Karlsruhe Institute of Technology

Software Design and Quality

A Methodology for Systematically Including Semantic Information in Threat Modeling and Analysis

Bachelor's Thesis of

Benoît Legien

At the KIT Department of Informatics
KASTEL – Institute of Information Security and Dependability

First examiner:	Prof. Dr. Ralf Reussner
Second examiner:	Prof. Dr.-Ing. Anne Koziolek
First advisor:	M.Sc. Nicolas Boltz
Second advisor:	Dr. Christopher Gerking

01. December 2025 – 01. May 2026

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Abstract

As data becomes increasingly digital and systems become more and more involved in people's lives, the impact of leaked, stolen or otherwise compromised data can be disastrous. To mitigate these risks, security and privacy by design can be implemented alongside threat modeling approaches and taxonomies. Many system-centric threat modeling approaches use Data Flow Diagrams (DFDs) to model a system's architecture. This model can then be used in combination with the previously mentioned taxonomies to analyze the system for vulnerabilities, before the system is even built. However, many threat modeling approaches still require considerable effort and expertise, and do not consider the flow of the data throughout the system. Our approach addresses these issues by building on the extensible architecture-based data flow analysis framework xDECAF. We have extended its web editor to enable software architects to easily include semantic information in the form of labels in Data Flow Diagrams (DFDs). These DFDs can then be checked against predefined constraints. Although our approach is designed to be taxonomy-independent, we define an initial set of constraints by translating the mature privacy threat taxonomy LINDDUN. To evaluate our approach, we performed a user study designed for comparability to a baseline study that evaluated LINDDUN. However, the evaluation's validity is limited due to the small dataset and the fundamental differences between the two approaches. While the results suggest that the approach can significantly decrease the time required for analysis, the translation seems to fail to identify most threats. Additionally, we were only able to translate about half of all LINDDUN threats and 70% of all leaf threats, due to the nature of our approach.

Zusammenfassung

Da Daten zunehmend digitalisiert werden und Systeme immer stärker in das Leben der Menschen eingebunden sind, können die Folgen von Datenlecks, Datendiebstahl oder anderweitig kompromittierten Daten katastrophal sein. Um diese Risiken zu mindern können „Security by Design“ und „Privacy by Design“ in Verbindung mit „Threat Modeling“ Ansätzen und Taxonomien eingesetzt werden. Viele system-zentrierte Threat Modeling Ansätze verwenden Datenflussdiagramme (DFDs), um ein System zu modellieren. Dieses Modell kann dann in Kombination mit den zuvor erwähnten Taxonomien verwendet werden, um das System auf Schwachstellen zu analysieren, noch bevor es überhaupt implementiert wurde. Allerdings erfordern jedoch viele Threat Modeling Ansätze nach wie vor erheblichen Aufwand und Fachwissen und berücksichtigen nicht den Fluss von Daten durch das System. Wir präsentieren einen Ansatz, der diese Probleme lösen soll, indem er auf dem erweiterbaren, architektur-basierten Datenflussanalyse-Framework xDECAF aufbaut. Dazu haben wir dessen Web-Editor erweitert, damit Softwarearchitekten auf einfache und gezielte Weise semantische Informationen in Form von Labels in DFDs einbinden können. Diese DFDs können dann anhand vordefinierter „Constraints“ überprüft werden. Obwohl unser Ansatz taxonomie-unabhängig konzipiert ist, definieren wir eine erste Sammlung von Constraints, indem wir die ausgereifte Taxonomie LINDDUN übersetzen, die Bedrohungen für die Privatsphäre von Nutzern digitaler Systeme definiert. Um unseren Ansatz zu bewerten, haben wir eine Nutzerstudie durchgeführt, die auf Vergleichbarkeit mit einer Basisstudie ausgelegt war, in der LINDDUN evaluiert wurde. Die Aussagekraft der Bewertung ist jedoch aufgrund des kleinen Datensatzes und der grundlegenden Unterschiede zwischen den beiden Ansätzen begrenzt. Während die Ergebnisse darauf hindeuten, dass der Ansatz den Zeitaufwand für die Analyse deutlich verringern kann, scheint die Übersetzung die meisten Bedrohungen nicht zu identifizieren. Zudem konnten wir aufgrund der Art unseres Ansatzes nur etwa die Hälfte aller LINDDUN Bedrohungen, bzw. 70% aller Bedrohungen die Blätter sind, übersetzen.

Contents

List of Figures

List of Tables

1. Introduction

This chapter presents the context in which this thesis is written. Section 1.1 establishes the context and explains the existing problems this thesis aims to solve. In Section 1.2, we provide an overview of the contribution of this thesis. Finally, Section 1.3 outlines the structure of the thesis.

1.1. Motivation

As more and more digital systems come into existence, an ever-growing amount of data is stored and transferred digitally. These systems have become deeply involved in people's personal lives, collecting an immense amount of data about their users in the process. Subsequently, if this data is leaked, stolen or otherwise compromised, companies and individuals alike face serious consequences. To reduce the impact and risk of such a scenario, preventative practices are required, e.g., *security by design*. There have also been many efforts in legal domains to enforce the security and privacy of systems. For example, the General Data Protection Regulation (GDPR) demands the data protection by design and by default in the EU. This means that data protection is required to be a major influencing factor during the entire development lifecycle for companies in the EU.

However, practices like security by design still require processes and frameworks that define how to systematically analyze a system. A class of these frameworks is unified under the term *threat modeling*. It refers to a variety of approaches intended to proactively identify and address potential issues at early stages of the development cycle. One of the most prominent threat modeling approaches is the *Security Development Lifecycle (SDL)* [18, 44] by Microsoft. It defines a taxonomy of threats named *STRIDE* that focuses on the security of systems and protection against mostly external adversaries.

Although *security threat modeling* has been performed for quite a while, this methodology has only gained traction in the context of privacy in the past decade. This has been intensified in recent years due to the high demand for data necessary to train newer AI models. Companies have been observed training their models with data from their users, including data that may be perceived as private [20]. Nowadays, there is a growing number of approaches to improve the accessibility of threat modeling in the context of privacy. For example, one of the pioneers of this topic is LINDDUN [9], the privacy equivalent to STRIDE.

Despite the efforts to make threat modeling more accessible [9, 50], successfully executing an analysis still requires considerable effort and expertise [49, 38]. One of these challenges is

that many approaches require security or privacy specialists to perform it, since the topic is often quite complex and requires extensive knowledge. Another challenge is that analyzing a system takes a lot of time. For example, even a simple process still requires modeling the system, determining threats based on extensive taxonomies, prioritizing the threats and selecting appropriate countermeasures. Furthermore, this process must be repeated for each iteration of the development process in order to maintain the security and privacy requirements.

Approaches that aim to automate and simplify certain steps in the threat modeling process have been developed. These approaches are capable of analyzing systems based on their abstract components and compute possible threats. However, these approaches still face challenges. For example, *threat explosion* describes the phenomenon that such applications detect a large number of irrelevant threats, i.e. false positives. Each suggested threat must subsequently be filtered and handled appropriately by a professional.

1.2. Contribution

This thesis presents an approach aimed at improving the automated analysis of systems in the context of threat modeling. We investigate how to systematically include semantic information in system models, and how this semantic information can improve the accuracy of the computed threats. To this end, we leverage the capabilities of the data flow analysis framework *xDECAF* and implement an extension to its web editor. The extension streamlines the incorporation of semantic information in the form of labels in DFDs. Importantly, we designed the extension to be as generic as possible so that the approach can be used independently of any taxonomy.

To showcase the approach, we translated the privacy threat taxonomy LINDDUN into a data structure that the extension can use. We further used the translated taxonomy to evaluate the approach's capabilities in a user study.

1.3. Outline

The remainder of this thesis is structured as follows: Chapter 2 provides an overview over the foundations our approach is based on, including data flow analysis, DFDs, and threat modeling. Next, we summarize related work that address similar issues in Chapter 3. We describe our approach in Chapter 4 and explain how we transformed the LINDDUN taxonomy to fit our needs in Chapter 5. Chapter 6 presents the evaluation of the approach in comparison to a baseline study. Finally, we summarize and conclude the thesis in Chapter 7.

2. Foundations

This chapter provides an overview of the fundamental concepts that form the basis of this thesis. Section 2.1 summarizes the topic of data flow analysis and the application xDECAF, which this thesis is based on. The topic of threat modeling is introduced in Section 2.2.

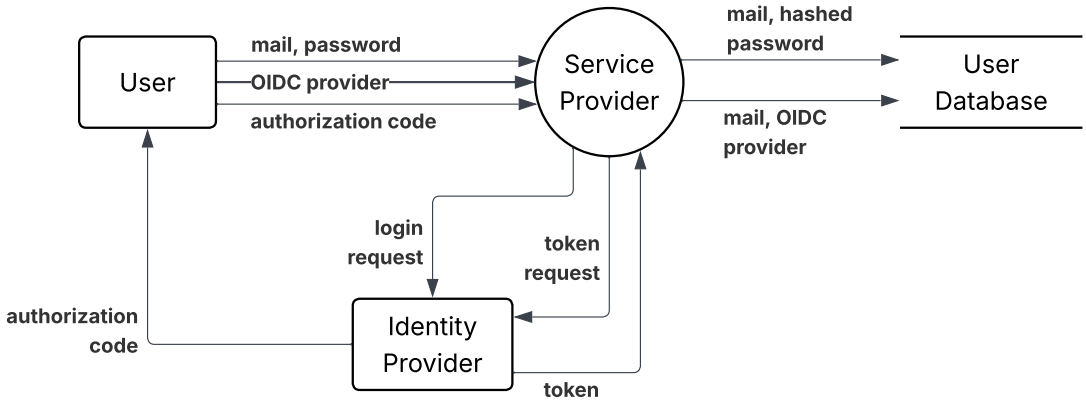
2.1. Data Flow Analysis

The ability to analyze the flow of data in a system or program without executing it can provide significant benefits. Data Flow Analysis is a static analysis technique capable of inferring runtime behavior of programs [17]. This technique has been applied in many different areas, such as compiler optimization and software verification. In the context of this thesis, data flow analysis is used to analyze architectural models, namely DFDs, during the early stages of the development lifecycle. Although data flow analysis can use many internal representations [17], our approach is based on directed acyclic graphs (DAGs).

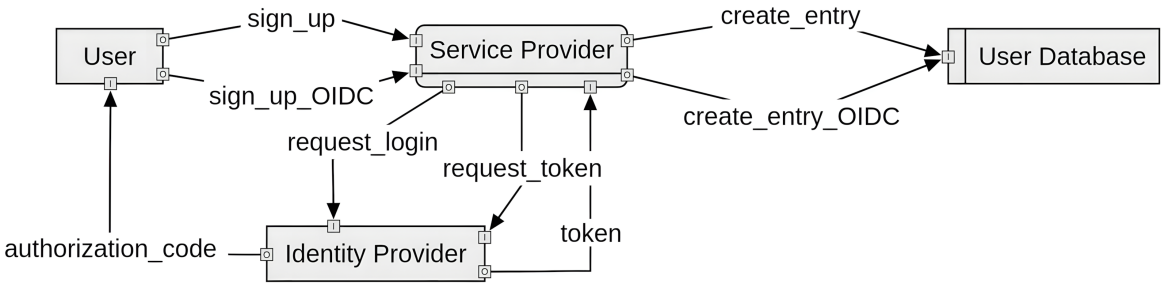
2.1.1. Data Flow Diagrams

One common method for modeling the flow of data in a system are Data Flow Diagrams (DFDs). Because of their simple notation, they can be used by developers and stakeholders alike, and are commonly used for architectural analyses, e.g., threat modeling [2, 40, 15]. Figure 2.1a shows an example of a DFD in DeMarco notation visualizing a simplified sign up sequence of a *service provider*. A user can sign up for the service provider using either their email and a password (i.e. the upper data flow from User to Service Provider), or using a third-party *identity provider* via the OpenID Connect (OIDC) protocol (i.e. the middle data flows from User to Service Provider). If the user selects an email to sign up with, the service provider can immediately create an entry in the user database, hashing the password in the process. However, if the user selects an identity provider to sign up with, the service provider first has to request a login from the identity provider, which in turn sends an *authorization code* via the user to the service provider. The service provider then has to request a token using the previously received authorization code to securely request a token from the identity provider. Only then, the service provider can create an entry in the user database, linking it to the identity provider.

Figure 2.1a demonstrates the original notation by DeMarco [8], and Figure 2.1b shows the extended notation used in this thesis. The original notation consists of only four elements.



(a) Sign up sequence in DeMarco notation.



(b) Sign up sequence with input and output pins.

Figure 2.1.: DFD example showing a sign up sequence.

- *Processes* (or *function nodes*) represent parts of the system that can interact with and transform data. They are displayed as circles or as rounded squares with a horizontal line, e.g., the node *Service Provider*.
- *External entities* (or *input/output nodes*) are sources or sinks of data and, as their name suggests, are usually not part of the system. They are displayed as squares, e.g., the nodes *User* and *Identity Provider*.
- *Data stores* (or *storage nodes*) represent stored data that can be retrieved at a later time. They are displayed as two parallel horizontal lines (or as squares with a vertical line), e.g., the node *User Database*.
- Finally, *data flows* (or *edges)* visualize the flow of data. They are displayed as arrows between nodes and can be labeled with relevant information about the flow, e.g., *mail, password*. Importantly, processes can change the labels of data flows since they can modify the data.

