinction between different types of sensitive data ranging from, for example, to medical information, contact information (e.g., home address), or non-personal information.

Number of Records (NR) This factor scales the impact of a privacy threat along the number of records that are processed or stored. This factor can be used for modeling two types of cases. (i) Scaling the privacy impact when multiple records of the same data type are being processed or collected. (ii) Scaling the impact down when data of a certain type is only present for a fraction of the data subjects.

Data Subject Type (DST) The *Data Subject Type* specifies the risk inherent to the type of data subject whose data is being processed. It is used to consider vulnerable data subject types (e.g., children).

Number of Data Subjects (NDS) The *Number of Data Subjects* is a scaling factor, analogous to the Number of Records (NR), to scale the impact of a privacy threat to the number of involved data subjects.

4.2.2 Impact on Threat Modeling

The impact of the risk model on the threat modeling process starts with an overview of the parameters that influence the presented risk factor values. Afterwards, this section discusses the inclusion of the risk information in the DFD model. Capturing this information in the model supports the automated threat prioritization using risk indicators.

Parameters

The presented risk components in Section 4.2.1 do not directly map to single DFD elements. Instead, they are influenced by a number of different elements from the threat modeling inputs. Below, each of the five parameters that influence the risk components are listed. Figure 4.3 shows, for each risk component, the relevant parameters.

System Context (SC) The first parameter is the system context which expresses the system design information that is taken into account in the risk assessment. It includes information on the element types affected by the threat, the security and privacy solutions that are present, and the values of any assets involved.

Threat Type (TT) The second parameter is the considered threat type. The presence of security or privacy threats depends on their type and the system context. An explicit threat type parameter also enables aggregations over threat type categories (i.e. STRIDE and LINDDUN). This aggregation can provide insight into which threat types are the most problematic in the system under consideration.

Attacker Profile (AP) Third, to perform a risk assessment that incorporates the effectiveness of the security and privacy countermeasures against different attacker types, the analysis needs to consider additional information on the attacker capabilities in manifesting different threat types. Furthermore, the parameterization of attacker profiles enables several analyses such as assessing the impact against a single profile or decomposing risk into the different profiles for comparison. This enables evaluating which type of attacker would pose the highest risk.

Data Subject Type (DST) Fourth is the data subject type. The risk of privacy threats needs to take into account the type of the data subject being threatened. As parameters in the risk assessment, the risk value can be decomposed in order to analyze the system's privacy risk from the perspective of different data subjects.

Data Type (DT) Finally, the privacy risk is also influenced by the data types being processed, which also depends on the data subject those data belong to. For invalid combinations (i.e. data types that are unrelated to data subject type being considered), the risk is zero. This is explicitly different from the Data Type Sensitivity (DTS), which is just one risk component that is influenced by the data type. This is again a parameter that can be used in the risk analysis in order to compare and evaluate how high the risk is for the different data types that are being processed.

In order to calculate the total risk of the system, the risk values for each combination of the above five parameters are combined. Other types of aggregations, as mentioned in the individual parameter descriptions, are also possible in order to enable different types of risk analyses.

DFD Resources and Model Enrichments

In order to obtain the required risk information inputs for performing the risk assessment, this information needs to be represented in the modeling inputs for the threat elicitation and analysis. This ensures the risk information is stored close to the elements they pertain to and it ensures that the information is readily available during the threat elicitation and the analysis of the elicited threats. This section will go into detail on which existing threat modeling artifacts (e.g., DFD model elements, security or privacy solutions) need to be enriched with additional information to support the risk analysis. These model enrichments are not discussed in Chapter 3, which focuses on modeling the system as-is without the complementary risk analysis information.

Since the required information for the risk assessment is often impossible to determine with absolute certainty (e.g., the effectiveness of a countermeasure which degrades over time). Our approach has therefore been designed to explicitly support uncertainty in the risk inputs. More specifically, all the required numerical properties are expressed as estimates. Chapter 5 will go into detail how these estimates are used in the risk calculation while taking into account the uncertainties expressed in the input values.

Attacker Models Since the risk analysis includes an assessment to which degree the countermeasures present in the model are sufficient to resist a certain type of attacker, the risk analysis needs to be performed in the context of a certain attacker profile. The attacker profile information is included in the threat elicitation activity in the form of a set of attacker models. Such an attacker model provides three properties for the risk analysis: (i) the capability of the attacker (Threat Capability (TC)), (ii) the probability of action (Probability of Action (PoA)), and (iii) the frequency of contact (Contact Frequency (CF)). A final risk component influenced by the attacker profile is whether or not the attacker is an insider. This information is represented as a mapping from the attacker profile to a set of DFD elements, and indicates for these elements whether the attacker is able to bypass the countermeasures. To reduce the required effort in constructing multiple different attacker profiles, a set of common profiles can be provided and reused across multiple models and analyses. Only the insider link will have to be constructed for each individual model, although heuristics could be applied to facilitate this by, for example, adding all the elements inside a trust boundary to the list of the attacker profile.

Security and Privacy Solutions In order to obtain information about the effectiveness of the security and privacy solutions in the model, the strength of the solution's individual countermeasures has to be taken into account during risk assessment. This way, the differences in the effectiveness of the countermeasures can be captured. For example, the application of TLS with large public keys but a weak encryption cipher may provide strong authentication guarantees but weak confidentiality guarantees. To extract this information from the model, the security countermeasures are enriched with a *strength* estimate to use for the risk assessment.

DFD Model Asset Values In order to assess the damage of security threats, asset value information is required for the risk assessment. This information can be obtained from the DFD model by enriched the DFD model elements with *value* estimates that quantify the potential damage. Because the damage to an asset may vary depending on the type of the threat, it possible to assign different value estimates for different threats types. This granular value specification supports the modeling of, for example, scenarios in which a denial of service threat inflicts a lower damage than an information disclosure threat on a certain data store.

Data Types and Data Subject The final source of information that is required for assessing the impact of privacy threats, is the information on the data subjects and the data types of their data. This information is not present in a traditional DFD used in threat modeling. To obtain this information, the complementary data protection model [SDVL+19] can be used. This model captures all the data processing activities, data types, and data subject types; all of which are linked to the corresponding DFD elements. By navigating the links between these two models, the necessary data type and data subject type information can be gathered for privacy risk analysis.

4.3 Evaluation and Discussion

The first part of this section provides the evaluation of the interaction-based elicitation in Section 4.3.1 and the evaluation of the risk prioritization applied in the context of a real-world application for whistleblower submissions to journalists (Section 4.3.2). The second part discusses the interaction-based threat elicitation (Section 4.3.3) and the risk model (Section 4.3.4).

4.3.1 Evaluation of the Interaction-Based Elicitation

We qualitatively assess the interaction-based LINDDUN approach through a direct comparison with the element-based LINDDUN variant. The assessment considers the following aspects: (i) the *expressivity* of the interaction-based approach; (ii) the a-priori *elimination* of inapplicable privacy threats; (iii) the prevention of *undiscovered* threats; and (iv) the *effort-precision* trade-off.

Expressivity

First, we assess whether the interaction-based table for considering a threat's applicability is at least as expressive as the element-based table. More specifically, that there are no threats resulting from the element-based elicitation that the interaction-based elicitation cannot uncover. The construction of the backwards-compatible interaction-based table from the element-based one for the comparison explicitly confirms the fact that every element-based threat type condition can be captured in the interaction-based table. Not only is the interaction-based representation at least equally expressive as the element-based one, it can also be used to express semantically different threat types that are associated to the source, data flow, or destination elements involved in an interaction. Table 4.5 illustrates these differences. Any cell that differs for an element depending on its role as sender or recipient in an interaction cannot be represented in the element-based table as this table does not support making a distinction based on the element's role in the interaction. We argue that this leads to the identification of more precise and fine-grained threats as the threat elicitation can explicitly consider this contextual information.

Table 4.5: Differences between Interaction- and Element-Based

Source		**Destination**	**L**	**I**	**N**	**D**	**D**	**U**	**N**
Process		Process	×	×	×	×	×	–	×
Process		DataStore	×	×	×	×	×	–	×
Process	flow	ExternalEntity	×	×	×	×	×	–	×
DataStore		Process	×	×	×	×	×	–	×
ExternalEntity		Process	×	×	▨×	▨×	▨×	×	–
Process		***Process***	×	×	×	▨–	▨–	–	×
DataStore		***Process***	×	×	×	▨–	▨–	–	×
ExternalEntity	flow	***Process***	×	×	×	▨–	▨–	–	×
Process		***DataStore***	×	×	×	▨–	×	–	×
Process		***ExternalEntity***	×	×	▨×	–	–	×	▨×
Process		Process	×	×	×	×	×	–	▨–
Process		DataStore	×	×	×	×	×	–	▨–
Process	***flow***	ExternalEntity	×	×	×	×	×	–	▨–
DataStore		Process	×	×	×	×	×	–	▨–
ExternalEntity		Process	×	×	×	×	×	–	▨–

The elements in the first three columns highlight the element to which the privacy threat is associated (using a ***colored and emphasized*** notation). Note that invalid DFD element combinations (such as *DataStore-flow-DataStore* or *ExternalEntity-flow-ExternalEntity*) are not included in this table. Colored cells are used to mark the differences (e.g., ▨–) from the element-based table.

Elimination of inapplicable threats

The detectability threat is an example of a threat which only applies for element as the sender of interaction, but not as the recipient. The interaction-based threat type table encodes this applicability information, which allows for the up-front elimination of this threat in the contexts where it is not applicable. Since the element-based table does not support making this distinction, a threat modeler could spent significant effort on trying to elicit this threat when it was not applicable in the first place. We argue that the up-front elimination of inapplicable threats in specific interactions improves the overall efficiency of the threat modeling activities as it reduces the number of threats that have to be analyzed and prioritized after the elicitation.

Undiscovered threats

If the threat modeler focuses on a single type of role (e.g., sender) during the threat elicitations, certain threats corresponding to other potential roles of the considered elements may remain undiscovered. The interaction-based elicitation avoids this problem by explicitly forcing the threat modeler to systematically consider every applicable role of the element under consideration to ensure the inclusion of the other applicable threat types. Hence, the more fine-grained level of granularity leads to better guidance during the elicitation.

Effort-Precision Trade-off

Instead of specifying the applicability of threat types at the level of DFD element types, the interaction-based approach specifies the applicability at the level of a 'source-data flow-destination' configuration in which the threat is located on one of the elements. Figure 4.1 illustrates the difference between the element-based and interaction-based elicitation approaches. It shows how interact-based approach reduced the total number of iterations from 10, the number of DFD elements, to 6, the number of data flows. However, each of the remaining 6 iterations requires more effort because the three different roles involved in that interaction have to be considered, depending on the applicability as specified in the interaction table (Table 4.2). While the interaction-based approach increases the effort required in a single iteration, we argue that it provides an efficiency gain because it encodes and reuses expertise (i.e. the fine-grained determination of a threat's applicability in the context of a specific interaction), instead of having to rely on a human expert to know and apply this knowledge.

Applying any threat elicitation approach with an increased amount of contextual information in the consideration of a threat's applicability involves an inherent trade-off between the time and effort spent during the analysis of the threats afterwards versus the precision and efficiency of the threat elicitation. Dhillon [Dhi11] described positive experience of this trade-off in the application of interaction-based STRIDE for eliciting security threats in an industrial context.

4.3.2 Evaluation of the Risk Assessment Model

We evaluate the threat risk assessment model by applying it on the SecureDrop whistleblower submission system [Fre18a], a real-world application which allows whistleblowers to anonymously contact journalists and submit documents to them and which has strict security requirements in order to protect the identity of the whistleblowers.

The evaluation assesses whether the risk-based prioritization of the elicited threats is effective in identifying the most important threats. For the evaluation, a threat's importance is determined by the presence of security countermeasures in the SecureDrop documentation or implementation. Hence, the evaluation relies on the assumption that solutions have been implemented for the most critical threats.

The SecureDrop application was chosen for evaluation purposes because: (i) it is an open-source system enabling inspection of the source code for security and privacy countermeasures, (ii) it has a publicly available DFD model, (iii) it has detailed documentation on the security assumptions and the attacker capabilities that are considered [Fre18b], and (iv) it has strong security and privacy requirements.

The following SecureDrop sources are used in the evaluation to determine the presence of solutions: (i) the threat modeling documentation with their DFD [Fre18b]; (ii) the user, administration, and developer documentation; (iii) the source code; and (iv) the documentation of other components explicitly referred to by the other documentation or source code (for example, the recommended hardware firewall or software libraries used for authentication).

Figure 4.4 shows the SecureDrop DFD used for the evaluation. This DFD contains 81 elements (8 trust boundaries, 6 external entities, 17 processes, 7 data stores, and 43 data flows).[4] As part of the evaluation, the DFD model is enriched with 36 security solutions and assumptions.[5] These assumptions contain security-relevant information that does not reflect in specific solutions in the system or is external to the system. An example of this is how the media organizations communicate the URL of the SecureDrop instance to potential whistleblowers.

Methodology

Assignment of value estimates To minimize any bias in assigning the values, we use the following static value assignment scheme which considers the importance of protecting the identities and data of the involved users: 3 if the data of a whistleblower is involved, 2 if the data of an administrator is involved, 1 if the data of a journalist is involved. Each estimate has the same deviation of ± 1 and a default confidence of 4. The values are combined when multiple user types are involved. These values are used exclusively for the Asset Value (AV) for the prioritization of the security threats.

Adding security Solutions All the security solutions and assumptions from the threat model [Fre18b], documentation, and source code are encoded in the SecureDrop DFD model as solutions in the meta-model from Chapter 3. We assign a generic countermeasure strength to the

4 Appendix A.1 provides a detailed account on the construction of the SecureDrop DFD model.

5 These are not necessarily separate solutions. Some solutions (e.g., a firewall) are instantiated multiple times in the model between different pairs of elements.

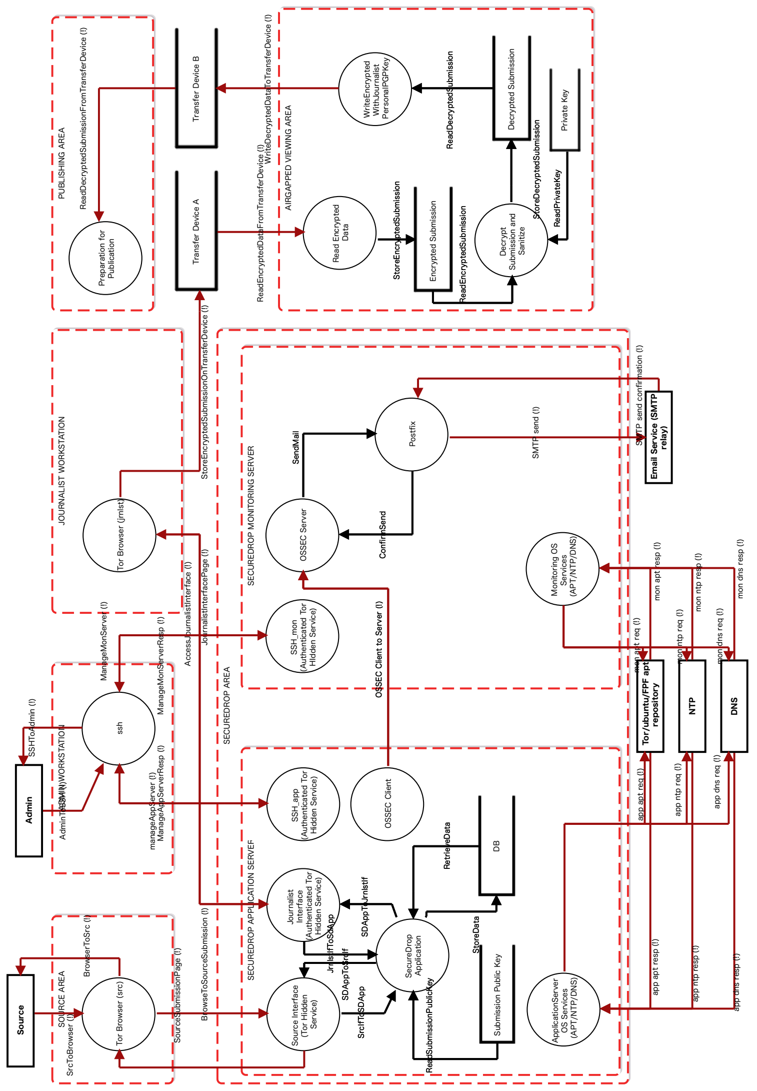


Figure 4.4: SecureDrop DFD model
The model is largely based on the SecureDrop threat modeling document. Since the threat modeling document refers to an earlier version, any inconsistencies were resolved by referring to the latest version of the documentation and source code.

different solutions, as only the presence of a countermeasure is relevant for the evaluation and not the actual strength of the countermeasure.

Risk Analysis & Data Collection We ran the interaction-based STRIDE [Sho14] threat elicitation and risk analysis on the modeled SecureDrop DFD to obtain the risk scores per elicited threat. The evaluation records two elements of data for each threat: (i) the calculated risk using the above risk model, and (ii) whether any type of countermeasure is present (to indicate the importance of the threat according to the SecureDrop developers).

Relating Mitigation Status and Potential Risk We evaluated the effectiveness of the resulting threats' risks as follows. The presence of security countermeasures or assumptions for a specific threat indicates that the threat is considered important enough by the developers to spend effort on providing the countermeasure or recording the assumption in the design documentation. Hence, to assess whether important threats are assigned higher risk scores we evaluate whether the threats with elevated risk scores have actual countermeasures or assumptions present in SecureDrop or its threat model documentation. Focused threat mitigation efforts by the developers of the system would thus result in a high coverage of the high-risk threats and, given the limited resources available, a lower coverage of low-risk threats. The opposite results would indicate that either the risk analysis information is a sub-optimal predictor of the importance of threats, or that the project misdirected its security efforts towards unimportant threats.

Results

Figure 4.5 shows the results of the threat elicitation on risk analysis of the SecureDrop DFD. The figure shows the density plot of mitigated (top) and unmitigated (bottom) threats by their potential risk. These density plot illustrates how the high-risk threats at the right-hand side of the plot are often explicitly mitigated in SecureDrop by security countermeasures or explicit assumptions in

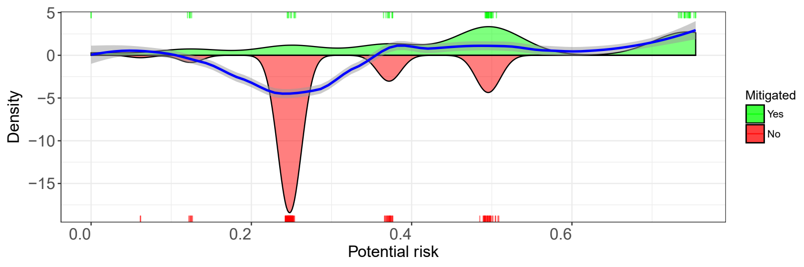


Figure 4.5: Threat Distribution by Potential Risk
The threats are grouped by the presence or absence of security solutions in SecureDrop that mitigate them. The distributions show a high mitigation-rate for the high-risk threats, indicating the estimated high-risk threats are considered important enough to counter them, and a lower mitigation-rate for low-risk threats.

the design documentation, while the lower-risk threats at the left-hand side of the plot are less frequently mitigated. Assuming that the SecureDrop developers are serious about the project's security, which is corroborated by the extensive security design documentation of the project, our results—even with the simple value assignment scheme used above—provide strong indications that the risk-based threat prioritization generated with the risk assessment model is able to identify the most important threats.

Threats to Validity

While the evaluation is limited to a single application, the risk model (FAIR) applied in the threat modeling context is a pre-existing risk decomposition which already found use in other application contexts. The current applications show that, with a reasonable set of asset value assignments, the application of this risk model in a threat modeling context can lead to realistic prioritization of security threats. This result does rely on the important assumption that the SecureDrop developers mitigated the important threats. This, however, is not an unreasonable assumption given the security-sensitive nature of the application and the extensive amount of threat modeling, security and attacker assumption documentation that are made available by the project. Furthermore, the fact that the mitigations rely on the presence of implemented countermeasures or explicitly documented assumptions provides strong evidence that these threats were actually considered by the developers.

4.3.3 Discussion on Interaction-Based Threat Elicitation

As previously mentioned in the interaction-based elicitation section, the interaction-based approach addresses several problems with the element-based threat elicitation. Here we provide a discussion on the lessons learned and future work in this space, covering: (i) the semantics and ambiguities of threat types; (ii) the role that knowledge representations, such as threat trees, can have in this context; and (iii) whether the current system abstraction is the most appropriate for all threat types.

Semantics and Ambiguities of Threat Types

Multiple elaborate discussions on the semantics of threats accompanied the construction of the interaction-based LINDDUN mapping table. More specifically, many conflicting views concerned ambiguities in the interpretation of what specific privacy threat on specific element type entails. Most of these revolved around the distinction between the location of a threat (i.e. where a threat would manifest itself), where it would be elicited (as applicability conditions in the table), and the target of a threat (what or who is threatened).

Especially for privacy threats, delineating these properties is a non-trivial endeavor, as they differ fundamentally from the more traditional interpretations in security threat modeling. Security threat modeling performs its analysis from the point of view of a deliberate malicious actor that targets a specific element of the system. Such an actor does not need to be present for a privacy threat to manifest itself. Indeed, the processing of personal data without user consent is an example of such a threat that does not have a malicious adversary unless the owner of the system is considered to be the adversary. Furthermore, the threatened entity is in all cases one or more data subjects, i.e. the person whose data the system processes. This threatened entity is not necessarily involved in the considered interaction when eliciting privacy threats. Indeed, this entity does not even have to be present in the model at all. To complicate matters even further, it is also not a single element that is attacked to manifest a privacy threat, but it can be a combination of multiple elements together that realize the privacy threat. For example, a system with a database with insufficiently anonymized data that shares this data with third parties poses a privacy threat, without the data subject being involved.

The interaction-based privacy threat elicitation table resolves these ambiguities by codifying the expert consensus in the privacy threat table. However, as the complexity of the threat types for which to analyze the system increases, these ambiguities will reappear, as larger configurations of involved DFD elements make it even harder to locate and assign a threat to a single element.

To resolve this problem, future approaches can move away from the manual systematic assessment of the system and, instead, rely on automation to do comprehensive assessment of the system for more complex configurations or patterns of DFD elements to determine the applicability of more complex threat types or design flaws. In such scenarios, to concrete 'location' of a threat is no longer an issue, as the approach no longer relies on a systematic iteration over these potential 'locations' in order to uncover the security and privacy threats.

Threat Trees

The existing security and privacy threat trees reflect the structured followed by the approach when eliciting threats. More specifically, the structure of the tree follows the DFD element types to closely align with the element-based elicitation approach. Since the interaction-based threat elicitation approach structures the applicability of the threat types in a more fine-grained manner to take the system's interaction context into account, these threat knowledge base structures will have to be refactored in order to support the representation of these updated threat types. We expect these knowledge structures to migrate away from trees in order to facilitate the incorporation of more complex applicability conditions that make the causal relations between threat types explicit such as, for example, *linkability* leading to *identifiability*.

Granularity for Threat Elicitation

We argued how the context of an interaction is better suited than individual elements for the elicitation of LINDDUN privacy threats. However, the pattern-based elicitation section illustrates how even the interaction context is insufficient for the detection of more complex design flaws. Arguably, these threats cannot be discovered with a plain element- or interaction-based threat analysis. They indicate a complex trade-off exercise between the systematicity of the approaches that iterate over the all the model elements to ensure a comprehensive assessment versus approaches that are more reliant on the extensiveness of their knowledge bases but, in return, can detect more complex flaw patterns in the system under analysis.

4.3.4 Discussion on the Risk Model

In this section we discuss open issues and planned future extensions to our risk-based threat elicitation approach.

Risk component units

To evaluate whether an attacker can defeat a countermeasure, the comparison of their strength and capability values requires them to use the same scale. However, there is no reusable absolute scale with clearly defined units to measure the strength of solutions and capabilities of attackers. To overcome this problem, they can be assigned relative values on an abstract scale. With these relative values, the risk assessment can compare their relative strengths and incorporate the effectiveness of the countermeasures in the analysis. The downside of this approach is that with an increasing number of solutions and attacker profiles, it can become harder to assign new values and ensure they make sense in relation to all the previously assigned values.

The impact of this problem can be reduced by providing an extensive catalog of security and privacy solutions and attacker profiles in which properly verified relative strength values are already assigned.

Effort trade-off between adding information and prioritizing threats

After performing automated threat elicitation, the manual processing of the (often) large lists of threats requires considerable effort. The inclusion of a risk analysis activity in the threat elicitation context can reduce the amount of post-processing effort required in filtering and prioritizing the threat list. Such a risk analysis activity, however, requires some additional information to be present, as illustrated by the risk decomposition in Figure 4.3. While the integration of the risk analysis can reduce the overhead of separate isolated risk assessment activities, the threat modeler still must provide this information. This information requirement introduces a trade-off exercise between enriching the threat model with more information for the risk analysis and afterwards manually performing a prioritization assessment of the resulting threats. The integration of the required risk inputs provides some additional benefits beyond the integrated risk assessment, it also documents the input information in the same model instead of relying on separate (or even implicit) information for assessing the risk. While this does impose some extra effort on the threat modeler, some optimizations or heuristics can reduce this effort such as, for example, the value assignment scheme used in the evaluation (Section 4.3.2).

Difficulty in determining the risk component estimates

An issue closely related to the effort in adding the risk input information to the model, is the difficulty in determining these values. The easiest solution for obtaining this information is by relying on business stakeholders to provide this information. Otherwise, the threat modeler will have to assign estimates for the model values. The presented risk model purposefully uses estimates specifically for supporting this uncertainty in determining the values for the risk model. The aforementioned optimizations can assist in this context as well to systematically assign values to all the model elements for providing impact information for risk analysis.

4.4 Related Work

In this section, we discuss the related work in three parts. First, the elicitation of threats, especially on the context of privacy threat elicitation, is discussed in light of the presented extension on interaction-based privacy threat elicitation. Next, expanding the context of the elicitation further, generic design flaw detection is covered. Finally, different approaches to threat prioritization and risk assessment are discussed to assist in triaging the identified security and privacy threats.

4.4.1 Threat Elicitation

The broader field of privacy engineering focuses on systematically identifying and addressing privacy threats [GGD11, GTD15], but also the development of design strategies and architectural patterns [CHH16], Privacy-Enhancing Technologies (PETs), and cryptographic enablers.

Methods and approaches that involve systematic threat modeling represent a suitable candidate for implementing the end-to-end risk-based privacy analysis demanded by the General Data Protection Regulation (GDPR) [Eur16]. These methods can be considered the technical component (complementary to the legal, organizational, and business-oriented measures) of the Privacy Impact Assessment (PIA) which involves estimating and resolving the overall privacy risks identified in the system.

Microsoft originally introduced the STRIDE threat modeling approach as part of its SDL [HL06, SS04]. Originally, this approach involves element-based threat elicitation. More recently, STRIDE has evolved towards interaction-based threat elicitation [Sho14], also in tool support [Mic16, Mic20]. Tuma and Scandariato [TS18] empirically evaluated the difference between the element-based and interaction-based threat elicitation performed by human analysts. They observe slightly lower productivity and a lower recall for the interaction-based elicitation, which could be explained by the complexity of the elicitation procedure. Automated application of interaction-based elicitation can avoid these productivity issues and elicitation errors by human analysts.

In addition to the LINDDUN methodology [DWS+11, Wuy15] that was extended above, other initiatives to privacy by design are noteworthy. PRIAM [DL16] is a framework for conducting a systematic privacy risk assessment, not at the level of technical architecture but at the level of data catalogs. It uses harm trees to link privacy weaknesses and risk sources to known harms. Oliver [Oli14] proposes an approach based on data flow modeling that is enriched with ontologies and classifications to annotate and describe information flows. This approach is element-based in nature. Tuma et al. [TSB19] presented a more formal end-to-end analysis which also relies on DFDs but does not elicit the STRIDE threats. Instead, they focus on data integrity and confidentiality objectives. Oetzel and Spiekermann [OS14] propose a step-by-step PIA starting from legal requirements (privacy targets), leading to a list of control recommendations. This approach does not systematically zoom in on a technical architecture. Shapiro [Sha16] introduced the System-Theoretic Process Analysis for privacy engineering process (STPA-Priv), which is similar to STPA-Sec, an earlier adaptation of such approach focused on security [YL14]. These approaches do not involve eliciting threats at the level of a technical architecture, but at the more coarse-grained level of a blackbox system and its context. Finally, Shapiro [Sha17] also introduced STECA-Priv which instantiates Systematic-Theory Early Concept Analysis in a privacy engineering domain. Contrary to the STPA-Priv or threat modeling, it does not yet require a system design, but instead starts from the system specification to infer a privacy control structure [Sha17].

4.4.2 Design Flaw Detection

The identification of problematic areas in design-level representations of a system with architectural smells [GPEM09, MCKX15, BLP09, TL18] or anti-patterns [NCFS17, MCK+19] has previously been used to identify software engineering issues, such as maintainability, and to assist in refactoring. For example, Garcia et al. [GPEM09] introduce a catalog of architectural bad smells specified with UML diagrams. Similarly, Bouhours et al. [BLP09] contribute with a catalog of 23 "spoiled patterns" or architectural design anti-patterns. Yet, the existing literature about architectural design flaws [GPEM09, BLP09, TL18, MCK+19, NCFS17] lacks a systematized knowledge base about security-relevant architectural design flaws in support of automated assessment.

Targeting security specifically, the report of the IEEE Center for Secure Design [ADD+14] provides a set of 10 key security design flaws to avoid. A similar goal is pursued by the OWASP Top 10 Application Security Risks [OWA17a]. A more extensive collection can be found in the Common Architectural Weakness Enumeration (CAWE) catalog by Santos et al. [SPM+17], constructed by extracting the architecturally relevant weaknesses from the CWE database by MITRE [CWE20] and assessing their impact on security tactics [BCK12]. Another resource from MITRE is CAPEC [CAP18] which provides this information from an attacker perspective. Finally, the observation that these catalogs contain similar flaws have led Tuma et al. [THMS19] to do a re-evaluation of the security catalog and they suggest several improvements to reduce overlap between the flaws. While the above catalogs provide an extensive set of weaknesses to identify, applying that knowledge a concrete application's design model requires translating this knowledge to practical detection rules, linked to a suitable system description that supports automatic assessment. None of these catalogs are currently amenable to that level of automation.

The approach of Berger et al. [BSK16] leverages Microsoft's threat modeling approach [SS04, HLOS06] to detect architectural flaws by translating CWE [CWE20] and CAPEC [CAP18] entries to graph queries, relying upon element attributes. Chapter 3 discussed the benefits of explicitly representing complete solutions [SYVJ18c] in order to ensure the effect of complex solutions on multiple elements are correctly and consistently incorporated into the model, showing positive improvements in terms of semantic quality, traceability, separation of concerns, and dynamism.

Almorsy et al. [AGI13] presented an approach for formalizing attack scenarios and security metrics in OCL and validated the approach by translating NIST security principles [SHF01] and attack scenarios from CAPEC [CAP18] in OCL signatures. For the analysis, they rely on a system description model in UML, together with a security specification model to capture security objectives, requirements, architecture, and controls. This contrasts with our approach which relies on DFDs.

A final common class of design-level analysis approaches for security is of course threat modeling, which starts from a DFD-based abstraction of the system to elicit security [HL06, Sho14] or privacy [DWS+11, Wuy15] threats. Both the security and privacy threat modeling approaches support a systematic analysis of the system under consideration by iterating over every element (element-based [HL06, Sho14, DWS+11]) or interaction (interaction-based [Sho14, SWY+18]). The knowledge-bases used in these approaches can also be extended for detecting additional threat types, but the element- or interaction-based approach limits the complexity of the criteria to assess as they remain limited to a single element or interaction. The security design flaws from the catalog, however, can involve multiple interactions as sensitive data traverses through the system, and can verify the presence of multiple required solutions together to ensure the absence of a single security design flaw. While such an approach loses the systematicity of threat modeling, it does enable identifying more complex design flaws.

4.4.3 Threat Prioritization

Türpe [Tür17] discussed how security needs arise from the interactions of three dimensions: design, goals, and threats, and observed how many efforts focus only on a single dimension. We structure this section according to the threat–design interactions and the threat–

goal interactions, as threat modeling and risk analysis focus on the interactions between these dimensions.

In existing approaches and applications [SS04, Tor05, HLOS06, HL06, Sho08, Mic16], data flow diagrams remain largely security and privacy agnostic models, with only minor, and often ad-hoc, additions for security or privacy. However, recently there have been several proposals for extensions to these data flow diagrams in order to have a more systematic representation of security- and privacy-relevant information in order to elicit the most relevant security and/or privacy threats [ASS16a, TSWS17, SYVJ18c, BSK16]. However, thus far, the influence of these solutions in the resulting threats is only binary. This means that, while the solutions can eliminate certain threats, they do not assist in prioritizing the threats that do remain. Any assessment to do so must be manually performed afterwards.

Risk analysis, on the other hand, focuses the threat–goal interactions. Risk analysis approaches elicit security requirements starting from security goals, anti-goals, such as in CORAS [LSS10], or attack trees [Sch99], and can be used in a complementary fashion to threat modeling [RKK16]. Instead of conducting such analyses in isolation, we presented an integrated approach that includes the risk information in the DFD model used for threat modeling, thereby integrating both activities and enabling them to reinforce each other's results.

A related approach that applies risk analysis in the context of threat modeling is Process for Attack Simulation and Threat Analysis (PASTA) [UM15]. It starts with the definition of business objectives and scope, and it includes attack tree modeling and risk analysis steps at the end. It supports the use of existing threat modeling tools, such as the Microsoft Threat Modeling Tool [Mic20], as part of its process. The risk analysis phase is numerical, but assigns only a limited set values (i.e. 90%, 50%, and 10%) to the risk parameters according to the low, medium, or high options offered for them. The percentages are averaged to determine the threat probability, which is used together with vulnerability, impact, and countermeasures to calculate the risk. Our approach collects this information in the model up-front to enable its reuse across multiple threats for the risk-driven prioritization.

Beckers [Bec12] compared multiple privacy requirements engineering approaches. None of the considered approaches support the notion of risk, despite being explicitly required by privacy regulations [Eur16].

Heckle and Holden's [HH06] findings suggest that neither privacy impact assessments nor classic risk analysis models are sufficient for privacy risk assessment in the context of voting systems. Abu-Nimeh and Mead [AM09] propose combining them by the IRS PIA [Int96] in Security Quality Requirements Engineering (SQUARE) [MS05]. While such a PIA [Int96] supports a detailed assessment of the realization of privacy-by-policy in the framework of Spiekermann and Cranor [SC09] given its focus on assessing compliance with privacy principles, a set of questions may not be the best approach. Alshammari and Simpson [AS18] make the case for a model-based approach for privacy compliance checking. The incorporated data protection view [SDVL$^+$19] supports such a model-based compliance assessment. Furthermore, its integration in the risk assessment provides support for assessing the risk of privacy threats such as identifiability and linkability, supporting the realization of privacy-by-architecture [SC09].

PRIAM [DL16] provides a very detailed description of information that needs to be collected for privacy risk assessment. The risk assessment itself requires the construction of harm trees, in which the risk is assessed with the combination of privacy weaknesses and risk sources for feared events which can lead to the harm at the top of the tree. Our approach can be considered a kind of instantiation of this approach, but explicitly requires the assignment of numerical estimates for the risk factors. By requiring such numerical assignments, a completely automated assessment can be performed.

Hong et al. [HNLL04] presented a privacy risk model specifically developed for ubiquitous computing systems, focusing on the selective disclosure of personal information (*personal privacy*). Similar as the IRS PIA [Int96], a set of questions is used for the privacy risk analysis, after which the risks are prioritized.

4.5 Conclusion

Element-based threat elicitation involves iterating over all the elements in a DFD model without considering the system context. In this chapter we: (i) argued why this approach sub-optimal for eliciting threats, (ii) extended the LINDDUN privacy threat modeling framework to take the interaction context into account in the threat elicitation, and (iii) show the further expansion of the architectural context for the detection of more complex design flaws.

After the elicitation of the security and privacy threats, we considered the lacking support for prioritizing the elicited threats which would require additional overhead before being able to obtain actionable results and would happen in a disconnect from the concrete system the threats originated from.

To address this problem, we presented a risk model to prioritize the elicited security and privacy threats, grounded in concrete data on the system, its security and privacy solutions, and the relevant attacker profiles. The risk analysis embedded in a threat modeling approaches resolves the disconnect by directly relying on the DFD model elements enriched with relevant risk information, enabling the on-the-fly per-threat risk assessment. From this risk-enhanced threat elicitation, the resulting list of threats can be triaged according to their calculated risk to support mitigation efforts on the most important and high-risk threats first.

Furthermore, by measuring the risk of all the elicited threats automatically, the total risk reduction progress can be measured and managed. In the future work, we will explore how the effectiveness of the countermeasures in these security and privacy solution catalogs can be dynamically updated with information on newly discovered vulnerabilities to enable continuous reassessments of existing models in light of this new information.

Chapter 5 Outline

5 Advanced Tool Support

"Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" [...] I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

— Charles Babbage [Bab64]
Passages from the Life of a Philosopher

This chapter is an extended version of the originally published article:
Laurens Sion, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. SPARTA: Security & privacy architecture through risk-driven threat assessment. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 89–92. IEEE, 2018

The previous chapters presented the contributions to support an in-depth and rigorous security and privacy threat assessment (Section 4.1) and risk analysis (Section 4.2) to elicit and prioritize security and privacy threats in solution-enriched Data Flow Diagram (DFD) models (Chapter 3). This chapter targets the disconnect between these analysis activities by combining them together in integrated tool support. The tool implementation aims to achieve the following objectives.

Automation. A key goal of the tool support prototype is to provide proof of concept implementation for the automated elicitation and prioritization of security and privacy threats. The automated threat elicitation and prioritization does not require additional end-user inputs other than the enriched DFD model that captures the information on the software system to analyze, its security and privacy solutions, and the relevant asset values. Automation is also key to ensure that the threat elicitation and analysis renders reproducible results regardless of the threat modeler performing the analysis.

Fine-grained threat applicability. Traditional threat modeling approaches rely on binary assessment of a threat's applicability. However, not every security or privacy solutions is equally effective at preventing a certain threat type. Security and privacy threat elicitation tools should therefore support a more nuanced assessments of a threat's applicability by taking into account the effectiveness of existing security and privacy countermeasures.

Reuse of security and privacy knowledge. Security and privacy threat modeling approaches rely on a common set of knowledge on the types of security and privacy threats that are relevant for specific DFD elements. However, this knowledge may be modified and adapted to the end-user's organizational context. Automated tool support should explicitly capture this knowledge and enable its reuse across multiple analyses. Other relevant security and privacy knowledge is that on security and privacy solutions. Tool support should also capture this knowledge explicitly and support reuse across multiple models.

Immediate modeling feedback. Since the main goal of threat modeling tool support is to assist the threat modeler in analyzing the system to identify and mitigate security and privacy threats, immediate analysis feedback during the modeling is very useful. Such feedback enables the end-user to perform what-if analyses by modeling alternative design and evaluate the impact of these designs on the elicited security and privacy threats.

Measuring progress. Finally, given the large amount of security and privacy threats that can be elicited from system descriptions, managing

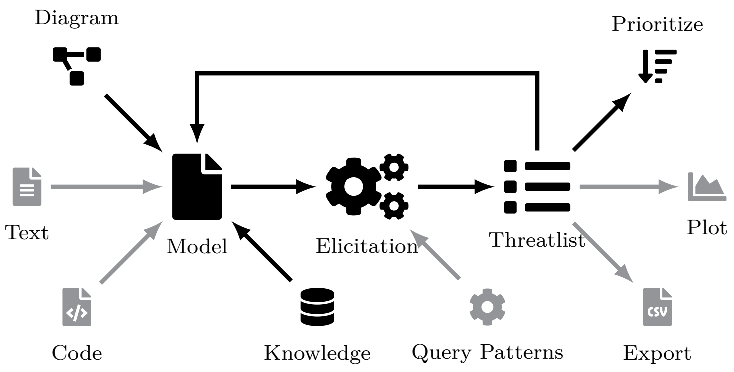


Figure 5.1: Sparta Approach Overview
High-level overview of the sparta *approach with the three key activities in the center and potential input and outputs on the left- and right-hand side. The grayed-out items show potential extensions of the approach such as relying on different types of inputs (e.g., a textual* dfd *representation), new types of analysis, and future exports based on the analysis results.*

these results to keep track of the progress in mitigating the identified threats is an essential part of tool support.