The simplicity of the DeMarco notation has some drawbacks [39, 35]. Some of these shortcomings are addressed by the extended notation introduced by Seifermann et al. [35]. Specifically, this notation allows modeling multiple, alternate data paths in DFDs by defining

input and *output pins* (or *ports).* In Figure 2.1b, the small squares on the edges of nodes containing the letters `I` or `O` are the input and output pins, respectively. These pins indicate alternative data flows, enabling a deterministic interpretation and traversal of the DFD. Subsequently, DFDs in the extended notation can be algorithmically transformed into DAGs.

2.1.2. xDECAF

One framework that can analyze the flow of data in a DFD is the extensible, open-source framework xDECAF [5, 51]. It consists of multiple services, most importantly a backend which analyzes the DFDs, and a web editor, i.e. frontend, which allows users to model, import, export, and include labels into DFDs. This thesis builds on the web editor, which is built using TypeScript.

xDECAF is based on the notation introduced by Seifermann et al. [35] for transforming the DFD into analyzable DAGs. Additionally, xDECAF allows the definition of labels, which add semantic information to the otherwise text-only labeled flows. Labels can also be assigned to nodes in order to add semantic information too. Furthermore, one can define a set of rules the system has to adhere to. These rules are called constraints. Altogether, the framework can propagate the semantic labels through the traversable DAGs, representing the flow of data in the system. The DAGs can then be checked against the constraints to determine which DFD elements violate which constraint.

The semantic labels are grouped into *label types*, with their values called *label type values*. Henceforth, semantic labels may be written as *[Label Type].[Label Type Value]* in order to simplify the understanding. Furthermore, we use the term *label* as a term for concrete semantic labels, i.e. combinations of a label type and a label type value, in this thesis. If the label type is clear from the context, the term label may also be used with only the label type value.

To define what labels are propagated, one can define the behavior of output ports. Their behavior can be defined using a combination of three possible assignments. First, *forward [InputName]* defines that all data coming from the input flow named [InputName] shall be forwarded. Second, *set [Label]* defines that the flow leaving the output port in question gains the label [Label]. This assignment is equivalent to saying the label [Label] is assigned to the flow that leaves the output port in question. Third, *unset [Label]* is the inverse of *set [Label]*, defining that any flow that previously had the label [Label] assigned now does not. Each output port can have one *forward* assignment and unlimited *set* and *unset* assignments. For example, a node has an input port that receives the flow "password", and an output port with the following assignments.

```
forward password set Hashed.Hashed unset Hashed.NotHashed
```[1]

[1]Technically, each assignment must be written in its own line. To improve readability, we omitted the new lines in the example.

This assignment states that the flow leaving the output port forwards all labels coming through the input "password". Furthermore, this flow additionally receives the label *Hashed.Hashed* and looses the label *Hashed.NotHashed*.

After the label are propagated, the DAG can be checked against a set of defined constraints. The constraints are defined by using a domain-specific language (DSL) [15], which uses the following basic syntax:

```
[Source Selector] neverFlows [Destination Selector]
```

In most cases, a constraint determines whether a label is propagated to nodes of a certain label. In this case, the source selector is defined as `data [LabelType].[LabelTypeValue]` and the destination selector as `vertex [LabelType].[LabelTypeValue]`. For example, the constraint `data DataForm.Raw neverFlows vertex Location.nonEU` determines whether a flow propagates the label *DataForm.Raw* to a node with the label *Location.nonEU*. Furthermore, it is possible to define the destination selector not just for nodes with a label, but the node type, e.g., *process* or *data store*. In this case, the destination selector is defined as `vertex type [NodeType]`. For example, the constraints `data DataForm.Raw neverFlows vertex type STORE` determines whether a flow propagates the label *DataForm.Raw* to a data store. Additionally, it is possible to combine multiple selectors using implicit *AND* or *OR* notations. For example, if a constraint determines whether two or more specific labels are propagated at the same time, the source selector must be defined by concatenating the two or more source selectors in question, e.g., `data [LabelTypeOne].[ValueOne] data [LabelTypeTwo].[LabelTwo]`. Similarly, if a constraint determines whether any of two or more labels is propagated, the source selector must be defined by joining the two or more specific labels, e.g., `data [LabelTypeOne].[ValueOne],[LabelTypeTwo].[LabelTwo]`. Furthermore, any of the mentioned selectors may be inverted by adding an exclamation symbol to the front of the label or node type. Altogether, the constraint

```
data Sensitivity.Sensitive,Sensitivity.HighlySensitive
                        neverFlows
     vertex type STORE vertex !Access.Restricted
```

checks whether either *sensitive* or *highly sensitive* data is stored in data stores that do not restrict access.

2.2. Threat Modeling

Another area that uses DFDs is *threat modeling*. It refers to a variety of approaches intended to proactively identify and address potential issues at early stages of the development cycle. Most of these approaches consist of four steps [29, 43]. The first step is to *scope your work*, meaning to gain an understanding of what one is working on, e.g., by drawing diagrams.

Secondly, one must *determine threats* which can be supported by threat taxonomies. Third, one must *determine countermeasures and mitigation*. The final step is to *assess your work*, meaning to reflect on one's work and iterating.

Threat modeling approaches can be grouped into two categories. *Attacker-centric* approaches focus on potential negative scenarios and analyze under which conditions these scenarios can become reality. These approaches often use *fault trees* (or *attack trees*) to visualize and document potential threats. *System-centric* approaches focus on the system under analysis and try to predict what vulnerabilities this system may have. These approaches often use DFDs to model systems, since "problems tend to follow the data flow, not the control flow" [37]. Software like the *Microsoft Threat Modeling Tool* [23] and *OWASP Threat Dragon* [28] use DFDs in combination with the security taxonomies to automatically generate possible threats depending on the system under analysis. These application use pattern matching to predict threats, leading to "*threat explosions*", which is the phenomenon that they generate a huge amount of irrelevant threats, i.e. false predictions. Subsequently, all generated threats have to be filtered and treated by a human.

One of the most prominent taxonomies for threat modeling is *STRIDE*. This taxonomy was defined alongside the SDL [18] and consists of the six threat types *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service*, and *Elevation of Privilege*. As the threat types suggest, STRIDE focuses on threats *to* a system, i.e. security.

In contrast to security threat modeling, where threats commonly originate outside the system and organization that builds it, *privacy threat modeling* focuses on threats to user's privacy. It assesses the threats against the data of individuals, including from the system itself and the organization that builds the system in question.

LINDDUN [9, 38] is a system-centric, DFD-based framework inspired by STRIDE. It provides a taxonomy of goals and threats in the context of privacy. The acronym LINDDUN stands for the seven identified threat types *Linking, Identifying, Non-repudiation, Detecting, Data Disclosure, Unawareness and Unintervenability, and Non-compliance*. Each of these threat types contain multiple distinct threats, some of which contain further sub-threats. This hierarchy is visualized in form of *threat trees* [31]. Figure 2.2 shows such a threat tree, namely the one for the treat type *Unawareness and Unintervenability* [31]. Moreover, all threat trees can also be viewed with more details, such as examples and impact.

Since LINDDUN was first introduced in 2010 [9], a number of papers have been published that have improved or extended the taxonomy. In 2020, a new variant called *LINDDUN Go* [50] was introduced, followed by a new version in 2025 [38] that refactored the original taxonomy and combined it with LINDDUN Go, leading to a unified taxonomy. Along with the taxonomy, two techniques have been defined. The first technique based on the standard variant of LINDDUN is *LINDDUN Pro* and defines how to use the taxonomy to execute a privacy threat analysis. This technique is *per-interaction*, meaning that every combination of (node, flow, node) must be checked for any of the threats. In contrast, the second technique *LINDDUN Go* [50, 38] comes in form of a card deck [21] and is executed *per-threat*. The card deck includes 33 cards of possible threats with their respective hotspot, source and other detailed information. It is intended to be used in a structured brainstorming session

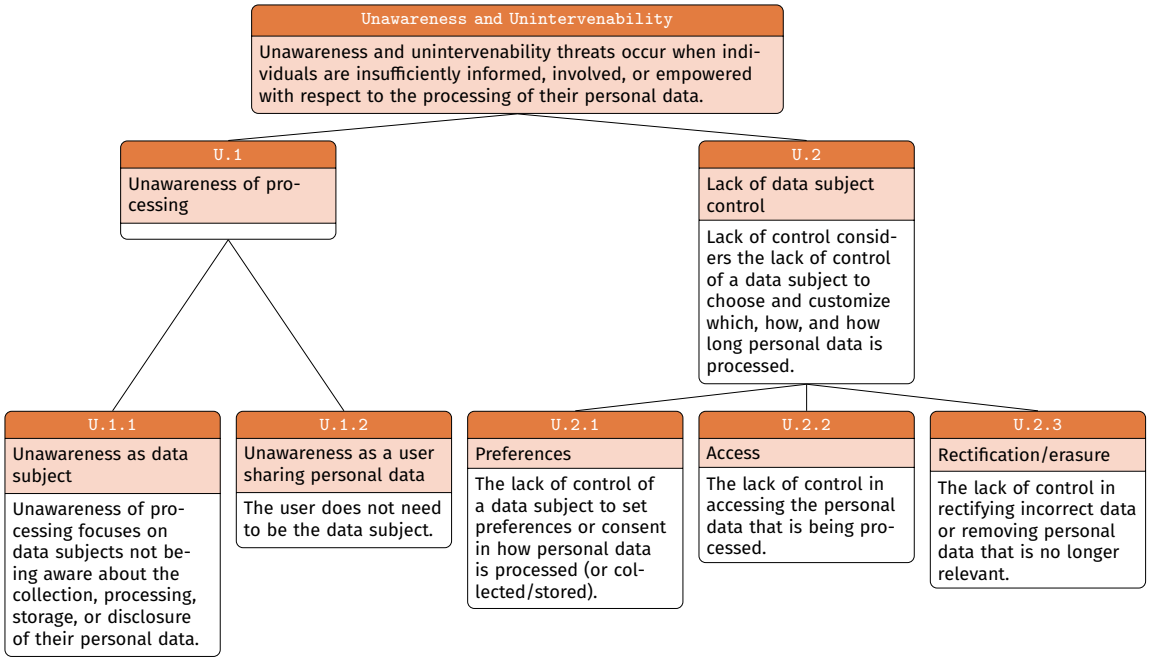


Figure 2.2.: Threat Tree *Unawareness and Unintervenability* [31].

to help with the privacy assessment process. Industry professionals stated that LINDDUN Go simplifies the process, and that the cards make the content less heavy and easier to understand [50].

3. Related Work

This chapter presents the work that partly relates to this thesis. We also discuss how this thesis differs from the related work. In Section 3.1, we discuss related work in the context of threat modeling. We then discuss related work on the topic of data flow analysis in Section 3.2.

3.1. Threat Modeling

A number of papers have been published on the topic of threat modeling. For example, there are various threat modeling approaches and taxonomies. *PRIAM* [7] defines a two-phase process to conduct a Privacy Risk Analysis (PRA) systematically and traceable. Its focus is on customization and open-endedness in order to be adopted to any case. However, it does not provide a concrete threat taxonomy.

The PANOPTIC™ Privacy Threat Model [16] defines a privacy threat taxonomy consisting of two parts: Privacy Contextual Domains (PCs) and Privacy Activities (PAs). PCs reflect various aspects of the socio-technical environment, while PAs relate to the different types of potential privacy attack components. However, this approach does not define how to elicit threats using its taxonomy. Furthermore, the taxonomy is too extensive and detailed to be used for this thesis.

UsersFirst [47] describes an iterative approach that focuses on notices and choices presented to an individual. The approach consists of two phases: In the first phase (design phase), organizations identify and design notice and choice interfaces. In the second phase (analysis phase), organizations evaluate the resulting designs against the provided taxonomy of usability threats. Since UsersFirst focuses on a very specific type of privacy threat, it is not used in this thesis.

PITA [45] is a attacker-centric approach to assess potential privacy impacts. Its main artifacts are privacy impact trees, which are enriched with a *privacy footprint*. These trees can be aggregated to compute an overall privacy footprint. Additionally, PITA provides a taxonomy that distinguishes between privacy impact types that pertain data subject identity, data subject treatment, data subject control and treatment of personal data. However, as PITA is not based on DFDs, it is not applicable to our approach.

However, providing extensive taxonomies is only one part of improving threat modeling. There are significant efforts to simplify and automate certain steps of the threat modeling process. For example, Berger et al. presented an approach that automatically extracts threats

from DFDs [3]. This approach allows splitting the responsibility of analyzing a system between security experts, who define knowledge bases with rules based on a graph query language, and system experts, who create DFDs with their Extended Data Flow Diagrams (EDFDs) notation. The extended DFDs are automatically transformed into graphs and checked against the rules of the knowledge bases. Our approach differs as the underlying framework propagates the labels through the DFD, simulating the flow of data instead of analyzing a graph directly.

Sion et al. presented an approach that improves the semantic quality, traceability, separation of concerns, and dynamism in threat elicitation [40]. This is achieved by defining security solutions and enabling the traceability of security effects to the security solutions. Furthermore, the approach enables independent evolution of the security solution catalog and continuous threat assessment in their developed tooling. *SPARTA* [41] extends this approach by adding an automated risk based threat prioritization. However, this approach differs from ours as it uses a graph-based pattern language instead of label propagation.

PILLAR [24] is a tool that leverages large language models (LLM) to analyze a system in all four essential phases of the threat modeling process based on the LINDDUN taxonomy. For example, it can create DFDs from unstructured textual inputs and elicit privacy threats. Furthermore, it simulates multi-agent collaboration to improve its results, and is capable of prioritizing risks and recommending privacy controls. With these features, PILLAR is evaluated with a case study, showing an overall precision of 85.71% and a false-positive-rate of 19.05%. Although the results seem promising, PILLAR is based on LLMs and therefore inherit their weaknesses.

3.2. Data Flow Analysis

Similar to this work, integrating other research topics with data flow analysis may be beneficial. Since xDECAF is designed for extensibility, it has previously received such work. For example, *ABUNAI* [14] combines data flow analysis with architecture-based uncertainty propagation. It can classify types of uncertainty, propagate the uncertainty sources through the model and then analyze it. *MDPA* [4, 6] integrates legal information from the GDPR to analyze a system and make statements about its data privacy. It divides the approach into two views. One for legal experts and the other for software engineers, enabling both to work together while sharing a common system description. Niehues et al. present an approach that assists in mitigating confidentiality violations under uncertainty [26, 27]. To this end, it uses machine learning techniques to evaluate the severity of identified violations and to automatically repair them. This approach differs from all three described approaches, as it aims to automate certain steps of the threat modeling process, thus investigating the applicability of xDECAF for threat modeling.

4. Systematic Labeling Approach for Automatic System Analysis

In this chapter, we present our approach for systematically including semantic information in DFDs for system-centric threat analysis. We begin with an overview of the approach and its features in Section 4.1. Section 4.2 presents the *threat knowledge interface (TKI)* , which we have defined to enable the implementation of the features independently of any specific taxonomy. Section 4.3 then presents the implementation of the approach.

4.1. Overview

Our approach aims to improve system-centric threat modeling at design time by leveraging the propagation and constraint-checking capabilities of xDECAF. To this end, we developed an extension for xDECAF's web editor that streamlines the process of including labels into DFDs. Since the approach is primarily user-facing, the extension is implemented only on the frontend. In order for the approach to be independent of any taxonomy, we define an abstract *threat knowledge interface (TKI)*. The TKI includes three additions to the existing data model, enabling us to improve the process further. We list and detail the features of the approach in the order of user-interactions within the extension.

F1: Extend existing data model

F2: Import threat knowledge

F3: Streamlined inclusion of labels into existing DFDs

- F3.1: Iterative labeling process
- F3.2: Labeling process navigation
- F3.3: Easy label assignment to nodes
- F3.4: Easy label assignment to output ports
- F3.5: Encourage assignment of labels to intended elements
- F3.6: Prohibit assignment of mutually exclusive labels at the same time
- F3.7: Show additional information
- F3.8: Automatic export when labeling process is complete

F4: Export threats

In the context of threat modeling, we identified three additions to the existing data model that help streamline the labeling process (Feature F1). First, we noticed that labels are typically intended to be assigned to either nodes or flows. Reducing the set of elements a user must label reduces the overall time of the approach. Second, some labels are mutually exclusive. For example, it generally does not make sense for data to be both in plaintext and encrypted simultaneously. Because the approach can model such exclusive relationships, invalid states can be prevented. Third, because labels usually consist of only a few words, they can easily be misinterpreted, causing nuances can be lost. To mitigate possible misunderstandings, the approach displays detailed information per label.

In order for the approach to be independent of any specific threat taxonomy, we define the TKI that allows the use different threat knowledge bases. Subsequently, the web editor of xDECAF also allows to import such a interface file (Feature F2). This interface is required to be in JavaScript Object Notation (JSON) because the main programming language of the web editor is TypeScript.

Once a user has imported a TKI file to a DFD, they must add labels to its elements in order for the framework to analyze the flow of data in the system (Feature F3). We refer to the process of systematically including labels as *labeling process*. The labeling process presents the main part of the approach. Thus, its features are split into multiple sub-features.

The process itself is inspired by LINDDUN Go [50, 36] and its per-threat iterative process. Similarly, the labeling process works by iteratively presenting the user with a single label to be included at relevant elements throughout the DFD, keeping the approach lightweight and ensuring that no label is overlooked (Feature F3 and F3.1). The labeling process is started by clicking a button; and advanced to the next label by clicking a button as well (Feature F3.2). Once the labeling process is started, the user must assign labels to elements in the DFD. Assigning labels to nodes and flows (Features F3.3 and F3.4) is one of the main factors contributing to the ease of use of our extension. This is because the number of times this action is performed correlates to the number of labels and DFD elements. We greatly simplify this action greatly by automatically assigning the current label of the labeling process with a single right-click. Although right-clicking instead of left-clicking contradicts common practice, left-clicking on an output port is already reserved for opening its forwarding behavior. To visualize the progress made per label, the labeling process highlights DFD elements with the current label. We decided against using a different color to indicate that a label can *not* be assigned to a DFD element to reduce visual clutter.

Furthermore, the extension leverages the new additions to the data model to further improve the labeling process. For example, it encourages the assignment of labels to their intended element by highlighting said elements in a bright green color (Feature F3.5). It also highlights elements that have colliding labels, i.e. labels that exclude the current label of the labeling process, assigned to it in a bright red color signaling a warning. However, users may still want to change their mind. Therefore, the extension allows a user to click on elements with colliding labels, which prompts the user to decide for either label. This ensures that mutually exclusive labels are never assigned at the same time (Feature F3.6) while allowing

users freedom in their usage of the extension. In order to mitigate misunderstandings and to educate the user on the usage of labels, the extension displays the provided additional information on user interaction (Feature F3.7). Specifically, when the user hovers over the help-icon, the information is displayed. This allows the user to have easy, repeated, on demand access to the information, without permanently obstructing the view.

Finally, once the labeling process is complete, it shows a button that finishes the process by downloading the DFD file and analyzing the labeled DFD (Feature F3.8). It also downloads a csv file containing a table of the analyzed threats and the element where the violation happens (Feature F4). This table embeds the approach in the threat modeling cycle by allowing users to use the analyzed threats outside of the web editor.

We further discuss the usage of colors to visualize the state of a DFD element. In order to improve intuition, we selected a traffic light color scheme: Green elements visualize that a label may be assigned; yellow elements visualize that the label is already assigned; and red elements visualize that the label will collide with a previously assigned label. The described order also reflects the hierarchy of the colors, e.g., an element that is both intended and contains colliding labels is always colored red.

4.2. Threat Knowledge Interface

As explained in Section 4.1, the approach extends the xDECAF web editor's data model for threat modeling purposes. Additionally, the approach is designed to be independent of any specific taxonomy. To this end, the threat knowledge interface (TKI) is based on the *threat knowledge metamodel (TKM)*, which was introduced in [38]. The TKM provides support for capturing the threat knowledge unambiguously in a structured format. Moreover, it is defined generically enough to capture threat knowledge bases other than LINDDUN.

Figure 4.1 shows the metamodel of the TKI. Although both the TKM and xDECAF web editor's data model contain many more classes and attributes, this metamodel provides a simplified view that is sufficient for the TKI. The gray classes in the metamodel are directly inspired by the TKM. Specifically, the *ThreatKnowledge* provides the identity of a threat type collection, e.g., LINDDUN. *ThreatType* and *Threat* are the elements of threat trees. For example, in the threat tree *Unawareness and Unintervenability* (Figure 2.2) is the root node the threat type, and all remaining nodes are threats. Furthermore, the class *Characteristic* represents the base class that contains most information about the context of the threats, e.g., *Criterion*, *Impact*, etc.

The elements that are part of xDECAF web editor's data model are on the left side of the metamodel. The simplified class *DataFlowDiagram* represents the main class containing every label. Each combination of *LabelTypeValue* and respective *LabelType* is a single label. Importantly, constraints are not part of the DFD itself, but rather reference the labels of the DFD. Consequently, labels and constraints must be defined in accordance to each other.

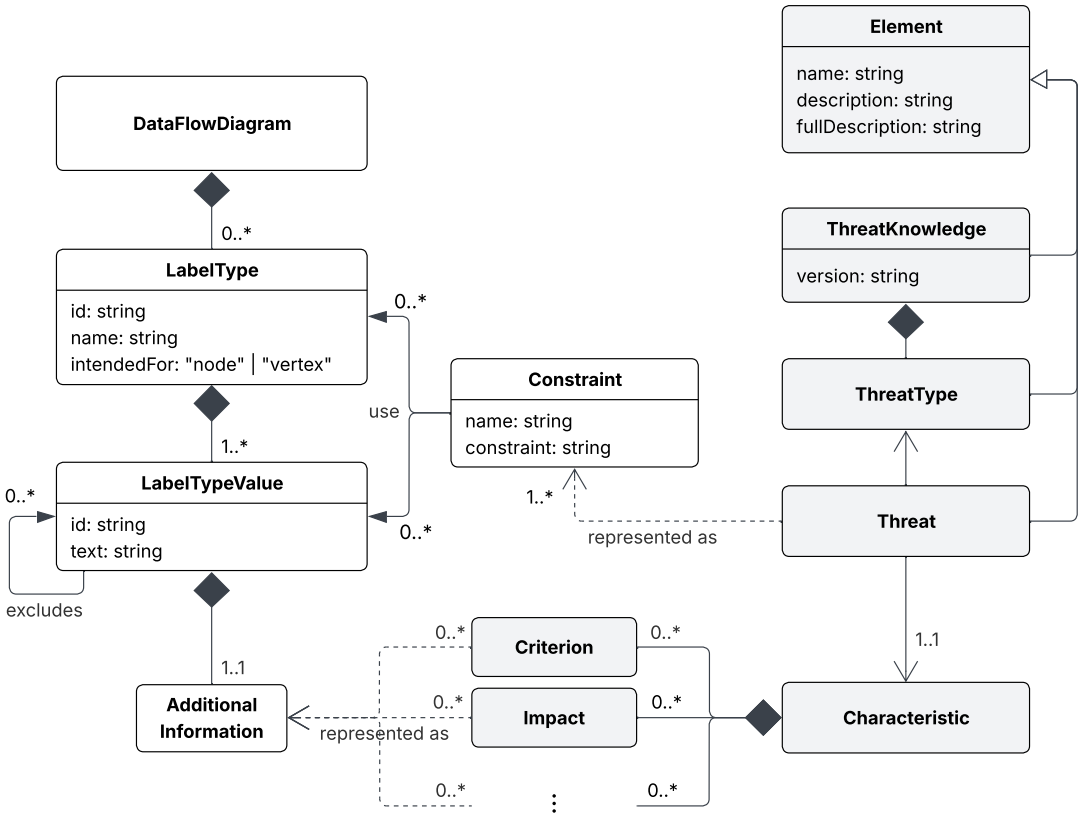


Figure 4.1.: Informal metamodel of the TKI. Gray elements are inspired by the TKM.

As previously mentioned, we are extending the existing data model by adding three attributes. First, we add the attribute *intendedFor* to the class *LabelType* to store the DFD element that a label is intended for. Second, since labels can exclude other labels, the attribute *excludes* of the class *LabelTypeValue* stores this information. Third, the attribute *AdditionalInformation*, which is part of the class *LabelTypeValue*, serves as an extensive description of the label.

The most important aspect of the metamodel is understanding how the flexible TKM is translated into a data structure that can be used by xDECAF's web editor. First, each characteristic may reference one or more labels as part of its criterion, impact, etc. From these characteristics, we can extract the labels, i.e. *LabelType* and *LabelTypeValue*, and add the context to the label's additional information. To define a constraint based on a threat, the threat's characteristic is used as a reference.

From the TKI metamodel, we derive the TKI itself. Listing 1 shows an example TKI in JSON syntax. The attributes *threatKnowledgeName* and *threatKnowledgeVersion* represent the class *ThreatKnowledge* of the metamodel. These attributes are strings that allow for the unique identification of the threat knowledge variants and complex versioning, e.g., semantic versioning [19]. The string *xDecafVersion* is also included, since the web editor may be further developed, possibly changing attributes or the constraint DSL.

```
{
    "threatKnowledgeName": "LINDDUN",
    "threatKnowledgeVersion": "1.0.1",
    "xDecafVersion": "1",
    "labels": [
        {
            "id": "IqL9df0D",
            "name": "DataIntegrity",
            "intendedFor": "Flow",
            "values": [
                {
                    "id": "AcyIOqAF",
                    "text": "Signed",
                    "excludes": [],
                    "additionalInformation": "..."
                },
            ]
        }
    ],
    "constraints": [
        {
            "name": "NonRepudiationThroughSignature",
            "constraint": "data DataIntegrity.Signed neverFlows
                vertex NodeControl.Organization"
        },
    ]
}
```

Listing 1: Example TKI in JSON syntax.

The array *labels* contains an array of *LabelType*s. Objects of the class *LabelType* contain a string attribute called *id* that uniquely identifies them, as well as a string attribute called *name* that stores their display name. *LabelType*s also contain the attribute *intendedFor*, which can only be one of two allowed string values: *"Flow"* or *"Vertex"*. Furthermore, *LabelType*s contain an array *values* that contains the concrete *LabelTypeValue*s.

Objects of the class *LabelTypeValue* are identified by their string attribute *id*, and displayed by their string attribute *text*. They also contain an attribute *excludes*, which is an array of objects containing both the *id* of a *LabelType* and the *id* of a *LabelTypeValue*. The attribute *additionalInformation* of the class *LabelTypeValue* is a string, but must be compatible with Markdown syntax [13] to allow for rich formatting with dynamic content.