The following sections will elaborate on the different components of the prototype to discuss the design and implementation of the goals in the tool. Section 5.1 first outlines the approach. Section 5.2 then discusses the concrete implementation of the prototype. After that, Section 5.3 provides the evaluation and discussion. Section 5.4 presents the related work. Finally, Section 5.5 concludes the chapter.

5.1 Approach

This section provides a high-level overview of the approach followed in the implementation of the sparta prototype.

The key idea in the sparta approach is to (i) rely on a model-based representation of the system, (ii) analyze this representation to identify security and privacy threats, and (iii) capture the security and privacy solutions in response to these results back in the model to verify that the identified threats have been correctly resolved.

Figure 5.1 graphically represents the approach with the three main steps in the center: model, analysis, and the results. The resulting identified security and privacy threats subsequently lead to the inclusion of new security and privacy solutions in the model.

This approach is intended to be extendable in each dimension. It relies on a model representation of the system, but there may be multiple different ways to populate such a model. The straightforward way is to rely on a graphical diagram editor to construct this model. Other ways to construct the model-based representation could be a text-based representation of the model, or reconstructing it from the source code or source-code annotations [Thr19].

The analysis in the approach relies on a knowledge base on the threat types to identify. These knowledge bases could be further extended, but the approach also supports the inclusion of completely new analysis activities to perform on the input models.

Finally, there are a number of extensions possible on processing and presenting the analysis results to the end-users. These can range from the initial presentation of the resulting threat list to more advanced graphical representations such as different plots or overlays to situate the high-risk areas on a diagram model of the system.

The following section will go into detail on how these three main parts are implemented in the SPARTA prototype.

5.2 Solution

This section discusses the implementation of the SPARTA prototype in the following parts: (i) the graphical modeling support, (ii) the threat elicitation and the risk-based threat prioritization, (iii) the underlying meta-model, and, finally, (iv) using the SPARTA prototype.

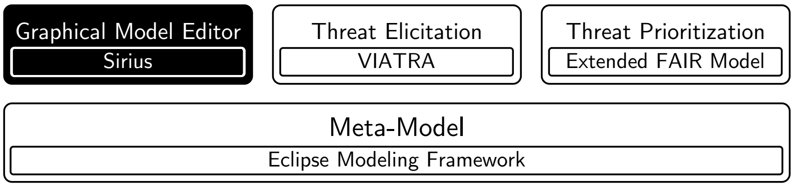
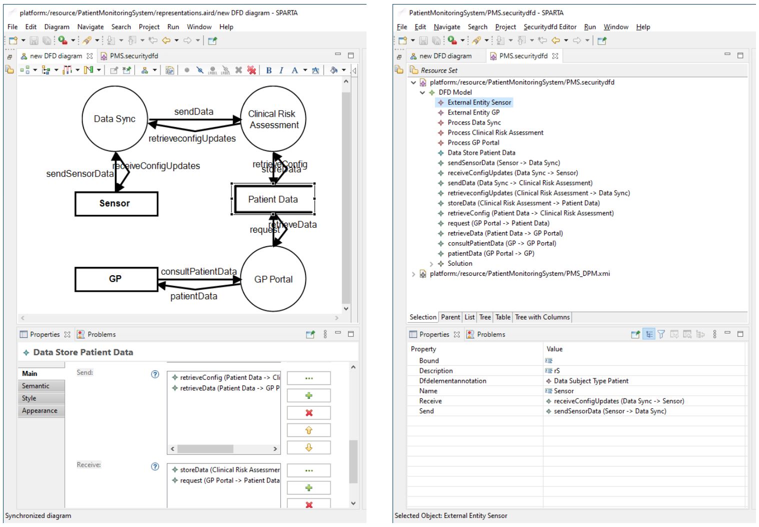


Figure 5.2: Sparta Modeling Support
This figure highlights the sparta *components for graphical modeling support.*

5.2.1 Modeling

Figure 5.2 shows the components from sparta that provide graphical modeling support. The graphical support in sparta is focused exclusively on the modeling of dfds. All the other information in the models is accessible through a tree-based model editor view but there is no graphical visualization of this information. The existing dfd visualization can be extended in the future to support multiple different visualizations of security and privacy solutions. The next parts first discuss the graphical model editor, after which the creation and editing of the catalogs in the tree-based editor is discussed.

Graphical Model Editor On top of the meta-model specified in the Eclipse Modeling Framework (emf), is the dfd viewpoint specification created in Sirius. This viewpoint enables the creation of basic dfd models containing processes, external entities, data stores, data flows, and trust boundaries. Since security and privacy solutions do not have any a dfd-based visualization, the graphical dfd editor does not yet support their instantiation. The regular tree-based model editor can add these solutions to the models. In addition to the constraints that are enforced by the meta-model, the dfd model editor can support a number of additional soundness checks on the created models as outlined in Section 3.3.1 to assist users during the creation or modification of models in the graphical editor. These criteria for sound models, both warnings and errors, are expressed in the aql model constraints. Some examples of these checks are: (i) direct connections



Data Store Patient Data

Send:
- retrieveConfig (Patient Data -> Cli
- retrieveData (Patient Data -> GP P

Receive:
- storeData (Clinical Risk Assessmer
- request (GP Portal -> Patient Data

platform:/resource/PatientMonitoringSystem/PMS.securitydfd
- DFD Model
  - External Entity Sensor
  - External Entity GP
  - Process Data Sync
  - Process Clinical Risk Assessment
  - Process GP Portal
  - Data Store Patient Data
  - sendSensorData (Sensor -> Data Sync)
  - receiveConfigUpdates (Data Sync -> Sensor)
  - sendData (Data Sync -> Clinical Risk Assessment)
  - retrieveconfigUpdates (Clinical Risk Assessment -> Data Sync)
  - storeData (Clinical Risk Assessment -> Patient Data)
  - retrieveConfig (Patient Data -> Clinical Risk Assessment)
  - request (GP Portal -> Patient Data)
  - retrieveData (Patient Data -> GP Portal)
  - consultPatientData (GP -> GP Portal)
  - patientData (GP Portal -> GP)
  - Solution

platform:/resource/PatientMonitoringSystem/PMS_DPM.xmi

Property	Value
Bound	
Description	rS
Dfdelementannotation	Data Subject Type Patient
Name	Sensor
Receive	receiveConfigUpdates (Data Sync -> Sensor)
Send	sendSensorData (Sensor -> Data Sync)

Selected Object: External Entity Sensor

(a) DFD Model Editor (b) Model Tree

Figure 5.3: SPARTA Modeling Support Screenshots
The above screenshots illustrate the modeling support of the SPARTA prototype. Figure 5.3a shows the graphical support for creating DFD models, while Figure 5.3b shows the model tree editor that is available for creating and editing both the DFD models and the catalogs of security and privacy solutions and threat types.

between data stores, (ii) empty trust boundaries, (iii) data store sinks, (iv) completely disconnected elements.

Catalogs To support the modeling and analysis, there are two types of catalogs available. These do not have a visualization, but the base DFD model loads them so that they can be used in the model itself and leveraged in the subsequent analyses performed on the model.

The first catalog is the *security and privacy threat types catalog*. This catalog contains all the threat types that need to be detected in the created DFD models. Instead of relying on some specific version of tool support, the models explicitly refer to the threat type catalog on the threats to elicit to ensure reproducible results.

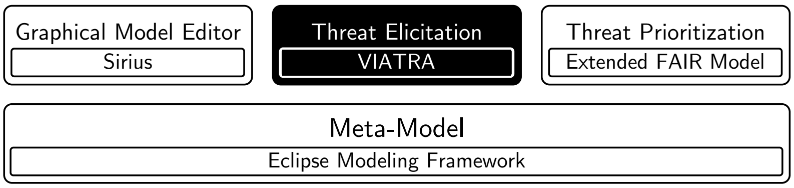


Figure 5.4: Sparta Threat Elicitation Components
Focus on the threat elicitation components in sparta.

The second catalog is the *security and privacy solutions catalog*. This catalog provides a reusable resource on security and privacy solutions that can be instantiated in the DFD models and can be reused across multiple models. The effects of these solutions are taken into account in the subsequent threat elicitation step. Multiple different solution catalogs can be used together.

5.2.2 Elicitation

The threat elicitation analysis in SPARTA (Figure 5.4) relies on the user model complemented with the used solution catalogs and threat type catalogs. The SPARTA tool uses VIATRA query patterns to query the user model to find the threats. In the case of the classical interaction-based threat elicitation, this involves querying the different types of interactions and using the threat type catalog to lookup whether a threat type is applicable for the specific interactions. Though, given the flexibility of the VIATRA pattern query language, more complex design flaws can also be discovered with the appropriate patterns (as elaborated upon in Section 4.1.3). Sparta currently relies on fixed set of interaction-based query patterns to do the threat elicitation. However, VIATRA supports dynamically loading new patterns. Future extensions could load user-created pattern catalogs for the elicitation.

The threat elicitation steps followed in the manual threat modeling approach [Sho14] are outlined in Snippet 5.1, which shows both the element-based and interaction-based elicitation. Sparta uses a different approach to elicit the threats: Instead of mechanistically iterating

```
public void elicitPerElement(DFDModel m) {
  for (DFDElement e : m.elements()) {
    for (Threat t : threatTypes) {
     if (t.isApplicable(e)) {
     // threat is triggered
}}}}
public void elicitPerInteraction(DFDModel m) {
  for (DataFlow e : m.dataflows()) {
    for (Threat t : threatTypes) {
      if (t.isApplicable(e.from(), e, e.to())) {
        // threat is triggered
}}}}
```

Snippet 5.1: Threat Modeling Elicitation Steps
This snippet shows the threat elicitation steps for both the element-based and interaction-based threat elicitation approaches with the meta-model from Chapter 3.

```
pattern processToProcess(threatened:Process, p2:
    Process, df:DataFlow, t:ThreatType, ts:
    ThreatSpecification, mitigationStatus:EString)
{
  // check if threat type applies
  find threatApplicable(threatened,t,ts,
        "ProcessToProcess");
  // check if threatened and p2 communicate
  find communication(threatened,df,p2);
  // check if mitigated in context of df
  find mitigation(threatened,df,t,
        mitigationStatus);
}
```

Snippet 5.2: VIATRA Interaction Pattern
This snippet shows the VIATRA *pattern description for detecting threats on interactions between two processes. The* mitigationStatus *parameter can be used to separately query for mitigated or unmitigated threats. However, because a pattern match is binary, finding partially mitigated threats requires matching regardless of the mitigation status and using the risk-based analysis to incorporate the strength of countermeasures when assessing the vulnerability to the considered threat type.*

over all model elements or interactions, the SPARTA approach involves modeling the applicability conditions as VIATRA patterns for which the DFD model is queried. Snippet 5.2 shows an example of such a VIATRA model query pattern for interactions between two processes. The pattern checks the types of the involved elements (in the pattern signature), whether the threat type is applicable (according to the dynamically loaded threat type pattern catalog), whether the elements communicate in the required configuration, and whether any mitigations are present to protect against the considered threat type.

This approach offers a number of benefits over the traditional iteration-based assessments. First, expressing the threat type applicability conditions as separate patterns enables reuse of these expressions across multiple threat types preventing the need to replicate this knowledge multiple times. Second, it improves the maintainability of these patterns as there is only a single location where these patterns need to be updated. Third, it enables efficient assessments of DFD models by relying on the VIATRA model query engine to automatically trigger multiple threat types for the matched patterns.

However, there are a number of downsides to this approach as well. First, the impact of changes to these patterns is harder to assess. There may be multiple different threat types that rely on a specific pattern for detection and these are all impacted by changes to the pattern description. This is not a problem for the interaction-based threat elicitation because all the conditions are very similar (i.e. the type of the sending element, the data flow, and the type of the receiving element). For the pattern-based design flaws, assessing the impact may become more complex as multiple flaws could be affected when modifying the pattern. Second, if a pattern also requires some threat type-specific exclusion criteria, these can become more complex as they have to be expressed generically. This is illustrated with the `find mitigation` condition in Snippet 5.2, which is parameterized with the threat type, instead of referring to a specific type such as, for example, tampering. To avoid the up-front specification of excluded threats, the exclusion criteria have to be written generically in the pattern. Furthermore, the threat elicitation in SPARTA avoids up-front exclusion of security or privacy threats (unless theoretically impossible), but, instead, relies

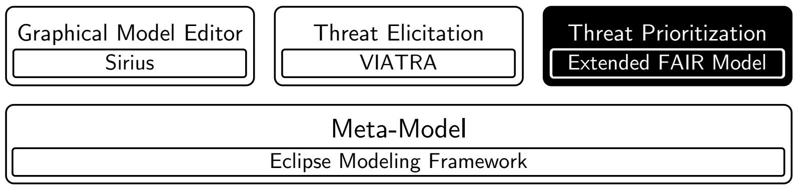


Figure 5.5: SPARTA Threat Prioritization Components
Focus on the threat prioritization components in SPARTA.

on the risk-driven prioritization to determine to which extent the application is vulnerable. This avoids the up-front elimination of threats that may increase in relevance when the threatened assets change in their valuation.

5.2.3 Prioritization

For the risk assessment (Figure 5.5), SPARTA implements the risk model as presented in Chapter 4. SPARTA uses some additional statistical libraries to sample and aggregate the results. The attacker profiles used in the risk assessment are provided as fixed list to choose from in the analysis. Ideally, these are also modeled as a user-editable resource that is loaded in the base user-model.

Performing the risk assessment of the elicited threats requires several inputs. Supplying this information is not always easy, as there may be considerable uncertainty in these values. The SPARTA risk assessment model's design specifically supports expressing this uncertainty in its inputs by relying on *estimates* to capture the uncertainty about the values used for the risk assessment (explained in Section 4.2.1).

Estimates

To explicitly support uncertainty in the risk input values, these inputs are described as estimates. An estimate for the risk input has four input values: (i) minimum, (ii) most probable (mode), (iii) maximum, and (iv) confidence (of the most probable value). These four estimate values are the parameters of a modified-PERT distribution [Vos08] and support expressing a wide range of uncertain values. Figure 5.6 shows the impact of different parameter values on the resulting distributions.

Sparta samples these different distributions for the risk components and calculates the risk using these samples. This calculation results in number of risk values equal to the sample size. The set of risk values can then be processed to obtain the lower and upper bounds for the risk values as well as the most probable value.

Estimating the Risk

This section revisits the risk model from Chapter 4 to show how these estimates are combined to calculate the final risk score. All risk calculations are performed on vectors of length S, which corresponds with the sampling size that is used in the calculations.

Choosing an appropriate sampling size involves a trade-off between the computational effort for performing the risk assessment and the accuracy of how closely the results match the distribution. Figure 5.7 shows the distributions for a number of different sample sizes to illustrate the impact of the sampling sizes on the results. Sparta uses a default sampling size of 2000. Given sparta's context of continuous risk assessment, future optimizations could involve continuously increasing the sample size after the initial set of threats is elicited and as long as no changes are being made that trigger changes in the threats lists. This enables gradually increasing the accuracy of the threat results while still providing early results for the threat modeler to act upon.

Below, the breakdown of a single threat's complete risk calculation using the estimates is provided. Figure 5.8 visualizes this calculation by showing how the different distributions for a single elicited threat are combined in a single Risk distribution.

Sparta calculates the risk with the entrywise product[1] of the Loss Magnitude (LM) and the Loss Event Frequency (LEF) vectors.

1 Also called Hadamard product

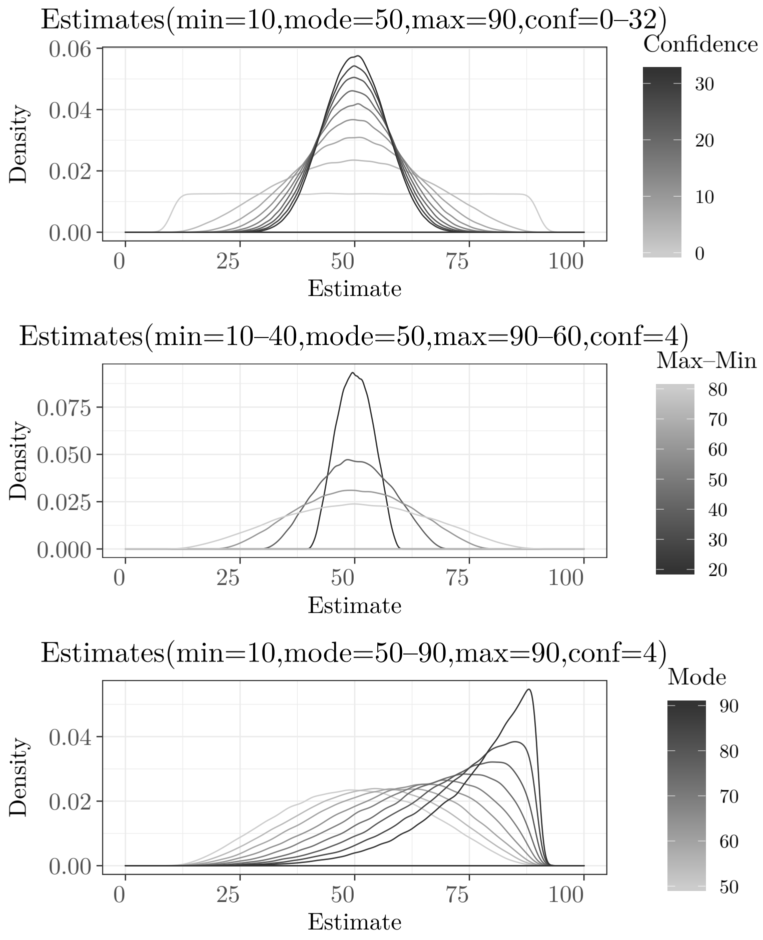


Figure 5.6: Impact of the estimate parameters
The above three modified-PERT distribution plots illustrate the impact of setting different, respectively, confidence, minimum and maximum, and most probable (mode) values. The parameters can be used to express a wide range of uncertainty.

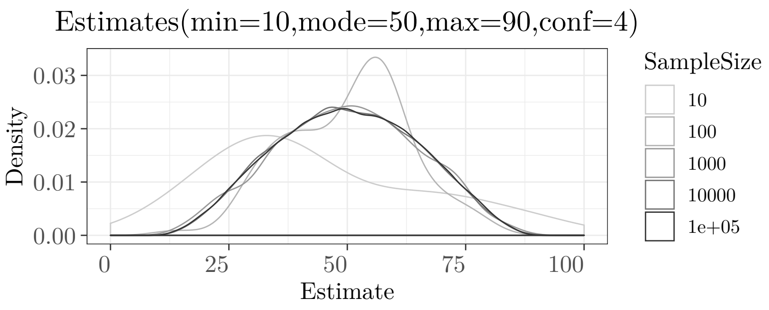


Figure 5.7: Impact of the Sample Sizes
This graph shows the density plots of different sample sizes to illustrate the impact of the sample size on the resulting density plots. From more than 1000 *samples, the differences in the density plots become small.*

$$\begin{aligned}\boldsymbol{Risk} &= \boldsymbol{LM} \odot \boldsymbol{LEF}\\ &= [\boldsymbol{LM}_1 \times \boldsymbol{LEF}_1, \boldsymbol{LM}_2 \times \boldsymbol{LEF}_2, \dots, \boldsymbol{LM}_S \times \boldsymbol{LEF}_S]\end{aligned}$$

The LM is calculated by combining the Asset Value (AV) and the Privacy Value (PV).

$$\boldsymbol{LM} = \boldsymbol{AV} + \boldsymbol{PV}$$

The PV follows from the entrywise product of its subcomponents: Data Type Sensitivity (DTS), Number of Records (NR), Data Subject Type (DST), and Number of Data Subjects (NDS).

$$\begin{aligned}\boldsymbol{PV} &= \boldsymbol{DTS} \odot \boldsymbol{NR} \odot \boldsymbol{DST} \odot \boldsymbol{NDS}\\ &= [\mathrm{PV}_1, \mathrm{PV}_2, \dots, \mathrm{PV}_S]\\ \mathrm{PV}_i &= \mathrm{DTS}_i \times \mathrm{NR}_i \times \mathrm{DST}_i \times \mathrm{NDS}_i\end{aligned}$$

The LEF vector is calculate by multiplying the entrywise product of the Retention Period (RP) and the Threat Event Frequency (TEF) with the Vulnerability (V).

$$\boldsymbol{LEF} = \mathrm{V} \cdot (\boldsymbol{RP} \odot \boldsymbol{TEF})$$

The TEF follows from the entrywise multiplication of the Probability of Action (PoA) with the Contact Frequency (CF).

$$\begin{aligned}\boldsymbol{TEF} &= \boldsymbol{PoA} \odot \boldsymbol{CF} \\ &= [\mathrm{PoA}_1 \times \mathrm{CF}_1, \mathrm{PoA}_2 \times \mathrm{CF}_2, \ldots, \mathrm{PoA}_S \times \mathrm{CF}_S]\end{aligned}$$

The V scalar is calculated as the maximum of the Countermeasure Defeated (CD) and the Countermeasure Bypassed (CB) parameters, as one of these factors is sufficient to be vulnerable.

$$\mathrm{V} = \max(\mathrm{CD}, \mathrm{CB})$$

While the CB is a binary value in its easiest form, i.e. the adversary is either able to bypass the countermeasure or not, it is also possible to express a likelihood as a fraction which will be taken into account against the vulnerability because of the countermeasure being defeated.

The CD is a fraction that is calculated by performing $\boldsymbol{S}$ attack simulations that the countermeasure under consideration has to resist. The CD value represents the fraction of successful attacks.

$$\mathrm{CD} = \frac{\sum_{i=1}^{S} \mathrm{CD}_i}{\boldsymbol{S}}$$

Each individual attack attempt CD_i is a binary value obtained by comparing a sample of the Threat Capability (TC) of the adversary and the Strength (S) of the countermeasures.

$$\mathrm{CD}_i = f(\mathrm{TC}_i, \mathrm{S}_i) \text{ with } f(x, y) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases}$$

The CD is calculated as a single value. However, instead of using the fraction of successful attacks, the set of attempts can be considered as a series of Bernoulli trials for which a confidence interval can be

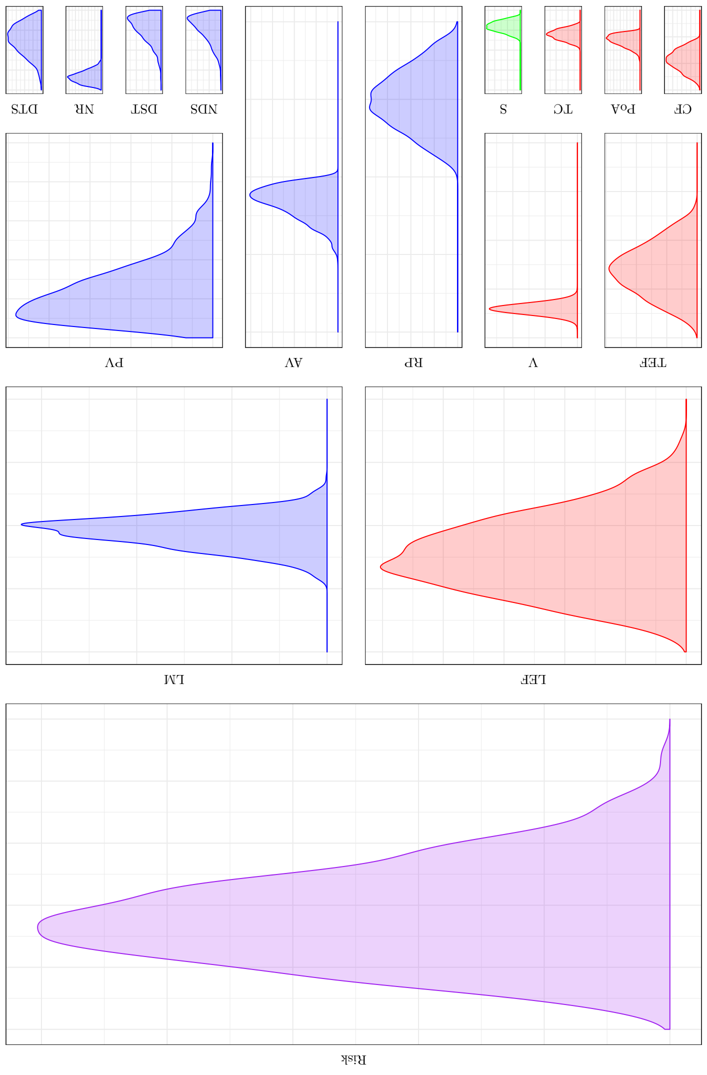


Figure 5.8: Risk Composition from the Distribution
This graph shows how the different risk components are combined into the final risk distribution.

calculated. SPARTA calculates the Clopper-Pearson 95% confidence interval to obtain the lower and upper bounds. When the number of successes (x) equals either 0 or the sample size $\boldsymbol{S}$, the interval is:

$$x = 0 \quad \left(0, 1 - \left(\frac{\alpha}{2}\right)^{\frac{1}{n}}\right)$$

$$x = \boldsymbol{S} \quad \left(\left(\frac{\alpha}{2}\right)^{\frac{1}{n}}, 1\right)$$

Otherwise, the interval is calculated with, respectively, the $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$ quantiles of the beta distribution for the lower and upper boundary.

$$\left(\mathrm{Beta}\left(\frac{\alpha}{2}; x, \boldsymbol{S} - x + 1\right), \mathrm{Beta}\left(1 - \frac{\alpha}{2}; x + 1, \boldsymbol{S} - x\right)\right)$$

The above formulas provide a lower and upper boundary of the CD value, which can also be used in the calculation of V to obtain its lower and upper boundaries.

The result of the risk calculation process is still a vector of sample size $\boldsymbol{S}$. An individual threat's risk is calculated as the median value:

$$\mathrm{Risk} = \mathrm{median}\,(\boldsymbol{Risk})$$

Using the aforementioned lower and upper boundaries of V, the lower and upper boundaries of the risk value can also be calculated if desired.

Granularity of the Calculations

A final aspect of the risk assessment in SPARTA is the granularity of the risk calculations. SPARTA calculates the risk values at a very fine-grained level and keeps track of these fine-grained risk values in a large risk matrix. The risk results are individually for every combination of: (i) threatened DFD element, (ii) the threat type, (iii) the attacker profile, (iv) the data subject type, and (v) the data type. Table 5.1 shows a small excerpt of what such a table looks like for the patient monitoring system shown in Figure 5.9.

Table 5.1: Excerpt of individual risk assessment results across the five parameters

System Context (DFD)	Threat Type	Attacker Profile	Data Subject Type	Data Type	Risk
storeData	Linkability	Motivated, lim. cap.	Patient	ECG Measurement	4.542
storeData	Identifiability	Motivated, lim. cap.	Patient	Risk level	6.632
storeData	Detectability	Opportunist	Patient	Risk level	10.409
patientData	Discl. of Info	Opportunist	Patient	Body temp measure.	3.66
retrieveData	Linkability	Motivated, capable	Patient	Risk level	1.084
retrieveData	Detectability	Disgruntled employee	Patient	Risk level	0.035
GP	Detectability	Motivated, cap., org.	General Practioner	Credentials	0.423
...	...	...	...	...	...
... *(6793 rows omitted)*					.

For a system with 16 DFD elements, 6 threat types, 5 attacker profiles, 2 data subject types, and 4 data types. These entries can be aggregated across the different dimensions to gain insights into which parameters are the biggest contributors to the privacy risk.

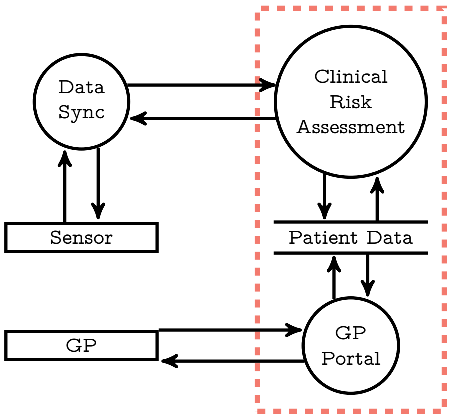
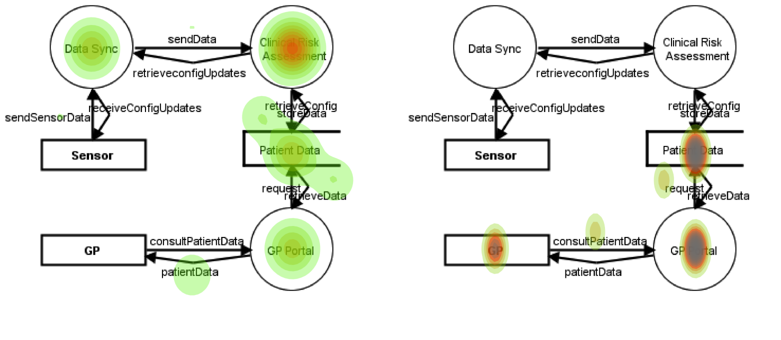


Figure 5.9: Patient Monitoring System Data Flow Diagram
Simplified DFD *representation of the Patient Monitoring System. This model is further enriched with the data subject types and their personal data types.*

After the data in Table 5.1 is calculated, it can be aggregated in multiple different ways to enable interesting insights into the impact of various parameters on the distribution of the risk in the application. This can support analysis activities by pinpointing which parameters are the biggest contributors to the overall risk of the system.

Figure 5.10 visualizes the distribution of the risk aggregated per system element (column 1 in Table 5.1) and data subject type (column 4 in Table 5.1) by adding the risk values for every combination of these two parameters. This distribution visualization provides an overview of where the risk is the highest for every combination of system context and data subject type. The visualization of this aggregated information in a heatmap makes it easily detectable where the highest risk associated with a data subject type is situated in the system. These types of visualizations can also be created for other parameter combinations to gain insight into the impact of the different parameter combinations on the risk distributions.



(a) Patient Risk Heatmap (b) General Practitioner Risk Heatmap

Figure 5.10: Heatmaps of data subject type risks
The two images illustrate the distribution of risk in the DFD *for different data subject types. The heatmaps are constructed by overlaying a 2D density plot on top of the* DFD. *Figure 5.10a shows the distribution of the patient risk. Note that the* sendData *and* sendSensorData *data flows in this diagram do have a small, but non-zero risk value. Figure 5.10b shows the distribution of risk from the perspective of the General Practitioner (GP).*

5.2.4 SPARTA Meta-Model

The SPARTA meta-model provides the foundation for the previous three components on modeling, elicitation, and risk-driven prioritization (Figure 5.11). Whereas, the full details on the meta-model itself are provided in Chapter 3, this section focuses on the implementation in the EMF and on how the different models are combined in SPARTA.

Sparta makes use of four types of models (all separate files): (i) the user model, (ii) the security and privacy solution catalog, (iii) the security and privacy threat catalog, and (iv) an optional threat specification catalog to specify deviations from the standard threat catalog. All the meta-model classes from Chapter 3 are implemented as *EClass*es in Eclipse Ecore. Because every model file requires a root object containing the elements, SPARTA introduces three catalog types to

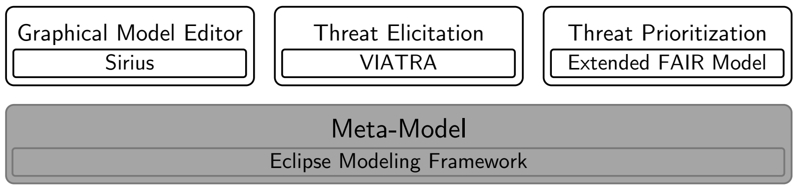


Figure 5.11: SPARTA Components
Overview of the main components that comprise the SPARTA prototype: the graphical model editor, the threat elicitation engine, and the risk-based threat prioritization. All the above components are built on top of the meta-model, which is realized in the Eclipse Modeling Framework (highlighted at the bottom).

contain all the relevant model elements: (i) a *DFD Model* type for containing the user model, (ii) a solution catalog type for the security and privacy solutions, and (iii) a threat catalog type for the threat types and the optional threat specifications.

A single *DFD Model* loads the security and privacy solution catalog(s) and threat type catalog(s) it needs, allowing an end-user to combine multiple different solution catalogs and threat types catalogs as desired.

5.2.5 Using SPARTA

This section briefly elaborates on the usage scenarios of the SPARTA tool prototype.

The first step is the modeling of the system under consideration in a DFD. The SPARTA tool provides two main views for the creation of the DFD system model. The first view is the graphical model editor, displayed in Figure 5.12, which allows graphically modeling the system under analysis. All changes made in the graphical editor are reflected in an underlying model, shown earlier in Figure 5.3b. This tree view can used for more advanced editing operations such as loading in a different threat type catalog.

After or even during the creation of the model, it can be analyzed by SPARTA for security and privacy threats. This analysis happens completely in the background, and does not require any additional inputs or feedback from the threat modeler.

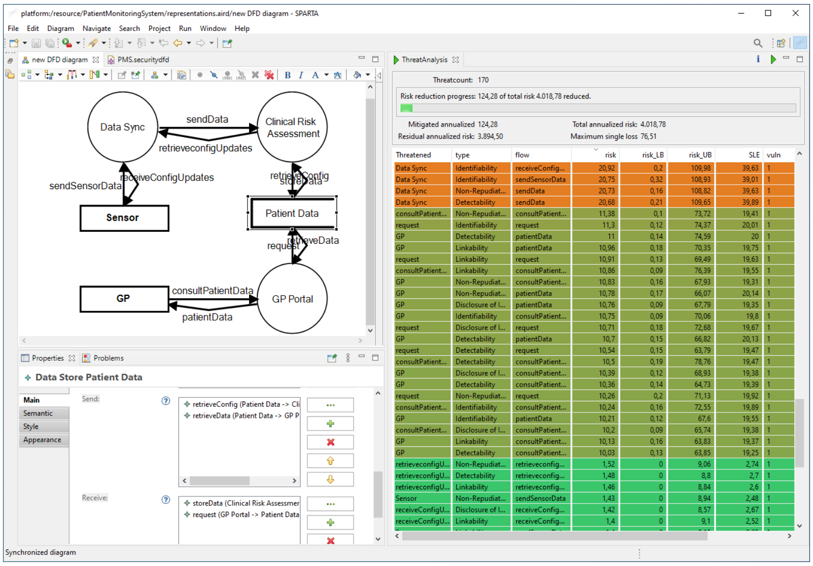
SPARTA presents the results of threat analysis in a table of threats in the threat analysis view (right-hand side of Figure 5.12). This view provides the full details on the elicited threats and will be automatically updated as changes are being made to the underlying model. The threats in this table are color-coded according to their current risk score. The scale ranges from 0 to the highest potential risk, i.e. the highest risk given no countermeasures are present. For each elicited threat, SPARTA included some additional information from the risk analysis: (i) the current risk (and upper and lower bounds), (ii) the damage of a single loss event, (iii) the vulnerability (to assess the effectiveness of the countermeasures). Besides the threat list, SPARTA also provides an overview of the total risk reduction progress. This supports the monitoring and keeping track of the global aggregated risk value when adding or modifying countermeasures. The SPARTA tool performs the analysis of the user model continuously, updating the resulting threat list on-the-fly as changes to the user model are made.

5.3 Evaluation and Discussion

This section first revisits the objectives from the introduction with a qualitative assessment of how they are realized in the SPARTA prototype. This is followed by the validation of the implemented risk model. Finally, a discussion is provided on the components in SPARTA and how its approach can be extended and leveraged in other types of analysis activities.

5.3.1 Assessment of the Objectives

Automation. After construction of the solution-enriched DFD model, SPARTA supports fully automated security and privacy threat elicitation and prioritization.



Data Store Patient Data

Send:
- retrieveConfig (Patient Data -> Cli
- retrieveData (Patient Data -> GP P

Receive:
- storeData (Clinical Risk Assessmer
- request (GP Portal -> Patient Data

Synchronized diagram

Threatcount: 170

Risk reduction progress: 124,28 of total risk 4.018,78 reduced.

Mitigated annualized 124,28 Total annualized risk: 4.018,78
Residual annualized risk: 3.894,50 Maximum single loss 76,51

Threatened	type	flow	risk	risk_LB	risk_UB	SLE	vuln
Data Sync	Identifiability	receiveConfig...	20,92	0,2	109,98	39,63	1
Data Sync	Identifiability	sendSensorData	20,75	0,32	108,93	39,01	1
Data Sync	Non-Repudiat...	sendData	20,73	0,16	108,82	39,63	1
Data Sync	Detectability	sendData	20,68	0,21	109,65	39,89	1
consultPatient...	Non-Repudiat...	consultPatient...	11,38	0,1	73,72	19,41	1
request	Identifiability	request	11,3	0,12	74,37	20,01	1
GP	Detectability	patientData	11	0,14	74,59	20	1
GP	Linkability	patientData	10,96	0,18	70,35	19,75	1
request	Linkability	request	10,91	0,13	69,49	19,63	1
consultPatient...	Linkability	consultPatient...	10,86	0,09	76,39	19,55	1
GP	Non-Repudiat...	consultPatient...	10,83	0,16	67,93	19,31	1
GP	Non-Repudiat...	patientData	10,78	0,17	66,07	20,14	1
GP	Disclosure of I...	patientData	10,76	0,09	67,79	19,35	1
GP	Identifiability	consultPatient...	10,75	0,09	70,06	19,8	1
request	Disclosure of I...	request	10,71	0,18	72,68	19,67	1
GP	Detectability	patientData	10,7	0,15	66,82	20,13	1
request	Detectability	request	10,54	0,15	63,79	19,47	1
consultPatient...	Detectability	consultPatient...	10,5	0,19	78,76	19,47	1
GP	Disclosure of I...	consultPatient...	10,39	0,12	68,93	19,38	1
GP	Detectability	consultPatient...	10,36	0,14	64,73	19,39	1
request	Non-Repudiat...	request	10,26	0,2	71,13	19,92	1
consultPatient...	Identifiability	consultPatient...	10,24	0,16	72,55	19,89	1
GP	Identifiability	patientData	10,21	0,12	67,6	19,55	1
consultPatient...	Disclosure of I...	consultPatient...	10,2	0,09	65,74	19,38	1
GP	Linkability	consultPatient...	10,13	0,16	63,83	19,37	1
GP	Detectability	consultPatient...	10,03	0,13	63,85	19,25	1
retrieveconfigU...	Non-Repudiat...	retrieveconfig...	1,52	0	9,06	2,74	1
retrieveconfigU...	Detectability	retrieveconfig...	1,48	0	8,8	2,7	1
retrieveconfigU...	Linkability	retrieveconfig...	1,46	0	8,84	2,6	1
Sensor	Non-Repudiat...	sendSensorData	1,43	0	8,94	2,48	1
receiveConfigU...	Disclosure of I...	receiveConfig...	1,42	0	8,57	2,67	1
receiveConfigU...	Linkability	receiveConfig...	1,4	0	9,1	2,52	1

Figure 5.12: SPARTA Prototype Screenshot
The above screenshot shows the model editor next to the threat analysis view. This view shows the overall risk reduction progress at the top, followed by the ordered elicited threat list based on the risk analysis of the identified threats.

Fine-grained threat applicability. Contrary to existing security and privacy threat elicitation approaches, the applicability of a security or privacy threat types is no longer a binary notion. SPARTA elicits every theoretically possible security or privacy threat. While the final risk score is used the prioritize the elicited threats, the intermediary *vulnerability* value provides a granular indication of a threat's applicability between [0, 1] by taking into account the contextual information on security and privacy solutions. This avoids an elicitation step where threats that were deemed inapplicable are eliminated and 'forgotten'. By still including these threats with the appropriate risk scores, SPARTA ensures that they are reconsidered when the assumptions that would have caused their elimination change over time. For example, a partially mitigated threat can become relevant again when the value of the threatened assets increases later on.