Finally, the array *constraints* of the TKI contains an array of *Constraint*s, which are identified via their string attribute *name*. The string attribute *constraint* stores the concrete constraint in the constraint DSL.

4.3. xDECAF Web Editor Extension

In Section 4.1, we presented an overview of the extension's main features. In Section 4.2, we then explained the TKI and the data it provides. This section explains the concrete details of the implementation, including how the TKI is used, in the order of the previously explained features.

The internal data types (F1) of the web editor were extended by creating a new interface that inherits the existing interfaces for label types and label type values. Since all labels are stored in a registry, the new interface can still be used with the existing registry implementation. Consequently, both types of labels can be used simultaneously. Another advantage of this implementation is that the new attributes can be imported and exported via the existing DFD import and export functionality without modifying any code. However, the new attributes must be removed before requesting an analysis from the backend in order to ensure compatibility.

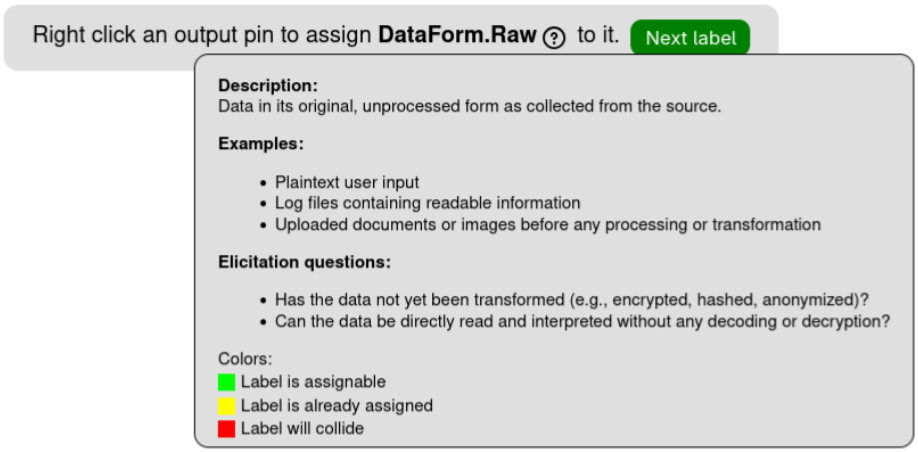
We also implemented the import of a TKI file for existing JSON (F2). In this process, the web editor removes all existing labels in the DFD, since the previously used labels may conflict with the new ones. This includes the behavior of the output ports, which is completely removed except for any *forward* statements. Then, the label registry is emptied and populated with the new labels. Additionally, all constraints are replaced by those defined in the TKI file.

The extension adds a button to the existing user interface (UI) that starts the labeling process when clicked (F3.2). Figure 4.2 shows the UI for the labeling process (F3.1) for the example label *DataForm.Raw*. It also shows the button that is used to advance to the next label (F3.2). Any implementation regarding the UI, particularly the labeling process, utilize existing styles to maintain visual consistency. Furthermore, the UI is kept simple to avoid visual clutter that could distract from the task.

Assigning labels to nodes (F3.3) is straightforward because only the reference to the label is required. In contrast, assigning labels to output ports is more complex because the internal representation is a string that stores the behavior (F3.4). The label is assigned to the port by appending it to the behavior string in a *set* statement. Additionally, since labels may exclude other labels, ports cannot forward excluded labels. Thus, the port behavior is also extended by one *unset* statement per excluded element on a label assignment.

At the start of an iteration, the label type is checked for the attribute *intendedFor*. Using this information, the DFD is traversed, coloring all matching elements in the process (F3.5). Whether a colliding label (F3.6) is assigned to an element is also checked at the start of an iteration. Similar to the label assignment, this process is straightforward for nodes, but challenging for output ports. First, the behavior of the port is analyzed for any *set* statements. Then, the found labels are looked up in the label registry. Only then can the port be checked for collisions. Colliding labels are also checked when a label is assigned. If a collision is detected, the extension presents the user with a dialog explaining the problem. Figure 4.3 shows such a dialog for an element with the label *NodeControl.User*, but where the user still wants to assign the label *NodeControl.Organization*.

(a) Labeling process UI with example label *DataForm.Raw*.



(b) Labeling process UI showing additional information.

Figure 4.2.: Screenshot of the labeling process UI.

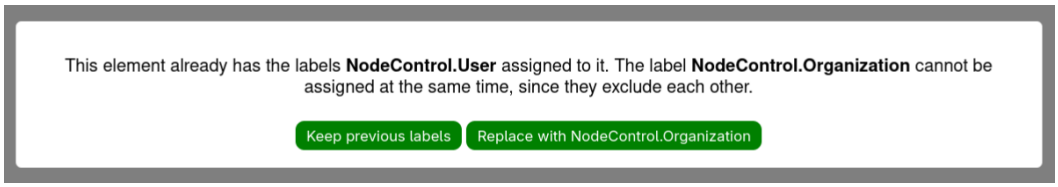


Figure 4.3.: Dialog when a colliding label is assigned.

Figure 4.2b shows the UI when hovering over the help symbol, i.e. question mark. The string containing the additional information is transformed from Markdown to HTML, which enables the extension to display the information in the correct format. Additionally, we include color information for the DFD elements at the bottom of the help interface in case a user wants to look them up.

Finally, once all labels have been iterated, the button on the labeling process is replaced by a new one. Clicking this button automatically downloads the DFD to a file, sends the labeled system to the backend, and exports the resulting threats to a CSV file as soon as the backend responds (F3.8, F4). The threat file includes the ID and name of any threatened element, as well as the name of the violated constraint.

5. Translation of the LINDDUN taxonomy

To showcase the approach explained in Chapter 4, we required an initial TKI. Although many taxonomies may be translated, we translated the unified version of LINDDUN (version `v241203`) [38] due to its comprehensive documentation, methodical evaluation and prominence in the field of privacy threat modeling. We iteratively derived the labels and constraints that are required for the TKI from its threat knowledge. We describe the iterative translation process in Section 5.1, followed by the derived labels in sections 5.2 and 5.3. Finally, we present the translated constraints in 5.4. In total, we derived 34 labels distributed across 9 label types as well as 24 constraints.

5.1. Translation Process

The iterative translation process is visualized in Figure 5.1. Each iteration consisted of four steps: First the LINDDUN threat trees and other literature is read, from which labels are extracted and stored in an intermediate data structure. Transforming the intermediate data structure into a TKI file is done automatically. The extracted labels are enriched with descriptions, examples and elicitation questions. The next step of the translation process is to group similar labels. These groups are then assigned an intended DFD element, i.e. node or flow. Consequently, the label groups are equivalent to label types. Afterwards, the labels are analyzed for relations, most importantly what other labels they exclude. The last step

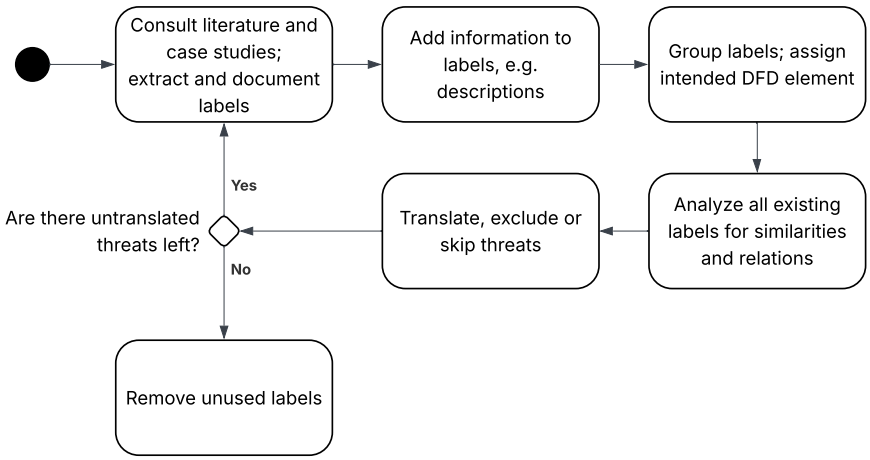


Figure 5.1.: Iterative taxonomy translation process.

of an iteration is the translation of the threats itself into constraints. If the labels are not sufficient to correctly translate a threat, it is skipped. However, some threats are impossible to translate at all. In this case, the threat is documented as *non-translatable* along with the reason. Such threats are subsequently excluded from future iterations. Once all threats are either translated or excluded, the translation process is concluded by removing irrelevant labels. A label is considered irrelevant if it is either used in no constraint or if no relevant label excludes the label in question.

5.2. Data Labels

In this subsection, we present all labels we derived as part of translating the LINDDUN taxonomy. Table 5.1 shows all labels that describe some aspect of data, and therefore are intended to be assigned to data flows. Even though there certainly are more possible values, the presented values suffice to analyze a system for the translated privacy constraints. These values also include the absence of specific values, e.g., *DataInterveniability.None* where a user is neither aware of data collection, nor can they access, control, edit or delete said data. These labels are included in the translation because they actively exclude other labels, and therefore may need to be assigned to remove another label. For example, a system collects data that a user can access and edit. However, this system also derives data from the user data, but the user has no control over the derived data.

Label Type	**Values**	**Sources**
Form	Raw, Encypted, Aggregated, Hashed	[31, 38, 7]
Identifiers	Revealing Attributes, Direct Identifier, Quasi Identifier, Anonym	[31, 38]
Integrity	Signed, Unsigned	[31, 38]
Interveniability	Rectification Possible, Control via Preferences, Accessible, Awareness, None	[31, 38]
Observability	Fully Visible, Metadata Only, Non Observable	[31, 38]
Precision	Strictly Necessary, Excessive Volume, Excessive Data Types, Excessive Frequency, Excessive Retention, Excessive Enrichment	[31, 38, 7]
Sensitivity	Public Data, Personal Data, Personal Data about Other People	[31, 38]
Other	Contains Hidden Data, Has Side Effects	[31, 38]

Table 5.1.: Data labels.

On another note, some of the presented data labels are imprecise. For example, all labels based on cryptographic functions, such as *DataForm.Encrypted* and *DataIntegrity.Signed*, do not cover the entire field of cryptographic research. The security of an encryption depends heavily on context which can currently not be modeled with this approach, e.g., who has access to the decryption key. In order to simplify the approach for the user, we opted to not include such differentiation and instead keep a simple label.

Most values of the label type *Precision* are similar in their extremity, e.g., *Excessive Volume* and *Excessive Retention*. In this translation, anything more than what is strictly necessary is treated as excessive. Furthermore, including more levels than *strictly necessary* and *excessive* would introduce more labels without any advantage in constraint expressiveness. Inversely, all *excessive* labels could be combined into a single label in order to further simplify the translation. However, the LINDDUN taxonomy makes a clear distinction between the presented variants, as all 10 leaf threats from DD.1 to DD.3.4 relate to the various types of unnecessary and excessive data collection and processing. We managed to address all 10 leaf threats with constraints using only the presented five labels, with the objective of simplifying the methodology without diminishing its expressiveness.

In literature and legislation, data *sensitivity* has a variety of values [33, 48, 32] that are not used in this translation, such as *confidential* or *restricted*. However, this approach does not aim to capture data sensitivity perfectly, but rather to simplify the analysis. Therefore, we opted for a simple term that encompasses all kinds of sensitive data.

The label type *Other* serves as a fallback category for labels that shared no similarities with other labels. Only the labels *Contains Hidden Data* and *Has Side Effects* ended up in this category.

5.3. Node Labels

Table 5.2 shows all labels that describe some aspect of nodes and therefore are intended to be assigned to DFD nodes, i.e. processes, data stores or external entities. A fundamental observation is that there are significantly less node labels than data labels. While there are certainly more nuances to who controls a DFD node, we believe that these four node label values are sufficient to reasonably differentiate control in most systems.

Label Type	Values	Sources
Control	User, Organization, Public, ThirdParty	[31, 38, 50]

Table 5.2.: Node labels.

There are certain aspects of nodes that cannot be semantically defined using these four abstract label type values. For example, some regions like the EU dictate stricter security policies than other regions. Such compliance issues are addressed in the LINDDUN taxonomy as part of the threat type *Non-compliance*. However, while this may be technically possible,

automated analysis of these topics would require rigorous legal research, which this thesis cannot achieve at a reasonable level. Moreover, since the TKI can use multiple knowledge bases simultaneously, we decided against a unified approach and instead leave regional differences to future work. Thus, we decided against incorporating regional node labels, such as *NodeRegion.EU* or *NodeRegion.USA*.

One notion of the defined labels that needs to be discussed is the blur of control on end-user devices, i.e. DFD nodes labeled *NodeControl.User*. Some modern operating systems collect an increasing amount of data, to the extend that governments are starting to move towards more privacy-friendly options [10, 46]. This calls into question whether a device owned by a user is really in the users control; and whether such channels actively threaten privacy. This depends greatly on the operating system and software installed on the device in question. Due to a vast number of options with varying levels of privacy implications, we decided to not include these options into our taxonomy. However, if required, the extensibility of the defined interface can be used to extend the existing threat knowledge to include such details.