Reuse of security and privacy knowledge. The post-processing of elicited threats to determine their applicability and priority is a process that involves making architectural security and privacy decisions. Performing this activity as a separate isolated activity after the threat elicitation causes that knowledge to be separately, if at all, recorded and potentially out of sync with the system design and its corresponding threats. The integration of threat elicitation and assessment, paired with the approach of continuously re-eliciting and re-assessing threats, forces the collection of these architectural design decisions into the supporting models. This ensures that the elicited security and privacy threats and their corresponding risks are consistent with the decisions.

Immediate feedback. The SPARTA prototype provides immediate modeling feedback to the end-user by re-assessing changes to the model and automatically updating the corresponding security and privacy threat list. This allows the end-user to perform what-if analyses by evaluating the impact of different design alternatives, through modeling the changes and verifying the impact on the resulting threat list.

Measuring progress. The risk integrations in SPARTA provide two main benefits. First, the integrated risk assessment enables the aggregation of the threat-specific risk scores in order to track the overall risk reduction process across multiple design revisions. This provides a useful metric to assist in choosing the most appropriate security and privacy solution to efficiently reduce the risk. Second, the support for fine-grained risk calculation enables a number of different analysis scenarios where the risk can be aggregated across different dimensions such as the attacker profiles, the element types, specific threats, etc. to gain insight into the risk distribution for these parameters.

5.3.2 Validation of the Risk Model

The evaluate the practical feasibility of a continuous threat specific risk assessment using Monte Carlo simulations to calculate the risk, we conducted an initial performance evaluation of the SPARTA prototype by running a security threat and risk analysis on the WebRTC [Goo17] reference architecture as presented in Figure 3.1. The DFD of this reference architecture contains 42 DFD elements and triggers 194 threats (when using STRIDE per interaction).

The total analysis time in SPARTA is 3.35 s averaged over 10 runs. This total analysis time includes loading the model (including threat type catalog and solution catalog), the query engine, performing the threat elicitation, the risk analysis, and presenting the results to the users. Running the threat elicitation and risk analysis in isolation results for 100 runs results in an analysis time of 842 ms (95% CI: 718 ms–966 ms). These results were obtained with a distribution sampling size of 2000 from an implementation without any performance optimizations.

The qualitative evaluation of the modeling support for security solutions in SPARTA was already covered in Section 3.4 with a comparison of the property-based solution representations (such as the Microsoft Threat Modeling Tool 2016 [Mic16]), indicating positive improvements in terms of semantic quality, traceability, separation of concerns and dynamism [SYVJ18c]. The evaluation of the risk-based prioritization for the security threats was discussed in Section 4.3, where the prioritization was applied on the open source SecureDrop whistleblower submission system [Fre18a], showing correspondences between provided security countermeasures by the developers and the associated risk score assigned by SPARTA [SYVJ18b].

5.3.3 Discussion

This section briefly discusses the SPARTA prototype in two main parts. The first three sections discuss the use of SPARTA with the estimates for the prioritization, the possibility to extend SPARTA with different prioritization methods, and the visualization of the analysis results. The last three sections cover more elaborate extensions: leveraging the framework in the context of data protection by design, reconstructing the DFD models from code, and the visualization of security and privacy solutions in the graphical models.

Providing estimates for the prioritization

As mentioned in the discussion on the risk model in Section 4.3.4, the risk-driven threat prioritization requires the specification of a number of input values in the model. This involves an effort trade-off between providing additional information in the input model versus requiring additional effort afterwards in prioritizing the identified threats. There are number of opportunities to reduce the up-front overhead of introducing additional information: (i) specifying this information up-front enables its reuse across multiple analysis activities, while changes in the resulting threat list makes it harder to reuse these results; (ii) the input for the different security and privacy solution strengths and attacker profiles only have to be provided once and can be re-used across analyses and applications; and (iii) to reduce the effort in assigning the values of the assets, simple schemes can be used to assign values to the elements as applied in the evaluation in Section 4.3.2. Tool support such as SPARTA can assist the user with assigning values on the model elements.

Different Threat Prioritization Methods

The current implementation of the threat prioritization in SPARTA relies on the extended FAIR risk assessment model presented in Chapter 4. However, the elicitation and risk assessment logic in SPARTA is not specific for the presented risk assessment model. The SPARTA threat types could be extended to use a different risk assessment model for prioritization. The only constraint SPARTA enforces on this risk model is that it should be able to provide a sortable value to enable ranking the elicited threats in the threat results view.

Visualizing Threat Analysis Results

The results of the threat risk assessment in SPARTA are current presented as a sorted and color-coded list according to the magnitude of the threat's risk. However, as Figure 5.10 illustrated with heatmaps, a number of different graphical visualizations are possible of these risk results. The benefits of these alternative visualizations is that they enable the exploration of the impact of different parameters (e.g., the type of adversary or the type of data subject) on the distribution of the risk in the application. SPARTA is currently limited to the color-coded list, but future extensions could support provide more dynamic visualizations to explore this risk space interactively.

Reconstructing Models from Code

Any analysis of a software system in SPARTA requires the construction of that application's DFD model in SPARTA. This requires both initial effort in constructing the model and modeling effort to ensure the model remains consistent with the source code of the system.

To reduce the up-front modeling effort in constructing these models, software architecture reconstruction techniques [DP09] could be applied to enable the reconstruction of a DFD-based representation of the system. Initial attempts to aid the construction of these model can rely on source-code based annotations, such as used in ThreatSpec [Thr19]. These annotations still require the manual effort of constructing the model but, because of their close proximity to the code, can simplify the effort in maintaining a model that is consistent with the code.

Visualizing Solutions

Finally, in addition to the semantic support for representing security and privacy solutions which enables tool support to leverage this information in subsequent analyses, is the graphical syntax of these solutions for communicating this information to the human-end users. Constructing graphical representations for solutions involves a number of challenges to find appropriate visualizations that meet the different criteria for effective visual notations [Moo09] such as: one-to-one mapping between concepts and graphical representations, semantically transparent representations that suggest their meaning, etc.

5.4 Related Work

Threat modeling was introduced by Microsoft as part of its security development life cycle [SS04, Tor05, HLOS06, HL06, Sho08] and has proven popular since with multiple real-world applications in industry [Tor05, Sho08, Dhi11]. In this section, we revisit the state of existing threat modeling tool support to discuss to which degree the existing threat modeling tools support the solution-aware and risk-driven prioritization of security and privacy threats.

5.4.1 Threat Elicitation Tool Support

The Microsoft Threat Modeling Tool [Mic16, Mic20] is the most common and readily available tool that can automatically elicit security threats based on the data flow diagram drawn in the tool. The effect of security properties in the diagram is binary. They influence whether a threat is generated or not. There is no way to partially take into account the effect of solutions for assessing the severity of the elicited threats. Any prioritization in the resulting threat list has to be performed manually by the user. Another publicly available tool is OWASP's ThreatDragon [OWA18]. ThreatDragon does not allow any automatic threat elicitation. The identification and subsequent prioritization of threats has to be performed manually. The Irius Risk tool from Continuum Security [Con18] does support the automatic elicitation of threats based on the information provided about the components (e.g., used technological components, sensitivity of the data). The reduction in the risk of the threats elicited by Irius Risk is not derived from the combined strength of the countermeasures and the capability of an attacker profile. Instead, the threat specification (risk pattern in Irius Risk terminology) contains a list of the relevant countermeasures with the percentage reduction they provide. The next threat modeling tool is OVVL [Res19, SR19], which supports the elicitation of threats. There is, however, no support for prioritizing them. This again has to be manually performed by the user. Finally, there is the SecuriCAD tool from Foreseeti [For20]. It runs attack simulations on a custom model. The results of these attack simulations are presented in a prioritized list according to the amount of affected high-value assets. Because of the current lack of tool support for privacy threat elicitation, there is also no automated support to prioritize the elicited privacy threats.

5.4.2 Other Threat Modeling Approaches

In addition to these DFD-based threat elicitation tools, a number of different threat modeling tools are relevant in this context. The next two tools are both more focused on code, either as annotations or by representing the models in code. The third tool comes from a security and privacy requirements context and has been extended into threat modeling.

The first tool is ThreatSpec [Thr19], which provides a set of source code annotations that have to be applied by the developers in code comments. ThreatSpec analyzes the application source code with these annotations in order to extract a DFD, a list of threats, and a list of countermeasures. This approach does not support automated analysis of the source code for eliciting threats, as these have to be manually added; it does, however, enable the specification of this security information close to the threatened code or provided countermeasures.

The second tool is PyTM [Tar20]. PyTM does not use source code annotations but, instead, represents the DFD model as code, to enable this information to be stored with the source code as well. PyTM does support the elicitation of threats based on the model.

CAIRIS [Fai18] is the tool that accompanies the IRIS (Integrating Requirements and Information Security) approach. CAIRIS focuses mainly on usability and security requirements, but has recently been extended with support for DFDs. It does not directly model these with a graphical DFD editor, but, similarly to ThreatSpec and PyTM, generates a graphical view of the DFD. CAIRIS does not automatically elicit threats, but they can be manually documented.

5.4.3 Risk Assessment and Prioritization

In addition to the threat prioritization in the context of threat modeling, there are also separate risk analysis approaches changethat are more decoupled from threat modeling activities. These approaches, however, lack any integration with threat modeling activities: (i) they do not use the same concepts at the same abstraction level, which can require multiple translation steps to reuse the elicited threats in the risk assessment context; (ii) vice versa, the results of the risk assessment may not translate back to enable the prioritization of the elicited threats; (iii) they require considerable re-assessment effort when the threat results or the system change.

The first risk analysis framework is CORAS [LSS10]. The CORAS approach relies on a number of graphical models of the asset, threat, risk, and treatment diagrams to support a risk analysis exercise, which involves a number of analysis activities: preparation, determining the target and scope, refining the target and assets, risk identification, risk estimation, risk evaluation, and risk treatment. The construction of the different diagrams to create an overview of threat (agents), scenarios, vulnerabilities, and unwanted incidents is similar to the creation of attack trees but instead of the primary node being the attack to realize, it is more asset-centered by focusing on the unwanted incidents that harm the assets initially the determined. The CORAS approach focuses mainly on analyses of 150 to 300 man-hours. This makes the approach unsuitable for lightweight and continuous re-assessment in the context threat modeling activities.

There are a number of tree-based risk analysis techniques from safety engineering which are closely related. A common technique and international standard in the safety domain is Fault Tree Analysis (FTA) [IEC06], which relies on the construction of a fault tree. The root of the tree is the failure that has to be prevented. the tree is constructed using multiple logic gates that represent how combinations of faults in different systems can lead to the undesired event at the root. Another related technique is Failure Mode and Effects Analysis (FMEA), which performs in the analysis in the opposite direction by investigating the impact of the failure of single component. The difference between the above two approaches is that the first one is deductive (starting from a failure to find out which elements most likely contributed to that failure) versus inductive (assuming a failed element to determine its effect on the system) [VGRH81].

For assessing privacy risk, the PRIAM [DL16] methodology can be used. Similar to the attack trees, PRIAM employs harm trees to assess the risk. A harm tree has a specific harm as root node, followed by a number of feared events combined through AND/OR nodes, and, finally, at the bottom weaknesses and risk sources. As is the case for CORAS, the manual construction of the harm trees hinders the continuous re-assessment in the context of privacy threat modeling. Our approach can be considered an instantiation of this approach, but with an explicit assignment of numerical estimates for the risk factors to support the completely automated assessment.

Hong et al. [HNLL04] developed a privacy risk model for ubiquitous computing systems, focusing on the selective disclosure of personal information (*personal privacy*). Similar to the IRS Privacy Impact Assessment (PIA) [Int96], it relies on a set of questions as part of the privacy risk analysis, after which the identified risks are prioritized.

Abu-Nimeh and Mead [AM09] propose the combination of privacy risk assessment with the IRS PIA [Int96] in Security Quality Requirements Engineering (SQUARE) [MS05]. While such a PIA [Int96] supports a detailed assessment of the realization of privacy-by-policy in the framework of Spiekermann and Cranor [SC09], its question-based assessment is less suited for frequent automated re-assessment.

5.5 Conclusion

There is a dichotomy between threat modeling approaches to identify security and privacy threats and risk analysis approaches to support triaging and prioritizing the identified threats. The makes the results from the risk assessment quickly outdated as the identified threats may be irrelevant due to changes to the system while other newly identified threats lack a risk score to prioritize them.

The SPARTA tool resolves this disconnect by combining these two analysis activities in a single framework, enabling the integration of the threat elicitation and risk assessment activities. This integration enables the risk assessment to use the risk input information directly from the relevant concrete elements in the DFD model, supporting the construction of the prioritized list of threats directly from the model data.

The approach embodied in SPARTA provides a non-binary assessment of a threat's applicability, which corresponds more closely with the real-world context of the system under design in which nothing is perfectly secure but many countermeasures do aid in reducing the risk of security threats. Additionally, the integration of this information in the threat list supports a number of threat management activities such as keeping track of the overall risk reduction process.

In future extensions, the integration of security and privacy solution catalogs with existing vulnerability and weaknesses resources can support automated updates of the effectiveness of these solutions to dynamically reassessing existing systems as new information on the effectiveness of existing solution is released, thereby taking an important step towards continuous security and privacy assessment.

Chapter 6 Outline

6 Conclusion

"The only people who see the whole picture," he murmured, "are the ones who step out of the frame."

— Salman Rushdie [Rus99]
The Ground Beneath Her Feet

This chapter concludes the dissertation with a summary of contributions presented in the previous chapters. This is followed by a discussion of the applicability of the SPARTA approach and framework in other contexts such as data protection by design and a summary of the accomplished contributions in this complementary research area. After that, the opportunities for future work are presented. Finally, the chapter closes the dissertation with some concluding remarks.

6.1 Summary

Security and privacy are critical properties of today's software systems. While there is considerable effort and focus spent on the implementation-level security, this is not the case for design-level security. Design-level security and privacy flaws get considerably less attention despite their high impact and costly resolution. These flaws require solutions that should be incorporate at design time to address often re-occurring problems that demand design decisions and choices in an early stage of the software development process. A comprehensive assessment of the security and privacy of a system can therefore only be achieved by analyzing the security and privacy properties of the design.

This dissertation examines the application and extension of security and privacy threat modeling for the design-level analysis of software systems, which illustrated its potential with the multiple practical applications in industry [Sho08, Dhi11]. Threat modeling for design analysis is improved by extending modeling support for a richer system representation, expanding the elicitation analysis, and integrating the comprehensive risk assessment of the identified threats. Threat modelling can be applied as part of the requirements engineering stage, and can be re-applied when assessing the design of the software system to determine: (i) whether previously identified threats are properly mitigated, and (ii) whether new threats are introduced as a result of design decisions. This dissertation provided the following three contributions:

1. **Semantic support for representing security and privacy solutions.** We provide explicit modeling support for the first-class representation of security and privacy solutions in Data Flow Diagram (DFD) models. This enables: (i) the explicit representation and thus documentation of previously made security and privacy design decisions (traceability), (ii) an extensive representation of applied solutions taking into account their effects, but also their limitations in, for example, scope, (iii) separating the security and privacy solution concepts from system concepts, allowing for independent evolution (separation of concerns), and (iv) future dynamic updates to these solutions for scenarios such as continuous threat assessment and re-evaluation in light of changes in effectiveness of existing, already-applied, security solutions. Furthermore, the model representation provides the necessary inputs for the subsequent analysis and risk assessment activities.

2. **Supporting model-based security and privacy analysis activities.** To counteract the problems with manual threat elicitation, we provide an approach for automating the security and privacy threat elicitation using patterns, which allows further

extending the supporting security and privacy threat knowledge bases with more advanced threat types and design flaws that can be automatically detected. Furthermore, to avoid the manual prioritization effort commonly required in traditional threat elicitation approaches, an integrated security and privacy threat risk assessment model is proposed. This risk assessment model leverages the security and privacy information embedded in the supporting DFD model, reusing this information in the risk analysis and supporting a wide range of risk analysis activities.

3. **Tool support implementing the above two contributions in a threat modeling framework.** We presented SPARTA: Security and Privacy Architecture through Risk-driven Threat Assessment, a prototype implementation of the threat modeling framework that integrates the modeling of security and privacy solutions, and the automated threat elicitation and risk assessment to support comprehensive and automated design analysis for security and privacy.

6.2 Applicability

This section revisits the contributions presented earlier and discusses their relevance and applicability beyond the immediate context of security and privacy threat modeling.

Applying the security and privacy solution extension mechanisms to other model representations

Chapter 3 presented modeling support for representing security and privacy solutions in DFD models. While the extensions are applied and illustrated in the context of DFDs, they do not have strong dependencies on DFD concepts. Technically, the only dependency on DFDs is used in the ability to limit the binding of a security and privacy solution's roles to DFD elements of a specific type. This mechanism is very generic and can be reused when leveraging the solution representation support in other modeling languages. For example, instead of scoping a solution's roles to, for example, DFD processes, these limitations can also be applied in, for example, a Unified Modeling Language (UML) modeling context by limiting bindings to UML element types (e.g., components, classes, lifelines). Combining the solution support with appropriate threat type knowledge would enable the use of these other modeling languages for threat elicitation.

Threat prioritization alternatives

The threat prioritization presented in Section 4.2.1 leveraged the extended FAIR-based risk model. The approach embodied in SPARTA supports the application of other threat prioritization approaches. Every elicited security or privacy threat provides the threat type and includes a reference to the relevant DFD elements indicating where the threat occurs. This information can be used by other threat prioritization approaches to determine a score to sort the elicited threats.

Leveraging the SPARTA framework for other analyses

Generically, SPARTA provides a compelling framework for analyzing problems in different application domains by: (i) constructing a model-based representation; (ii) analyzing this model for a number of problematic patterns; (iii) analyzing the identified problems to assess their severity; and (iv) presenting the results. Indeed, a similar approach is useful in the context of data protection by design where the data processing operations can be modeled [SDVL+19, SDVL+] and analyzed to assist in performing data protection impact assessments.

We have successfully leveraged the SPARTA approach in the context of data protection by design [SDVL+19, SDVL+]. To enable data protection analyses in support of meeting the obligations imposed by the General Data Protection Regulation (GDPR), we provided modeling support for creating data protection models. These data protection models represent the data processing operations in the system using relevant concepts from the GDPR such as: collection, processing, controllership, data subject type, personal data type, etc. These concepts provide the necessary information for analyzing the data processing operations from a legal compliance perspective.

The analysis of the data protection models involves the automated application of a number of legal assessments to identify problematic data processing operations that are incompatible with the GDPR. The identified problematic situations can subsequently be analyzed to prioritize them and present them to the user to resolve them.

6.3 Future work

The exploration and development of the contributions in this dissertation are fertile ground for new research ideas and opportunities. This section highlights a subset of these opportunities.

Empirical studies

Advanced threat modeling tool enables a number of complex automated analyses, but it comes at the cost of some additional modeling effort such as the instantiation of security and privacy solutions and the assigning the values for the risk-driven threat prioritization. With the SPARTA tool, these cost-benefit trade-offs can be evaluated: (i) the additional effort for adding the risk inputs to the models versus the manual prioritization of the threats afterwards; (ii) whether users correctly create and instantiate solutions and what the impact is of incorrect solutions and instantiations on the resulting threats; and (iii) how users make the trade-off between alternative solutions.

Structuring Threat Knowledge

The current interaction-based security and privacy threat elicitation approach as described in Chapters 3 and 4 relies on a practical representation of the STRIDE and LINDDUN threat knowledge. There is, however, a lot of information in these threat knowledge bases that is currently not used in threat elicitation approaches because this information is not available in a structured and reusable form. Future extensions would involve the construction of a meta-model for representing security and privacy threat knowledge, including the relations between different threat types. By capturing such knowledge in a structured and reusable format, threat elicitation approaches can leverage this knowledge for more advanced and defense-in-depth analysis scenarios such as evaluating the impact of existing threats on other threats (i.e. how combinations of different threats can be used to realize other threats deeper down in the system). For example, a spoofing threat when accessing the system can subsequently lead to an information disclosure threat. The information disclosed in this threat may then again be used to spoof a different user. By spoofing this user, some other information can be modified, thereby realizing another tampering threat. Reasoning about these chains of causation requires modeling the relations between the different threat types.