The label *NodeControl.Public* requires further explanation as well. Although the label type implies that the public has control over a DFD node (however that may be possible), it rather means that nobody can control who controls or accesses the node in question. For example, if an organization publishes user-derived data on their website, this doesn't mean that the website is under public control, but that anyone with internet access can view it. The decision to group the label *Public* into the type *Control* was made on the basis that there were no similar labels it can be grouped with. Another aspect of the label *NodeControl.Public* is that it allows to model and therefore analyze HTTP connections, which is challenging in DFDs [39]. This requires a single DFD node "Internet" labeled with *NodeControl.Public*, as the internet is an unsecure channel [11]. Furthermore, if the HTTP connection is over Transport Layer Security (TLS) (i.e. HTTPS), the flow may be modeled by assigning the labels *DataForm.Encrypted* and *DataObservability.MetadataOnly* to the incoming flow.

5.4. Constraints

When translating the LINDDUN threats, our goal was to stay as close as possible to the threat definition. Subsequently, we did not consider additional literature in order to further specify the constraints. However, since the LINDDUN version this chapter is based on does not only provide the threat trees but also the LINDDUN Go cards, we also considered LINDDUN Go. We did not find a perfect mapping between the threat trees and cards, but their descriptions and examples matched for the most part. From this, we were able to add the card information to the individual threats, which we mostly used to determine the destination selector of constraints. The reason for this is that most threat descriptions do not reveal by themselves where the threat in question originates from, but the cards provide a treat source, e.g., *ORGANIZATIONAL*. However, we want to note that the threat source of LINDDUN Go cards and the destination selectors of threats do not match perfectly in their implications.

	Constraint	Threat(s)
C1	data DataSensitivity.PersonalDataAboutOtherPeople neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	L.2.1.2
C2	data DataIdentifiers.DirectIdentifier neverFlows vertex NodeControl.Organization	I.1.1, I.2.1.1
C3	data DataIntegrity.Signed neverFlows vertex NodeControl.Organization	Nr.1.2
C4	data DataSensitivity.PersonalData neverFlows NodeControl.Public	DD.4.2
C5	data !DataInterveniability.RectificationPossible neverFlows vertex NodeControl.Organization	U.2.3
C6	data DataPrecision.ExcessiveDataTypes neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	DD.1.1, DD.1.2, DD.3.2, DD.1.3, Nc.1.1.2

Table 5.3.: Excerpt of the translated constraints.

Table 5.3 shows a excerpt of the translated constraints and what LINDDUN threats they can identify. The full list of constraints can be viewed in Section A.1. The table omits the name of the constraints to improve readability. Furthermore, the names usually only describe the constraint and are therefore insignificant to this thesis. However, the constraints in the excerpt are enumerated in order to reference them. For example, constraint C1 determines if personal data from non-users flows to a node controlled by the organization or a third party. This constraint covers threat L.2.1.2 which states that "[t]he combination of records from multiple different sources enables building a more comprehensive profile on an individual" [31]. For example, uploading "contact lists [] to services allow service providers to indirectly reconstruct the social graph of the involved individuals" [31].

Some constraints cover multiple threats. For example, constraint D6 can identify multiple threats by determining whether a service collects more data types than what is strictly necessary for the functionality. This covers the data disclosure threats DD.1.1, DD.1.2 and DD.1.3 by determining whether the service itself collects the data, and threat DD.3.1 by determining whether this data is "propagated to other services where it is not needed" [31]. Furthermore, it covers threat Nc.1.1.2 which determines violations of the data minimization principle for legislations such as the GDPR.

Table 5.4 shows all the constraints that could not be translated. All 18 non-leaf threats were not translated because they do not provide enough information. The threat trees visualize this particularly well. While leaf threats often have multiple detailed descriptions, including examples, criteria and additional information, non-leaf threats have typically only an ID, name and short description. Although the non-leaf threats are certainly helpful when consulting the LINDDUN taxonomy itself, our approach does not require constraints to be grouped into categories.

Reason	Threat(s)
Threat is not a leaf	L.1, L.2, I.1, I.2, I.2.1, DD.1, DD.2, DD.3, DD.4, DD.4.1, U.1, U.2, Nc.1, Nc.1.1, Nc.1.1.3, L.2.1, L.2.2, Nr.1
Threat cannot be modeled in a DFD	Nc.2, Nc.3, Nc.4, I.2.3, L.2.2.3, L.2.2.2, D.3, D.1
Threat is too broad	Nc.1.1.1, Nc.1.1.3.1, Nc.1.2, Nc.1.1.3.2, Nr.1.1, Nr.1.3

Table 5.4.: Non-translated constraints.

Some threats require information that cannot be modeled within a DFD and therefore be analyzed in a reasonable way. For example, LINDDUN threat I.2.3 "*The data subject is distinguishable from others*" determines whether the anonymity set is too small, which would allow singling out an individual data subject [31]. Determining whether this threat applies requires knowledge about both the data set and the uniqueness of the individuals. While this may be possible to model within a DFD, it certainly is challenging to accurately reason about.

Finally, some threats are described too broadly. For example, threat Nc.2 "Generic regulatory non-compliance" does not specify any policies precisely, but more so in a general way. Its criteria states "Does the system, or its processing activities, violate one or more rules in [jurisdictions with specific rules for personal data processing (e.g., the EU)]?" [31]. This certainly is important to mind in the general privacy threat modeling context, but far more complex than what this thesis is capable of doing with only a few constraints. Furthermore, the proposed approach can be extended to include such compliance constraints as previously discussed.

6. Evaluation

In this chapter, we present the evaluation of the proposed approach. This evaluation is designed to be comparable to the study of LINDDUN [38], in order to evaluate its improvement to the threat modeling process. We summarize the design of the aforementioned baseline study in Section 6.1. Section 6.2 explains the goals, questions and metrics (GQM) [1] which were used to derive the design of the evaluation. In Section 6.3, we present the design of the user study We present the findings and discuss them in comparison to the baseline study in Section 6.4. We discuss the threats to validity in Section 6.5, followed by the limitations of the proposed approach in Section 6.6.

6.1. Baseline Study

Article [38] presents an empirical study conducted in 2023 that evaluated the new unified LINDDUN threat knowledge. In the study, eleven master's students applied the LINDDUN threat knowledge to an application case. The students had no prior experience with privacy threat modeling. However, they received introductory lectures on privacy by design and LINDDUN before conducting the privacy threat analysis. Furthermore, they worked extensively on the application case in the context of the educational program before the study. To ensure data quality, three reports were excluded from the final dataset.

From this data, the baseline study used Fleiss' kappa to measure the inter-rater agreement on the elicited threats with a confidence interval of $p \leq 0.05$. Additionally, Fleiss' kappa was also computed for each threat type. The kappas are summarized in Table 6.1. The baselines study also shows that the participants identified a mean of 30 and a median of 27 threats overall.

6.2. GQM

The evaluation is designed to allow the comparison of the proposed approach to the baseline study. We use a *Goal-Question-Metric (GQM)* [1] plan in order to ensure the evaluation's quality. The goals, questions and metrics of the GQM plan are summarized in Table 6.2 and explained in more detail in subsections 6.2.1, 6.2.2, 6.2.3.

	Fleiss κ	p-value
Linking	0.0894	0.0766
Identifying	-0.00179	0.972
Non-repudiation	0.316	4.01e-10
Detecting	0.13	0.0131
Data Disclosure	0.252	6.25e-07
Unawareness and Unintervenability	0.286	0.000723
Non-compliance	0.345	4.41e-05
All types	0.187	0

Table 6.1.: Fleiss' kappa scores of the baseline study [38].

G1	**Evaluating the efficiency of the proposed approach.**
Q1.1	How long does it take to analyze a system using the proposed approach?
	M1.1.1 Mean completion time and standard deviation
	M1.1.2 Mean labels per hour and standard deviation
	M1.1.3 Mean threats per hour and standard deviation
G2	**Evaluating the accuracy of the proposed approach and translation.**
Q2.1	How relevant are the identified threats of the proposed approach?
	M2.1.1 Precision (Micro-Averaging)
	M2.1.2 Recall (Micro-Averaging)
Q2.2	How different are different users' resulting threats of the same system?
	M2.2.1 Mean and median amount of threats
	M2.2.2 Agreement of threats (Fleiss' kappa)
G3	**Evaluating the completeness of the translation.**
Q3.1	How many LINDDUN threats can be identified with the translation?
	M3.1.1 Coverage of LINDDUN threats by xDECAF constraints

Table 6.2.: The Goals (ID starting with G), Questions (ID starting with Q) and Metrics (ID starting with M).

6.2.1. Efficiency

The proposed approach aims to improve upon the time-intensive nature of threat modeling approaches. Thus we determine its success in this regard (**G1**). In order to estimate whether the proposed approach reduces the required time to analyze a system, we determine the completion time of the task per participant. The per-participant values are used to compute the mean and standard deviation (M3.2.1). Additionally, we determine the amount of labels and threats per hour (M3.1.2, M3.1.3). These two metrics allows us to estimate how much effort the proposed approach requires for smaller or larger DFDs.

6.2.2. Accuracy

Determining the accuracy (**G2**) of the proposed approach is important, as this approach should only identify relevant threats. To this end, we determine the multi-class classification measures *precision* (M2.1.1) and *recall* (M2.1.2) with *micro-averaging* [42] per threat type and for all threats together. The metrics are calculated as follows:

$$Precision_\mu = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fp_i)} \qquad Recall_\mu = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fn_i)}$$

The values tp_i, fp_i and fn_i are the true positive, false positive and false negative counts of class C_i. The confusion matrix (or error matrix) for binary classification can be seen in Table 6.3 to visualize the classes true positive, false positive and false negative. *Predicted Positive* and *Predicted Negative* represent whether a participants solution identifies a threat or not. *Real positive* and *Real negative* represent whether the system actually experiences this threat or not. Calculating the real values requires a reference solution, which we explain in Subsection 6.3.3.

	Predicted Positive	**Predicted Negative**
Real Positive (*P*)	True positive (tp)	False negative (fn)
Real Negative (*N*)	False positive (fp)	True negative (tn)

Table 6.3.: Confusion Matrix.

Furthermore, we want to quantify how different users' results differ on the level of identified threats (M2.2.2). Although there are multiple different ways to quantify inter-rater agreement, we compute Fleiss' kappa [12, 25] because it was computed as a result of the baseline study. Subsequently, we can compare the agreement of the participants who used the *per-interaction* variant of LINDDUN to the agreement of the participants who used the proposed approach. Fleiss kappa κ and its values p_o and p_e are computed as follows.

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$
$$p_o = \frac{1}{N \cdot n \cdot (n-1)} \left(\left(\sum_{i=1}^{N} \sum_{j}^{k} n_{ij}^2 \right) - N \cdot n \right)$$
$$p_e = \sum_{j}^{k} p_j^2$$

The value p_o refers to the observed agreement, p_e to the expected agreement. Furthermore, N corresponds to the total number of elements, n to the number of ratings per element, and k to the number of categories. The element n_{ij} represents the number of raters who assigned the i-th element to the j-th category; and $p_j = \frac{1}{N \cdot n} \sum_{i}^{N} nij$ the relative amount the j-th answer was selected. The values of κ range from -1 to 1, with values ≤ 1 showing that there's no agreement between the raters other than what can be expected by chance, and 1 showing perfect agreement between all raters.

For metric M2.2.2, N represents each pair of (ThreatType, ThreatenedElement). Furthermore, n is the number of participants. Importantly, we reduce the categories, i.e. k, to binary decisions analog to the baseline study. Specifically, the resulting threats are reduced to whether the participant agrees that a DFD element is threatened or not.

6.2.3. Completeness

Quantifying the completeness (**G3**) of the proposed approach is important, as non-identifiable threats greatly hinder the applicability of the approach. As a measure of completeness, we calculate the relative amount of threats that can be identified by the proposed approach (M3.1.1). A threat is considered to be identifiable by our approach if it is covered by at least one constraint in the translation. The coverage is calculated as follows:

$$\text{Coverage} = \frac{|T_{\text{covered}}|}{|T|}$$

The variables T represent the set of all threats, while $T_{\text{covered}} \subseteq T$ represent the set of covered threats. Subsequently, the calculated value lies between the values 0 and 1, with 1 representing complete coverage. We further distinguish the metric by reducing the set T to include only the threats of one specific threat type. Since the constraints do not cover any non-leaf threats, we also calculate the coverage with the base set T of only leaf threats. In order to contextualize the representativeness of the results of the evaluation, we also calculate the coverage only considering threats that are part of the reference solution. Importantly, we determine the completeness without empirical data, since it only requires the threats and constraints.

6.3. Study Design

This study was conducted via an online questionnaire, which was sent to several people, including students, research assistants and academic researchers. The subsections are oriented towards the structure of the questionnaire. Starting with a self-assessment, which is described Subsection 6.3.1. Next, we describe the task of the questionnaire in Subsection 6.3.2. In order to determine whether this approach improves upon the threat modeling process, the task of the questionnaire is oriented towards the 2023 LINDDUN baseline study. We describe how we calculate the reference solution based on the baseline study in Subsection 6.3.3. Finally, in Subsection 6.3.4 we name our criteria of excluding data points from the collected questionnaire data.

6.3.1. Self-Assessment

Because we did not have access to a homogeneous group of participants as the baseline study did, the participants were asked to fill out a self-assessment prior to analyzing the system. This enables us not only to account for differences in expertise in our participant group, but also to better reason about comparability between our results and that of the baseline study. As part of the self-assessment, the participants were asked to rate their prior experience in the three areas *threat modeling*, *LINDDUN* and *DFDs* on a scale from 0 (no prior experience) to 4 (expert). Furthermore, the participants were asked to select their current role among the following options: Student (Undergraduate), Student (Graduate), Research Assistant, Industry Practitioner or Academic Researcher. In addition, if neither role sufficed, they had the option to specify a custom role.