Visualization

Visualization in the context of threat modeling has two main dimensions. First, the communication of information on security and privacy solutions to the user and providing an effective visual representation for this information. Second, the communication of the analysis results and enabling the end-user to navigate and interpret the results.

Visual Syntax for the Model Extensions. Besides the tool support used for analyzing the system DFD models, human designers are also users of these models. To improve the communication of the relevant information on the system's security and privacy solutions, different graphical representations of this information can be explored in order to discover the most optimal representations for communicating this information to designers. These representations are complementary to the semantic representation of these solutions in the underlying models as presented in Chapter 3.

Visual Representations of the Analysis Results. Not only the representation of the system and its security and privacy solutions need to be communicated to human designers. The results of the security and privacy analyses also have to be communicated and visualized to the analysts to enable them to act on this information. Different visualizations of this analysis information can be explored such as the risk heatmaps that were used as an illustration in Figure 5.10 to evaluate which visualizations of analysis results are the most effective. Furthermore, interactive visualizations could support a user in exploring the impact of different parameters (e.g., attacker strength, data type sensitivity). These visualizations can aid in the decision-making.

Decision Trade-Off Analysis Support

While the contributions presented in the previous chapters support the evaluation of the impact of instantiating different countermeasures in a design by the continuous re-assessment of the threats and their corresponding risks, the exploration of all the potential alternative solutions for mitigating the identified threats represents a very large solution space. This problem of identifying appropriate solutions can be considered as a search problem that can be addressed by systematically generating and evaluating different design variants to explore and navigate this design space. Future extensions can leverage the existing SPARTA tool support infrastructure to evaluate these generated variants and guide the user in making the trade-off decisions between different suggested design alternatives.

Source Code as Input to Threat Modeling

In order to uncover design-level structural flaws in a system, a design representation of the system is required. While user-friendly tool support can provide considerable assistance in the construction of these design-level representations, they do not solve the problem of a potential disconnect, or drift over time, of the design representation of the system and its actual structure as expressed in the source code. To resolve this problem, a number of different approaches can

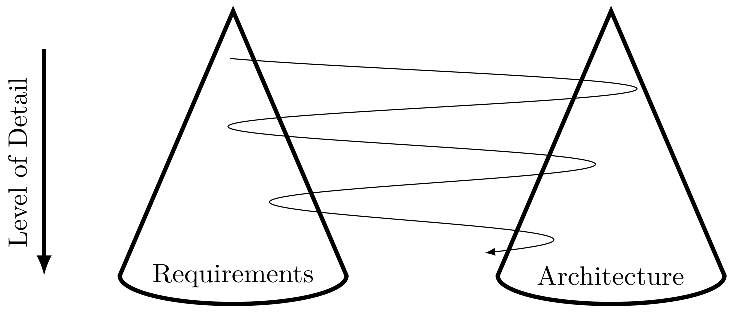


Figure 6.1: Twin Peaks Model
Adapted view of the Twin Peaks model from Nuseibeh [Nus01] which shows the iterative and progressive increase in detail in both requirements and architecture.

be explored ranging from code-level annotations [Thr19] to ensure consistency between the design and the implementation, towards more automated design reconstruction techniques that reverse engineer a design representation of the system based on its implementation. These approaches can significantly reduce the initial effort in constructing the design representation and provide opportunities for automation in the context of continuous integration by enabling automated analysis.

6.4 Concluding remarks

We close the dissertation with a number of concluding remarks. Chapters 3 to 5 presented three contributions in advancement of the modeling of systems, the analysis and prioritization of security and privacy threats, and tool support that implements these in a security and privacy threat modeling framework.

Revisiting Twin Peaks We started this dissertation with a discussion on security and privacy by design and the application of these security and privacy analysis techniques in the early phases of the development lifecycle, aligning with the Twin Peaks [Nus01, HYS$^+$11] model. Indeed, the presented design analysis techniques fit into the iterative process of co-developing requirements and architecture. The initial high-level design description can be analyzed early on to identify security and privacy threats. These threats provide relevant information to feed back into the requirements process as they can be interpreted as requirements formulating what is not allowed to happen. These can be translated into security or privacy objectives. The analysis of the identified security and privacy threats can then, in turn, lead to modifications and further refinement of the overall system design and architecture (Figure 6.1).

Case Studies Throughout the dissertation, we have used a number of case studies to illustrate, validate, and evaluate the contributions. Next, we discuss a number of observations drawn from those cases.

First, there is very limited public documentation on the security and privacy design aspects of software systems, due to the inherent sensitive nature of such documentation. This makes it very challenging to rely on a common baseline or ground truth (i.e. a thorough and well-documented threat analysis which is carried by consensus among experts) of existing software systems to use in evaluation of security and privacy analysis techniques. In this regard, the thorough and detailed documentation from the SecureDrop project is very welcome. In future work, it would be particularly beneficial to deliberately create exemplars and repositories of threat models to assist in the evaluation of security and privacy analysis techniques with a common baseline.

Second, most solutions and assumptions encountered in the documentation of the applications were straightforward to model in the DFD with the solution extension. There were some solutions, which are harder to capture because the DFD notation itself provides limited support. Examples of this are the types of solutions that rely on functionality provided by the underlying OS, and how that OS interacts with other processes in the system. These solutions are hard to capture because DFDs do not offer support to model the dependencies between these different layers. More specifically, while a solution can capture the effect (e.g., logging for some process), it does not capture the details of how that mechanism is realized (e.g., a host intrusion detect client that uses OS logging functionality for monitoring some other process).

Third, the reuse of catalogs can greatly reduce the modeling efforts. The threat type catalogs were created once and reused across all the case studies. The catalogs on security or privacy solutions were not elaborate or complete. As a result, a significant part of the modeling efforts is spent on creating every newly encountered solution the catalog. With more elaborate catalogs on security and privacy solutions, those modeling efforts can be significantly reduced.

Fourth, a simple value assignment strategy was used for assigning the Asset Values (AVs) in the SecureDrop model. This scheme reduces the effort of determining the values for every element in the model.

Fifth, while the elicitation and prioritization of security and privacy threats can be fully automated with SPARTA, the construction of the initial DFD remains a manual task. The available of design documentation simplifies the creation of the model. If such documentation is not available, the creation of the model requires significant investment in exploring the code and trying to reconstruct the design to model.

Finally, while the graphical representation of DFDs is simple and straightforward. There is no single format for exchanging these models. To improve the reuse of case studies across multiple threat modeling tools, a standard exchange format for DFD models is instrumental.

Documentation The presented DFD modeling extensions requires some additional effort from the threat modeler to capture the security and privacy solutions in the DFD models. However, the return of this modeling effort can be improved, beyond the immediate benefits in a threat modeling context highlighted in this dissertation, as the captured information can serve multiple purposes. For example, the GDPR imposes a number of obligations on organizations that process personal data. Among those obligations are: (i) keeping track of which types of personal data is processed and for which purpose(s), and (ii) being able to provide appropriate documentation to national supervisory authorities on the types of measures taken to protect the personal data. Documentation on the security and privacy solutions and the processed personal data types in the DFD models can thus assist in meeting those obligations of the GDPR.

Automation The manual application of threat elicitation and analysis techniques is effort-intensive and error-prone. Therefore, such a manual analysis does not align well with contemporary iterative development practices as it makes frequent re-assessments very costly.

To resolve these problems, this dissertation has presented three contributions to support the automated application of security and privacy threat elicitation and analysis techniques: (i) the modeling support ensures security and privacy design decisions are recorded in the models and consistently documented and this information is leveraged in the threat analysis activities later on; (ii) the threat elicitation is extended by further concretizing the system context that is considered when determining a threat's applicability and leveraging tool support to avoid manual elicitation errors; and (iii) the integrated support for risk-driven threat prioritization based on information captured in the models to ensure the results can be explained from the inputs and updated when the assumptions change. All these contributions are implemented in tool support to enable automated and frequent re-assessment of the models and provide a solid basis for further extensions in populating the models from code, solution trade-off analysis, etc. towards even more integrated design analysis activities in better alignment with contemporary continuous integration and deployment practices.

Appendix A Outline

A

Application Case Descriptions

This appendix provides a more detailed description of the different case studies used throughout this dissertation.

A.1 SecureDrop

SecureDrop is an open source application developed by the Freedom of the Press Foundation to support news organizations in enabling whistleblowers to anonymously submit documents to the journalists. The application is run by 35 news organizations worldwide.

The description of the application case starts with an overview of the application as provided by the developers, followed by a detailed description of how this case was modeled in SPARTA for the evaluation.

We start the case description with an overview of a SecureDrop deployment provided by the developers [Fre18a, Fre18b].

The modeling of the SecureDrop application in SPARTA is presented in the following parts: (i) construction of the Data Flow Diagram (DFD) from the existing diagram depicted in Figure A.1, (ii) modeling the different security countermeasures as solutions, (iii) modeling the security-relevant assumptions of the developers as solutions, and (iv) assigning element values in support of the prioritization.

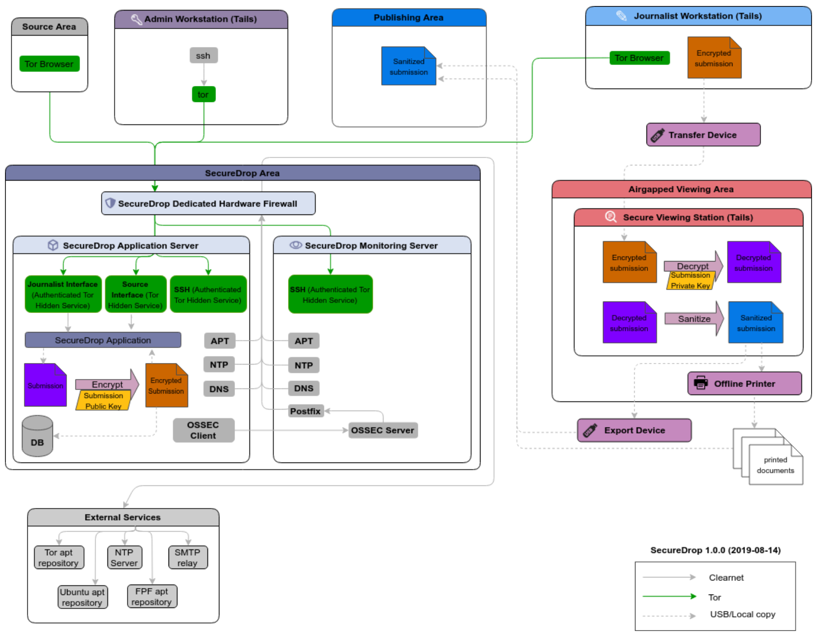


Figure A.1: SecureDrop Diagram
This diagram shows the DFD *from the SecureDrop documentation [Fre18b] and serves as the primary input for the construction of the* DFD *in* SPARTA.

A.1.1 Data Flow Diagram

Modeling the SecureDrop application involves from the different element types to the set DFD model elements. This mapping of the elements is relatively straightforward. Figure A.1 shows the resulting DFD.

- every type of application, process, or OS service is mapped to DFD processes;
- every type of storage (e.g., file, database, USB drive) is mapped to DFD data stores;
- every external Internet service (e.g., application repositories, mail servers) are mapped to external entities;

- the areas and physical devices are represented as DFD trust boundaries, and
- the human end-users and external services (e.g., apt repositories) are added as external entities.

A.1.2 Solutions

This section describes the list of security solutions that were modeled in the SecureDrop case study. Some of these solutions are instantiated multiple times in the system.

Tor Communication of the source, journalist, and administrator happens over the Tor network.
Signed APT updates APT update services uses PGP signatures.
Encrypted Storage device The storage device is encrypted.
Firewall Firewalls are applied for communications with the source (whistleblower), journalist, admin, monitoring, application services, monitoring services.
File encryption Database file encryption.
Sumbission encryption The submissions are encrypted.
Submission sanitization Submissions are sanitized on the secure viewing station of the journalist.
Session data signing Session data is signed to ensure its integrity.
XSS Protection XSS protection mechanisms.
Airgap Physical airgap for the viewing area.
Access control For source and journalist interfaces.

A.1.3 Assumptions

In addition to the solutions outlined above, the model also includes several assumptions that are explicitly documented by the developers in the threat modeling documentation. These assumptions are also modeled as '*solutions*' in order to represent their effect in excluding certain types of threats in certain locations of the system.

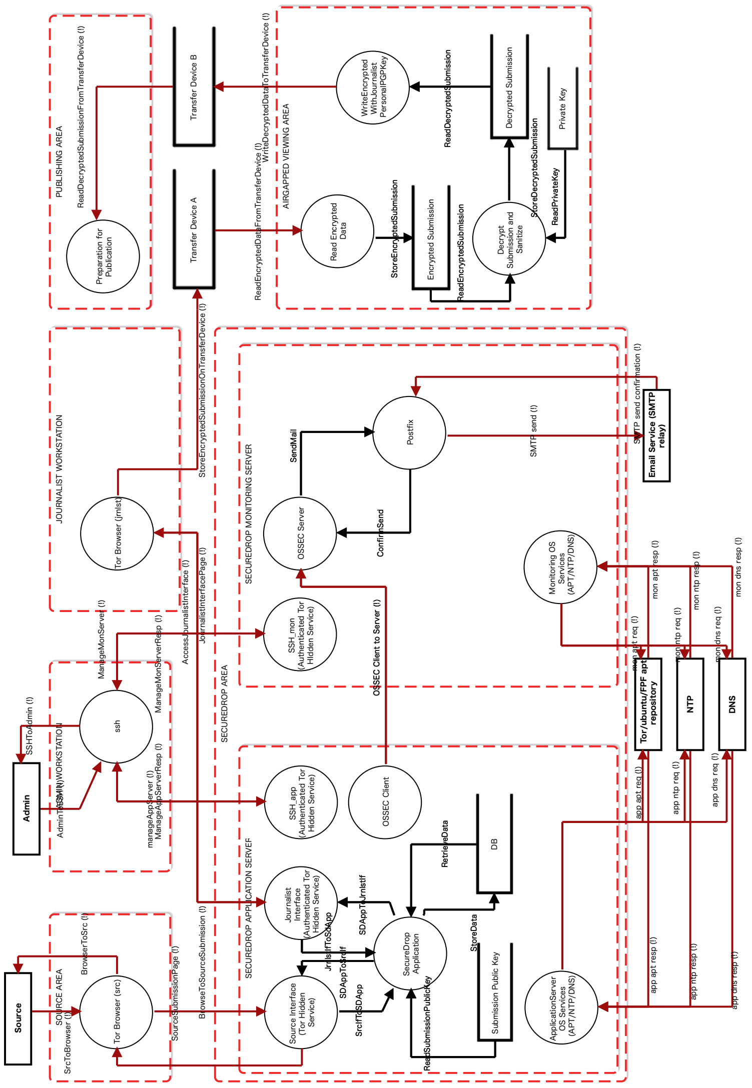


Figure A.2: SecureDrop DFD model
The model is largely based on the SecureDrop threat modeling document. Since the threat modeling document refers to an earlier version, any inconsistencies were resolved by referring to the latest version of the documentation and source code.

Authentication of the submission site Submission site URL is shown on a news organization site with authentication.

Source follows usage guidelines The source follows the usage guidelines when submitting documents.

Source repudiation not applicable Repudiation by the source is explicitly considered to be inapplicable.

SQLite DB repudiation not applicable Repudiation by the SQLite database is explicitly considered to be inapplicable.

Guidelines followed Administration, deployment setup, and isolation guidelines followed. The system is not compromised.

A.1.4 Assigned values for risk-driven prioritization

To support the prioritization of the identified threats on the SecureDrop model, the DFD elements are assigned values. The assigned values depend on the type of the involved users (whisltleblower, administrator, or journalist). A static value assignment scheme is used in the SecureDrop model:

- 3 if the data of a whistleblower is involved,
- 2 if the data of an administrator is involved, and
- 1 if the data of a journalist is involved.

These values can be combined when multiple user types are involved. The focus in this model is exclusively on the Asset Value (AV) for the prioritization of the security threats.

A.2 Patient Monitoring System

The Patient Monitoring System is an e-health system used for the monitoring of patients with cardiovascular diseases. These patients are equipped with wearable sensors and a smartphone application which synchronizes the sensor data with the back-end system. The back-end system stores the sensor data and performs a clinical risk assessment in order to identify high-risk patients that require further follow-up or medical intervention. The processed data and calculated clinical risk

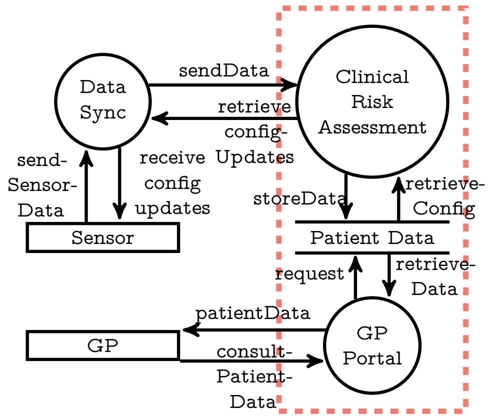


Figure A.3: Patient Monitoring System Data Flow Diagram
Simplified DFD representation a Patient Monitoring System. This model is further enriched with information on the data subject types and their personal data types.

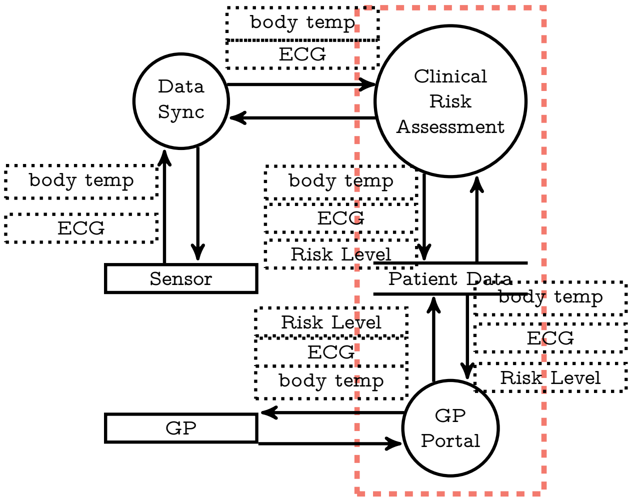


Figure A.4: Patient Monitoring System with Personal Data Types
This diagram extends DFD from Figure A.3 with the personal data types of the patient. The data from the General Practitioner (GP) is limited to the GP's credentials that are transferred to the portal to retrieve the patient data.

levels are made available to the patients' General Practitioners (GPs). Figure A.3 shows the DFD model a strongly simplified instance of the patient monitoring system. For more information on the context of this system, we refer the reader to the literature [DBOV+16].

A.2.1 Personal Data Types

The support the elicitation and prioritization of privacy threats, the DFD model of the Patient Monitoring System is extended with personal data types of the patient and the general practitioners. These data types are linked to the DFD elements which transfer or store the data. Figure A.4 visualizes which data flows transfer personal data of the patient. The data store *Patient Data* stores all the personal data types of the patient, but this is not visualized on the diagram in Figure A.4.

A.2.2 Assigned values for risk-driven prioritization

Given the limited set of personal data types and data subject types, each of them can be assigned a value individually and no assignment scheme is needed as used in the previous case study (Appendix A.1).

The following assignments were used for the personal data types:

- Body temperature measurement: 1
- Ecg measurement: 2
- Risk level: 3
- Gp Credentials: 2

The following assignments were used for the data subject types:

- Patient: 2.5
- Gp: 1.5

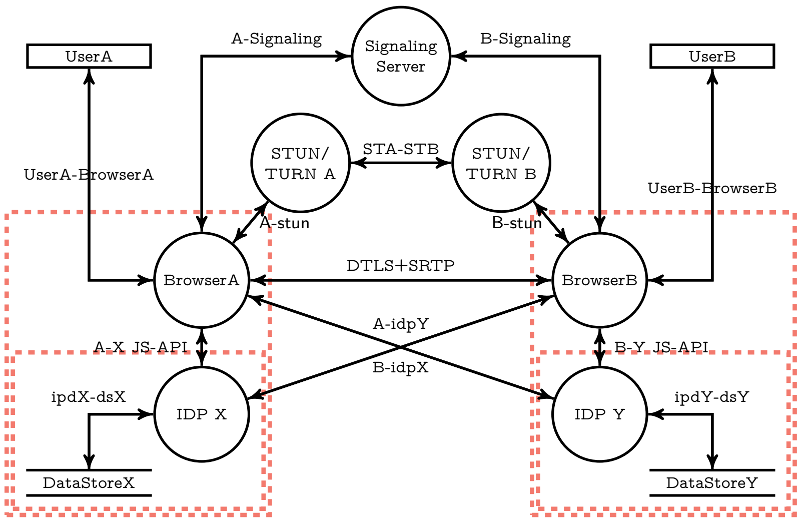


Figure A.5: WebRTC Data Flow Diagram
Simplified DFD representation of the WebRTC reference architecture [Goo17]. Bidirectional flows are modeled as two separate flows in opposite directions.

A.3 WebRTC

The third and final case study is the WebRTC case. The WebRTC case was developed in the context of the Pro-Flow research project. This model does not represent a concrete software application or product. Instead, a more generic model of the technology was constructed, which can be instantiated in concrete applications afterwards. This enables the up-front security threat analysis of this building block.

The WebRTC model provides a moderately sized DFD, consisting of: 7 processes, 2 external entities, 2 data stores, 28 data flows, and 4 trust boundaries (2 of them nested). The WebRTC DFD model was used to test the practical feasibility of the threat elicitation mechanisms and to demonstrate the limitations of the specialization-based representations of security solutions. The WebRTC model has already been reused as a case study for other DFD-based analysis techniques [TSB19].

A.3.1 Data Flow Diagram

The WebRTC DFD is highly symmetrical. Each side has: an external entity for the user, a browser process for setting up the connection and communicating with the other party, a STUN/TURN process to assist in traversing NAT gateways, an identity provider, and the identity provider's data store. The only shared component is the signaling server process to coordinate and assist in setting up the sessions between the browser processes. Figure A.5 shows the resulting DFD model.

Bibliography

[ABPW99] Christopher Alberts, Sandra Behrens, Richard Pethia, and William Wilson. Operationally critical threat, asset, and vulnerability evaluation (octave) framework, version 1.0. Technical Report CMU/SEI-99-TR-017, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1999.

[AC09] American Institute of Certified Public Accountants Inc. and Canadian Institute of Chartered Accountants. Generally Accepted Privacy Principles. Technical Report August, American Institute of Certified Public Accountants, Inc. and Canadian Institute of Chartered Accountants, 2009.

[ADD$^+$14] Iván Arce, Neil Daswani, Jim Delgrosso, Danny Dhillon, Christoph Kern, Tadayoshi Kohno, Carl Landwehr, Gary Mcgraw, Brook Schoenfield, Margo Seltzer, Diomidis Spinellis, Izar Tarandach, and Jacob West. Avoiding the Top 10 Software Security Design Flaws. Technical report, IEEE Center for Secure Design, 2014.

[ADSW03] Christopher Alberts, Audrey Dorofee, James Stevens, and Carol Woody. Introduction to the OCTAVE Approach. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

[AGI13] Mohamed Almorsy, John Grundy, and Amani S Ibrahim. Automated software architecture security risk analysis using formalized signatures. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 662–671. IEEE Press, 2013.

[AIS+77] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A pattern language: towns, buildings, construction.* Oxford University Press, 1977.

[AK03] Colin Atkinson and Thomas Kühne. Model-driven development: A metamodeling foundation. *IEEE Software*, 20(5):36–41, 2003.

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, jan 2004.

[AM09] Saeed Abu-Nimeh and Nancy R. Mead. Privacy risk assessment in privacy requirements engineering. *2009 2nd International Workshop on Requirements Engineering and Law, RELAW 2009*, pages 17–18, 2009.

[And08] Ross Anderson. *Security Engineering.* Wiley, 2nd edition, 2008.

[AS16] Majed Alshammari and Andrew Simpson. Towards a Principled Approach for Engineering Privacy by Design. 2016.

[AS18] Majed Alshammari and Andrew Simpson. A model-based approach to support privacy compliance. *Information & Computer Security*, 2018.

[ASRJ18] Amir Shayan Ahmadian, Daniel Strüber, Volker Riediger, and Jan Jürjens. Supporting privacy impact assessment by model-based privacy analysis. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, pages 1467–1474, New

York, NY, USA, 2018. Association for Computing Machinery.

[ASS16a] Thibaud Antignac, Riccardo Scandariato, and Gerardo Schneider. *A Privacy-Aware Conceptual Model for Handling Personal Data*, pages 942–957. Springer, Cham, 2016.

[ASS$^+$16b] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: Breaking TLS with SSLv2. In *25th USENIX Security Symposium*, August 2016.

[Bab64] Charles Babbage. Passages from the life of a philosopher, 1864.

[BCK12] Lenn Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Addison-Wesley, 2012.

[BDL06] David Basin, Jürgen Doser, and Torsten Lodderstedt. Model driven Security: From UML Models to Access Control Infrastructures. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1):39–91, 2006.

[BDLvV09] Muhammad Ali Babar, Torgeir Dingsøyr, Patricia Lago, and Hans van Vliet, editors. *Software Architecture Knowledge Management.* Springer Berlin Heidelberg, 2009.

[Bec12] Kristian Beckers. Comparing privacy requirements engineering approaches. *Proceedings - 2012 7th International Conference on Availability, Reliability and Security, ARES 2012*, pages 574–581, 2012.

[BLP09] Cédric Bouhours, Hervé Leblanc, and Christian Percebois. Bad smells in design and design patterns. *The Journal of Object Technology*, 8(3):43–63, 2009.

[Box79] George E.P. Box. Robustness in the Strategy of Scientific Model Building. In Launer, Robert L. and Wilkinson, Graham N.", editor, *Robustness in Statistics*, pages 201 – 236. Academic Press, 1979.

[BS91] K. Yusuf Billah and Robert H. Scanlan. Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks. *American Journal of Physics*, 59(2):118–124, February 1991.

[BSK13] Bernhard J. Berger, Karsten Sohr, and Rainer Koschke. Extracting and analyzing the implemented security architecture of business applications. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pages 285–294, 2013.

[BSK16] Bernhard J. Berger, Karsten Sohr, and Rainer Koschke. Automatically extracting threats from extended data flow diagrams. *Lecture Notes in Computer Science*, 9639:56–71, 2016.

[CAP18] CAPEC - Common Attack Pattern Enumeration and Classification. Available from MITRE, 2018.

[Cav06] Ann Cavoukian. Creation of a Global Privacy Standard. pages 1–4, 2006.

[CFH$^+$76] G. R. Craig, L. E. Frey, W. L. Hetrick, M. Lipow, E. C. Nelson, T. A. Thayer, J. A. Yoxtheimer, J. A. Whited, and R. B. White. Software Reliability Study. Technical report, TRW Defense & Space Systems Group, 1976.

[CHH16] M Colesky, J H Hoepman, and C Hillen. A Critical Analysis of Privacy Design Strategies. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 33–40, may 2016.

[Con18] Continuum Security. Irius Risk. https://community.iriusrisk.com, 2018.

[CVE14] CVE-2014-0160. Available from MITRE, 2014.

[CVE19a] CVE - Common Vulnerabilities and Exposures. Available from MITRE, 2019.

[CVE19b] CVE-2019-13450. Available from MITRE, 2019.

[CWE19] CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). Available from MITRE, 2019.

[CWE20] CWE - Common Weakness Enumeration. Available from MITRE, 2020.

[DBOV+16] Femke De Backere, Femke Ongenae, Frederic Vannieuwenborg, Jan Van Ooteghem, Pieter Duysburgh, Arne Jansen, Jeroen Hoebeke, Kim Wuyts, Jen Rossey, Floris Van den Abeele, et al. The OCareCloudS project: Toward organizing care through trusted cloud services. *Informatics for Health and Social Care*, 41(2):159–176, 2016.

[DeM79] Tom DeMarco. *Structured Analysis and System Specification.* Yourdon Press, 1979.

[DGFRLP04] Nelly Delessy-Gassant, Eduardo B Fernandez, Sajeed Rajput, and Maria M Larrondo-Petrie. Patterns for application firewalls. In *Proceedings of the Pattern Languages of Programs (PLoP) Conference*, 2004.

[Dhi11] Danny Dhillon. Developer-Driven Threat Modeling: Lessons Learned in the Trenches. *IEEE Security Privacy*, 9(4):41–47, jul 2011.

[DL16] Sourya Joyee De and Daniel Le Métayer. PRIAM: A privacy risk analysis methodology. *Lecture Notes in Computer Science*, pages 221–229, 2016.

[DP09] S. Ducasse and D. Pollet. Software architecture reconstruction: A process-oriented taxonomy. *IEEE Transactions on Software Engineering*, 35(4):573–591, July 2009.

[DWS+11] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1), 2011.

[DWS+19] Pierre Dewitte, Kim Wuyts, **Laurens Sion**, Dimitri Van Landuyt, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. A comparison of system description models for data protection by design. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1512–1515, New York, NY, USA, 2019. Association for Computing Machinery.

[Eur16] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016. *Official Journal of the European Union*, 59(L 119):1–88, may 2016.

[Fai18] Shamal Faily. *Designing Usable and Secure Software with IRIS and CAIRIS*. Springer, 2018.

[FB13] Eduardo Fernandez-Buglioni. *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.

[FBJ+16] Michael Felderer, Matthias Büchler, Martin Johns, Achim D Brucker, Ruth Breu, and Alexander Pretschner. Security Testing: A Survey. In *Advances in Computers*, volume 101, pages 1–51. Elsevier, 2016.

[Fir04] Donald Firesmith. Specifying reusable security requirements. *Journal of Object Technology*, 3(1):61–75, 2004.

[FJ14] Jack Freund and Jack Jones. *Measuring and managing information risk: a FAIR approach*. Butterworth-Heinemann, 2014.

[For20] Foreseeti. SecuriCAD. https://www.foreseeti.com/securicad/, 2020.

[Fra92] Robert B. France. Semantically Extended Data Flow Diagrams: A Formal Specification Tool. *IEEE Transactions on Software Engineering*, 18(4):329–346, 1992.

[Fre18a] Freedom of the Press Foundation. SecureDrop | The open-source whistleblower submission system, 2018.

[Fre18b] Freedom of the Press Foundation. Threat Model – SecureDrop 0.5.2 Documentation, 2018.

[FSS$^+$20] Shamal Faily, Riccardo Scandariato, Adam Shostack, **Laurens Sion**, and Duncan Ki-Aries. Contextualisation of data flow diagrams forsecurity analysis. In *The Seventh International Workshop on Graphical Models for Security (GraMSec)*, 2020.

[GGD11] Seda Gürses, Carmela Gonzalez Troncoso, and Claudia Diaz. Engineering privacy by design. In *Proceedings of the Conference on Computers, Privacy & Data Protection (CPDP 2011)*, page 25, 2011.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software Addison-Wesley.* Addison-Wesley, 1994.

[GJF06] Carlos M. Guitierrez, William Jeffrey, and Cita M. Furlani. FIPS PUB 200: Minimum Security Requirements for Federal Information and Information Systems. *FIPS Publication 200*, (March), 2006.

[Goo17] Google. WebRTC Project. https://webrtc.org/, November 2017.

[GPEM09] Joshua Garcia, Daniel Popescu, George Edwards, and Nenad Medvidovic. Toward a catalogue of architectural bad smells. In *International Conference on the Quality of Software Architectures*, pages 146–162. Springer, 2009.

[GS79] Christopher Gane and Trish Sarson. *Structured Systems Analysis: Tools and Techniques.* Prentice Hall Ptr, 1979.

[GTD15] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering Privacy by Design Reloaded. *Amsterdam Privacy Conference 2015*, pages 1–21, 2015.

[Hai05] Yacov Y Haimes. *Risk Modeling, Assessment, and Management*, volume 40. John Wiley & Sons, 2005.

[HH06] Rosa R Heckle and Stephen H Holden. Analytical tools for privacy risks: Assessing efficacy on vote verification technologies. In *Symposium On Usable Privacy and Security*, 2006.

[HL06] Michael Howard and Steve Lipner. *The Security Development Lifecycle*. Microsoft Press, 2006.

[HLMN08] Charles B. Haley, Robin Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008.

[HLOS06] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Threat Modeling: Uncover Security Design Flaws Using The STRIDE Approach. *MSDN Magazine*, 6, nov 2006.

[HNLL04] Jason I. Hong, Jennifer D. Ng, Scott Lederer, and James A. Landay. Privacy risk models for designing privacy-sensitive ubiquitous computing systems. *Proceedings of the 2004 conference on Designing interactive systems processes, practices, methods, and techniques - DIS '04*, page 91, 2004.

[HSS12] Bernhard Hoisl, Stefan Sobernig, and Mark Strembeck. Modeling and enforcing secure object flows in process-driven SOAs: an integrated model-driven approach. *Software & Systems Modeling*, pages 1–36, 2012.

[HYS+11] Thomas Heyman, Koen Yskout, Riccardo Scandariato, Holger Schmidt, and Yijun Yu. The security twin peaks. In ´ Ulfar Erlingsson, Roel Wieringa, and

Nicola Zannone, editors, *Engineering Secure Software and Systems*, volume 6542 of *Lecture Notes in Computer Science*, pages 167–180. Springer Berlin Heidelberg, 2011.

[IEC06] IEC. 61025:2006 fault tree analysis (FTA). Technical report, 2006.

[IEC08] IEC. 60812:2008 failure modes and effects analysis (FMEA and FMECA). Technical report, 2008.

[Int96] Internal Revenue Service. Internal Revenue Service Model Information Technology Privacy Impact Assessment. Technical report, IRS, 1996.

[ISO11] ISO/IEC/IEEE. ISO/IEC/IEEE Systems and software engineering – Architecture description. *ISO/IEC/IEEE 42010:2011(E)*, 2011.

[ISO12a] ISO/IEC. ISO/IEC 19505-1:2012 Information technology - Object Management Group Unified Modeling Language (OMG UML), Infrastructure. Standard 19505-1:2012(E), ISO/IEC, April 2012. http://www.omg.org/spec/UML/.

[ISO12b] ISO/IEC. ISO/IEC 19505-2:2012 Information technology - Object Management Group Unified Modeling Language (OMG UML), Superstructure. Standard 19505-2:2012(E), ISO/IEC, April 2012. http://www.omg.org/spec/UML/.

[JAS15a] JAS Global Advisors. JASBUG: Cutting through the fear uncertainty and doubt (FUD). https://www.jasadvisors.com/additonal-jasbug-security-exploit-info/, 2015.

[JAS15b] JAS Global Advisors. JASBUG fact sheet. https://www.jasadvisors.com/about-jas/jasbug-security-vulnerability-fact-sheet/, 2015.

[JB05] A. Jansen and J. Bosch. Software Architecture as a Set of Architectural Design Decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pages 109–120. IEEE, 2005.

[JSP+11] Bart Jacobs, Jan Smans, Pieter Philippaerts, Frédéric Vogels, Willem Penninckx, and Frank Piessens. Verifast: A powerful, sound, predictable, fast verifier for c and java. In *NASA Formal Methods Symposium*, pages 41–55. Springer, 2011.

[Jür05] Jan Jürjens. *Secure Systems Development with UML*. Springer Berlin Heidelberg, 2005.

[Ker83a] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, Jan 1883.

[Ker83b] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:161–191, Feb 1883.

[KG99] Loren Kohnfelder and Praerit Garg. The threats to our products. *Interface (Microsoft Corporation)*, 1999.

[Lan13] Carl E. Landwehr. A Building Code for Building Code: Putting What We Know Works to Work. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 139–147, New York, NY, USA, 2013. ACM.

[LB03] C. Larman and V. R. Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, June 2003.

[LeB07] David LeBlanc. Dreadful. 2007.

[Lei19] Jonathan Leitschuh. Zoom Zero Day: 4+ Million Webcams & maybe an RCE? Just get them to visit your website! `https://medium.com/bugbountywriteup/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5`, 2019.

[LNI+03] L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett. Introducing abuse frames for analysing security requirements. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, pages 371–372, 2003.

[Loc12] John Locke. *Some Thoughts concerning Education*. A. & J. Churchill, 1712.

[LPM+16] Tong Li, Elda Paja, John Mylopoulos, Jennifer Horkoff, and Kristian Beckers. Security attack analysis using attack patterns. *Proceedings - International Conference on Research Challenges in Information Science*, 2016-Augus, 2016.

[LSS10] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[McG06] Gary McGraw. *Software Security: Building Security In*. Addison-Wesley, 2006.

[MCK+19] Ran Mo, Yuanfang Cai, Rick Kazman, Lu Xiao, and Qiong Feng. Architecture anti-patterns: Automatically detectable violations of design principles. *IEEE Transactions on Software Engineering*, 2019.

[MCKX15] R. Mo, Y. Cai, R. Kazman, and L. Xiao. Hotspot patterns: The formal definition and automatic detection of architecture smells. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*, May 2015.

[MDK14] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This poodle bites: exploiting the ssl 3.0 fallback. *Security Advisory*, 2014.

[MF99] J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pages 55–64, 1999.

[MH17] Kyriakos Malamas and Danial Hosseini. Design flaws as security threats. Master's thesis, 2017.

[Mic15] Microsoft Corporation. Microsoft security bulletin ms15-011 - critical:. https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2015/ms15-011, 2015.

[Mic16] Microsoft Corporation. Microsoft Threat Modeling Tool 2016. http://aka.ms/tmt2016, 2016.

[Mic18] Microsoft Corporation. Sdl security bug bar (sample). https://docs.microsoft.com/en-us/security/sdl/security-bug-bar-sample, 2018.

[Mic20] Microsoft Corporation. Microsoft Threat Modeling Tool 7. http://aka.ms/tmt, 2020.

[MIT19] MITRE. 2019 CWE Top 25 Most Dangerous Software Errors, 2019.

[MMW18] Gary McGraw, Sammy Migues, and Jacob West. BSIMM9. Technical report, 2018.

[Moo09] Daniel L. Moody. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, nov 2009.

[MS05] Nancy R Mead and Ted Stehney. Security Quality Requirements Engineering (SQUARE) Methodology. *SIGSOFT Softw. Eng. Notes*, 2005.

[NCFS17] Tayyaba Nafees, Natalie Coull, Ian Ferguson, and Adam Sampson. Vulnerability anti-patterns: a timeless way to capture poor software practices (vulnerabilities). In *Proceedings of the 24th Conference on Pattern Languages of Programs*, page 23. The Hillside Group, 2017.

[NIS12] NIST. Guide for Conducting Risk Assessments. *NIST special publication*, (800-30 Rev.1), 2012.

[NIS19a] NIST. National Vulnerability Database (NVD), 2019.

[NIS19b] NIST. NIST Privacy Risk Assessment Methodology (PRAM), February 2019.

[Nus01] B. Nuseibeh. Weaving together requirements and architectures. *Computer*, 34(3):115–119, 2001.

[NVBM$^+$17] Job Noorman, Jo Van Bulck, Tobias Mühlberg, Frank Piessens, Pieter Maene, Bart Preneel, Ingrid Verbauwhede, Johannes Götzfried, Tilo Müller, and Felix Freiling. Sancus 2.0: A low-cost security architecture for iot devices. *ACM Transactions on Privacy and Security*, 20(3):1–7, 9 2017.

[OEC80] OECD. OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. *OECD*, 1980.

[OEC13] OECD. The OECD Privacy Framework. *Organisation for Economic Co-Operation and Development*, pages 1–154, 2013.

[Oli14] Ian Oliver. *Privacy engineering: A dataflow and ontological approach.* CreateSpace Independent Publishing Platform, 2014.