6.3.2. Task

Next, the participants were asked to analyze a system using the proposed approach. The system under analysis was the *eConsent* system that was used in the baseline study, as it is the only system that was previously analyzed using the LINDDUN variant the proposed approach is based on. Prior to analyzing the system itself, the participants were shown a video that explains the extension. Afterwards, they were provided with a description of the system, which provides some more context. This description also contains a visual representation of the system, i.e. the DFD seen in Figure 6.1. The full system description can be found in Section A.2. We sourced the DFD and system description from the baseline study.

In order to be able to analyze the system using the approach, we provided the analyzable xDECAF-internal representation of the DFD. To this end, we recreated the DFD of the eConsent application case in the web editor of xDECAF. Since there are no alternate flows, each flow receives its individual output and input port. We also defined the forwarding behavior of the flows by carefully reading the system description and accordingly defining

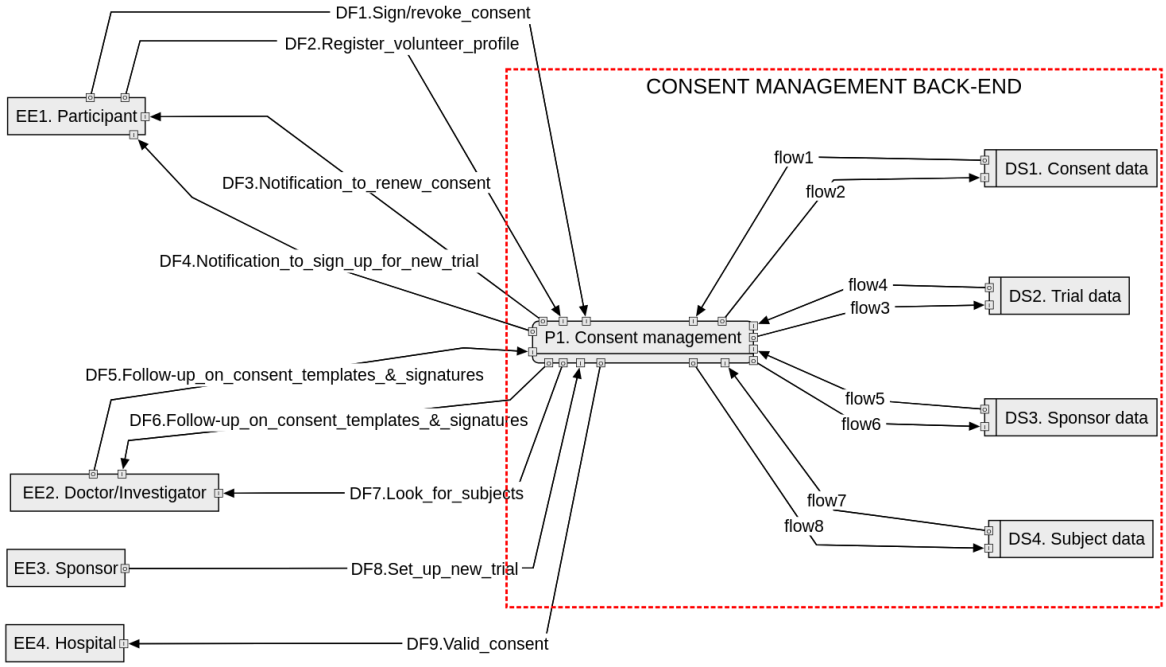


Figure 6.1.: DFD of the eConsent system.

the behavior. Additionally, any flow leaving a data store forwards the data of all incoming flows. Finally, we included the derived labels and translated constraints in the DFD file.

During the analysis, participants were asked to document assumptions they made about the system. This allows us to contextualize their decisions when adding labels to the system. Once the participants were done with the task, they had to upload a DFD file that includes their labels. From this file alone, we can extract both the labels and the analyzed threats, and thus determine metrics M2.1.1 through M2.2.2. In contrast, metrics M1.1.1, M1.1.2 and M1.1.3 require the knowledge about how long each participant spent on the task. In order to improve participant experience, the questionnaire did not determine the time participants spent on the task explicitly. However, we use the time spent on each questionnaire page as an approximation, which is collected by the questionnaire software implicitly.

6.3.3. Reference Solution

Since the eConsent system was previously analyzed, we can use the data of the baseline study to approximate a reference solution. This solution is required so that we can calculate the metrics precision and recall. To determine the reference solution, we first determined the relative number of study participants that assigned a threat to a DFD element for each combination of LINDDUN threat and threatened DFD element. From these assignment rates, we calculated the mean assignment rate per threat type. Only those threats that have a higher assignment rate than their respective mean are part of the reference solution. Consequently, the reference solution contains only threats that are selected above average.

Furthermore, since the proposed approach can not identify threats on data flows, we mapped the threats of the data flows onto their destination node.

It is also important to note that the data of the baseline study does not contain any information about threats to flow1 through flow8, nor about any of the four data stores. Therefore, for metrics M2.1.1 and M2.1.2, we adapted the data of our evaluation to the reference solution by ignoring any threats to the data stores.

6.3.4. Dataset Exclusion Criteria

In order to ensure dataset integrity, some participants answers must be excluded from the dataset. If a participant's answer fits one of the following criteria, it is excluded from the final dataset.

1. **Questionnaire was not completed:** The questionnaire software records all answers, even if a participant does not finish the questionnaire.
2. **No file uploaded or file without changes uploaded:** If a participant does not upload a file or uploads a file without any labels, we cannot draw significant conclusions from their results.
3. **Participant does not want to have their data used for analysis:** As part of the questionnaire, participants are asked whether their data can be used for this study. If they select 'No', their answers must be excluded.

6.4. Evaluation Results and Discussion

In this section, we present and discuss the findings of the study. The remainder of this section describes the results of the self-assessment, while subsections Subsection 6.4.1, Subsection 6.4.2 and Subsection 6.4.3 describe the evaluated results of goals G1, G2 and G3 respectively.

The questionnaire had a total number of 12 submission. However, after filtering the submissions according to the criteria described in Subsection 6.3.4, only five submissions remained. This limited amount of data points greatly limits the validity of this evaluation. Subsequently, all conclusions drawn from this evaluation must be treated with caution. We further discuss this point in Section 6.5.

The roles of the participants based on the self-assessment are shown in Table 6.4. The sum of all roles exceeds the number of participants since participants had the option to select multiple roles. These values show that the participants represent a variety of possible roles. Table 6.5 shows the prior knowledge of the participants based on their self-assessment. As a participant had to select a single value per knowledge area, the sum of all percentages per knowledge area is 100%. In contrast to the roles, the distribution of participant expertise

is mostly concentrated at a specific expertise level, rather than being spread across the levels.

	Count	% of participants
Student (Undergraduate)	2	40%
Student (Graduate)	2	40%
Research Assistant	2	40%
Professional Developer	0	0%
Academic Researcher	1	20%
Other	0	0%

Table 6.4.: Distribution of self-assessed participant roles.

Knowledge Area	No Knowledge	Novice	Competent	Proficient	Expert	∑
Threat Modeling	0%	20%	60%	20%	0%	1
LINDDUN	0%	20%	80%	0%	0%	1
DFDs	0%	0%	20%	20%	60%	1

Table 6.5.: Distribution of self-assessed participant expertise.

6.4.1. Efficiency

Table 6.6 shows the metrics calculated as a result of goal G1. On average, the participants spent 32.6 minutes on the task page of the questionnaire, with a standard deviation of 15.56 minutes. Although we do not have concrete measurements of the comparison study, it states that "[participants] will have spent more [than 2.5h]" [38] on the threat elicitation. This suggests that the proposed approach can improve the efficiency compared to the *per-interaction* variant of LINDDUN. However, the standard deviation also suggests that the efficiency can vary quite a lot from participant to participant.

We also calculate the mean amount of labels the participants and the standard deviation. 2.72 labels per minute implies that users of the proposed approach can add labels to a DFD relatively quickly. This also suggests that our approach may scale well, although this must be further evaluated. Furthermore, we calculate the mean amount of threats per minute, which is 1.89 threats/min. With a standard deviation of 0.5 threats/min, this value suggests that the approach should identify at least one threat per minute. Although an evaluation of LINDDUN measured threats per minute, this measurement was based on the old threat knowledge and can therefore not be compared to the proposed approach reasonably. Furthermore, since these two metrics (labels/hour and threats/hour) are highly dependent on the amount of translated labels and constraints, they can only provide rough estimates of the approach's overall efficiency.

	Mean	Standard Deviation
Completion Time	32.60 minutes	15.56 minutes
Labels per minute	2.72 labels/min	1.37 labels/min
Threats per minute	1.89 threats/min	0.50 threats/min

Table 6.6.: Evaluated efficiency metrics (G1).

6.4.2. Accuracy

Table 6.7 presents the evaluated metrics precision and recall with micro-averaging. As can be seen, the precision is much greater than the recall. Overall, a bit more than 60% of total positive predictions were actually correct. This value varies per threat type, reaching as much as 90.24% for the threat type *Identifying*. However, this measure also reaches as little as 0% for the threat type *Non-compliance*, since no non-compliance threats were identified at all. The overall recall is 14.71%, meaning that of all true positive threats only a few were actually predicted. Although this value also varies per threat type, no recall exceeds 40% for any threat type. As no non-compliance threats were identified, recall also reaches 0% for this threat type. These values suggest that overall the approach requires much further work.

	$Precision_{\mu}$	$Recall_{\mu}$
Linking	68.42%	27.37%
Identifying	95.24%	17.39%
Non-repudiation	80.00%	12.31%
Detecting	50.00%	7.50%
Data Disclosure	50.00%	6.47%
Unawareness and Unintervenability	50.00%	38.18%
Non-compliance	0.00%	0.00%
All types	64.03%	14.71%

Table 6.7.: Evaluated precision (M2.1.1) and recall (M2.1.2).

Table 6.8 shows the agreement on identified threats. It also shows the mean and median identified threats per participant for further context. As one can see, the computed kappa scores can vary per threat type. Some threat types, e.g., Detecting, show an agreement that is about as good as chance, while others, e.g., Linking, show a relatively high agreement. Furthermore, the p value of the two highest kappas are almost zero, suggesting that these kappa scores are unlikely to achieve by chance. In contrast, the lowest kappas also show the highest p-values, suggesting statistical insignificance. The kappa and p-value of the threat

type Non-compliance are impossible to compute because there are no identified threats for this threat type. Overall, the agreement is decent with a low p-value.

When comparing the κ scores of the approach to those of the baseline study, we can see that the overall agreement more than doubles. However, the only threat types where the approach's evaluation shows higher κ are *Linking* and *Identifying*. In contrast, the kappa scores for the threat types *Non-repudiation*, *Detecting* and *Data Disclosure* are less than those of the baseline study. For the threat type *Unawareness and Unintervenability*, both studies show very similar agreement. Since no threats were identified in the threat type *Non-compliance*, no statements can be made about the agreement. It is important to note that the comparison of the κ scores is threatened by the fact that the two evaluations elicit threats for different DFD elements. However, it is impossible to determine κ scores that represent both approaches at their best and enable comparison between them without further evaluation. We discuss this further in Section 6.5

	Threats per participant		Agreement	
	Mean	Median	Fleiss κ	p-value
Linking	5.6	5.0	0.622	3.65e-09
Identifying	2.8	3.0	0.482	4.91e-06
Non-repudiation	2.4	2.0	0.205	0.0523
Detecting	1.6	0.0	0.0118	0.911
Data Disclosure	3.2	3.0	-0.0668	0.526
Unawareness and Unintervenability	5.0	5.0	0.28	0.0079
Non-compliance	0.0	0.0	NaN	NaN
All types	20.6	21.0	0.401	0

Table 6.8.: Evaluated agreement on threats (Q2.2).

6.4.3. Completeness

Table 6.9 shows the amount of threats per threat type and the relative amount of treats that are covered by constraints. Since the proposed approach does not define constraints for non-leaf threats, the coverage is the same or greater when only considering leafs. As shown, the coverage can vary a lot, depending on the threat type. Overall, the coverage of all threats is about half of the threats, reaching about 70% for only leafs. This shows that our approach can identify a majority of the threats, but that there are also a significant number of threats it can not identify.

Turning to Table 6.10, we present the coverage of threats included in the reference solution. As can be seen, the reference solution includes the majority of all threats. Notably, all but

Treat Type	Number of Threats	Number of Leafs	Coverage (Threats)	Coverage (Leafs)
Linking	10	6	40.00%	66.67%
Identifying	9	6	55.56%	83.33%
Non-repudiation	6	5	50.00%	60.00%
Detecting	3	3	33.33%	33.33%
Data Disclosure	18	13	72.22%	100%
Unawareness and Unintervenability	7	5	71.42%	100%
Non-compliance	12	9	16.67%	22.22%
All types	65	47	50.77%	70.21%

Table 6.9.: Coverage of LINDDUN threats.

five leaf threats are present, while 12 non-leaf threats are missing. Consequently, the overall coverage of non-leaf threats is higher than that of all LINDDUN threats, while the overall coverage of leaf threats remains similar. These values suggest that the evaluated metrics precision (M2.1.1) and recall (M2.1.2) should be similar to those of a system in which all threats appear.

Treat Type	Number of Threats	Number of Leafs	Coverage (Threats)	Coverage (Leafs)
Linking	8	6	50%	66.67%
Identifying	7	6	71.43%	83.33%
Non-repudiation	4	4	50%	50%
Detecting	3	3	33.33%	33.33%
Data Disclosure	13	11	84.61%	100%
Unawareness and Unintervenability	5	5	100%	100%
Non-compliance	8	7	25%	28.57%
All types	48	42	62.50%	71.43%

Table 6.10.: Coverage of LINDDUN threats that are part of the reference solution.

6.5. Threats to Validity

Due to the empirical nature of this study, we discuss the threats to construct, internal and external validity as characterized by Runeson et al. [34] in the following paragraphs.

Construct Validity determines whether our metrics can answer our goals. The measurement of efficiency of the proposed approach is determined by the metric of completion time. This metric is an indicator of the efficiency, as it directly measures the time it took a participant to complete the task. However, since we do not measure the exact task completion time but the time the participant spent on the task page of the questionnaire, the results should rather be interpreted as an upper limit. We also use the metric label per hour and threats per hour to estimate how the approach handles system of different sized. However, labels per hour and threats per hour do not take into account additional complexities when analyzing larger systems. Therefore, these metrics may only be used as an approximation.

As a metric of accuracy, we computed the precision and recall with micro-averaging, two metrics which are common metrics for determining accuracy [42]. However, the accuracy of the derived reference solution depends on the accuracy of the data of the baseline study, which we can not determine. Furthermore, we do not correct for chance in the calculation of the reference solution. Therefore, the accuracy of metrics may contain errors. Additionally, we also determine the agreement in the form of Fleiss' kappa of the labels and threats among the participants as a measure of accuracy. Although Fleiss' kappa is an imperfect metric for measuring agreement [52], it is used by the baseline study [38]. We mitigate this threat by further contextualizing the evaluated agreement with the additional metric of mean and median amounts of labels and threats, which is also done in related work [38].

The completeness of the proposed approach is determined by the coverage of the threats. The coverage metric is a commonly used metric for such a goal due to its normalized, interpretable and monotonic definition.

Internal Validity determines whether the results of the evaluation are genuinely attributable to the proposed approach. The validity of the measurement of efficiency is most threatened by the web editor of xDECAF. Since the approach is only an extension, the efficiency of the base application influences the efficiency of the approach. Furthermore, the differences in study setup between the baseline study and this study, e.g., graded study assignment vs. voluntary online questionnaire, further threatens internal validity.

The validity of the accuracy measurement is threatened by a number of factors. First, the metrics precision and recall are based on the estimated reference solution and are therefore influenced by previous data. Second, the baseline study and this evaluation elicit threats for different DFD elements. In part, we mitigate this threat for precision and recall by adapting the data of our evaluation to that of the baseline study. However, this threat remains when comparing the agreement on threats between the baseline study and our evaluation. Third, it is impossible to derive many implementation details from the system description and DFD. Consequently, the participants labels may differ greatly, which affects the level of agreement. Fourth, the participants in the evaluation had a different set of prior knowledge

and training to those of the baseline study. This difference may affect the level of agreement in comparison to that of the baseline study.

The completeness of the proposed approach is determined without any study participation. Therefore, the approach itself is the only influencing factor of the threat coverage.

External Validity determines how generalizable the results of the evaluation are. This validity is greatly threatened by the limited number of participants and their prior knowledge. Furthermore, the evaluation is performed on a single system. Therefore, it is only possible to draw limited conclusions about other systems of different sizes and complexities. Additionally, the DFD provides only a high-level view of the system. The evaluation does not determine whether the results hold for DFDs that model systems closer to the implementation.

6.6. Limitations

Although this work provides a contribution towards the simplification of the privacy threat modeling process, some nuances of other approaches are missing. Importantly, the translation fails to identify a lot of threats. This is partly due to the fact that not all threats were possible to translate into constraints, but also that the existing constraints are not accurate enough to predict the threats correctly.

Another problem that arises in this work is the generalization of certain labels. For example, the label *encrypted* is treated as secure, i.e. impossible to derive data from. While this might be true in certain cases, it does not apply for all systems, e.g., a system that also stores the decryption key along with the data does not provide the same security as a system that stores the decryption key in a secure hardware component on the users device.

7. Conclusion

This chapter concludes this thesis. Section 7.1 summarizes the contribution and evaluation of the thesis, followed by an overview of possible future work in Section 7.2. Finally, we express our thanks in Section 7.3.

7.1. Summary

In this thesis, we presented our approach of using xDECAF for threat modeling purposes. We developed an extension to xDECAF's web editor allowing the user to easily add labels to DFDs. Inspired by LINDDUN Go, the approach is designed to be lightweight while ensuring that the user does not overlook labels. To further aid the process, we extended the data model of xDECAF's web editor to include three additional attributes. This allows the approach to identify DFD elements to which labels should be assigned to, ensure that labels that do not make sense when assigned simultaneously are not assigned simultaneously, and provide users with essential information about labels.

Additionally, we described the TKI, an interface that allows the approach to remain independent of any specific taxonomy. Consequently, the approach may be adopted by more taxonomies. However, to demonstrate the approach, we translated the privacy threat taxonomy LINDDUN for the TKI. We present the translation process, which resulted in 9 label types, 34 labels, and 24 constraints.

To evaluate the approach, we present a study that is designed to allow for comparison to the evaluation of LINDDUN [38]. It measures the efficiency of the proposed approach, as well as the accuracy and completeness of our initial translation. The results suggest that the approach can analyze a system in about 32 minutes, where LINDDUN Pro would require more than 2.5h [38]. However, there are problems with the initial translation, as shown by the results. While the evaluated overall precision is decent with 64.03%, the overall recall is significantly worse with 14.71%. The evaluated overall agreement on threats is 0.401 and therefore higher than that of LINDDUN Pro. However, the agreement per threat type shows that the approach only outperforms the baseline in a few threat types. The initial translation is evaluated to cover 50.77% of all threats and 70.21% of leaf threats, leaving a significant amount of threats unidentifiable.

7.2. Future Work

As previously discussed, the proposed approach has some limitations that remain to be solved. The greatest point of future work is improving the labels and constraints, as the approach seems to not identify a lot of threats. This may be improved by iteratively detailing the labels and constraints. For example, the mapping table [22] or hotspot information [50, 21] from LINDDUN can be included to further specify existing LINDDUN constraints. However, this would require an extension of the DSL, as the current version does not yet support using vertexes for source and destination selectors in the same constraint.

Another point of future work is the improvement of the TKI. For example, the ability to model key ownership in the approach may improve the approach's accuracy. This could be achieved by using variables in xDECAF similar to role-based access control (RBAC), though the two do not behave exactly the same.

Furthermore, improving the approach's embedding in the greater threat modeling process may prove fruitful. Currently, it can only export threats which still have to be looked at, filtered and treated by security professionals. If the tool can suggest fitting Privacy Enhancing Technologys (PETs) or privacy pattern [30] based on its analysis, this could improve the threat modeling process. Additionally, the described translation process could be applied to other taxonomies, for example STRIDE [18].

7.3. Acknowledgments

I want to thank DistriNet Research Unit, KU Leuven and especially Dimitri Van Landuyt for their cooperation, which made the evaluation of this thesis possible. Additionally, I sincerely thank my advisor Nicolas Boltz for his continuous support, advice and feedback throughout this thesis.

Bibliography

[1] Victor R Basili and David M Weiss. "A methodology for collecting valid software engineering data". In: *IEEE Transactions on software engineering* 6 (1984), pp. 728–738.

[2] Leonard J. Bass, Paul C. Clements, and Rick Kazman. "Software architecture in practice". In: *SEI series in software engineering*. 1999. URL: https://api.semanticscholar.org/CorpusID:108931126.

[3] Bernhard J. Berger, Karsten Sohr, and Rainer Koschke. "Automatically Extracting Threats from Extended Data Flow Diagrams". In: *Engineering Secure Software and Systems*. Cham: Springer International Publishing, 2016, pp. 56–71. ISBN: 978-3-319-30806-7.

[4] Nicolas Boltz et al. "A Model-Based Framework for Simplified Collaboration of Legal and Software Experts in Data Protection Assessments". en. In: (2022). ISSN: 1617-5468. DOI: 10.18420/INF2022_44. URL: http://dl.gi.de/handle/20.500.12116/39544 (visited on 05/01/2026).

[5] Nicolas Boltz et al. "An Extensible Framework for Architecture-Based Data Flow Analysis for Information Security". In: *Software Architecture. ECSA 2023 Tracks, Workshops, and Doctoral Symposium*. Springer Nature Switzerland, 2024, pp. 342–358. ISBN: 9783031663260. DOI: 10.1007/978-3-031-66326-0_21.

[6] Nicolas Boltz et al. "Enabling a model-driven workflow for ongoing interdisciplinary collaboration in legal threat modeling". en. In: *Information and Software Technology* 195 (July 2026), p. 108121. ISSN: 09505849. DOI: 10.1016/j.infsof.2026.108121. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950584926001102 (visited on 05/01/2026).

[7] Sourya Joyee De and Daniel Le Métayer. "PRIAM: A Privacy Risk Analysis Methodology". In: *Data Privacy Management and Security Assurance*. Springer International Publishing, 2016, pp. 221–229. ISBN: 9783319470726. DOI: 10.1007/978-3-319-47072-6_15.

[8] Tom Demarco. "Structured analysis and system specification". In: *New York: Yourdon Inc* (1978).

[9] Mina Deng et al. "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements". In: *Requirements Engineering* 16.1 (Nov. 2010), pp. 3–32. ISSN: 1432-010X. DOI: 10.1007/s00766-010-0115-7.

[10] "Denmark moving away from Microsoft". In: *The Nordic Times* (July 2025). URL: https://nordictimes.com/the-nordics/denmark/denmark-moving-away-from-microsoft/.

[11] D. Dolev and A. Yao. "On the security of public key protocols". In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 198–208. DOI: 10.1109/TIT.1983.1056650.

[12] Joseph L. Fleiss. "Measuring nominal scale agreement among many raters." In: *Psychological Bulletin* 76.5 (Nov. 1971), pp. 378–382. ISSN: 0033-2909. DOI: 10.1037/h0031619. URL: http://dx.doi.org/10.1037/h0031619.

[13] John Gruber. *Markdown Syntax Documentation*. URL: https://daringfireball.net/projects/markdown/syntax (visited on 04/22/2026).

[14] Sebastian Hahner. *Architecture-Based and Uncertainty-Aware Confidentiality Analysis*. Medium: PDF. 2025. DOI: 10.5445/IR/1000178700. URL: https://publikationen.bibliothek.kit.edu/1000178700 (visited on 05/01/2026).

[15] Sebastian Hahner et al. "Modeling Data Flow Constraints for Design-Time Confidentiality Analyses". In: *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, Mar. 2021, pp. 15–21. DOI: 10.1109/icsa-c52384.2021.00009.

[16] Samantha Katcher et al. "The PANOPTIC™ Privacy Threat Model". In: *Twentieth Symposium on Usable Privacy and Security (SOUPS)*. 2024.

[17] Uday Khedker, Amitabha Sanyal, and Bageshri Sathe. *Data flow analysis: theory and practice*. CRC Press, 2017.

[18] Loren Kohnfelder and Praerit Garg. "The threats to our products". In: *Microsoft Interface, Microsoft Corporation* 33 (1999), pp. 1–8.

[19] Patrick Lam, Jens Dietrich, and David J. Pearce. "Putting the semantics into semantic versioning". In: *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2020. Virtual, USA: Association for Computing Machinery, 2020, pp. 157–179. ISBN: 9781450381789. DOI: 10.1145/3426428.3426922. URL: https://doi.org/10.1145/3426428.3426922.

[20] Lauren Leffer. "Your personal information is probably being used to train generative AI models". In: *Scientific American* 34.1s (Mar. 2023), p. 48. URL: https://www.scientificamerican.com/article/your-personal-information-is-probably-being-used-to-train-generative-ai-models/.

[21] *LINDDUN Go Cards*. URL: https://downloads.linddun.org/linddun-go/default/v241203/go.pdf (visited on 04/28/2026).

[22] *LINDDUN Pro Mapping Table*. URL: https://linddun.org/instructions-for-pro/#mappingtable (visited on 04/28/2026).

[23] *Microsoft Threat Modeling Tool*. Aug. 2022. URL: https://learn.microsoft.com/de-de/azure/security/develop/threat-modeling-tool (visited on 04/28/2026).

[24] Majid Mollaeefar et al. "PILLAR: LINDDUN Privacy Threat Modeling Using LLMs". In: *2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2025, pp. 278–286. DOI: 10.1109/EuroSPW67616.2025.00038.

[25] Filip Moons and Ellen Vandervieren. “Measuring agreement among several raters classifying subjects into one or more (hierarchical) categories: A generalization of Fleiss’ kappa”. In: *Behavior Research Methods* 57.10 (Sept. 2025). ISSN: 1554-3528. DOI: 10.3758/s13428-025-02746-8. URL: http://dx.doi.org/10.3758/s13428-025-02746-8.

[26] Nils Niehues, Sebastian Hahner, and Robert Heinrich. “An Architecture-Based Approach to Mitigate Confidentiality Violations Using Machine Learning”. In: *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. Odense, Denmark: IEEE, Mar. 2025, pp. 107–118. ISBN: 979-8-3315-2090-8. DOI: 10.1109/ICSA65012.2025.00020. URL: https://ieeexplore.ieee.org/document/10978930/ (visited on 05/01/2026).