[OS14] Marie Caroline Oetzel and Sarah Spiekermann. A systematic methodology for privacy impact assessments: A design science approach. *European Journal of Information Systems*, 23(2):126–150, 2014.

[OWA17a] OWASP Top Ten Project, 2017.

[OWA17b] OWASP. Software Assurance Maturity Model Version 1.5. Technical report, OWASP, 2017.

[OWA18] OWASP. OWASP Threat Dragon. `https://www.owasp.org/index.php/OWASP_Threat_Dragon`, 2018.

[OWH15] Donald W. Olson, Steven F. Wolf, and Joseph M. Hook. The tacoma narrows bridge collapse. *Physics Today*, 68(11):64–65, November 2015.

[PAC19] Marco Patrignani, Amal Ahmed, and Dave Clarke. Formal Approaches to Secure Compilation: A Survey of Fully Abstract Compilation and Related Work. *ACM Comput. Surv.*, 51(6):125:1–125:36, February 2019.

[Pet13] Marian Petre. UML in practice. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 722–731, May 2013.

[Pet14] Marian Petre. “no shit” or “oh, shit!”: responses to observations on the use of uml in professional practice. *Software & Systems Modeling*, 13(4):1225–1235, Oct 2014.

[PP03] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing.* Prentice Hall, 3rd edition, 2003.

[Pri20] PrivacyPatterns.org, 2020.

[Res19] Tobias Reski. Conception and development of a threat modeling tool. Bachelor thesis, Hochschule Offenburg, Feb 2019.

[RH13] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[RKK16] Tobias Rauter, Nermin Kajtazovic, and Christian Kreiner. Asset-Centric Security Risk Assessment of Software Components. *2nd International Workshop on MILS: Architecture and Assurance for Secure Systems*, 2016.

[Rus99] Salman Rushdie. *The Ground Beneath Her Feet.* Henry Holt & Company, 1999.

[SBMP08] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: Eclipse Modeling Framework.* Pearson Education, 2008.

[SC09] Sarah Spiekermann and Lorrie F Cranor. Engineering privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, 2009.

[Sch91] Arthur Schopenhauer. *The Art of Literature*. Swan Sonnenschein & Co., Limited, 1891.

[Sch99] Bruce Schneier. Attack trees. 1999.

[Sch03] Markus Schumacher. Firewall patterns. In *EuroPLoP*, pages 417–430, 2003.

[SDVL$^+$] **Laurens Sion**, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Peggy Valcke, and Wouter Joosen. DPMF: A Modeling Framework for Data Protection by Design. *Enterprise Modelling and Information Systems Architectures (EMISAJ). (accepted).*

[SDVL$^+$19] **Laurens Sion**, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. An architectural view for data protection by design. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 11–20. IEEE, 2019.

[SFBH$^+$06] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns*. Wiley, 2006.

[Sha16] Stuart S. Shapiro. Privacy Risk Analysis Based on System Control Structures: Adapting System-Theoretic Process Analysis for Privacy Engineering. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 17–24, may 2016.

[Sha17] Stuart S. Shapiro. Addressing Early Life Cycle Privacy Risk. In *2017 International Workshop on Privacy Engineering - IWPE'17*, 2017.

[SHF01] Gary Stoneburner, Clark Hayden, and Alexis Feringa. Nist sp 800-27 engineering principles for information technology security (a baseline for achieving security). Technical report, NIST, 2001.

[Sho08] Adam Shostack. Experiences threat modeling at microsoft. In *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*, 2008.

[Sho14] Adam Shostack. *Threat Modeling: Designing for Security.* John Wiley & Sons, Indianapolis, Indiana, 2014.

[Sho20] Adam Shostack. Personal communication/feedback from Adam Shostack on the EnCyCriS2020 paper: Security Threat Modeling: Are Data Flow Diagrams Enough?, 2020.

[Sil12] Nate Silver. *The Signal and the Noise.* Penguin Books, 2012.

[SMC74] W P Stevens, G J Myers, and L L Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, 1974.

[SNL05] Christopher Steel, Ramesh Nagappan, and Ray Lai. *Core Security Patterns: Best Pratices and Strategies for J2EE, Web Services, and Identity Management.* Prentice Hall Ptr, 2005.

[SO05] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.

[SPM$^+$17] J. C. S. Santos, A. Peruma, M. Mirakhorli, M. Galstery, J. V. Vidal, and A. Sejfia. Understanding software vulnerabilities related to architectural security tactics: An empirical investigation of chromium, php and thunderbird. In *International Conference on Software Architecture*, 2017.

[SR19] Andreas Schaad and Tobias Reski. "Open Weakness and Vulnerability Modeler" (OVVL): An Updated Approach to Threat Modeling. In *Proceedings of the 16th International Joint Conference on e-Business and*

Telecommunications - Volume 2: SECRYPT,, pages 417–424. INSTICC, SciTePress, 2019.

[SS75] J.H. Saltzer and M.D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[SS04] Frank Swiderski and Window Snyder. *Threat modeling.* Microsoft Press, 2004.

[STM17] J. C. S. Santos, K. Tarrit, and M. Mirakhorli. A catalog of security architecture weaknesses. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, pages 220–223, April 2017.

[STY$^+$19] **Laurens Sion**, Katja Tuma, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. Towards automated security design flaw detection. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, SEAD '19, pages 49–56, 2019.

[SVLJ20] **Laurens Sion**, Dimitri Van Landuyt, and Wouter Joosen. The Never-Ending Story: On the Need for Continuous Privacy Impact Assessment. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020.

[SVLJdJ16] **Laurens Sion**, Dimitri Van Landuyt, Wouter Joosen, and Gjalt de Jong. Systematic Quality Trade-off Support in the Software Product-line Configuration Process. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, page 164–173, New York, NY, USA, 2016. Association for Computing Machinery.

[SVLWJ19a] **Laurens Sion**, Dimitri Van Landuyt, Kim Wuyts, and Wouter Joosen. Poster: Privacy risk assessment for data subject-aware threat modeling. 40th IEEE Symposium on Security and Privacy, 2019.

[SVLWJ19b] **Laurens Sion**, Dimitri Van Landuyt, Kim Wuyts, and Wouter Joosen. Privacy risk assessment for data subject-aware threat modeling. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 64–71. IEEE, 2019.

[SVLYJ16] **Laurens Sion**, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. Towards systematically addressing security variability in software product lines. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, pages 342–343, New York, NY, USA, 2016. Association for Computing Machinery.

[SVLYJ18] **Laurens Sion**, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. SPARTA: Security & privacy architecture through risk-driven threat assessment. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 89–92. IEEE, 2018.

[SWY+18] **Laurens Sion**, Kim Wuyts, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Interaction-based privacy threat elicitation. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 79–86. IEEE, 2018.

[SYSJ17] **Laurens Sion**, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. A modular meta-model for security solutions. In *Companion to the first International Conference on the Art, Science and Engineering of Programming*, Programming '17, pages 1–5, New York, NY, USA, 2017. Association for Computing Machinery.

[SYvdB+15] **Laurens Sion**, Koen Yskout, Alexander van den Berghe, Riccardo Scandariato, and Wouter Joosen. MASC: Modelling Architectural Security Concerns. In *Proceedings of the Seventh International Workshop on Modeling in Software Engineering*, MiSE '15, pages 36–41. IEEE Press, 2015.

[SYvdB+20] **Laurens Sion**, Koen Yskout, Alexander van den Berghe, Dimitri Van Landuyt, and Wouter Joosen. Security Threat Modeling: Are Data Flow Diagrams Enough? In *EnCyCris '20: Proceedings of the First International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. IEEE, 2020.

[SYVJ18a] **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Poster: Knowledge-enriched Security and Privacy Threat Modeling. In *2018 IEEE/ACM 40th International Conference on Software Engineering Companion (ICSE-C)*, pages 290–291, May 2018.

[SYVJ18b] **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Risk-based design security analysis. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*, SEAD '18, page 11–18, New York, NY, USA, 2018. Association for Computing Machinery.

[SYVJ18c] **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Solution-aware Data Flow Diagrams for Security Threat Modelling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1425–1432, New York, NY, USA, 2018. Association for Computing Machinery.

[Tar20] Izar Tarandach. PyTM. `https://github.com/izar/pytm`, 2020.

[TCS18] K Tuma, G Calikli, and R Scandariato. Threat analysis of software systems: A systematic literature review. *Journal of Systems and Software*, 144:275–294, 2018.

[THMS19] Katja Tuma, Danial Hosseini, Kyriakos Malamas, and Riccardo Scandariato. Inspection guidelines to identify security design flaws. In *Proceedings of the 13th European Conference on Software Architecture - Volume 2*, ECSA '19, pages 116–122, New York, NY, USA,

2019. Association for Computing Machinery. https://doi.org/10.1145/3344948.3344995.

[Thr19] ThreatSpec. ThreatSpec. https://threatspec.org/, 2019.

[TK91] Yonglei Tao and Chenho Kung. Formal definition and verification of data flow diagrams. *The Journal of Systems and Software*, 16(1):29–36, 1991.

[TL18] Davide Taibi and Valentina Lenarduzzi. On the definition of microservice bad smells. *IEEE software*, 35(3):56–62, 2018.

[Tor05] Peter Torr. Demystifying the threat modeling process. *IEEE Security & Privacy Magazine*, 3:66–70, 2005.

[TS18] Katja Tuma and Riccardo Scandariato. Two Architectural Threat Analysis Techniques Compared. In Carlos E. Cuesta, David Garlan, and Jennifer Pérez, editors, *Software Architecture*, pages 347–363, Cham, 2018. Springer International Publishing.

[TSB19] K. Tuma, R. Scandariato, and M. Balliu. Flaws in flows: Unveiling design flaws via information flow analysis. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 191–200, March 2019.

[TSSY20] Katja Tuma, **Laurens Sion**, Riccardo Scandariato, and Koen Yskout. Automating the Early Detection of Security Design Flaws. In *23rd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2020.

[TSWS17] Katja Tuma, Riccardo Scandariato, Mathias Widman, and Christian Sandberg. Towards security threats that matter. In *3rd Workshop On The Security Of Industrial Control Systems & Of Cyber-Physical Systems (CyberICPS 2017)*, 2017.

[Tür17] Sven Türpe. The Trouble With Security Requirements. *25th IEEE International Requirements Engineering Conference*, 2017.

[UFF12] Anton V. Uzunov, Eduardo B. Fernandez, and Katrina Falkner. Engineering Security into Distributed Systems: A Survey of Methodologies. *Journal of Universal Computer Science*, 18(20):2920–3006, 2012.

[UM15] Tony UcedaVelez and Marco M Morana. *Risk Centric Threat Modeling.* Wiley, 2015.

[VGRH81] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. *Fault Tree Handbook (NUREG-0492).* U.S. Nuclear Regulatory Commission, 1981.

[VJNB06] Jan Salvador Ven, Anton G J Jansen, Jos A G Nijhuis, and Jan Bosch. Design Decisions: The Bridge between Rationale and Architecture. In AllenH. Dutoit, Raymond McCall, Ivan Mistrik, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 329–348. Springer Berlin Heidelberg, 2006.

[VLSVJ19] Dimitri Van Landuyt, **Laurens Sion**, Emiel Vandeloo, and Wouter Joosen. On the applicability of security and privacy threat modeling for blockchain applications. In Sokratis Katsikas, Frédéric Cuppens, Nora Cuppens, Costas Lambrinoudakis, Christos Kalloniatis, John Mylopoulos, Annie Antón, Stefanos Gritzalis, Frank Pallas, Jörg Pohle, Angela Sasse, Weizhi Meng, Steven Furnell, and Joaquin Garcia-Alfaro, editors, *Computer Security. CyberICPS 2019, SECPRE 2019, SPOSE 2019, ADIoT 2019*, pages 195–203. Springer International Publishing, 2019.

[VM02] John Viega and Gary McGraw. *Building secure software: how to avoid security problems the right way.* Addison-Wesley, Boston, MA, USA, 1st edition, September 2002.

[Vos08] David Vose. *Risk analysis: a quantitative guide.* John Wiley & Sons, 2008.

[vSYJ17] Alexander van den Berghe, Riccardo Scandariato, Koen Yskout, and Wouter Joosen. Design notations for secure software: A systematic literature review. *Software & Systems Modeling*, 16(3):809–831, 2017.

[WB90] Samuel D. Warren and Louis D. Brandeis. Right to Privacy. *Harvard Law Review*, 4(5):193–220, 1890.

[WJ15] Kim Wuyts and Wouter Joosen. LINDDUN privacy threat modeling: a tutorial. 2015.

[WSJ20] Kim Wuyts, **Laurens Sion**, and Wouter Joosen. LINDDUN GO: A Lightweight Approach to Privacy Threat Modeling. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020.

[WSVLJ19] Kim Wuyts, **Laurens Sion**, Dimitri Van Landuyt, and Wouter Joosen. Knowledge is power: Systematic reuse of privacy knowledge for threat elicitation. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 80–83. IEEE, 2019.

[Wuy15] Kim Wuyts. *Privacy Threats in Software Architectures.* PhD thesis, KU Leuven, Jan 2015.

[WVHJ18] Kim Wuyts, Dimitri Van Landuyt, Aram Hovsepyan, and Wouter Joosen. Effective and Efficient Privacy Threat Modeling through Domain Refinements. In *SAC2018: 1st track on Privacy by Design (PbD)*, 2018.

[XL19] Wenjun Xiong and Robert Lagerström. Threat modeling – A systematic literature review. *Computers & Security*, 84:53–69, July 2019.

[YC75] Edward Yourdon and Larry Constantine. *Structured Design.* 1975.

[YC79] Edward Yourdon and Larry L Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1979.

[YHSJ06] Koen Yskout, Thomas Heyman, Riccardo Scandariato, and Wouter Joosen. A system of security patterns. Report CW 469, Department of Computer Science, KULeuven, December 2006.

[YHVL+20] Koen Yskout, Thomas Heyman, Dimitri Van Landuyt, **Laurens Sion**, Kim Wuyts, and Wouter Joosen. Threat modeling: from infancy to maturity. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*. IEEE, 2020.

[YL14] William Young and Nancy G Leveson. An Integrated Approach to Safety and Security Based on Systems Theory. *Communications of the ACM*, 57(2):31–35, feb 2014.

List of Publications

Journal papers

- **Laurens Sion**, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Peggy Valcke, and Wouter Joosen. DPMF: A Modeling Framework for Data Protection by Design. *Enterprise Modelling and Information Systems Architectures (EMISAJ). (accepted)*

International conference and workshop papers

- 2020 Dimitri Van Landuyt, **Laurens Sion**, Pierre Dewitte, and Wouter Joosen. The Bigger Picture: Approaches to an Inter-Organizational Perspective on Data Protection In *SPOSE: Workshop on Security, Privacy, Organizations, and Systems Engineering*, 2020.

- 2020 Katja Tuma, **Laurens Sion**, Riccardo Scandariato, and Koen Yskout. Automating the Early Detection of Security Design Flaws. In *23rd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2020

- 2020 **Laurens Sion**, Dimitri Van Landuyt, and Wouter Joosen. The Never-Ending Story: On the Need for Continuous Privacy Impact Assessment. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020

- 2020 Kim Wuyts, **Laurens Sion**, and Wouter Joosen. LINDDUN GO: A Lightweight Approach to Privacy Threat Modeling. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2020
- 2020 **Laurens Sion**, Koen Yskout, Alexander van den Berghe, Dimitri Van Landuyt, and Wouter Joosen. Security Threat Modeling: Are Data Flow Diagrams Enough? In *EnCyCris '20: Proceedings of the First International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*. IEEE, 2020
- 2020 Koen Yskout, Thomas Heyman, Dimitri Van Landuyt, **Laurens Sion**, Kim Wuyts, and Wouter Joosen. Threat modeling: from infancy to maturity. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*. IEEE, 2020
- 2020 Shamal Faily, Riccardo Scandariato, Adam Shostack, **Laurens Sion**, and Duncan Ki-Aries. Contextualisation of data flow diagrams forsecurity analysis. In *The Seventh International Workshop on Graphical Models for Security (GraMSec)*, 2020
- 2019 **Laurens Sion**, Katja Tuma, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. Towards automated security design flaw detection. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, SEAD '19, pages 49–56, 2019
- 2019 Dimitri Van Landuyt, **Laurens Sion**, Emiel Vandeloo, and Wouter Joosen. On the applicability of security and privacy threat modeling for blockchain applications. In Sokratis Katsikas, Frédéric Cuppens, Nora Cuppens, Costas Lambrinoudakis, Christos Kalloniatis, John Mylopoulos, Annie Antón, Stefanos Gritzalis, Frank Pallas, Jörg Pohle, Angela Sasse, Weizhi Meng, Steven Furnell, and Joaquin Garcia-Alfaro, editors, *Computer Security. CyberICPS 2019, SECPRE 2019, SPOSE 2019, ADIoT 2019*, pages 195–203. Springer International Publishing, 2019
- 2019 **Laurens Sion**, Dimitri Van Landuyt, Kim Wuyts, and Wouter Joosen. Privacy risk assessment for data subject-aware threat

modeling. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 64–71. IEEE, 2019

- 2019 Kim Wuyts, **Laurens Sion**, Dimitri Van Landuyt, and Wouter Joosen. Knowledge is power: Systematic reuse of privacy knowledge for threat elicitation. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 80–83. IEEE, 2019

- 2019 **Laurens Sion**, Pierre Dewitte, Dimitri Van Landuyt, Kim Wuyts, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. An architectural view for data protection by design. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 11–20. IEEE, 2019

- 2019 Pierre Dewitte, Kim Wuyts, **Laurens Sion**, Dimitri Van Landuyt, Ivo Emanuilov, Peggy Valcke, and Wouter Joosen. A comparison of system description models for data protection by design. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1512–1515, New York, NY, USA, 2019. Association for Computing Machinery

- 2018 **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Risk-based design security analysis. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment*, SEAD '18, page 11–18, New York, NY, USA, 2018. Association for Computing Machinery

- 2018 **Laurens Sion**, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. SPARTA: Security & privacy architecture through risk-driven threat assessment. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 89–92. IEEE, 2018

- 2018 **Laurens Sion**, Kim Wuyts, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Interaction-based privacy threat elicitation. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 79–86. IEEE, 2018

- 2018 **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Solution-aware Data Flow Diagrams for Security Threat

Modelling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1425–1432, New York, NY, USA, 2018. Association for Computing Machinery

- 2017 **Laurens Sion**, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. A modular meta-model for security solutions. In *Companion to the first International Conference on the Art, Science and Engineering of Programming*, Programming '17, pages 1–5, New York, NY, USA, 2017. Association for Computing Machinery
- 2016 **Laurens Sion**, Dimitri Van Landuyt, Wouter Joosen, and Gjalt de Jong. Systematic Quality Trade-off Support in the Software Product-line Configuration Process. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, page 164–173, New York, NY, USA, 2016. Association for Computing Machinery
- 2016 **Laurens Sion**, Dimitri Van Landuyt, Koen Yskout, and Wouter Joosen. Towards systematically addressing security variability in software product lines. In *Proceedings of the 20th International Systems and Software Product Line Conference*, SPLC '16, pages 342–343, New York, NY, USA, 2016. Association for Computing Machinery
- 2015 **Laurens Sion**, Koen Yskout, Alexander van den Berghe, Riccardo Scandariato, and Wouter Joosen. MASC: Modelling Architectural Security Concerns. In *Proceedings of the Seventh International Workshop on Modeling in Software Engineering*, MiSE '15, pages 36–41. IEEE Press, 2015

International abstracts/presentations/posters

- 2019 **Laurens Sion**, Dimitri Van Landuyt, Kim Wuyts, and Wouter Joosen. Poster: Privacy risk assessment for data subject-aware threat modeling. 40th IEEE Symposium on Security and Privacy, 2019

2018
- **Laurens Sion**, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Poster: Knowledge-enriched Security and Privacy Threat Modeling. In *2018 IEEE/ACM 40th International Conference on Software Engineering Companion (ICSE-C)*, pages 290–291, May 2018

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
IMEC-DISTRINET
Celestijnenlaan 200A box 2402
B-3001 Leuven
Laurens.Sion@cs.kuleuven.be
https://distrinet.cs.kuleuven.be