[27] Nils Niehues, Sebastian Hahner, and Robert Heinrich. “Mitigation strategies for confidentiality violations in software architecture using ranked feature importance”. en. In: *Journal of Systems and Software* 235 (May 2026), p. 112761. ISSN: 01641212. DOI: 10.1016/j.jss.2025.112761. URL: https://linkinghub.elsevier.com/retrieve/pii/S0164121225004303 (visited on 05/01/2026).

[28] *OWASP Threat Dragon*. URL: https://owasp.org/www-project-threat-dragon/ (visited on 04/28/2026).

[29] *OWASP Threat Modeling Project*. URL: https://owasp.org/www-project-threat-modeling/ (visited on 04/28/2026).

[30] *Privacy Patterns*. URL: https://privacypatterns.org/ (visited on 04/30/2026).

[31] *Privacy threat trees | linddun.org*. URL: https://linddun.org/threat-trees/ (visited on 04/18/2026).

[32] Paul Quinn and Gianclaudio Malgieri. “The difficulty of defining sensitive data—The concept of sensitive data in the EU data protection framework”. In: *German Law Journal* 22.8 (2021), pp. 1583–1612.

[33] John M.M. Rumbold and Barbara K. Pierscionek. “What Are Data? A Categorization of the Data Sensitivity Spectrum”. In: *Big Data Research* 12 (2018). Big Data Centric Computational Intelligence in Bioinformatics and Healthcare, pp. 49–59. ISSN: 2214-5796. DOI: https://doi.org/10.1016/j.bdr.2017.11.001. URL: https://www.sciencedirect.com/science/article/pii/S2214579617302010.

[34] Per Runeson et al. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[35] Stephan Seifermann et al. “Detecting violations of access control and information flow policies in data flow diagrams”. In: *Journal of Systems and Software* 184 (Feb. 2022), p. 111138. ISSN: 0164-1212. DOI: 10.1016/j.jss.2021.111138.

[36] Adam Shostack. “Elevation of privilege: Drawing developers into threat modeling”. In: *USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. USENIX Association, 2014.

[37] Adam Shostack. *Threat modeling: Designing for security*. John wiley & sons, 2014.

[38] Laurens Sion et al. “Robust and reusable LINDDUN privacy threat knowledge”. In: *Computers & Security* 154 (2025), p. 104419. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2025.104419. URL: https://www.sciencedirect.com/science/article/pii/S0167404825001087.

[39] Laurens Sion et al. “Security Threat Modeling: Are Data Flow Diagrams Enough?” In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ICSE ’20. ACM, June 2020, pp. 254–257. DOI: 10.1145/3387940.3392221.

[40] Laurens Sion et al. “Solution-aware data flow diagrams for security threat modeling”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 1425–1432.

[41] Laurens Sion et al. “SPARTA: Security & Privacy Architecture Through Risk-Driven Threat Assessment”. In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 2018, pp. 89–92. DOI: 10.1109/ICSA-C.2018.00032.

[42] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. In: *Information processing & management* 45.4 (2009), pp. 427–437.

[43] *Threat Modeling Process | OWASP Foundation*. URL: https://owasp.org/www-community/Threat_Modeling_Process (visited on 04/30/2026).

[44] Peter Torr. “Demystifying the threat modeling process”. In: *IEEE Security & Privacy* 3.5 (2005), pp. 66–70.

[45] Dimitri Van Landuyt. “Privacy Impact Tree Analysis (PITA): A Tree-Based Privacy Threat Modeling Approach”. In: *IEEE Transactions on Software Engineering* 51.7 (July 2025), pp. 2102–2124. ISSN: 2326-3881. DOI: 10.1109/tse.2025.3573380.

[46] “Von LibreOffice bis Linux: Schleswig-Holstein setzt konsequent auf OSS”. In: *heise online* (Sept. 2025). URL: https://www.heise.de/hintergrund/Interview-Wie-die-OSS-Umstellung-in-Schleswig-Holstein-laeuft-10629991.html.

[47] Tian Wang et al. “UsersFirst: A User-Centric Privacy Threat Modeling Framework for Notice and Choice”. In: (2024).

[48] *What personal data is considered sensitive? - European Commission*. URL: https://commission.europa.eu/law/law-topic/data-protection/rules-business-and-organisations/legal-grounds-processing-data/sensitive-data/what-personal-data-considered-sensitive_en (visited on 04/30/2026).

[49] Kim Wuyts, Riccardo Scandariato, and Wouter Joosen. “Empirical evaluation of a privacy-focused threat modeling methodology”. In: *Journal of Systems and Software* 96 (2014), pp. 122–138.

[50] Kim Wuyts, Laurens Sion, and Wouter Joosen. *LINDDUN GO: A Lightweight Approach to Privacy Threat Modeling*. eng. 2020.

[51] *xDECAF - An extensible data flow diagram constraint analysis framework for information security*. 2024. URL: https://dataflowanalysis.org/ (visited on 11/20/2025).

[52] Antonia Zapf et al. “Measuring inter-rater reliability for nominal data – which coefficients and confidence intervals are appropriate?” en. In: *BMC Medical Research Methodology* 16.1 (Dec. 2016), p. 93. ISSN: 1471-2288. DOI: 10.1186/s12874-016-0200-9. URL: http://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-016-0200-9 (visited on 04/28/2026).

A. Appendix

A.1. All Constraints

Table A.1 shows all 24 derived constraints.

Constraint	Threat(s)
data DataPrecision.ExcessiveFrequency neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	DD.2.2, DD.1.2, DD.3.2
data DataPrecision.ExcessiveVolume neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	DD.2.1, DD.3.2, DD.2.3
data DataIntegrity.Signed neverFlows vertex NodeControl.Organization	Nr.1.2
data DataPrecision.ExcessiveDataTypes neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	DD.1.1, DD.1.2, DD.3.2, DD.1.3, Nc.1.1.2
data DataPrecision.ExcessiveEnrichment neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	DD.3.1
data DataSensitivity.PersonalData DataPrecision.ExcessiveRetention neverFlows vertex NodeControl.Organization	DD.3.4, Nc.1.1.4
data !DataInterveniability.ControlViaPreferences neverFlows vertex NodeControl.Organization	U.2.1
data !DataInterveniability.Accessible neverFlows vertex NodeControl.Organization	U.2.2
data !DataInterveniability.RectificationPossible neverFlows vertex NodeControl.Organization	U.2.3
data !DataInterveniability.Awareness neverFlows vertex NodeControl.Organization	U.1.2, U.1.1
data DataIdentifiers.DirectIdentifier neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	L.1.1
neverFlows vertex NodeControl.ThirdParty,NodeControl.Public	DD.4.1.1, DD.4.1.2
data DataIdentifiers.QuasiIdentifier neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	L.2.1.1
data DataSensitivity.PersonalDataAboutOtherPeople neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	L.2.1.2
data DataSensitivity.PersonalData neverFlows vertex NodeControl.Organization,NodeControl.ThirdParty	L.2.2.1
data DataSensitivity.PersonalData neverFlows vertex NodeControl.Public	DD.4.2

data Data.HasSideEffects neverFlows	Nr.2, D.2
data Data.ContainsHiddenData neverFlows vertex NodeControl.Organization	Nr.1.4
data Data.HasSideEffects neverFlows vertex type STORE	DD.3.3
data DataIdentifiers.DirectIdentifier neverFlows vertex NodeControl.Organization	I.1.1, I.2.1.1
data DataIdentifiers.DirectIdentifier DataObservability.MetadataOnly neverFlows vertex NodeControl.Organization	I.1.2
data DataIdentifiers.QuasiIdentifier neverFlows vertex NodeControl.Organization	I.2.1.2, I.2.1.1
data DataIdentifiers.RevealingAttributes neverFlows vertex NodeControl.Organization	I.2.2
data DataIdentifiers.DirectIdentifier,DataIdentifiers.QuasiIdentifier neverFlows vertex type STORE	I.2.1.2

Table A.1.: Translated constraints.

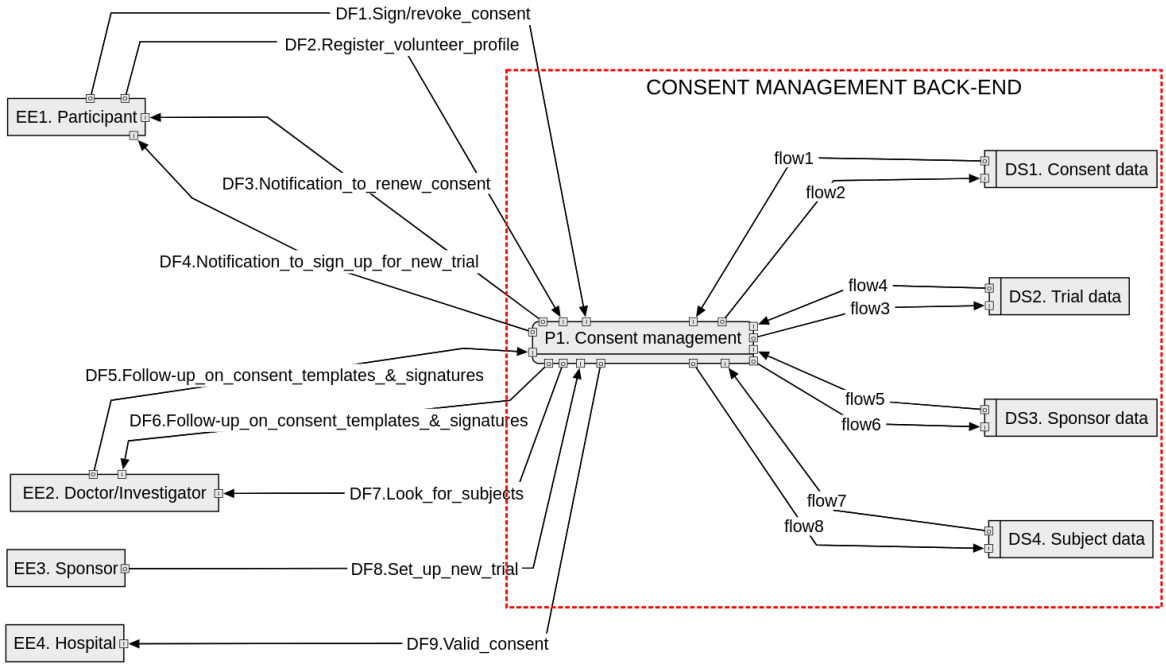
A.2. eConsent System Description

Freely-given informed consent is required for all clinical trials, as regulated via the Nuremberg Code (1947), declaration of Helsinki (1964), and other regulations, e.g., Good Clinical Practice (GCP). The goal is to inform participant on purpose, procedures, risks, benefits, confidentiality, etc of a clinical trial in an objective and understandable manner. These consent templates are verified by an ethical committee whether information is adequate, and external auditors (may) check whether consent procedure is properly adhered to.

Participant must sign a consent form before they enter the trial. The Sponsor or Investigator must countersign as well. Signed consent forms must be stored for several years after the end of the trial. A typical consent form on paper is 10-40 pages, and provides a mix of general and study-specific information. Participants must re-read and sign on every change. Information must be updated when major changes are introduced and participants need to re-consent after changes.

The e-consent platform is a solution that allow retrieving past consent forms, remote consenting, withdrawing, etc. A typical procedure is that a doctor gives the participant a tablet to read and sign consent form after a conversation. Once signed, everything is converted into a (signed) PDF and stored.

DFD Description



The data flows of the e-consent platform are as follows:

The "EE1. (Potential) participant" is an external entity that interacts with the "P1. Consent management" process to invoke a "DF2. Register volunteer profile" data flow. The "P1. Consent management" process subsequently registers the participant in the "DS1. Consent data" data store which records patient info and medical trial info and in the "DS4. Subject data" data store which records medical history, and interests/preferences. The "P1. Consent management" process sends "DF3. Notification to renew consent" to the "EE1. (Potential) participant" The "EE1. (Potential) participant" is offered also the ability to "DF1. Sign/ revoke consent" to the "P1. Consent management" which will be sent to "DS1. Consent data" data store which holds dates of consent and revoked consent. The "P1. Consent management" process also sends "DF4. Notification to sign up for a new trial" to the "EE1. (Potential) participant"

The "EE3. Sponsor" is able to invoke "DF8. Set up a new trial" to the "P1. Consent management" process. The "P1. Consent management" process registers this in the "DS2. Trial data" data store which records a template, required criteria and time period information. Additionally, the "P1. Consent management" process records sponsor information (organization, team and trials) in the "DS3. Sponsor data" data store.

After that, the "EE2. Doctor/investigator" is able to invoke "DF7. Look for subjects" to the "P1. Consent management" process. The "P1. Consent management" process collects this information from the "DS4. Subject data" data store (based on medical history, interests/preferences). The "EE2. Doctor/investigator" is also enabled to "DF5. Follow-up on consent templates & signatures" to the "P1. Consent management" process. Vice versa is the "P1. Consent management" process enabled to invoke "DF6. Follow-up on consent templates & signatures" to the "P1. Consent management" process.

Finally, upon conducting a clinical study, the "EE4. Hospital" is able to invoke "DF9. Valid consent" to the "P1. Consent management" process. The "P1. Consent management" process will then verify consent validity via the "DS1. Consent data" data store.